# Building Knowledge Graphs

## Processing Infrastructure and Named Entity Linking

Klang, Marcus

2019

*Document Version:*
Publisher's PDF, also known as Version of record

# Building Knowledge Graphs:

## Processing Infrastructure and Named Entity Linking

**Marcus Klang**


Doctoral Dissertation, 2019

Department of Computer Science
Lund University

ISBN 978-91-7895-286-1 (printed version)
ISBN 978-91-7895-287-8 (electronic version)
ISSN 1404-1219
LU-CS-DISS: 2019-04
Dissertation 64, 2019

Department of Computer Science
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: marcus.klang@cs.lth.se
WWW: http://cs.lth.se/marcus-klang

Typeset using LaTeX
Printed in Sweden by Tryckeriet i E-huset, Lund, 2019

# Abstract

Things such as organizations, persons, or locations are ubiquitous in all texts circulating on the internet, particularly in the news, forum posts, and social media. Today, there is more written material than any single person can read through during a typical lifespan. Automatic systems can help us amplify our abilities to find relevant information, where, ideally, a system would learn knowledge from our combined written legacy. Ultimately, this would enable us, one day, to build automatic systems that have reasoning capabilities and can answer any question in any human language.

In this work, I explore methods to represent linguistic structures in text, build processing infrastructures, and how they can be combined to process a comprehensive collection of documents. The goal is to extract knowledge from text via things, *entities*. As text, I focused on encyclopedic resources such as Wikipedia.

As knowledge representation, I chose to use graphs, where the entities correspond to graph nodes. To populate such graphs, I created a named entity linker that can find entities in multiple languages such as English, Spanish, and Chinese, and associate them to unique identifiers. In addition, I describe a published state-of-the-art Swedish named entity recognizer that finds mentions of entities in text that I evaluated on the four majority classes in the Stockholm-Umeå Corpus (SUC) 3.0.

To collect the text resources needed for the implementation of the algorithms and the training of the machine-learning models, I also describe a document representation, *Docria*, that consists of multiple layers of annotations: A model capable of representing structures found in Wikipedia and beyond. Finally, I describe how to construct processing pipelines for large-scale processing with Wikipedia using Docria.

# Contents

# Preface

## List of Included Publications

I **Named entity disambiguation in a question answering system.**
*Marcus Klang and Pierre Nugues*
In *Proceedings of the The Fifth Swedish Language Technology Conference (SLTC 2014)*, Uppsala, November 13-14 2014.

II **WIKIPARQ: A tabulated Wikipedia resource using the Parquet format.**
*Marcus Klang and Pierre Nugues.*
In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4141–4148, Portoro, Slovenia, May 2016.

III **Docforia: A multilayer document model.**
*Marcus Klang and Pierre Nugues.*
In *Proceedings of The Sixth Swedish Language Technology Conference (SLTC 2016)*, Umeå, November 2016.

IV **Multilingual supervision of semantic annotation..**
*Peter Exner, Marcus Klang, and Pierre Nugues.*
In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1007–1017, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

V **Langforia: Language pipelines for annotating large collections of documents.**
*Marcus Klang and Pierre Nugues.*
In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 74–78, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

VI **Overview of the Ugglan entity discovery and linking system.**
*Marcus Klang, Firas Dib, and Pierre Nugues.*

In *Proceedings of the Tenth Text Analysis Conference (TAC 2017)*, Gaithersburg, Maryland, November 2017.

VII **Linking, searching, and visualizing entities in Wikipedia.**
*Marcus Klang and Pierre Nugues.*
In *Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors,* Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3426–3432*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA).

VIII **Comparing LSTM and FOFE-based architectures for named entity**
*Marcus Klang and Pierre Nugues.*
In *Proceedings of the The Seventh Swedish Language Technology Conference (SLTC 2018)*, pages 54–57, Stockholm, October 7-9 2018.

IX **Docria: Processing and storing linguistic data with Wikipedia.**
*Marcus Klang and Pierre Nugues.*
In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, Turku, October 2019.

X **Hedwig: A Named Entity Linker**
*Marcus Klang*
To be submitted.

# Contribution Statement

Marcus Klang is the main contributor to all the papers included in this doctoral thesis when listed as first author. He was the main designer and implementor of the research experiments and responsible for most of the writing.

In the paper *Overview of the Ugglan Entity Discovery and Linking System*, Firas Dib contributed the FOFE-based named entity recognizer that was used as part of the mention detection system. Peter Exner was the main contributor in the papers *A Distant Supervision Approach to Semantic Role Labeling* and *Multilingual Supervision of Semantic Annotation*, with Marcus Klang contributing the named entity linker used to produce part of the input to the system. In the paper *Linking, Searching, and Visualizing Entities for the Swedish Wikipedia*, Marcus Klang contributed infrastructure tools and resources.

The supervisor Prof. Pierre Nugues contributed to the design of the experiments, writing of articles, and reviewed the content of the papers.

# List of Additional Publications

The following papers were related, but not included in this thesis. Specifically, papers XI, XIII, XIV were succeeded by paper IV.

XI **Using distant supervision to build a proposition bank.**
*Peter Exner, Marcus Klang, and Pierre Nugues.*
In *Proceedings of the The Fifth Swedish Language Technology Conference (SLTC 2014)*, Uppsala, November 13-14 2014.

XII **A platform for named entity disambiguation.**
*Marcus Klang and Pierre Nugues.*
In *Proceedings of the workshop on semantic technologies for research in the humanities and social sciences (STRiX)*, Gothenburg, November 24-25 2014.

XIII **A distant supervision approach to semantic role labeling.**
*Peter Exner, Marcus Klang, and Pierre Nugues.*
In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 239–248, Denver, Colorado, June 2015. Association for Computational Linguistics.

XIV **Generating a Swedish semantic role labeler.**
*Peter Exner, Marcus Klang, and Pierre Nugues.*
In *Proceedings of The Sixth Swedish Language Technology Conference (SLTC 2016)*, Umeå, November 2016.

XV **Linking, searching, and visualizing entities for the Swedish Wikipedia.**
*Anton Södergren, Marcus Klang, and Pierre Nugues.*
In *Proceedings of The Sixth Swedish Language Technology Conference (SLTC 2016)*, Umeå, November 2016.

XVI **Pairing wikipedia articles across languages.**
*Marcus Klang and Pierre Nugues.*
In *Proceedings of the Open Knowledge Base and Question Answering Workshop (OKBQA 2016)*, pages 72–76, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

# Acknowledgements

First, I would like to express my deepest gratitude to my supervisor, Professor Pierre Nugues. For his steadfast support, inspiration, patience, and outstanding ability to bring clarity when you need it the most. I am most grateful that he introduced me to the field of natural language processing.

I also would like to thank my colleagues in the Department of Computer Science. For their inspiration, many interesting discussions, and their ability to sustain a wonderful, curious atmosphere that makes it an excellent workplace. Specifically, I want to thank Peter Möller, Lars Nilsson, Anders Bruce, and Tomas Richter for all their support, invaluable technical advice, and rewarding discussions.

Vetenskapsrådet, *The Swedish Research Council*, funded my research through the *Det digitaliserade samhället* program, grant number 340-2012-5738, and made this work materially possible. I also benefited from a gift from Nvidia in the form of a graphical processing unit.

Special thanks to Dr. Peter Exner and my roommate Dr. Dennis Medved for their invaluable advice, inspiration, and stimulating discussions.

Finally, I would like to profoundly thank my parents Carina and Tommy Klang for their unwavering support, optimism, warmth, and encouragement.

*Marcus Klang*
*Lund, September 2019*

# Popular Science Summary in Swedish

# Att lära datorn hitta kunskap i text genom namngivna saker

**Marcus Klang**
Institutionen för datavetenskap
Lunds universitet
Lund, Sverige
marcus.klang@cs.lth.se

Det finns idag massvis med kunskap på Internet och det ökar konstant, långt mer än en enskild människa kan ta till sig under en livstid. Hur lär vi datorn att hitta kunskap och på så sätt förstärka människans förmåga?

## 1 Namngivna saker

I det svenska språket så använder vi namn för att hänvisa till saker och ting. Ta meningen:

> Jag köpte kaffe på LED-café i E-huset.

Här finns två namngivna saker: *LED-café* och *E-huset*. Utifrån meningen kan vi också utläsa att LED-café är en plats där man kan köpa kaffe. Det finns en angivelse om var detta café finns: i E-huset. Men denna beskrivning är inte komplett. Läsaren förväntas känna till var E-huset finns. Låt säga att vi kände till alla platser i världen och hur dessa förhåller sig till varandra, ett nätverk av saker. Då kan vi slå upp E-huset och utläsa att det finns i Lund. Genom detta nätverk så finns information som att staden Lund finns i Skåne som tillhör landet Sverige på planeten Jorden och så vidare.

Kunskap om namngivna saker kan alltså användas för att utöka och komplettera information som kommer i bitar och på så viss tillföra mer information än vad som fanns från början. Detta kan användas för att hjälpa datorer att förstå innehåll och kopplingar i en text. Människor kan också hjälpas genom att datorn kan påvisa kopplingar och tillföra bakgrund till ett inlägg eller nyhetsartikel som en läsare möjligen inte kände till.

Andra användningsområden inkluderar att sammanfatta vad en nyhetsartikel handlar om baserat på vilka namngivna saker som nämns, som t ex personer, organisationer och platser. Ett nätverk av saker kan också användas för att ge svar på frågor som genom att koppla fakta till saker:

> Hur många är bosatta i Lund?

Namngivna saker fungerar som ingångar i ett nätverk som kan användas för tillföra fakta och kopplingar mellan olika saker.

### 1.1 Uppslagsbok

Att manuellt skapa en uppslagsbok och ett nätverk av namngivna saker och ting tar mycket tid. Mitt arbete handlar om att göra detta automatiskt från information som redan finns och det börjar med att lära en dator känna igen namngivna saker i en text.

Det som då behövs är ett uppslagsverk över namngivna saker. Wikipedia[1], är en fri encyklopedi online med artiklar skrivna av användare runtom i världen och täcker idag en stor mängd ämnen samt finns i 294 editioner, mestadels motsvararande riktiga språk.

Många artiklar på Wikipedia berör namngivna saker som personer, platser, organisationer, skapade verk och mycket mer. Genom Wikipedia kan vi skapa en grundläggande katalog av namngivna saker. Artiklar på Wikipedia innehåller också länkar till andra artiklar vilket möjliggör att en dator automatiskt kan finna kopplingar mellan saker bara för att de ofta nämns ihop.

[1] https://www.wikipedia.org/

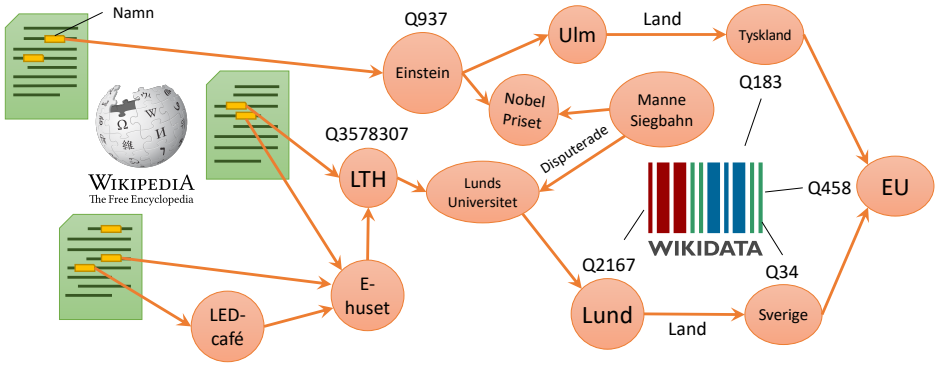



Figure 1: Att gå från artiklar på Wikipedia till ett nätverk av saker.

Tillsammans med Wikipedia finns ett systerprojekt vid namn Wikidata[2] som har som mål att skapa ett nätverk av saker genom manuellt arbete, en s.k. kunskapsgraf. Wikidata ger unika namn till artiklar på Wikipedia med något som liknar ett personnummer, ett s.k. Q-nummer. I Wikidata benämns Sverige som `Q34` och Lund `Q2167`. Wikidata innehåller fakta som exempelvis antalet personer bosatta i Lund.

Wikidata kopplar också ihop artiklar skrivna på olika språk vilket möjliggör att vi kan ta del av kunskap från många olika språk. Wikidata är ett pågående projekt som inte är komplett där en del information endast finns som brödtext inuti en artikel.

Vi använder Wikidata som en uppslagsbok över vilka saker som kan kopplas och använder det för att slå samman information från artiklar skrivna på 9 språk i Wikipedia[3]. Denna sammanslagningen möjliggjorde att vi kunde skapa ett system som kan hitta och koppla namngivna saker i språk som engelska, spanska och kinesiska.

En egenskap som Wikipedia, Wikidata, och världen i stort har är att det finns namn som kan betyda flera saker. Exempelvis, stadsnamnet ”Lund” är ett namn som använts på städer i Norge, England, Kanada och USA. Att finna vilken namngiven sak som passar ett namn är kärnpunkten i mitt arbete och beror på omgivningen där namnet nämns.

För att förenkla problemet något, så har fokuset varit att länka namn av utvalda kategorier: personer, organisationer, platser, byggnader, flygplatser, m.m.

## 2 Hitta namn

Att lära en dator hitta namn kan initialt ske genom att skapa ett antal enkla regler, åtminstone för språk som liknar svenska. Men förr eller senare så kommer antalet regler bli så stort att det kräver mycket tid för att utforma dessa. Som alternativ kan denna process automatiseras genom att låta en dator hitta mönster och på så sätt komma på regler själv. Det som krävs är exempel på namn och var de finns i en mening. Detta kallas för maskininlärning. Den metod som vi tillämpat är s.k. artificiella neurala nätverk som är gjorda för sekvenser[4] anpassade för att lära sig från ord i följd.

Genom dessa neurala nätverk så kan vi lära en dator hitta namn som finns i vanliga meningar. Dessutom kan nätverken säga om namnet speglar en person, plats, eller organisation, m m.

### 2.1 Koppla namngivna saker

När väl namn har identifierats, så ska dessa kopplas till vår uppslagsbok bestående av Q-nummer i Wikidata. Detta gör vi genom att

[2] `https://www.wikidata.org`

[3] Svenska, engelska, tyska, spanska, kinesiska, ryska, franska, danska och norska

[4] Eng. Recurrent Neural Network, specifikt LSTM

analysera vilka saker som ofta förekommer ihop eller hur saker specifikt namnges. På så sätt väljer vi det Q-nummer som är mest relevant bland på de alternativ som finns.

## 3 Infrastruktur

Genom att kombinera alla 9 språk som vi använt så får vi en stor samling skriven text. Bara engelska Wikipedia för sig själv innehåller vid skrivande stund 5.9 miljoner artiklar och över 3 miljarder ord.

För att kunna samla in relevant information från textkälla som denna krävs en infrastruktur. En viktig aspekt är hur man kan representera dessa artiklar och hur man kan tillföra information som namn och kopplingar till saker.

En viktig bit när man har mycket text som ska bearbetas är att det kräver mycket beräkningskraft, vilket innebär att flera datorer behövs för att lösa uppgiften inom rimlig tid.

### 3.1 Struktur

Vårt språk består av ord i meningar, som kan samlas i stycken och i flera nivåer tills det exempelvis kan utgöra en bok. Orden vi använder och vilken följd dessa har är inte slumpmässig, det finns en ordning. Man kan t ex tillföra ordklass, alltså om ordet är ett substantiv som ”Lund” eller ett verb som ”hitta”, preposition som ”under”, pronomen som ”hen” osv. Vi kan utöver detta t ex koppla ihop namn som ”Anna” och ”hon”, ”Björn” och ”han” som nämns upprepade gånger i en text.

Dessa *strukturer* som återfinns i vårt språk kan vi lära en dator att hitta. Tyvärr är det en uppgift som ofta kräver att flera människor och forskare arbetar tillsammans. Konkret innebär det att vi delar program och lösningar med varandra som är gjorda för att infoga olika strukturer. Kruxet är att alla dessa program använder olika språk för att säga hur en struktur ska se ut.

Vi har utvecklat en metod som kan spara dessa strukturer och så att säga förena olika språk och samtidigt hantera successiv utökning av nya strukturer med fokus på stora mängder text. Idén går ut på att konvertera strukturen till lager. Tänk dig ett kalkerpapper som ligger ovanpå varandra. Ett kalkerpapper där man har understrukit var ord finns med ordklasserna undertill substantiv, verb, osv. Ett annat papper innehåller var namnen finns, ett beroende mellan ord, osv.

Detta sätt att lagra struktur kan användas för att tillföra viktig kunskap i text som sedan kan delas, förändras, och utökas med ny information givet andra metoder.

## 4 Kunskap från nätverk

Vi kan slutligen använda namngivna saker och identifiera att t ex ”Måns Zelmerlöv” är son till ”Birgitta Sahlén”. Att ”Lund” hade en befolkningsmängd på 91 940 (2018) och att Sverige tillhör EU. Denna information har vi lärt en dator att känna igen och kan i framtiden använda för att besvara frågor.

## 5 Sammanfattning

Genom att använda namngivna saker i vanlig text så kan vi tillfoga kunskap och bakgrund genom ett nätverk av saker. Ett nätverk som kan hjälpa datorer och människor att förstå innehållet. Genom en infrastruktur som kan bearbeta miljarder de ord som finns i Wikipedia så kan vi hitta namn och koppla saker till ett nätverk av saker som Wikidata.

# Introduction

## 1 Introduction

This thesis deals with the construction of repositories containing facts or things derived from collections of texts. In the past 30 years, the online availability of such collections, also called *corpus*, plural *corpora*, literally "body" in Latin, has increased considerably. Their style, genre, and quality vary from the free-form nature of tweets, comments, and forum posts to the stricter style of books, encyclopedic, and news articles.

By reading and processing texts, we can gather a wealth of knowledge. In addition, using prior structured knowledge, in the form of *graphs*, allows us to reference known things and add new connections between them.

The focus of this thesis is to identify things in text, such as persons, organizations, locations, works of art, etc. and disambiguate them into unique references. We can then use these things, also called *entities*, as nodes in a graph to create a *web of knowledge*. Building automatic systems to carry this out requires first some prior structured facts on the entities and how they relate. Then, mapping entities in text to nodes in graphs requires knowledge from corpora. For open-domain identification, this entails that corpora must cover a large range of topics. This problem is easier to solve, in practice, when the topics or entities are well delimited in the text, i.e. when the texts are divided into sections which loosely correspond to an entity.

Automatic systems use *machine learning*: Techniques allowing a machine, for instance, to create semantic models of entities and then identify them in a text. The machine learning models are trained to find patterns in natural language; they often rely on the manual annotation of structures in texts or on statistical patterns derived from large corpora. The size of the collected corpora is important as natural language evolves and changes. This thesis deals with contemporary language and this enabled us to use pretrained language models and other existing resources.

Wikipedia is a free online *encyclopedia* with articles on a large range of topics, hosted by the Wikimedia Foundation. Wikipedia's content is written by users

from all around the world; divided into 294 active editions[1], where most of them correspond to distinct languages. Wikipedia is permissively licensed and can be downloaded in bulk, making it one of the largest available textual resources with few restrictions online. This kind of corpora matches the openness, contemporary language, size, and variation needed to serve as the primary knowledge source for this thesis.

We used Wikipedia to bootstrap the identification of what an entity is and how entities relate. However, Wikipedia is not the golden standard of truth and one must always consider the risk of untruth as it is user-driven with few restrictions on what can be added. We assumed, nonetheless, that Wikipedia has enough content, users, and a proper moderation by the community itself, so that it is as close as we can get to the truth today in a freely accessible resource. Most importantly, it is a resource accessible and modifiable by anyone with an open internet access.

Processing and transforming large corpora with millions of documents such as Wikipedia requires a practical methodology, a *processing infrastructure*. Although some frameworks exist such as UIMA (Ferrucci & Lally, 2004), these frameworks introduce undesired complexity when used as an evolutionary and prototyping tool in research. The same applies to the construction of processing pipelines, introducing technical challenges in executing and combining the output from different tools within a pipeline. Therefore to process the encyclopedic corpora and identify entities, I designed and built new infrastructure tools. In this thesis, I focused on three essential parts of such an infrastructure: document representations, efficient storage techniques, and methods to create processing pipelines operating in parallel distributed across multiple computers.

## 1.1 Outline

In this part of my dissertation, I explain and connect the elements I designed to create graphs of entities from text. It consists of a machine learning pipeline for named entity recognition and entity linkage. In addition, as the volume of data requires a nontrivial architecture, I describe how I built the infrastructure and document data representations.

The outline of this introductory chapter is as follows:

- Section 2, *Natural language processing*, introduces the concept of a knowledge graph and the spectrum of structure complexity;
- Section 3, *Corpus*, describes the main text collections I used in this thesis;
- Section 4, *Infrastructure*, introduces frameworks I designed for large-scale computation and considerations when adding structure and processing text;

[1] `https://en.wikipedia.org/wiki/List_of_Wikipedias`

- Section 5, *Machine learning* (ML), describes ML concepts in relation to this work;
- Section 6, *Data representation*, describes how to structure text for machine learning;
- Section 7, *Models*, introduces machine learning models I used in this work;
- Section 8, *Document database*, introduces a document indexing tool and an online frontend to annotate documents.
- Section 9, *Named entity recognition*, introduces methods to find names in arbitrary text;
- Section 10, *Named entity linking*, takes the recognized names and links them to an entity repository.

# 2 Natural Language Processing (NLP)

The Internet has profoundly changed the way ideas and information reach us from distant places. The steady increase of daily text production throughout the 21st century has made it more difficult to overview and navigate available information. Its amount today is more extensive than any single human being can read under a typical lifespan. Machines can hopefully provide a way to amplify our intellectual abilities through the development of systems that understand natural language. It is already the case that these machines can process and transform text into structured resources that can serve as the basis for many applications.

Typical NLP applications include spell-checkers, word predictors, multilingual translation systems, grammar checkers, synonym generators, the automatic construction of knowledge bases, entity search, interactive assistants or agents such as Apple Siri, Amazon Alexa, Microsoft Cortana or, Google Assistant, and much more.

Systems built using NLP often start from small to large collections of text from various sources such as news articles, the web, or books. In this thesis, I mostly used multilingual versions of Wikipedia (Klang, Dib, & Nugues, 2017; Klang & Nugues, 2016a, 2016b, 2016c, 2017, 2018b, 2019b).

In the next sections, I outline the building blocks of machine reading from the lowest level of textual representation in machines, to syntax, semantics, and finally discourse or top-level structures.

## 2.1 Knowledge Graph

In this section, I discuss how knowledge graphs can represent entities and knowledge about them.

**Definition.** There are multiple definitions of what a knowledge graph is in the literature. Ehrlinger and Wöß (2016) provide a selection of plausible definitions and an attempt at a single definition. In my thesis, I used Wikidata, a Wikimedia Foundation project and I followed their definition of a knowledge graph.

Wikidata is a database describing *things* found in Wikipedia. These things frequently correspond to a single Wikipedia page as, for instance, the *city of Lund*. Each *thing* is called an *item* in the vocabulary of Wikidata. Figure 1 shows that various kinds of information that can be attached to these items: Factual statements that describe a property such as where a person has been educated, a short description of the item, possible aliases, references to these property-value claims, etc.

In Wikidata, each thing is uniquely identified using a naming system made up of a single letter and a number, e.g *Earth* is identified as Q2 and *Sweden* as Q34. Properties use a different letter: `P`, e.g. the *mass* property for *Earth* is `P2067`; the *official name* of Sweden is described by a statement having the type of `P1448`.

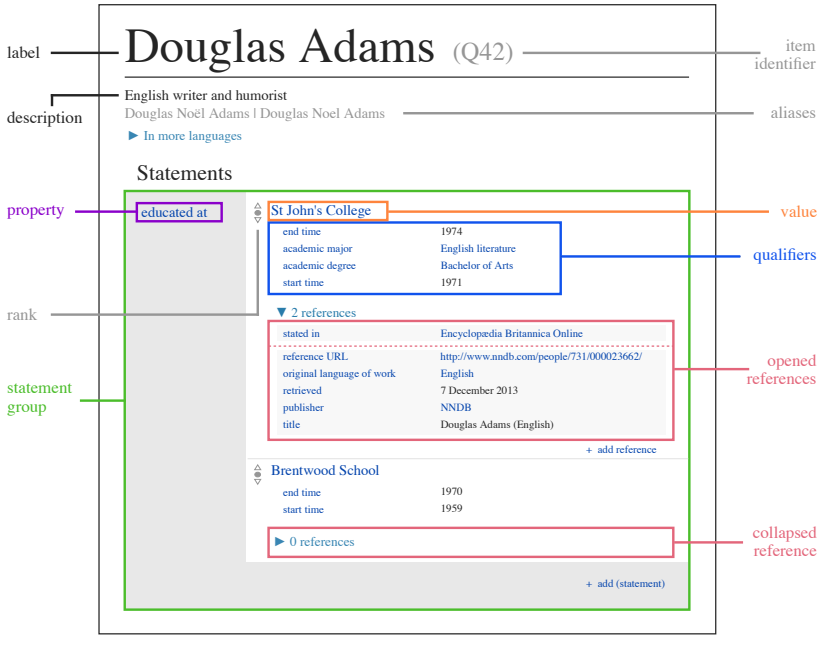



**Figure 1:** Wikidata data model, image source: `https://commons.wikimedia.org/wiki/File:Datamodel_in_Wikidata.svg`

**Example.** As a complete example of how a text is transcribed into a graph structure in Wikidata, the sentence:

> Douglas Adams was educated at St. John's College between 1971 and 1974.

is represented as a *statement*, see Figure 1. The statement consists of a *property* with a *value* and a list of references backing it up. In this example, the *property* `educated at` (`P69`) contains a *value* with a reference to another entity: St John's College (`Q691283`). In addition, metadata such as the academic degree of Bachelor of Arts (`Q1765120`) is also attached. One of the references backing up this claim refers to the entity *Encyclopædia Britannica Online* (`Q5375741`).

**Properties and entities.** The properties themselves, such as `educated at` (`P69`), are part of the graph. This property has attached examples, references to other relevant properties, a short description of its meaning, allowed values, and more. This higher level self-description of the graph structure is the *ontology* of Wikidata.

The key elements of a knowledge graph consist of uniquely identified entities, relationships between them, connected information such as facts, and a description of its structure: an *ontology*. In addition, a possible additional reasoning component may contain rules which accelerates queries and enables inference in the graph to extract usable information. All this is precisely the definition of Wikidata. I used it as a basis for the identification of entities over many languages and the classification of entity types (Exner, Klang, & Nugues, 2015, 2016; Klang et al., 2017; Klang & Nugues, 2019b).

## 2.2 Structure

Raw information can be categorized along a spectrum of structure levels. This section gives a background of the concepts used when discussing the structure of data sources such as the Wikipedia corpus and knowledge graphs such as Wikidata.

I divide information into three distinct categories of structure:

**Unstructured data** lacks structure for an effective analysis and requires packaging and processing beforehand. This includes plain text, raw audio samples, etc.

**Semi-structured data** is unambiguously packaged, but may be ambiguously structured requiring further processing for an analysis: e.g. XML, JSON.

**Fully structured data** is unambiguously structured information suitable for analysis, e.g. relational tables, typed messages, or column based data.

Wikipedia articles are written as plain text using natural language with few restrictions. The lack of structure puts Wikipedia articles into the unstructured category. Nonetheless, Wikipedia articles may contain some structure defined by the user, such as sections, paragraphs, lists, and tables.

An example of a semi-structured resource is Wikidata, parsable in a language-independent way. However, its structure can vary requiring further processing for a practical analysis and therefore not a sufficiently structured resource. Knowledge in Wikidata is incomplete, and some information only exists as plain text in natural language in Wikipedia articles.

Adding structure to Wikipedia represents a large part of the corpus processing step, where the extraction of information requires a language-dependent analysis.

**Adding structure to natural language.** In the papers we published, we utilized three layers of existing structure (Exner et al., 2015, 2016; Klang et al., 2017; Klang & Nugues, 2019b):

- The logical structure of text;
- Entities in the text;
- Co-occurring entities per paragraph.

**The logical structure of text.** The text contains subdivisions such as sections and paragraphs in Wikipedia articles. These subdivision are typically not independent and information flows from one section to another with bidirectional references.

**Entities in text.** Wikipedia articles contain links referencing other pages within Wikipedia and often refer to entities. In this context, an entity in the text consists of two parts: the word sequence within a sentence, the *mention* of a thing, frequently consisting of a name, a noun, or a pronoun, and the actual entity, called the target. This has the form `[[target|mention]]` in the wiki syntax, or simply `[[mention]]`, if the target and mention strings are equal.

In a long article, some of these references might be multiply mentioned but not linked. Automatically finding chains of mentions of a same entity is the task of *co-reference resolution*. Expanding co-reference resolution into linkage over multiple documents with name references is the task of *entity linking*.

**Finally, co-occurring entities.** In paragraphs, sentences often connect entities, e.g:

> The Swedish Government sold the state-driven train company SJ.

Using this sentence as an example, something is *sold* by the entity *The Swedish Government*, and the entity sold was *train company SJ*. This sentence fits the semantic *frame* of the word *sell*, with three independent bits of information:

- a predicate *sold*,
- a seller, *The Swedish Government*, and
- a thing sold, *train company SJ*.

The task of identifying and classifying bits into roles is the task of *semantic role labeling* (SRL) in which words are given a semantic role. In SRL, a frame consists of a predicate usually a verb, and arguments referencing mentions. Co-occurrences of entities within a paragraph can be used to find relations that are often mentioned together. If links are resolved to Wikidata Q-numbers, this information can be combined from multiple Wikipedia editions representing different natural languages.

# 3 Corpus

This section describes shortly the corpora I used throughout my thesis and their particulars.

## 3.1 CoNLL03: Language Independent Named Entity Recognition

The CoNLL 2003 shared task on Language-Independent Named Entity Recognition (Tjong Kim Sang & De Meulder, 2003) was a competition designed to evaluate the performance of named entity recognition (NER) in English and German. As with the others CoNLL tasks, the competition was open to any participant and the organizers published the data.

The CoNLL 2003 dataset consists of two corpora annotated with named entities, divided into training, development, and test sets and an evaluation script. The CoNLL 2003 corpora used news articles from the *Reuters Corpus Volume I* (RCV1) in English and the *Frankfurter Rundschau* in German. One year earlier, CoNLL02 (Tjong Kim Sang, 2002) ran a similar task, this time with Spanish and Dutch corpora.

Both CoNLL 2003 and 2002 datasets are frequently cited resources for training and evaluating NER systems. These tasks define four classes of named entities[2]:

**Person (PER):** Name of real and fictional individuals;

**Location (LOC):** Regions (cities, countries, continents, etc.), public places (airports, markets, hospitals, etc.), natural locations (mountains, rivers, beaches, etc.);

**Organizations (ORG):** Institutions, organizations, companies, etc.

**Miscellaneous (MISC):** Events, languages, titles of works, etc.

The CoNLL02/03 corpora consist of sequences of tokenized sentences annotated with their named entities. Both corpora use a tabular plain text format and the phrases mentioning the named entities, or chunks, are sequentially annotated with the IOBv1 or IOBv2 tagsets. These sets consist of three tags:

- The IOBv1 tags are inside (`I`), outside (`O`), and between (`B`), while
- (IOBv2) redefines `B` as the start of a chunk instead of marking the following chunk.

[2]For in-depth information, refer to official guidelines: `https://www.clips.uantwerpen.be/conll2003/ner/`

The CoNLL evaluation software is a Perl script that computes F1 scores with precision and recall with regards to the phrases.

The IOB tagset was further enhanced with the addition of two tags: `E`, meaning the end of a sequence, and `S`, for singleton tags. The IOBES tagset has been shown to perform slightly better than the IOB tagset (Ratinov & Roth, 2009). A lossless transform can be applied to convert between IOBv1, IOBv2, or IOBES[3]. A notable property of the annotation is that the outside `O` tag is assigned to the majority of tokens, reflecting the fact that most words are not part of a named entity.

This corpus was used in two papers: Klang et al. (2017) and Klang and Nugues (2018a).

## 3.2 Text Analysis Conference EDL Data

The Text Analysis Conference (TAC) arranged by the National Institute of Standards and Technology (NIST) consists of evaluation tasks similar in their organization to CoNLL. TAC includes an entity discovery and linking (EDL) track. This track provides annotated data of newswires and discussion forum texts in three languages: English, Spanish, and Chinese. The TAC EDL goal is to first recognize named entities and then link them to unique identifiers.

The task has been reoccurring at an annual basis for several years. Recent years 2014-2017 provided usable training data on an annual basis. Training and evaluation data is provided as XML files combined with a standalone gold-standard. While older versions provided tokenized data, newer versions are entirely based on referencing spans consisting of Unicode codepoint offsets[4].

**Named entities.** For the recognition step, the EDL track defines five categories of named entities:

**Person (PER)** is identical to the equivalent category in CoNLL02/03 with the exclusion of fictional characters;

**Organizations (ORG)** matches the CoNLL02/03 category;

**Location (LOC)** contains only natural locations and non-administrative regions

**Geo-political entities (GPE)** corresponds mostly to named locations that are governed by a political entity: cities, villages, states, countries, administrative regions, etc.

**Facilities (FAC)** corresponds to transportation infrastructure, man-made buildings, hospitals, airports, etc.

[3] Also called BILOU and BIOES in literature.

[4] Described in Sect. 4.3.

A notable difference with CoNLL02/03 is the absence of a `MISC` class meaning that TAC2017 excludes named entities corresponding to events, languages, works of art, etc.

In addition to named categories, TAC EDL includes a nominal category of entities. This category was introduced in the last years and can be described as hypernyms or phrases consisting of common nouns that reference named mentions (Klang et al., 2017; Klang & Nugues, 2019b).

**Entity linking.** The TAC EDL mentions are linked in two ways: to a knowledge base or, if the entity is not in the base, with a system-dependent unique number prefixed with NIL, for instance `NIL23` or `NIL768`.

TAC EDL was annotated and linked to the Freebase knowledge base. Freebase is Google's entity repository, which is similar to Wikidata. Google discontinued it and merged it with Wikidata. In the works I published, I used Wikidata as the underlying knowledge base and I converted the identifiers from and to Freebase using a Google provided conversion dataset (Klang et al., 2017; Klang & Nugues, 2019b).

The entities in the TAC EDL corpus not found in Freebase are encoded with a unique sequential number for each distinct annotated entity, for instance `NIL768`. In the linking step, different systems may produce different numbers, for instance `NIL23`, as long as the same distinct entity correspond to the same unique number.

**Notable properties.** A noteworthy difference, compared to CoNLL03, is the practice of multilingual end-to-end evaluation, i.e. the system is given the raw XML text without any sentence and token segmentation.

In the dataset of forum discussions, this raw XML contains the text of the discussion as well as metadata such as the author and timestamp, quotes from earlier posts, and links are structurally marked in the discussion form set. The newswires dataset similarly contains metadata with the addition of paragraphs being structurally annotated in the newswires text (Klang et al., 2017; Klang & Nugues, 2019b).

The TAC EDL corpora is also noisy: The discussions contain incorrect spelling variations and unconventional abbreviations. This makes entity linking more challenging as the mentions do not align with those found in Wikipedia.

Arguably, the TAC EDL corpus combines the best of both worlds: clean news articles with noisy posts in discussion forums. This is what entity linking systems will ultimately face when used on arbitrary web content in practice.

## 3.3 The Stockholm-Umeå Corpus (SUC)

The Stockholm-Umeå corpus (SUC) is a joint effort by the Department of Linguistics at Stockholm University and the Department of Linguistics at Umeå University to annotate approximately one million words in Swedish.

SUC is a balanced corpus, text with varying styles, collected from various sources. The corpus contains part-of-speech annotation, extended word features, and name annotations. To the best of my knowledge, SUC is the largest balanced Swedish gold annotated corpora freely accessible for research applications.

SUC is available in multiple formats: an original SGML variant, XML, and tab separated values (TSV) similar to the basic format used by CoNLL with different fields. It has sentence separators and is fully tokenized. I used this corpus to build a named entity recognizer for Swedish (Klang & Nugues, 2018a).

## 3.4 Wikidata

Wikidata is a knowledge graph project initiated by Wikimedia, the organization behind Wikipedia. The goals of Wikidata for their phase one were:

> to centralize interlanguage links across Wikimedia projects and to serve as a general knowledge base for the world at large[5].

Practically, this is translated into *items* bound to a unique identifier: the Q-number. Items can be linked to zero or more Wikipedia language editions, contain a collection of *statements* which consists of properties and values. For a person, common statements include `instance-of`, `date of birth`, `father`, `mother`, etc.

The statement values can be a date value, URL, or, commonly, another item reference using its Q-number identifier. The statement values can also have properties such as start and end times e.g. political election terms. This results in a graph of connected knowledge.

The set of available properties and their allowed values are determined by a continually evolving ontology. The common elements of the Wikidata ontology are the `instance-of` and `subclass-of` relations.

## 3.5 Wikipedia

Wikipedia is a large freely accessible online encyclopedia; It is a project within the nonprofit organization Wikimedia Foundation. Wikipedia is written and moderated by users around the world. As of July 2019 there is 294 active editions[6]; most correspond to a natural language. Exceptions exist such as simplified English and constructed languages: Esperanto and Volapük. Wikipedia received its first edit on 15 January 2001[7].

Wikipedia consists of pages, each with a unique name called a *label* in the Wikipedia vocabulary. In addition, each page is associated with a namespace

[5] `https://www.wikidata.org/wiki/Wikidata:Notability` retrieved 2019-04-08
[6] `https://en.wikipedia.org/wiki/List_of_Wikipedias`
[7] `https://en.wikipedia.org/wiki/History_of_Wikipedia` retrieved 2019-08-05

which serves different purposes. The default namespace contains articles, redirection pages, disambiguation pages, etc. The namespace association is reflected in the URL of the page, which uses prefixes to identify the namespace: For instance. category pages have the prefix of `Category:` in the English edition.

The article pages are written using the Wikitext markup, which is a plain text format that is automatically transformed into HTML. Wikitext has support for page inclusion, style formatting, common HTML structures such as paragraphs, tables, lists, and more. Wikitext can include source code written in a programming language such as Lua, commonly used to build templates. A template is a directive included in a Wikipedia page that is replaced by a specific content when viewing the page. Examples of templates include infoboxes containing factual information and unit conversions. All the Wikipedia editions run on the same underlying software: MediaWiki.

Parsing Wikitext is challenging as it is a diverse markup language. Speculatively, the original authors wanted to minimize the technical barrier of entry. Wikimarkup is therefore an error-tolerant markup language with regards to parsing failure. This does shift the burden from the user to the implementer of a parser, resulting in a complex software with many rules.

MediaWiki is open source and Wikimedia provides configuration settings to setup a mirror. However, for research projects, this is often a huge endeavor in terms of environment complexity and computation resources, both in time and space.

Most research projects using Wikipedia are more interested in a processed clean plain text version. As a result, an approximate conversion is often acceptable and many parsers exist to carry this out. The typical approach for such parsers is to apply heuristics which mimic the original MediaWiki parser implementation. However, corner cases frequently result in incorrect parsing.

The success of the Wikitext parsing approach varies with languages as each language has localized templates. For instance, English works relatively well with this approach. However, French uses templates more extensively, requiring more support from the parser to avoid losing too much content. To further complicate the implementation of approximate parsers, plug-ins are supported using custom tags.

As reproducing the Wikitext-to-HTML transform is difficult, MediaWiki provides a REST API[8], which converts Wikitext into HTML. This API is suitable for volume access and downloading the English Wikipedia using this it takes about three days.

Transforming HTML to plain text can be done using commonly available DOM parsers combined with rules. As approach, I used the JSOUP parser which tries to mimic a web browser behavior when the source document is incorrect.

[8] `https://en.wikipedia.org/api/rest_v1/`

I then applied rules to flatten and filter the hierarchy. These rules produce a raw, clean text. However, information provided in hierarchical form, such as paragraphs, sections, and anchors, is important to entity linking. I reconstructed it as layers on top of the clean text (Klang & Nugues, 2016a, 2016b, 2016d, 2017, 2018b, 2019a).

# 4 Infrastructure

A natural language processing infrastructure is, at its core, a set of tools and libraries used by applications to carry out large-scale processing of text.

In this thesis, the infrastructure focuses on storage, document representation, and the construction of processing pipelines that can be applied to large corpora. It is a problem with many aspects that I broke down into the following subproblems:

- Understand the corpus and the data representations required by NLP algorithms;
- Design a combined document representation to unify the corpus and the data representations produced by the algorithms;
- Design reusable multilingual pipelines that can be applied and distributed in parallel across multiple machines;
- Implement interactive pipelines that can be used for prototyping.

## 4.1 Motivation

Creating knowledge graphs with the goal of covering encyclopedic knowledge, broad or complete, must be based on textual sources reflecting the varied nature of knowledge. Wikipedia is a choice that meets this requirement as Wikipedia's goal is to be an online encyclopedia that can be extended by anyone, hence potentially benefiting from the knowledge of all the individuals on earth. As a consequence, this also entails that over time Wikipedia has become larger and processing it with resource-intensive algorithms requires considerable computing power.

An initial solution to this problem of scale is to reduce the size of the corpus. However, this reduction would affect the infrastructure design and results. Arguably, at some point, the algorithms must be tested on real scale. Otherwise, they would often yield different results and different performance characteristics, as implementations may have unintended weaknesses overlooked when working at small scale.

Accurate feedback is important to verify experimentally if hypothesized methods work when applied to large corpora. Also, there is a human element: Not all algorithms exist in ready-to-use software packages or may be technically difficult and inefficient to apply to large corpora. This forces researchers to implement or re-implement them. Usually, this leads to the following trade-off in complexity: quickly written or well-optimized software. In a research context, the ideal is to combine simple, quickly written software, while still maintaining the possibility to run it over a large corpus.

This is where distributed computing comes as a solution. The facts that support this kind of computing are:

1. Single machines produce results, with potentially long iteration cycles and, in some cases, prohibitively long;

2. Using a cluster consisting of multiple machines can linearly scale up computing power, reducing the iteration cycle. Clusters add constraints that increase complexity, but, combined with a functional programming model, this complexity can be reduced to acceptable levels, as when using Spark. See Sect. 4.2.

## 4.2 Distributed computing

When processing Wikipedia, typical applications run language pipelines, extract statistics, and aggregate information from documents. In this section, I describe methods for distributed computing.

**Parallelization.** Many frameworks have been developed with the promise of scaling software to run in parallel on multiple computers.

One method with a standard and many implementations is the Message Passing Interface (MPI) (Clarke, Glendinning, & Hempel, 1994). It provides messaging and synchronization over many machines, enabling programmers to write programs that run in parallel on many computers. Another more modern approach is the Akka toolkit[9] that runs on the Java Virtual Machine (JVM). Akka, with its modules, provides abstractions to schedule, distribute, and run computations in parallel. Akka modules can be used with a single machine or distributed over multiple computers in a cluster.

Both of these approaches abstract away the underlying complexity of managing communication with many machines. These abstractions enable a developer to write software that is optimized for different characteristics such as low-latency processing of real-time information.

**Hadoop and Map-Reduce.** Dean and Ghemawat (2004) described a programming model that enables efficient processing and aggregation over many computers: the Map-Reduce model.

Map-reduce is based on the use of key and values, transforming them into lists of key and values, ultimately sorting and grouping by key and transforming these grouped values into a list of values representing the final output. A notable property of the map-reduce programming model is that it enables the processing of datasets larger than the available working memory. It is only limited by the available storage space.

Hadoop was one of the first open-source Map-Reduce implementations designed to run on commodity hardware. Hadoop is written in Java and runs on the

[9] `https://akka.io/`

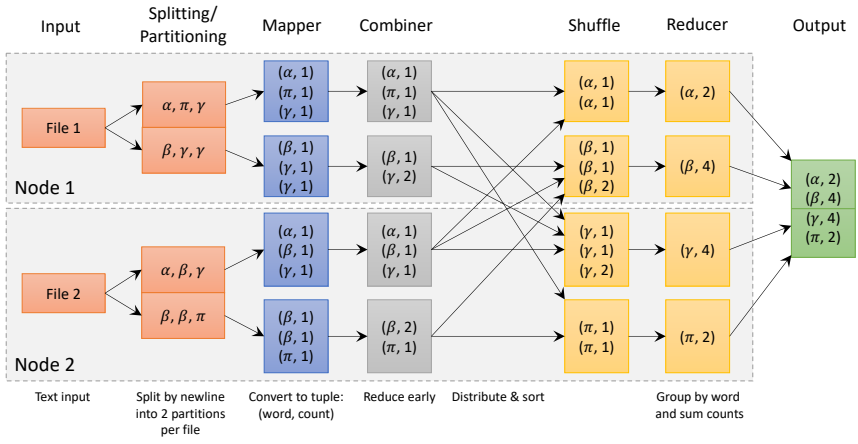



**Figure 2:** Map-Reduce word counting using two nodes.

JVM. Map-Reduce, as implemented in Hadoop, consists of the following parts, see Figure 2:

**Map function:** a function that is given a key and value pair, transforms them it into lists of key and value pairs. A map function, when computing word counts, transforms a document into a list of words as key and an initial count of one as the value.

**Reduce function:** a function that is given a key and a list of values with the same key and transform this information into the final output. Continuing with word counting, the reduce function computes the sum of occurrences from the list of values and emits a word and total count.

**Mapper:** The concrete worker or thread that executes the map-function over the data;

**Reducer:** The concrete worker or thread responsible for running the reduce function;

**Combiner function:** An optional function for partial reduction. The combiner function is only an optimization to reduce partial results early, minimizing the work required by the final reducer. Using the word count example, a combiner would sum up partial lists of counts.

**Partitioning function:** An optional function that computes an output partition given a key and a number of output partitions. The default implementation uses the hash value of the key.

**Shuffle operation:** The distributed synchronization and sorting step. Key and values emitted by the map function are first optionally reduced through the combiner and then assigned to a destination partition. The destination partition is sorted by key and the results streamed through the reducer.

Hadoop can scale to a large number of nodes[10] for massively parallel computation. However, writing software for the Hadoop ecosystem, or more specifically the classic Map-Reduce implementation, is verbose, repetitive, and often requires a deep knowledge of the internals to make sufficiently efficient software. In addition, many problems must be formulated as a sequence of map-reduce jobs, further adding to the amount of code required.

The classic map-reduce implementation is not memory bound like the original paper. The Hadoop map-reduce implementation streams key-value pairs and stores intermediate results on a disk. Storing key-value pairs to a disk requires serialization[11] and hence more storage. The consequence is reduced overall performance due to the increased volume of data to store.

**Distributed file systems.** In the context of this thesis, a typical cluster consists of ordinary computers (nodes) connected together with a standard Internet Protocol (IP)-based local network. Each machine consists of processors with multiple cores and independent storage systems.

Hadoop was designed to run on commodity hardware and, when using thousands of nodes, there is a high probability that one or more nodes fail during a long-running job. To avoid the loss of data, the solution is to have redundant storage and reschedule tasks to resume parts that failed. In addition, network bandwidth is limited and therefore mappers should ideally process what is available locally first before requesting data from other machines; increasing redundancy by duplication improves data locality and resilience to data loss, reducing overall network bandwidth requirements.

Hadoop include a file system called the Hadoop Distributed Filesystem (HDFS). This file system transparently balances stored data over many machines, maintaining sufficient duplication for locality and resilience to node loss by redundancy.

**Spark.** The Apache Spark project introduced a higher-level programming model, addressing three primary weaknesses of the classic map-reduce implementation: inefficient round-trips to the storage layer, no automatic job pipelining to enable fusing of mappers in sequence, and the verbose nature of writing map-reduce jobs. Spark is written in Scala running on the JVM, but provides API bindings for Java, Python, and R. I used Spark for the large-scale processing tasks of Wikipedia (Klang & Nugues, 2016b, 2016d, 2017, 2019a).

[10] Virtual or physical machines

[11] converting in-memory objects into a standalone representation that can be written to a storage medium

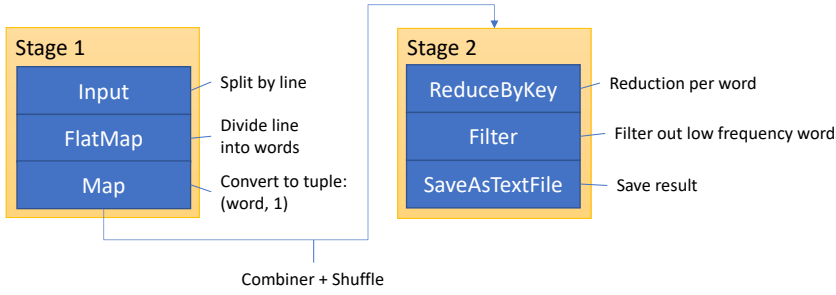



**Figure 3:** Spark word counting with added filtering at the end

First, the round-trips are reduced by in-memory reduction and caching, using the available memory to speed up reduction steps. Pipelines are optimized by introducing a functional programming model that modifies a Resilient Distributed Dataset (RDD) representing all data divided into partitions distributed over the machines. This programming model enables a developer to define pipelines as a sequence of RDD *transformations*. The RDD is not actually transformed until an *action*, such as saving to disk, is triggered.

This lazy evaluation enables Spark to compile a pipeline into a *directed acyclic graph* (DAG), fusing sequential operations together for optimal processing. This fusing reduces the number of shuffles required overall and lowers the number of round-trips to the storage layer by doing more in-memory. In addition, Spark is compatible with the data storage APIs found in the Hadoop ecosystem and makes use of them. The map-reduce based version of counting words (see Figure 2) is transformed into two stages in Spark (see Figure 3) with each stage running sequentially.

Spark also introduces the ability to share read-only data efficiently in the cluster and allows the developer to cache datasets in-memory for repeated processing to reduce the overhead of accessing the storage layer.

Spark has modules built on top of the core engine:

**Dataframes:** Structured columnar datasets with support for hierarchical structures. These dataframes, when used properly, reduce the memory footprint of a large dataset. In addition, they can make use of the Parquet format for saving and loading datasets efficiently;

**SQL:** Modified SQL query language applied to dataframes for scalable queries larger than memory;

**Streaming:** Real-time processing of data that is distributed over many machines;

**Machine learning:** Algorithms for various supervised machine learning models such as logistic regression and support vector machines, and unsupervised

methods such as word embeddings (Word2vec), clustering via K-means, and dimensionality reduction via a principal component analysis.

## 4.3 Document Representation

A core requirement for scalable text processing is a flexible and robust data model capable of representing the diverse sets of structures found in NLP applications. Moreover, good tools to extract, visualize, and inspect data are also important for the understanding and verification of a correct operation.

**Text segmentation.** Text is a sequence of characters. Before processing and transformation, software initially treats this sequence as a continuous stream of bytes with no apparent structure. However, many algorithms rely on a segmented stream of tokens and sentences. These tokens, i.e. the segmented units in a sentence, include the words, punctuation, numbers, symbols, emoticons, etc. Performing segmentation using simple rules works well for languages such as English or Swedish. However, logographic languages such as Chinese make minimal use of whitespaces or punctuation for word segmentation.

**Grammatical processing.** From segmented text, we can apply methods which use token sequences instead of raw characters. This includes part-of-speech tagging, assigning a category to each token e.g. noun, verb, preposition, punctuation, etc. In addition, *dependency parsing* can assign *labeled* dependencies between words.

Although, token sequences are useful, depending on the language, some semantic details can only be found by analyzing the *morphology* of the word. The simplest example is when going from singular to plural mentions, e.g. car and *cars*. More complex changes can also occur: run, runs, running, and ran are all variants of the canonical form or lemma *run*.

**Text representation in machines.** The byte stream discussed previously represents a sequence of *characters*. A single byte[12] could represent 255 characters with 0 as a special delimiter, which is more than enough for English. However, writing systems used by natural languages around the world are varied. They include the Arabic abjad, Devanagari with Hindi, Han characters with Mandarin Chinese, Cyrillic with Russian, etc. One of the first attempts to extend support was to use code-pages which divided the byte into a upper and lower region, with the lower 0-127 fixed, and the upper range of 128-255 could be changed depending on region or application. Remnants of this solution can still be found in common software such as Windows, for instance Windows-1252 is the code-page for Western Europe used in Sweden.

[12] unsigned 8-bit integer with 256 possible values

With the introduction of internet, and the rise of multilingual communication, this quickly became unfeasible as *scripts* such as Han used with Chinese do not fit into single code-pages, requiring specialized software support. To provide a common code, a standardized international *character set* was developed: Unicode. Unicode is also an international consortium founded 1991 that comprises various international technology companies, universities, and governments.

Concretely, the Unicode standard consists of a list of *code points*, integers, where each code point is associated with a *character* name and its visual representation, as well as a set of properties. The specific graphical rendering of the characters is determined by the *font* used.

Unicode is frequently implemented at the binary level using one of several predefined encodings: one popular choice being `UTF-8`, a variable length encoding method. In particular, `UTF-8` supports the fixed lower code-page region used by English without any special encoding. Today, `UTF-8` is widely supported in web browsers, operating systems, and application frameworks. However, it is not always the default option; web pages that use the Hypertext markup language (HTML) must explicitly be set to use `UTF-8` or web servers configured to use `UTF-8` by default for text transmission.

Wikipedia uses Unicode as the character set for text, as it allows multilingual text. However, Unicode contain ambiguous characters for certain classes such as letters, punctuation including dashes, periods, and numbers. This becomes an issue when the visual representation differ from the actual code-point sequences used requiring normalization to be unified over many languages.

**Requirements.** Over the time, the NLP research has produced quantities of quality tools with verified results. Reusing them reduces efforts in new projects, but also makes the research community dependent on the works of others. Locking longer term research into these works is inherently risky as it can be difficult to change a particular tool. For small scale and domain specific research, this is typically an acceptable risk as this kind of research has few moving parts.

However, when attempting to run a complex set of tools over a large number of documents, the number of moving parts and sources of issues increases. Possible strategies to mitigate this is to use pipelines, where multiple independent annotators are run in sequence as shown in Fig. 4. These tools are often research projects in themselves and move too quickly to be thoroughly tested. This requires a high degree of tolerance to variability in output and to software issues.

One of the key issue is to unify the output produced by these tools into a single representation with few hard dependencies on the original software. The general aspects that arguably cover most tool output are summarized in Figure 5. In this figure, we introduce concepts such as annotation, property, connections, spans, and offsets.

No model will be perfect for all the uses as there is a balance between the ease of use, from a researcher perspective, and rigidity imposed by the use of

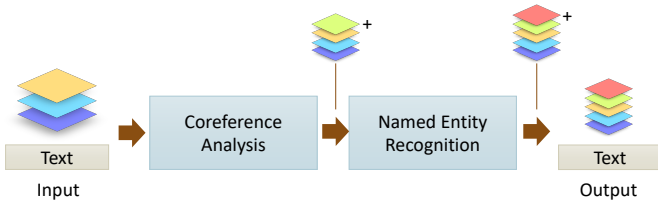



**Figure 4:** Annotation pipeline

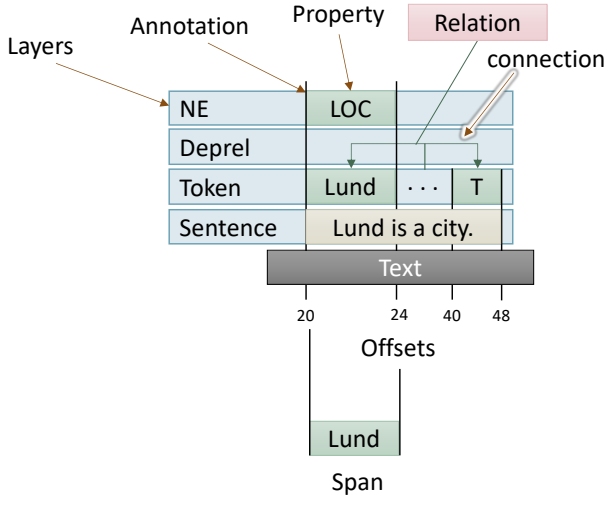



**Figure 5:** NLP data model

well-defined storage formats such with as FoLiA (van Gompel & Reynaert, 2013). Rigid type definitions add a layer of complexity, which is prohibitive for smaller, typically student projects: steep learning-curves, forced run-time environments, and resource requirements.

In this thesis, I designed Docria (Klang & Nugues, 2019a), which tries to create a middle-ground between rigidity and ease of use by defining data structures and a storage API in at least two languages: Java and Python. By applying inversion of control, most libraries and tools can be wrapped and adapted to fit this model. In addition, Docria has few hard dependencies.

To put the Docria model in context, we need to compare its representation to what is described in the literature. I have identified three families of formats: plain text, XML, and a variable representation family which Docria is part of. A major

class divider for these formats is the support for *stand-off annotations* (Thompson & McKelvie, 1997). The stand-off idea is to use offset references instead of modifying the text to fit the representation. Stand-off is primarily needed to:

- Deal with copyright issues when sharing original data;
- Support overlapping annotation; and
- Preserve the original text.

**Plain-text formats.** The flat text format is used extensively by CoNLL tasks, arguably for its simplicity in reading. However, these formats lack standardization and vary from task to task (or user to user).

Re-occurring conventions are double newlines for sentence separation, where each line is a separated list of fields; the fields usually correspond to what is needed for training or processing in a particular task. The oldest widespread usage of this format seems to be the task of CoNLL99 on chunking (Osborne, 1999).

Currently, plain-text format has one popular variant in the form of CoNLL-U to annotate the Universal Dependencies (Nivre et al., 2018). It is a variation of the format used for in CoNLL-X (Buchholz & Marsi, 2006). CoNLL-U defines extensions for stand-off reconstruction and subwords.

Plain-text formats are mostly centered around sentences, which renders higher-level structures hard to represent naturally.

**XML-based formats.** The eXtensible Markup Language (XML) is by far the most frequently-used format in the literature, typically for treebank annotation and other larger projects. Pustylnikov, Mehler, and Gleim (2008) give a short overview of some common treebanks, where most use XML-based formats in different variations. Notable ones are TIGERXML used for the TIGER corpus (Brants, Dipper, Hansen, Lezius, & Smith, 2002), the Text Encoding Initiative (TEI) XML encoding (TEI Consortium, 2019), and more recently FoLiA (van Gompel & Reynaert, 2013). TIGERXML, TEI, LAF, and FoLiA are primarily format specifications; tooling is secondary.

XML has a high degree of flexibility, well-proven and extensive tools for the general case. The primary weakness of XML based formats is their variability and flexibility. XML supports schemas which define the data layout, but this is not a requirement. In a real-world setting, format specifications such as TEI and FoLiA need automated verification tools to ensure compatibly in software tooling. The size of these specifications implies a high degree of effort to support all the aspects.

**Variable representation models.** The NLP Interchange Format (NIF) (Hellmann, Lehmann, Auer, & Brümmer, 2013) defines a way to attach linguistic information via the use of the Resource Description Framework (RDF) standard used

in the semantic web community. NIF defines a URI encoding to refer to spans, stand-off encoding.

As the original XML representation of RDF is very verbose, many competing encoding have been developed: Turtle, N-Triples, N3, JSON-LD, etc.

## 4.4 Creating a New Document Model: Docria

In this section, I explore three major revisions to the document representation I created. These revisions build on the ideas found in pieces throughout the literature. Many document representation models are annotation-based, models that add properties to ranges and divides these annotations into groups. Annotation-based data models are old, offset-based models can be found in TIPSTER (Harman, 1992) project, and then later in GATE (Cunningham, Humphreys, Gaizauskas, & Wilks, 1997). Encoding graph structures are secondary in pure annotation models and is frequently encoded using unique references, requiring an extra pass of processing to index and resolve. Graph structures are standard in problems such as dependency parsing, semantic role labeling, co-reference resolution, etc.

**WIKIPARQ.** The first iteration, WIKIPARQ (Klang & Nugues, 2016d), was based on the idea to use the RDF triple representation, i.e. a subject, predicate, and object. In our case, this meant transforming a multi-layer description with nodes and edges to RDF triples. A common issue with storing RDF naïvely is that it is highly redundant. RDF typically uses URLs and these might be long.

Parquet[13] is a storage format designed by Twitter to store logs and process them in a scalable way. This made Parquet the ideal choice as it was designed to reduce the effects of redundant information on disk. In combination with Spark SQL, a query language was provided which allowed many simple questions to be asked at once.

A typical English Wikipedia dump could hold up to 7 billion entries. Combined with the storage tricks used by Parquet, most data can be discarded when running a simple query by reading metadata or only reading columns required to determine if it should be included. This means that even though the number of rows is substantial, most can be excluded, yielding an acceptable performance. The assumption and requirement is that the query planner used by Spark SQL can transform the query into a sensible plan which minimizes a full shuffle.

In practice, query performance is uneven, making the real query cost hard to predict. Many operations when processing documents are inherently local, which increases query complexity. As a storage format, WIKIPARQ was a good choice; as a general purpose information storage, reprocessable resources and retrieval format, it was not.

[13] `http://parquet.apache.org/`

**Docforia.** The second revision was Docforia (Klang & Nugues, 2017), a first attempt to solve the drawbacks of WIKIPARQ, particularly to eliminate the risks of data explosions when the query planner produced a poor plan. Docforia is a programmer-oriented architecture which defines a storage model, serialization format, and some query APIs to process the content found in documents. It is built on the ideas of using a layered property graph for nodes and edges, and some heuristics to store token spans efficiently.

Docforia solved almost all the practical issues. However, the node-edge duality introduced duplication and redundancies into the implementation. Combined with a convention based schema, this increased the cost of implementing many performance optimizations.

**Docria.** This led us to the final revision: Docria (Klang & Nugues, 2019a). Docria was designed to reduce the effort required when porting to other programming languages. In addition, we removed the node-edge duality by using property hypergraphs that merge nodes and edges.

Docria separates text and layers that add linguistic information. Each layer consists of a variable number of nodes, where each node has a set of typed fields defined in a schema. The schema is required per layer and can define fields that reference spans in a text. Spans are encoded in a way that minimizes portability issues when implemented in different programming languages. Using a relational database terminology, a single Docria document is a database, in which each layer is a named table of rows with a fixed set of columns according to its schema. Additionally, the fields define types and spans for nodes.

Docria was designed to separate serialization from its in-memory API allowing future extensions. Docria defines and includes implementations for JSON, XML, and a binary format using MessagePack. The most compact encoding is MessagePack. MessagePack was ideal as it is self-describing, has many implementations in a diverse set of programming languages, has well-defined specifications, and is fast and compact enough.

Docria can be used on a per document basis or in a collection of documents in a file.

## 4.5 Language pipelines

The construction of large NLP systems frequently requires the assembling of different modules, where each module adds structured information to the overall representation. These modules may not conform to any particular specification and may be implemented in separate software packages.

I designed Langforia to build processing pipelines from modules, possibly disparate. My explicit goal was to make the process more concrete and with fewer software layers. Langforia enables a researcher to compose pipelines that can be

used interactively, in an online fashion, or be applied offline to a large corpus, i.e., in a scalable way.

### Previous Work

**GATE.** The General Architecture for Text Engineering (GATE) (Cunningham et al., 1997) consists of a document model that support stand-off annotations, a plug-in approach to constructing pipelines (CREOLE), and a graphical user interface that enables manual annotation, interactive pipeline construction, and to apply constructed pipelines to a collection of documents.

GATE's document model consist of:

**annotation type** that defines types such as tokens, sentences; similar to layers in Docria.

**annotation** that corresponds to a node with range reference in the original text. Annotations are associated with a type, and can contain properties. Annotations are similar to nodes in Docria that have a single span field.

**annotation set** that is a named group of annotation nodes.

GATE supports via plug-ins a variety of input formats and its default document output format is XML.

**UIMA.** The Unstructured Information Management Architecture (UIMA) is a framework that can be used to develop applications that require e.g. text analysis by combining reusable components (Ferrucci & Lally, 2004).

UIMA covers aspects such as design and research to pipeline construction and deployment at scale. UIMA was motivated by the observed difficulties in developing reusable technologies. Technologies that frequently start as research prototypes and require considerable effort to be converted into production ready pipelines. UIMA defines a software architecture that aligns development practices, an analysis model for structured information access, and a component model to discover and encapsulate analysis tools. The entire process from acquisition to analysis output is covered.

UIMA defines concepts such as:

**Text Analysis Engine (TAE)** is a recursive structure that consists of components that add annotation such as named entity recognizers.

**Common Analysis Structure (CAS)** is the storage model representing a document that contains the output from annotators.

**Annotator** is a component responsible for adding structured information to a document; it operates on the CAS.

**spaCy.** SpaCy[14] is a software package written for Python consisting of ready to use NLP pipelines for multiple natural languages. It was designed for performance and to be easy to embed into an application. Unlike GATE and UIMA, SpaCy was designed to support a set of concrete pipelines.

**CoreNLP.** CoreNLP (Manning et al., 2014) is a NLP software package developed at the Stanford University (Manning et al., 2014). Stanford distributes pretrained models for multiple languages. CoreNLP support training of new models and is similar to SpaCy in that it was not designed to abstract arbitrary external pipelines. Recently, the NLP group at Stanford introduced a neural pipeline written in Python[15] that also has an official wrapper for the Java based CoreNLP server (Qi, Dozat, Zhang, & Manning, 2018).

### Langforia

Langforia (Klang & Nugues, 2016b) was an attempt to abstract complex research-oriented pipelines using Docforia as the common ground in which to store the output. Another goal of this software was to embed it in a Spark pipeline to carry out the annotation using cluster computing.

Langforia was further developed with a frontend/backend architecture including a visualization component which can display the content of a Docforia document. The frontend can also be used for debugging, sending in documents, and running pipelines interactively over a web API.

The embeddable version of Langforia is similar to UIMA and GATE, but different from spaCy and CoreNLP in that it encapsulates external pipelines not part of the project. It is different from UIMA and GATE in that a pipeline must be known at compile-time and all dependencies are packaged with the embedding application. This implies that any change to a pipeline requires a recompilation.

We ultimately used Langforia to process large corpora such as the English Wikipedia with processing times as short as a few hours to a few days.

# 5 Evaluation

Before we deal with the techniques we developed and applied to recognize and link named entities, let us describe how we will evaluate their results. The evaluation metrics will then enable us to select the best techniques or models.

[14] `https://spacy.io/`
[15] `https://github.com/stanfordnlp/stanfordnlp`

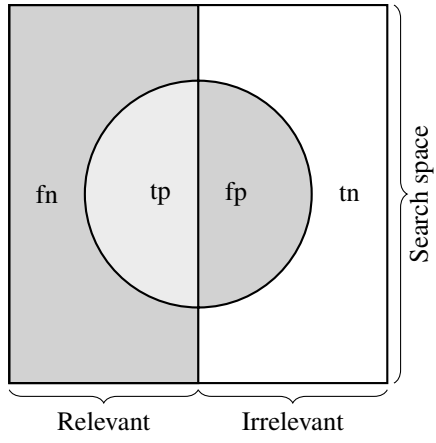



**Figure 6:** Search space for documents or observations

## 5.1 F-Measure

In this dissertation, the $F_1$-measure is used to evaluate named entity recognition and named entity linking. The $F_1$ measure is often used to evaluate the classification performance of a model. Originally, the $F_1$-measure was used to assess the performance of search in information retrieval. It then found its way as the evaluation metric of NLP systems in MUC-4 (Chinchor, 1992).

$F_1$ consists of two metrics *precision* and *recall* combined into one score.

**Counting.** Using the framework of information retrieval, the problem is to evaluate the performance of a search query applied to a set of documents. We can divide the search result into four classes, see Fig. 6 for a visual representation

**True positive (tp)** is the set of relevant documents that are correctly retrieved by the query;

**False positive (fp)** is the set of irrelevant documents that are incorrectly retrieved by the query;

**False negative (fn)** is the set of relevant documents that are incorrectly not retrieved by the query;

**True negative (tn)** is the set of irrelevant documents that are correctly not retrieved by the query.

Given the sizes of these sets, we can compute precision and recall, where in our case a document corresponds to an observation.

**Recall** measures the share of documents we should have found:

$$R = \frac{tp}{tp + fn}, \tag{1}$$

**Precision** measures the share of documents we found that are relevant:

$$P = \frac{tp}{tp + fp}, \tag{2}$$

**Harmonic mean.** The $F_1$ measure combines these two measures using the harmonic mean: *recall* $R$ and *precision* $P$:

$$F_1 = 2\frac{P \cdot R}{P + R}. \tag{3}$$

**Intuition.** Recall and precision tend to be opposites with a typical model capable of optimizing and performing well on either precision or recall; performing well on both is a more difficult problem.

Assume that a threshold is used when determining if a document matches the query or not. The motivation is then as follows: to optimize for recall, we can reduce the threshold for a match, which increases the number of false positives resulting in a decrease of precision. Conversely, to optimize precision, we just select the documents we are more confident in. This can be done by increasing the threshold, thereof reducing the number of false positives. The consequence is that the number of relevant documents drops, increasing the number of false negatives resulting in a reduction of recall.

The use of harmonic mean in the $F_1$ measure is to reduce the score when recall and precision are not in balance.

**Weighted F-measure.** In some applications, precision might be more important than recall, or the reverse. The weighted F-measure sets an emphasis on either precision or recall by adjusting the weight $\beta$:

$$F_\beta = \frac{(1 + \beta^2) \cdot P \cdot R}{(\beta^2 \cdot P) + R}. \tag{4}$$

Typical values for $\beta$ in a weighted F-measure are 0.5 or 2.

## 5.2 Mean Reciprocal Rank

The mean reciprocal rank (MRR) is a metric to evaluate the performance of a question answering system.

Given question $q_i \in Q$, a system returns a ranked list of answers. The rank of the correct answer $\text{rank}_i$ for each given question is used to compute MRR:

$$MRR = \frac{1}{|Q|} \sum_{i}^{|Q|} \frac{1}{\text{rank}_i} \tag{5}$$

If the correct answer is not found, typically you set the rank to a large number or set the reciprocal rank to zero.

## 5.3 Constrained Entity Alignment F-measure (CEAF)

The performance of entity linking or coreference resolution is more difficult to evaluate than that of named entity recognition. The MUC conferences introduced a coreference scoring scheme based on links (Vilain, Burger, Aberdeen, Connolly, & Hirschman, 1995), while B-cube (Bagga & Baldwin, 1998) is based on mentions. X. Luo (2005) proposed the *constrained entity alignment F-measure* (CEAF) as an improvement over the MUC link-based and B-cube F-measures. In this thesis, we followed TAC 2017 (Ji et al., 2017) and adopted $\text{CEAF}_m$ to rank the performance of competing entity linking systems.

The metric takes two sets of partitions: the reference set (gold standard), $r \in R$, and the system predicted set, $s \in S$. Each element in $R$ and $S$ represents a partition of keys, i.e. a set of tuples. In TAC 2017 (Ji et al., 2017), the sets were created by partitioning the mentions according to their entity identifiers and the final metric `CEAFmC+` defined the keys as tuples with mention span, entity target, and mention type as shown in Figure 7.

At the core of the CEAF metric, there is a requirement that each reference key in $R$ is aligned with at most one predicted key in $S$ resulting in a one-to-one mapping. This mapping $g^*$ is found by solving an assignment problem i.e. find the pairs of reference and system predicted partitions that globally maximizes the total similarity $\Phi(g^*)$. Concretely, the assignment problem is solved using the Kuhn-Munkres algorithm (Kuhn, 1955) and the similarity between partitions is given by a function $\phi(r, s)$.

Mathematically:

$$\Phi(g^*) = \sum_{r \in R} \phi(r, g^*(r)) \tag{6}$$

$$P_{\text{CEAF}_\phi} = \frac{\Phi(g^*)}{\sum_{s \in S} \phi(s, s)} \tag{7}$$

$$R_{\text{CEAF}_\phi} = \frac{\Phi(g^*)}{\sum_{r \in R} \phi(r, r)} \tag{8}$$

$$\begin{aligned} F_{\text{CEAF}_\phi} =& F_1(P_{\text{CEAF}_\phi}, R_{\text{CEAF}_\phi}) \\ =& 2 \cdot \frac{P_{\text{CEAF}_\phi} \cdot R_{\text{CEAF}_\phi}}{P_{\text{CEAF}_\phi} + R_{\text{CEAF}_\phi}} \end{aligned} \tag{9}$$

Two similarity functions were proposed by X. Luo (2005): a mention-based $\text{CEAF}_m$ and an entity-based $\text{CEAF}_e$:

$$\phi_{\text{CEAF}_m}(r, s) = |r \cap s| \tag{10}$$

$$\phi_{\text{CEAF}_e}(r, s) = 2\frac{|r \cap s|}{|r| + |s|} \tag{11}$$

Moosavi and Strube (2016) showed that CEAF and other commonly used metrics suffer from a "mention identification effect" with recall and precision "neither interpretable, nor reliable" as a result. Moosavi and Strube (2016) proposed the *link-based entity-aware* (LEA) evaluation metric to overcome these limitations. LEA was not used in this work and is provided as a reference for future work.

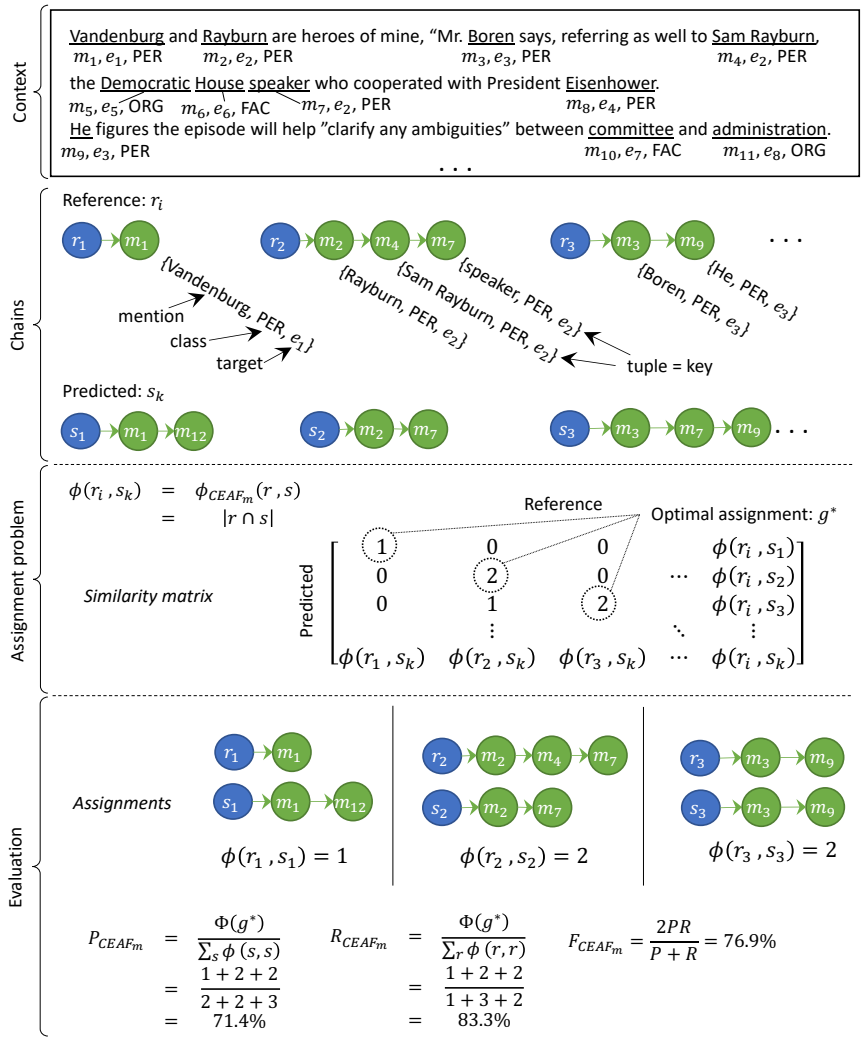



**Figure 7:** CEAFm example from a context in OntoNotes 5 used in CoNLL 2011 (Pradhan et al., 2011)

# 6 Machine Learning

## 6.1 Definition

Machine learning is the field of constructing models and algorithms that *learn* from data (*cf.* past experience), to make predictions. The word *learning* is used metaphorically and is frequently reduced to a gradient decent optimization problem in practice. Arguably, a model has *learned* something when it is capable of *generalizing* from known observed data, ultimately, producing *good* predictions on unseen data.

In this thesis, I applied machine learning and as a means to compute entity similarity to predict classes for entities:

- Find names in sentences;
- Determine their type;
- Improve entity disambiguation by predicting context compatibility;
- Build language models to define the word meaning based on context.

## 6.2 Concepts

This section provides the definition of the important concepts used throughout this thesis:

**Observation:** An instance or example, e.g. word, sentence of words.

**Prediction:** The computed output using an observation as input with a trained model.

**Feature:** Discrete unit of information as part of the observation, e.g. a word vector, scalar indicating if word is title cased. A set of features makes up the input;

**Label:** In many cases, each observation has a matching label, a category or value. This is also called the output, e.g. noun, verb, for the part-of-speech task, a tag per word which indicates first, part of, and last word of a name.

**Model:** An instance of a mathematical model: $\hat{y} = f_\theta(\mathbf{x})$, where $\hat{y}$ is the predicted label given the input $\mathbf{x}$ for a model $f_\theta$ with parameters $\theta$. For a dataset, we have $\hat{\mathbf{y}} = f_\theta(\mathbf{X})$;

**Architecture:** The high-level structure of a model. Predictive models discussed later such as feedforward, convolutional, or recurrent neural networks can be constructed in many ways.

**Loss:** The scalar value to optimize during training. Related variants include cost, error, or learning objective. Loss, in particular, is a value which should be minimized.

**Training set:** Full collection of observations with matching labels used when fitting a model.

**Test set:** Collection of observations with matching labels withheld during training. Used to evaluate the performance of the model.

**Validation set:** Same as test set, however used in situations in which we use evaluation data to improve the training process.

**Batch:** Limited collection of observations.

**Parameter:** Tunable weights which contribute to the output prediction given an input.

**Hyper-parameter:** High-level settings which alter aspects of the training and/or the structure of chosen model.

**Dense vector:** A vector where a substantial number of elements are nonzero.

**Sparse vector:** A vector where most elements are zero.

## 6.3 Learning Paradigm

In this thesis, I used two paradigms of learning:

**Supervised learning:** Given an input, predict and compare the predicted with the expected result and minimize the difference between predicted and expected result.

**Unsupervised learning:** Given an input, find patterns in the data.

Supervised learning requires a gold standard, a training set that maps an observation to the expected output. For instance, in the context of classifying names, given an observation *Lund University*, we expect an output in the form of a category such as *organization*.

Conversely, unsupervised learning finds patterns using only the observations. Word2vec is an example of a method that produces word representations, *embeddings*, from word co-occurrence observations. These embeddings can be used to measure word similarity or find patterns such as the fact that the words *man* and *woman* are analogous to *king* and *queen*.

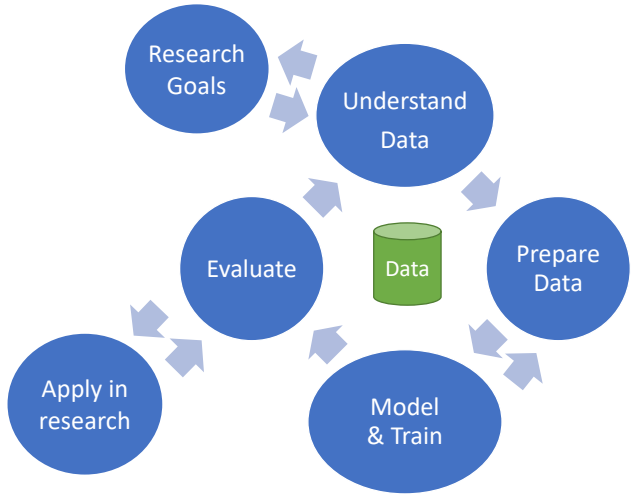



**Figure 8:** CRISP-DM (Wirth & Hipp, 2000) modified to a research context

## 6.4 Machine Learning Methodology

Cross industry standard process for data mining (CRISP-DM) (Wirth & Hipp, 2000) is a process model for carrying out data mining projects. A variant adapted to research and machine learning was used when creating machine learning models, see Figure 8.

The adapted process consists of the following parts:

**Research goals** such as finding the named entities in text in a way that improves the state of the art (and publishing it); then mark the named entity ranges for a subsequent named entity linker;

**Understand data** is a phase that attempts to bring clarity to what data is available and its particular properties. Most importantly, determine if the available data is suitable to solve the goal;

**Prepare data** is a cleaning and transformation phase, removing noise, and transforming raw data into a form that can be used to train models;

**Model & train** is the phase that attempts to find a suitable model, and train it to fit the prepared data;

**Evaluate** is the evaluation phase that determine how general a suggested model is. The evaluation phase often gives hints at whether we have sufficient data or not, or if some aspects have been misunderstood, suggesting an improvement in data understanding or acquisition;

**Apply in research** is the final phase when a trained model is judged to be good enough to be ready to be encapsulated for future research.

# 7 Data Representation for Machine Learning

Machine learning, with popular frameworks such as Pytorch and Tensorflow, use *tensors* for input and output. Tensors are a generalized representation to arrange numbers in a variable number of dimensions. Technically, tensors are multidimensional arrays combined with a set of computational rules and methods.

To process and learn from natural language using frameworks such as Pytorch or Tensorflow requires a tensor representation. These representations are built from features. The common feature types I used in this thesis are:

**Scalar:** A numeric value, for instance to represent the share of pages with a particular name;

**Embedding:** A dense vector representing a symbol in an abstract space, e.g. a word;

**Categorical:** Sparse vectors indicating a class such as person or location for a name;

**Bag-of-words:** Sparse vectors representing a set of symbols, frequently words.

In the following sections, I describe popular encoding methods for these features.

## 7.1 One-Hot

**Definition.** One-hot encoding is a method that encodes categorical features. The one-hot method maps all the symbols to elements of a vector. Converting a symbol into a vector is carried out by setting the mapped element to one.

Mathematically, a one-hot encoded vector $w_i$ for symbols $w_i \in W$, produces vectors of $\mathcal{R}^n$, where $n = |W|$, the number of unique symbols in the vocabulary $W$.

**Example.** Given the vocabulary $W = \{\text{Red}, \text{Green}, \text{Blue}\}$, with $n = 3$, produces $\mathcal{R}^3$ vectors:

$$\overrightarrow{\text{Red}} = \begin{pmatrix} 1, 0, 0 \end{pmatrix}$$
$$\overrightarrow{\text{Green}} = \begin{pmatrix} 0, 1, 0 \end{pmatrix}$$
$$\overrightarrow{\text{Blue}} = \begin{pmatrix} 0, 0, 1 \end{pmatrix}$$

A notable property of one-hot vectors is that they are orthogonal to each other. Also, the vectors are sparse with only a single nonzero element. Encoding a one-hot vector as an index is frequently carried out in optimized software, eagerly converting the index into a full dense vector when required.

## 7.2 Bag of Words

The one-hot method produces vectors for each symbol. The bag-of-word encoding is a technique to create a single representation of multiple symbols, such as all the words in a document.

As a word can appear in many forms such as *Car*, *car*, *Cars*, *cars;* these variants are often reduced to a single form in practice, the *term*. In English, words are often transformed into stemmed and normalized terms by removing frequent suffixes such as (*-s*, *-ing*, *-ed*), lowering the case of all characters, and more.

Building upon the one-hot method, documents can be represented through a sum operation:

$$\sum_{i}^{n} w_i.$$

This sum operation is a baseline method to encode bag-of-word features. If the words are allowed to repeat, this representation corresponds to a vector, where the axes are the term frequencies for all the words in the document.

A notable property of bags of words is that they ignore the word order.

## 7.3 TF-IDF

The term frequencies (TF) and inverse document frequencies (IDF) are scalars that introduce a measure of word significance in a document.

In natural language, some words are frequently used to fill out and connect sentences such as *the*, *and*, *in*, and *of*. In practice, less frequent words tend to be more salient and thus, be more important than frequent ones. The idea with the inverse document frequency it to reduce the significance of words mentioned in many documents $d \in \mathcal{D}$. To do so, it normalizes a word count with the number of documents the word appears in. This normalization mitigates the effect of frequent words dominating the representation.

To produce TF-IDF vectors, first *terms* in document $d$ are counted: $C(\text{term})$; this term may be scaled with respect to the total number of terms:

$$\text{TF}(\text{term}, d) = \frac{C(\text{term})}{\sum_{\text{term}_d \in d} C(\text{term}_d)} \tag{12}$$

The IDF weight counts the number of documents, where the term occurs and uses it to normalize the total number of documents:

$$\text{IDF}(term, D) = \log \frac{|\{d|d \in D\}|}{|\{d|\text{term} \in d, d \in D\}|} = \log \frac{N}{n_{\text{term}}} \tag{13}$$

Combining both terms, TF-IDF is computed as this:

$$\text{TF\_IDF}(\text{term}, d, D) = \text{TF}(\text{term}, d) \cdot \text{IDF}(\text{term}, D) \tag{14}$$

Combined with the bag-of-word technique, we can produce representations for multiple words. A bag-of-word TF-IDF often provides a strong baseline such as in the task of document classification (Yang et al., 2016).

**Example.** For simplicity, assume there are 4 terms (*LTH*, *Lund*, *Scania*, and *Sweden*) in three documents $d_1, d_2, d_3$.

The first step is to locally and globally count terms in documents. Local counting corresponds to the first three rows in Table 1, and the last row corresponds to the global count.

| **Term counts** | **LTH** | **Lund** | **Scania** | **Sweden** |
|---|---|---|---|---|
| $d_1$ | 0 | 2 | 0 | 2 |
| $d_2$ | 1 | 0 | 2 | 1 |
| $d_3$ | 3 | 1 | 0 | 3 |
| $\|\{d\|\text{term} \in d, d \in D\}\|$ | 2 | 2 | 1 | 3 |

**Table 1:** Local and global counting of terms

The final step is use the local and global counts to produce a final weight. In Table 2, each row corresponds to a vector and each component a term. The term *Sweden* is part of all the documents and therefore has a weight of zero.

| **TF-IDF** | **LTH** | **Lund** | **Scania** | **Sweden** |
|---|---|---|---|---|
| $d_1$ | 0 | $\frac{2}{4} \log \frac{3}{2} \approx 0.088$ | 0 | 0 |
| $d_2$ | $\frac{1}{4} \log \frac{3}{2} \approx 0.044$ | 0 | $\frac{2}{4} \log \frac{3}{1} \approx 0.239$ | 0 |
| $d_3$ | $\frac{3}{7} \log \frac{3}{2} \approx 0.075$ | $\frac{1}{7} \log \frac{3}{2} \approx 0.025$ | 0 | 0 |

**Table 2:** Example TF-IDF representation

## 7.4 Embeddings

The one-hot and TF-IDF methods produce sparse vectors when transforming bags of words and categorical features. Both models typically ignore co-occurrences

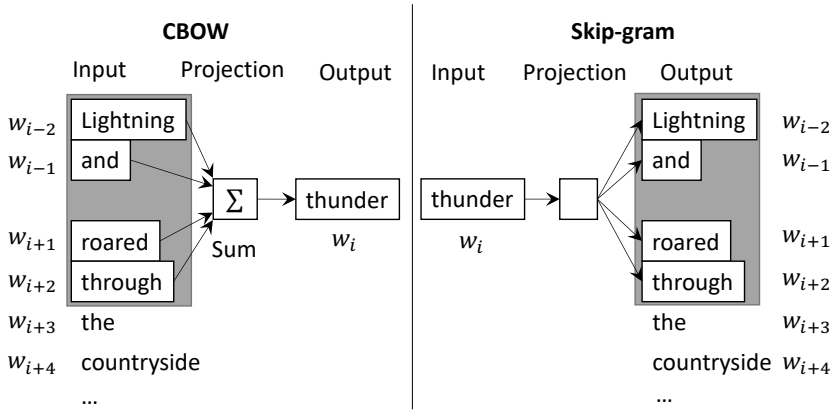



**Figure 9:** CBOW vs Skip-gram mode in Word2vec

and rely entirely on symbols overlapping. This makes these models perform poorly when synonymous words of different forms are present in the representation. Embeddings, specifically word embeddings, are an attempt to mitigate this problem by learning a fixed low dimensional dense representation that transforms each symbol into a vector space.

In the published papers in this thesis, I used word embeddings produced by two methods: Word2vec and GloVe. A newer method FastText builds upon these methods and includes the ability to create representations for words out of the known vocabulary by using subword embeddings (Bojanowski, Grave, Joulin, & Mikolov, 2017). Word embeddings are typically static; their representation is fixed per word regardless of the sentence. Some methods, that are more computationally intensive, overcome this limitation by contextualizing the word embedding based on where in a sentence the word appears. Examples of contextualized embedding methods include ELMo (Peters et al., 2018) and the Flair embeddings (Akbik, Blythe, & Vollgraf, 2018).

**Word2vec.** Word2vec is a model consisting of a shallow neural network that can operate in two modes: continuous bag-of-words (CBOW) and skip-gram (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013), see Figure 9. The first mode, CBOW, uses a context window that is continuously moved over sentences. The goal of CBOW is to learn representations by predicting the most probable word given a context window. Conversely, skip-gram learn representations by predicting nearby words given a focus word.

For a model such as Word2vec to work practically, various tricks are used such as: negative sub-sampling that limits the number of negative examples of words used during training and hierarchical softmax that reduces the computa-

tional complexity of large output-spaces. In addition, word embeddings rely on the *distributive hypothesis* (Harris, 1954) assuming that the meaning of a word tends to be defined by the words occurring with it.

GloVe differs from Word2vec in that word co-occurrences are used directly to learn representations, instead of the iterative process in Word2vec that moves a window over the training corpus. GloVe, as a consequence, requires more memory and a preprocessing step. However, Pennington, Socher, and Manning (2014) show that Glove produces better embeddings with higher accuracy in less time than Word2vec, regardless of mode given sufficient data.

**Intuition.** Embeddings make related symbols converge in vector space, essentially *embedding* relevant information into the vector representation.

**Practical use.** Word embeddings provide a practical solution to the problem of sparse vectors, and their associated huge spaces, by reducing these vectors into a representation in a vector space of related symbols. Using tensors with recurrent and/or encoding methods such as FOFE can process sequences of symbols. When training models for concrete tasks, embeddings are often initialized with pre-trained values to save time.

## 7.5 Sequence Compression Encoding

Single words in NLP are rarely useful: We need a sequence of them to define or convey an idea. Consequently, most models have to deal with sequences. Recurrent neural networks, discussed later, model this sequence behavior directly. The encoding is then embedded within the structure of the model.

Due to computational requirements, simplified methods are sometimes good enough. A baseline method using the bag-of-words technique is to compute the mean vector $\bar{\mathbf{e}}$, add all the vectors together, and normalize the sum by the vector length.

$$\bar{\mathbf{e}} = \frac{\sum_t \mathbf{e}_t}{\| \sum_t \mathbf{e}_t \|}$$

This approach may work for short sequences.

**FOFE.** Another approach is to use fixed-size ordinally-forgetting encoding (FOFE) (Zhang, Jiang, Xu, Hou, & Dai, 2015), which uses an exponential weighting scheme with a decay factor, $\alpha$ to produce an encoded vector $\mathbf{z}$:

$$\mathbf{z}_t = \mathbf{e}_t + \alpha \cdot \mathbf{z}_{t-1} \quad (1 \leq t \leq T)$$

This decay factor models the receptive field, and it can be proven that if $\alpha$ is tuned properly, it will result in unique vectors a neural network can successfully fit.

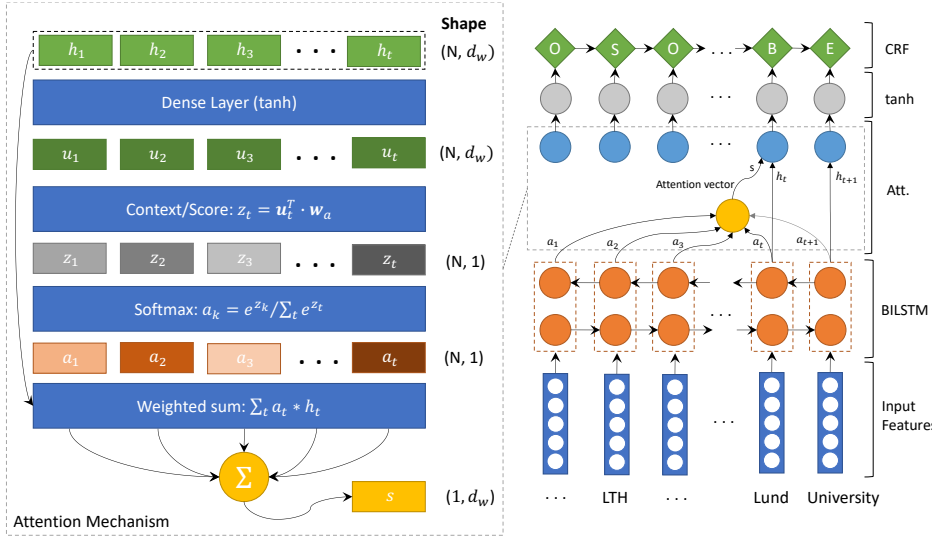



**Figure 10:** Attention mechanism in relation to a practical use in a BILSTM-CRF NER (L. Luo et al., 2017)

**Attention.** A more elaborate method uses a word attention mechanism, applied by Yang et al. (2016) to document classification.

Word attention is the transformation of a sequence of embeddings $\mathbf{e}_t$ into a single compressed vector representation $s$. The idea is to transform the sequence by passing each embeddings through a dense layer with a $\tanh$ activation $u_t$ with weights $W_a$ and $b_a$, and reduce it via a dot product with a context vector $u_a$ fitted during training. Ultimately, this produces a compressed vector representation $s$ as a softmax normalized $a_t$ weighted average of the embedding sequence $\mathbf{e}_t$.

$$
\begin{aligned}
u_t &= \tanh\left(W_a \mathbf{e}_t + b_a\right), \\
a_t &= \frac{\exp\left(\mathbf{u}_t^T \cdot \mathbf{w}_a\right)}{\sum_t \exp\left(\mathbf{u}_t^T \cdot \mathbf{w}_a\right)}, \\
s &= \sum_t a_t \, \mathbf{e}_t .
\end{aligned}
\tag{15}
$$

# 8 Models

In this section, I introduce the models I used in the published papers of this thesis.

## 8.1 Logistic Regression

Logistic regression is a a classification model used to determine the probability $p$ of class given the vector $\mathbf{x}$ by fitting weights $\mathbf{w}$.

Mathematically, $t = \mathbf{w} \cdot \mathbf{x}$ where $p = \sigma(t)$ is the sigmoid or logistic function:

$$\sigma(t) = \frac{1}{1 + e^{-t}} \tag{16}$$

Notable properties of the sigmoid function is that it is constrained to the range of 0 to 1 with 0.5 at the center.

## 8.2 Neural Network

An *artificial neural network* (ANN) is loosely modeled after the biological equivalent, and is a class of graphical models. These models were used in the design of the Ugglan and Hedwig entity linker and named entity recognition in Swedish, English, Spanish, and Chinese (Klang et al., 2017; Klang & Nugues, 2018a, 2018b, 2019b).

Figure 11 shows a simple neural network: It takes an input vector $\mathbf{x}$ and transforms it into a new vector $\mathbf{y}$. An ANN typically consists of multiple neurons, where each takes the input, weights the input with a neuron specific weight, and finally passes the weighted sum to an activation function.

Mathematically:

$$\mathrm{y}_k = \text{activation}(\sum_i \mathrm{x}_i \cdot \mathrm{w}_i^k + b_k) \tag{17}$$

or in matrix notation:

$$\mathbf{y} = \text{activation}(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{18}$$

where $\mathbf{x}$ is the input vector, $\mathbf{y}$ the output vector, and $\mathbf{W}$ is the weight matrix and $\mathbf{b}$ is a bias term for all connections to the input $\mathbf{x}$.

**Activation.** The choice of an activation is important as it introduces nonlinearity into the model, one distinct feature of neural networks.

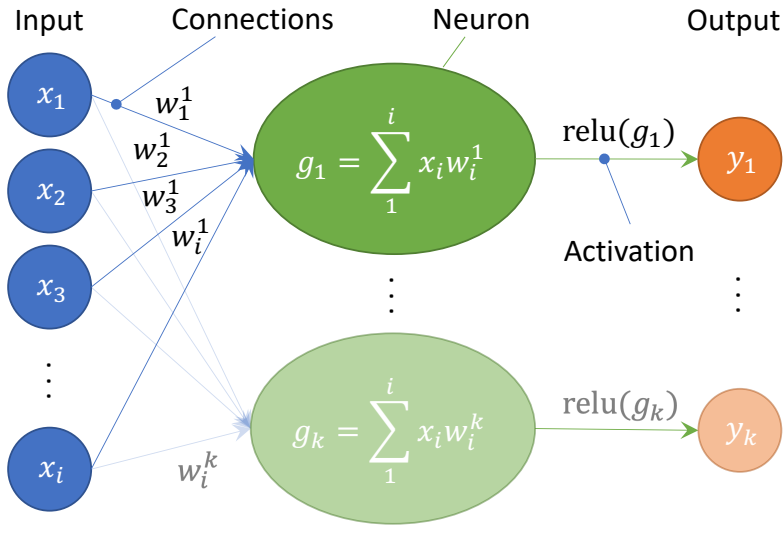



**Figure 11:** Artificial neuron with inputs $\mathbf{x}$ and outputs $\mathbf{y}$

Figure 12 shows common functions known to work well in practice:

$$\text{relu}(t) = \begin{cases} t & \text{if } t > 0, \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

$$\text{sigmoid}(t) = \sigma(t) = \frac{1}{1 + e^{-t}} \tag{20}$$

$$\tanh(t) = 2\,\sigma(t) - 1 \tag{21}$$

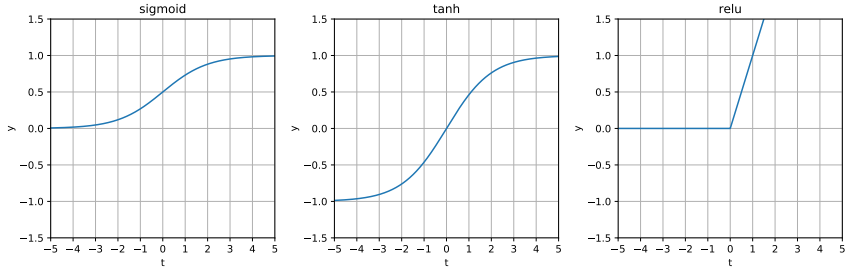



**Figure 12:** Sigmoid, tanh and relu activation functions

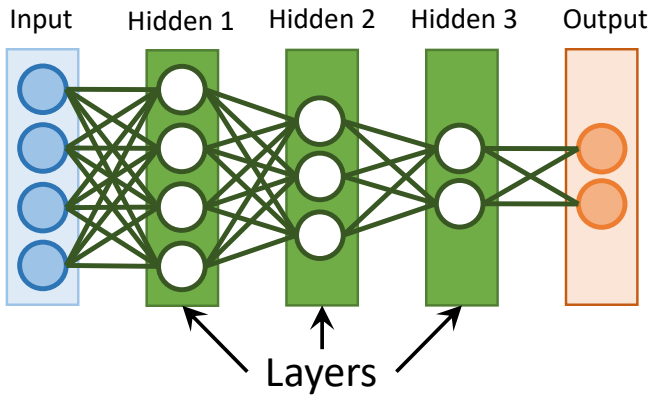



**Figure 13:** A artificial neural network, visualized as a stack of layers with a decreasing number of neurons in each layer.

**Layers.** Modelling large neural network architectures consists of reusing small models such as that in Sect. 8.2 and composing them together as a sequence of functions. Conceptualizing these models as layers simplifies the description of the encoding of the architecture of a neural network; see Figure 13.

The simple neural network described previously has multiple names, but in this dissertation, it will be called a dense layer as it transforms one dense vector into another.

There is a special type of layer called embedding layer, different from Word2vec and GloVe, in that it is only used to learn a vector representation for input symbols or contain a pre-trained embedding. Embedding layers are similar to standard dense layers, except for a few implementation details.

**Back-propagation.** Finding suitable parameters that make a neural network perform well on different tasks is the task of an *optimizer* combined with a *loss* function. Back-propagation is a popular method that exploits a derivative w.r.t. an error using the chain rule to optimize parameters during an iterative gradient descent.

This method consists of:

1. A forward pass, transforming an observation to a predicted output;
2. The computation of the loss by comparing prediction with the known correct answer;
3. A backward pass, using the error, i.e. the difference between expected and predicted output, and tuning the parameters going backwards through the

network using the derivative w.r.t. the error for each layer until the input has been reached.

Finding derivatives for complex models can be a time-consuming task. Popular machine-learning frameworks such as Tensorflow and Pytorch were designed to remove or reduce the need of doing this manually, requiring the user to only specify the forward pass. The backward pass relies on automatic differentiation to find the derivatives needed to optimize parameters.

## 8.3 Regularization

Training neural networks finds a local minimum of the error as given by the loss function. One key goal is to make the network generalize well, meaning that a network, given unseen observations, should produce reasonable predictions.

When networks do not generalize well, it can be due to the problem of *overfitting*, i.e. the network tunes its parameters to match precisely the outputs in the training set, but cannot generate meaningful predictions on unseen data.

To reduce the risk of overfitting, Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014) introduced a method called *dropout*, which randomly disables neurons. This dropout reduces the overfit risk associated with two neurons being tightly dependent on each other. The strength of the dropout is adjusted by specifying a share of neurons $p$ to drop randomly.

## 8.4 Recurrent Neural Networks

A recurrent neural network is an extension to ordinary feed-forward ANNs that incorporates a feedback mechanism to make use of previous time steps. Time steps can be rephrased as words in a sentence: Making prediction at step $i$ depends on predictions made previously from $i - 1$ to the start.

A recurrent neural network, simplified from Elman (1990) found in Pytorch[16] and in Keras as `SimpleRNN` (Chollet, 2017), has this mathematical definition:

$$y_t = \sigma(W_{ih}x_t + b_{ih} + U_{hh}y_{t-1} + b_{hh}), \tag{22}$$

where $\sigma$ is the activation function typically $\tanh$, the matrix $W_{ih}$ and its bias term $b_{ih}$ transforms the input features $x_t$ into a hidden representation, and finally the matrix $U_{hh}$ and its bias term $b_{ih}$ transforms the previous output $y_{t-1}$ prediction into a feature that is added onto the hidden representation.

In the litterature, two simple recurrent networks are mentioned: Jordan (1986) and Elman (1990) networks. The Elman network can be constructed using a `SimpleRNN` layer and a dense layer in sequence. The Jordan network is similar to the Elman network but with one difference: instead of using the `SimpleRNN` output as feedback, it uses the output of the dense layer as feedback forcing an intermediate step.

[16] `https://pytorch.org/docs/stable/nn.html#rnn` retrieved 2019-08-15

## 8.5 LSTM

Simple recurrent neural network models are hard to fit due the vanishing and exploding gradient problem (Hochreiter & Schmidhuber, 1997). The *long short-term memory* (LSTM) model was designed by Hochreiter and Schmidhuber (1997) to address and solve this problem.

There exist different variants of LSTM, the one I describe is based on the Pytorch[17] implementation.

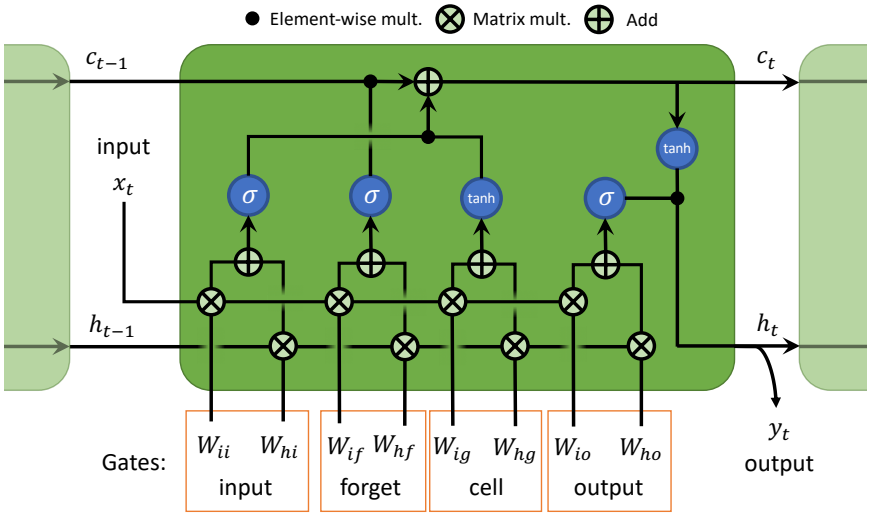



**Figure 14:** Visualization of one time step $t$ given input $x_t$ in a LSTM cell to predict output $y_t$, bias terms are excluded.

The core principle of a LSTM is a memory cell that is used to predict the output using four gates: input, forget, cell and output, as can shown in Figure 14.

Mathematically, this yields:

$$i_t = \sigma\left(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}\right) \tag{23}$$
$$f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right) \tag{24}$$
$$g_t = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right) \tag{25}$$
$$o_t = \sigma\left(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}\right) \tag{26}$$
$$c_t = f_t * c_{t-1} + i_t * g_t \tag{27}$$
$$h_t = o_t * \tanh c_t \tag{28}$$

[17] https://pytorch.org/docs/stable/nn.html#lstm

where $i_t$, $f_t$, $g_t$, $o_t$ are the input, forget, cell, and output gate, respectively, $h_t$ is the hidden state and the output in each step, and $c_t$ is the cell state. $\sigma$ and $\tanh$ are activation functions.

This construction allows the model to save long-term information found many steps earlier, but still allows short-term predictions that do not modify the cell state.

**Intuition.** The sigmoid activation function produces values with a range between 0 and 1, and can be used to choose what information to retain via an element-wise multiplication. The $\tanh$ activation can be used to add information. The cell-state $c_t$ could be called the long-term part of the model and the hidden state $h_t$, the short term. The cell state can remove information via the forget gate, and add information via the input gate and cell gate. As the model can block out input and short term information from the hidden state, the model can store information over many steps. The short term $h_t$ combines the input, short-term, and long-term to produce a final prediction.

**Related models.** The LSTM model is complicated and, as such, hard to optimize. It therefore requires heavy computations. Cho et al. (2014) introduced a simplified variant: the *gated recurrent units* (GRU). This model tries to simplify LSTM, reducing the overall complexity and was found competitive to LSTM in e.g. speech synthesis (Chung, Gulcehre, Cho, & Bengio, 2015).

## 8.6 Bidirectional Recurrent Neural Networks

Recurrence for sequences can be done for arbitrary orderings; a common trick is then to use bidirectional layers, in which we essentially train two independent layers. This is shown to perform better compared with using only a single direction (Schuster & Paliwal, 1997). We use different input directions and, in the end, combine the results. We used this method when training the LSTM based named entity recognizer (Klang & Nugues, 2018a).

# 9 Document Database

Docforia collections are accessed sequentially and each document is standalone, making search and random sampling of the collection difficult. To make search easier, I designed Panforia (Klang & Nugues, 2018b), a system to index Docforia dumps. Panforia makes large collection accessible by search and includes specialized tools for visualization of the layers contained within documents.

Panforia (see Figure 15) is built on top of a full-text search engine: Lucene, and includes a web browsed visualizer capable of rendering the potentially many nodes a document can contain. The visualization component also allows the dynamic selection of node and edge layers to visualize.

Panforia can index millions of documents in a few hours over multiple languages and could be run locally on a laptop. Panforia enables quick referencing of annotated layers produced by algorithms applied to large collections of text. The closest match to Panforia I could find is the Open Semantic Search[18], an open source project that can index millions of document with a modular architecture. However, it was not designed to support the complex layered nature of Docria documents that Panforia supports.

[18] `https://www.opensemanticsearch.org/`

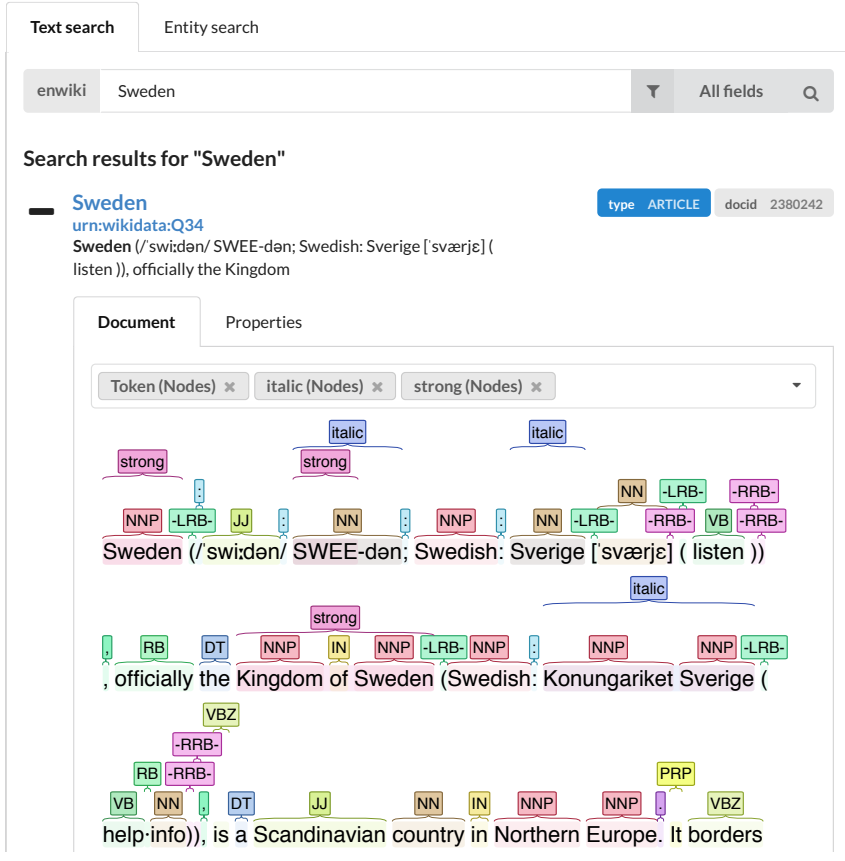



**Figure 15:** The Panoria frontend

# 10 Named Entity Recognition

Named entity recognition (NER) is the task of finding phrases referring to a *thing* with a name, a *named entity*, and assigning each phrase a category such as the class of persons.

Applications of named entity recognition can be found in areas such as:

- Information categorization, by adding tags using named entities as basis to e.g. news articles;
- Automatic document summarization, by using entities as a basis to determine what is mentioned and thus what may be important;
- Search, by using named entities as means of improve ranking relevance and reducing search space;
- Question answering, by identifying and aggregating relations between entities;
- Identification of major events, by finding and aggregating mentions of locations in tweets which could aid in earthquake detection or events requiring a fast first response from authorities;
- Brands and security, by tracking and aggregating mentions about an organization or persons.

We developed a multilingual named entity recognizer as a preliminary step to entity linking so that it marks potential mentions accurately (Klang et al., 2017; Klang & Nugues, 2019b). We designed it so that it could handle multiple languages with a flexible output of classes. From an input of words or word embeddings, the named entity recognizer predicts an output in the form of IOB or IOBES tags as described in Sect 3.1.

To ease the linking step, I focused on the elimination of potentially noisy mentions, i.e. I favored precision over recall so that the candidate link graph becomes smaller and less noisy. The goal was then to make the NER and linker work well together. This implied that we did not try to over-optimize the NER intermediate step at the expense of the linking score.

## 10.1 Related Work

The proceedings of the message understanding conference (MUC-5) (Sundheim, 1995) provide one of the earliest definitions of a named entity. In MUC-5, a named entity is a *thing*, which belongs to one of three categories: person, organization, or location. In addition, MUC-5 included temporal expressions (TIMEX), such as dates or times, and numerical expressions (NUMEX), essentially typed numbers, which include currencies and percentage values.

MUC-5 was also one of the first conferences with the F1 score as primary metric when conducting evaluations of NER systems. The best system in MUC-5 achieved a F1 score of 96.42% on a small annotated corpora of 30 articles.

Multilingual named entity recognition with larger corpora was attempted in CoNLL-03 (Tjong Kim Sang & De Meulder, 2003) with the newswires corpus RCV-1 (Sect. 3.1) from Reuters. CoNLL-03 excluded TIMEX and NUMEX used in MUC-5 and introduced a new class: miscellaneous (MISC). CoNLL-03 is frequently cited in the literature and has become one of the standards when evaluating English NER performance.

Florian, Ittycheriah, Jing, and Zhang (2003) achieved an F1 score of 88.76% with an ensemble of systems in CoNLL-03. This system included one classifier based on logistic regression, which, on its own, was the best classifier. Since then, Ratinov and Roth (2009) improved the score to 90.57% using hand-crafted features and dictionaries. Ma and Hovy (2016) used recurrent neural networks such as bidirectional LSTMs with word embeddings as input improving the F1 score to 91.21%. Peters et al. (2018) used a new contextualized embedding called ELMo to further improve the score to 92.22%. Finally, the FLAIR system (Akbik et al., 2018) reached 93.09%. This corresponds to the state of the art, as of today. As a side-note, the transformer-based $\text{BERT}_{\text{LARGE}}$ model achieved a competitive F1 score of 92.8% (Devlin, Chang, Lee, & Toutanova, 2019).

Dictionary methods using rules and statistical thresholds have proven successful in the context of entity linking (Eckhardt, Hreško, Procházka, & Smr, 2014; Lipczak, Koushkestani, & Milios, 2014). However, they tend to overgenerate mentions since many common words are names in specific contexts. Increasing thresholds, thus increasing precision in the dictionary method discards many relevant mentions. This increase can cause a poor recall, excluding mentions that have few entity candidates. Mentions with few candidates further aid in the resolution of other highly ambiguous mentions, because they may restrict possible entities due to their relations.

NER scores are language, corpus, and domain sensitive as can be seen in our experiments on Swedish and English (Klang & Nugues, 2018a). The annotated RCV-1 subset from CoNLL03 is based on news articles. These articles have been written by journalists proficient with the written word. But news articles are only one style of writing that we can find in the internet. The TAC EDL corpora, which are mixtures of news and discussion forum texts written by the general public in three languages, contain higher levels of ambiguity and more differences in style and domain.

## 10.2 Entity Classification

Classifying named entities into broad categories can help reduce ambiguity when linking, by excluding irrelevant hypotheses that are not things or do not conform to the annotation guidelines.

Wikipedia has a categorization system for pages corresponding to things such as persons, organizations, or locations. This is useful as we can select a specific set of entities based on their category, for instance all the persons. Unfortunately, Wikipedia categories are not completely coherent and are language-specific. In addition, in most annotated corpora, we have predefined sets of categories, as in CoNLL03 and TAC-EDL, that have no direct mapping to the Wikipedia nomenclature.

A first technique to standardize the categorization of the Wikipedia pages is to use Wikidata. Wikidata provides additional information, in this case, via the `instance-of`, and `subclass-of` property. However, this property is loosely defined and simple top-level Wikidata items, such as human settlement, will not match cities, villages, and metropolitan areas.

To solve this problem, I reformulated it this way: Given an entity in Wikidata, traverse the graph to create a set of features, and use these features to classify the entity into a class from a predefined set of classes, for instance that of CoNLL, MUC, or TAC-EDL. I applied this idea in Klang and Nugues (2019b) with in-domain data, i.e. the training corpus of TAC-EDL, and a training set constructed from entities extracted with manually defined rules.

The extraction rules consisted of queries that traversed the Wikidata graph, determining if a path exists between two entities using a breadth-first search. Rules matching a broad set of entities were filtered and sampled to keep the training set balanced. All unlabeled entities were assumed to be member of the outside `NONE` class. However, during training, the `NONE` class was sampled, with 10% per mini-batch. This sampling forces the classifier to assign `NONE` by default, assuming there is no feature overlap with other classes.

Finally, after training, I applied the fitted model to all the entities in Wikidata to create an entity class mapping that I subsequently used in the Hedwig linker.

# 11 Named Entity Linking

Named entity linking can be divided into two broad groups, supervised and unsupervised:

1. Supervised entity linking requires that all the linkable things are known and typically available in a entity repository. Wikidata is an example of such a repository that I used in this thesis.

2. Unsupervised entity linking attempts to construct a reference for all the detected names, not part of the repository. It links the mentions referring to such entities to unique generated references. This group corresponds to a coreference resolution across multiple documents (Bagga & Baldwin, 1998).

Supervised entity linking provides unique identifiers that can be referenced, such as the Q-numbers in Wikidata. Conversely, unsupervised entity linking does not have pre-defined identifiers.

In this thesis, I developed two entity linkers: Ugglan and Hedwig, both with supervised and unsupervised aspects:

1. Ugglan (Klang et al., 2017) is based on creating a mention-entity, entity-entity graph, running a Pagerank algorithm and then reranked with neural network trained on TAC-EDL data with features such as word context encoded with the FOFE method.

2. Hedwig (Klang & Nugues, 2019b) builds on Ugglan with a re-implemented Pagerank algorithm, additional features such as entity-mention and entity-word constructed using point-wise mutual information (PMI) in entity contexts from Wikipedia.

Named entity linking is frequently formulated as a ranking problem, i.e. given a list of potential candidate entities for each mention, rank them with respect to a score. The potential candidates are derived from a dictionary in the supervised case. This dictionary is frequently populated with names and aliases of known entities. In Klang et al. (2017) and Klang and Nugues (2019b), we used titles, redirect titles, the text of links in Wikipedia, and more to derive such a dictionary.

Entity candidates frequently incorporate *priors* in the computation of scores. A prior is a statistical or similarity measure derived from a larger linked corpora such as Wikipedia. Medelyan, Witten, and Milne (2008) proposed a measure called *commonness*, which is the conditional probability that a given mention refers to an entity candidate. Ranking by commonness is a strong baseline as shown in Klang and Nugues (2019b).

## 11.1 Related work

A variety of methods has applied to carry out entity linking such as simple classification models (Bunescu & Paşca, 2006; Cucerzan, 2007; Milne & Witten, 2008), end-to-end linkage with a voting scheme for linkage (Ferragina & Scaiella, 2010), graphical models (Guo & Barbosa, 2014; Hoffart et al., 2011), integer linear programming (Cheng & Roth, 2013), fully probabilistic models (Ganea, Ganea, Lucchi, Eickhoff, & Hofmann, 2016) to deeper neural models (Ganea & Hofmann, 2017; Sil, Kundu, Florian, & Hamza, 2018).

Wainwright, Jordan, et al. (2008)[19] showed that entity linking that tries to maximize local and global agreement jointly is a NP-hard problem. Therefore, most authors approximate or simplify this to make it feasible.

[19] cited in Globerson et al. (2016).

## 11.2 Agreement

Linking methods could broadly be divided into local and global methods. Local methods try to link mentions independently, based on their surrounding context. Global methods, in contrast, try to find a coherent, agreeable linkage, which satisfies a global objective, usually across a single document (Klang et al., 2017; Klang & Nugues, 2019b).

One approach used in Klang and Nugues (2019b) is to use the entity coreferences found in multiple Wikipedia editions as they can be traced via Wikidata Q-numbers. We populate the graph using entity candidates derived from mentions and add mention-entity and entity-entity edges similar to Hoffart et al. (2011). In addition, we add entity-mention and entity-word edges to introduce more of the local context as can be seen in Figure 16.

Concretely, we search for local features in context: words, other mentions, and other entity candidates. The graph is constructed by adding vertices such as mention, entity, and words, and directed edges representing the features found in a local context that bind these elements together. The features are precomputed lists per candidate entity and were derived using the context surrounding links in Wikipedia. The features consist of language-specific parts that include edges binding mention-entity and entity-word, and a multilingual part such as entity-entity. Once we populated the graph with all vertices and directed edges that represent connections to local features, we run the Pagerank algorithm over this graph. Pagerank assigns a weight to each vertex, specifically entity candidates, which is used to link mentions.

Depending on the evaluation framework and test set, the resulting linked document may contain entities that do not conform to the annotation guidelines, for instance TAC-EDL 2017 excludes concepts or references to works and events. To mitigate this, we assigned a class based on the mapping discussed in Sect. 10.2 and we removed entities with non-conforming classes from the linked graph.

## 11.3 Discussion

There is no consensus on the dataset to use for evaluation; this makes it hard to compare different approaches. GERBIL (Röder, Usbeck, & Ngonga Ngomo, 2018) is a online platform that enables a researcher to compare his or her system with others on a variety of English datasets. An English entity-linked version of CoNLL03 Hoffart et al. (2011) is another popular dataset.

Entity linking systems consist of many moving parts, namely: named entity recognizers, databases of entities, word embeddings, and more. The fact that most systems contain so many moving parts makes it hard to independently reproduce suggested systems or requires considerable effort. In addition, the computational resources required are not a factor when the goal is only to produce the best linkage on a small dataset. All these reasons combined made that the work in this

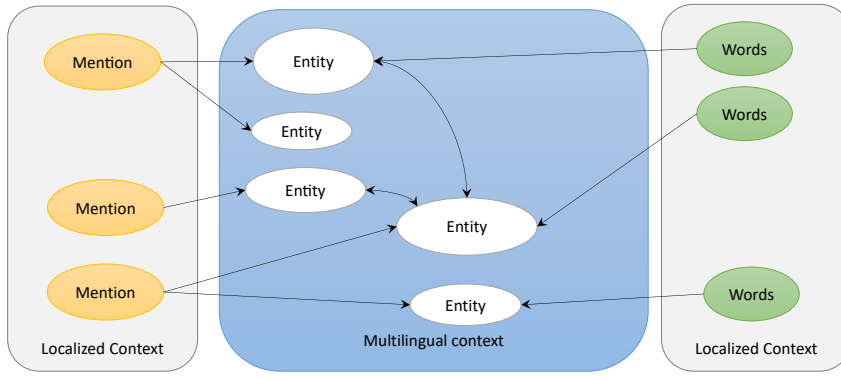



**Figure 16:** Localized and multilingual pagerank linkage

thesis opted for a page-rank method in linkage, simply because it is fast and produces reasonable results.

Ugglan achieved a `CEAFmC+` of 56.1%, and Hedwig 59.9% on the TAC 2017 multilingual dataset; recall is the weakness of both systems with 46.4% and 50.2%, while it reaches a high precision of 89.4% for Ugglan and 91.4% for Hedwig. If we exclude the nominal mentions, a peculiarity of TAC-EDL 2017, the recall increases to a `CEAFmC+` of 65.8% for Ugglan, and 69.0% for Hedwig with improvements in both recall and precision. Finding the correct mentions in Hedwig reaches an F1 score of 85.6%, and linkage 81.5%, an increase over Ugglan 80.8% to find names and 75.7% in linkage.

It should be noted that Ugglan used a neural reranker that improved the score of the baseline system. The results obtained by Hedwig advocate that a stronger NER in combination with a better Pagerank graph produce better scores than a re-ranker. In addition, it may be significantly faster than deep neural network systems that often are tricky to train properly.

# 12 Conclusion

Through this thesis, the common denominator is the development of concrete methods that can be used to store, transform, and process encyclopedic knowledge. I applied these methods in two main applications: named entity recognition and named entity linking.

I developed an infrastructure together with a suitable document representation to process Wikipedia and related resources. The document representation was refined through three revisions:

1. The RDF triplets inspired WIKIPARQ that can represent the structures in Wikipedia (Klang & Nugues, 2016d);

2. Docforia supporting more complex representations (Klang & Nugues, 2017); and finally

3. Docria that tries to preserve the best qualities from Docforia, while introducing a more rigid structure implemented in Python and Java (Klang & Nugues, 2019a).

In addition, mixing and matching commonly available libraries in a scalable way is time consuming. To alleviate this, I made an attempt to unify generic NLP libraries with the Langforia framework (Klang & Nugues, 2016b). Langforia abstracts libraries and can define pipelines for offline and online uses. The online version is used for prototyping and allows pipelines to be pluggable.

These infrastructure and document modeling tools enabled me to create named entity recognizers and named entity linkers. To the best of my knowledge, when it comes to published results, I have produced a state-of-the-art named entity recognizer on the four majority classes in SUC 3.0 using a BILSTM-CRF method (Klang & Nugues, 2018a). We have produced a competitive named entity linker applied to three languages in TAC EDL 2017, English, Chinese, and Spanish, reaching a final trilingual score of 59.9% CEAFmC+, 71.9% F1 on linkage, and 69.5% F1 in finding and classifying names correctly (Klang & Nugues, 2019b). As resources, I only used Wikipedia, Wikidata, and the datasets provided by the TAC organizers to train an in-domain named entity recognizer.

The entities resolved over multiple languages with a common repository served as distant supervisors to train a semantic role labeler (SRL) (Exner et al., 2016). While there are SRL tools for English, there are no large annotated datasets for many languages, including Swedish or French, and this means that we cannot apply them supervised learning techniques. Concretely, we applied the entity resolution to English, French, and Swedish texts; we aligned sets of identical entities across multilingual sentences, such as English and Swedish; we applied an existing SRL for English, and we transferred the semantic annotation from English to Swedish. Finally, we trained the SRL on this transferred annotation.

These tools, methods, and representations will serve as a foundation for future work.

## 12.1 Future Work

There are multiple possible avenues for future research: constructing a question answering system, training new semantic role labelling (SRL) systems for more languages expanding on the work in Exner et al. (2016), introducing the overall background of news articles through entities, either historical, geographical, political, for example, and improving search.

Constructing a question answering system can consist of the three following parts:

1. Find relations using entities as anchors with semantic roles aligned with multiple languages;
2. Enrich Wikidata with these relations resulting in a knowledge graph with information previously only available as text;
3. And finally, build a question answering system that transforms a free-form query into a lookup of the knowledge graph, generating a human understandable answer.

Another possibility is the construction of a system that introduces the historical, geographical, political, or scientific background in articles by the entities mentioned in the text. Such a system could potentially be used to identify factual errors and/or contradictions while cross-checking it against known information in e.g. Wikidata.

Finding and linking entities in queries for a search system with a huge collection of documents is a third possible future research area. Concretely, the system would use entity candidates and linked entities to reduce the search space by a progressive refinement. Such a system could be constructed as a sequence of questions to the user that would add constraints to exclude candidates. Eventually, the space would be small enough to apply computationally expensive methods for ranking, helping the user to find more relevant information.

# Bibliography

Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics* (pp. 1638–1649).

Bagga, A., & Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th annual meeting of the association for computational linguistics and 17th international conference on computational linguistics* (Vol. 1, pp. 79–85). Montréal, Québec, Canada: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/980845.980859 doi: 10.3115/980845.980859

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Brants, S., Dipper, S., Hansen, S., Lezius, W., & Smith, G. (2002). The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories* (Vol. 168).

Buchholz, S., & Marsi, E. (2006). CoNLL-x shared task on multilingual dependency parsing. In *Proceedings of the tenth conference on computational natural language learning* (pp. 149–164). New York City, New York: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=1596276.1596305

Bunescu, R., & Paşca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *11th conference of the European chapter of the association for computational linguistics.*

Cheng, X., & Roth, D. (2013). Relational inference for wikification. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1787–1796).

Chinchor, N. (1992, June, 16-18). MUC-4 evaluation metrics. In *Fourth message understanding conference (MUC-4), proceedings of a conference held in McLean, Virginia.* Retrieved from https://www.aclweb.org/anthology/M92-1002

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. (2017). *Deep learning with Python* (1st ed.). Greenwich, CT, USA:

Manning Publications Co.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015). Gated feedback recurrent neural networks. In *International conference on machine learning* (pp. 2067–2075).

Clarke, L., Glendinning, I., & Hempel, R. (1994). The mpi message passing interface standard. In K. M. Decker & R. M. Rehmann (Eds.), *Programming environments for massively parallel distributed systems* (pp. 213–218). Basel: Birkhäuser Basel.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Cunningham, H., Humphreys, K., Gaizauskas, R., & Wilks, Y. (1997). Software infrastructure for natural language processing. In *Proceedings of the fifth conference on applied natural language processing* (pp. 237–244). Washington, DC: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/974557.974592 doi: 10.3115/974557.974592

Dean, J., & Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th conference on symposium on operating systems design & implementation - volume 6* (pp. 10–10). San Francisco, CA: USENIX Association. Retrieved from http://dl.acm.org/citation.cfm?id=1251254.1251264

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies* (Vol. 1 (Long and Short Papers), pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/N19-1423

Eckhardt, A., Hreško, J., Procházka, J., & Smr, O. (2014). Entity linking based on the co-occurrence graph and entity probability. In *Proceedings of the first international workshop on entity recognition & disambiguation* (pp. 37–44).

Ehrlinger, L., & Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, *48*.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179-211.

Exner, P., Klang, M., & Nugues, P. (2015). A distant supervision approach to semantic role labeling. In *Proceedings of the fourth joint conference on lexical and computational semantics (*SEM 2015)* (pp. 239–248).

Exner, P., Klang, M., & Nugues, P. (2016). Multilingual supervision of semantic annotation. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 1007–1017).

Ferragina, P., & Scaiella, U. (2010). Tagme: on-the-fly annotation of short text fragments (by Wikipedia entities). In *Proceedings of the 19th ACM international conference on information and knowledge management* (pp. 1625–1628).

Ferrucci, D., & Lally, A. (2004). UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, *10*(3-4), 327–348.

Florian, R., Ittycheriah, A., Jing, H., & Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003 volume 4* (pp. 168–171).

Ganea, O.-E., Ganea, M., Lucchi, A., Eickhoff, C., & Hofmann, T. (2016). Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th international conference on world wide web* (pp. 927–938).

Ganea, O.-E., & Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*.

Globerson, A., Lazic, N., Chakrabarti, S., Subramanya, A., Ringaard, M., & Pereira, F. (2016). Collective entity resolution with multi-focal attention. In *Proceedings of the 54th annual meeting of the association for computational linguistics* (Vol. 1: Long Papers, pp. 621–631).

Guo, Z., & Barbosa, D. (2014). Robust entity linking via random walks. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management* (pp. 499–508).

Harman, D. (1992, October). The DARPA TIPSTER project. *SIGIR Forum*, *26*(2), 26–28. Retrieved from `http://doi.acm.org/10.1145/146565.146567` doi: 10.1145/146565.146567

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146-162. Retrieved from `https://doi.org/10.1080/00437956.1954.11659520` doi: 10.1080/00437956.1954.11659520

Hellmann, S., Lehmann, J., Auer, S., & Brümmer, M. (2013, October). Integrating

NLP using linked data. In *Proceedings of the 12th international semantic web conference.* Sydney, Australia.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780. Retrieved from `https://doi.org/10.1162/neco.1997.9.8.1735` doi: 10.1162/neco.1997.9.8.1735

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., ... Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 782–792).

Ji, H., Pan, X., Zhang, B., Nothman, J., Mayfield, J., McNamee, P., & Costello, C. (2017). Overview of TAC-KBP 2017 13 languages entity discovery and linking. In *Proceedings of the tenth text analysis conference (TAC 2017).*

Jordan, M. I. (1986). Serial order: A parallel distributed processing approach, ics report 8604. *Institute for Cognitive Science, UCSD, La Jolla*.

Klang, M., Dib, F., & Nugues, P. (2017, November). Overview of the Ugglan entity discovery and linking system. In *Proceedings of the tenth text analysis conference (TAC 2017).* Gaithersburg, Maryland.

Klang, M., & Nugues, P. (2016a). Docforia: A multilayer document model.. (Sixth Swedish Language Technology Conference (SLTC 2016) ; Conference date: 17-11-2016 Through 18-11-2016)

Klang, M., & Nugues, P. (2016b). Langforia: Language pipelines for annotating large collections of documents. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: System demonstrations* (pp. 74–78).

Klang, M., & Nugues, P. (2016c, 12). Pairing Wikipedia articles across languages. In *Proceedings of the open knowledge base and question answering workshop (OKBQA 2016)* (pp. 72–76). The COLING 2016 Organizing Committee.

Klang, M., & Nugues, P. (2016d, 5). Wikiparq: A tabulated wikipedia resource using the parquet format. In *Proceedings of the 10th international conference on language resources and evaluation (LREC 2016)* (pp. 4141–4148). European Language Resources Association (ELRA).

Klang, M., & Nugues, P. (2017). Docforia: A multilayer document model. In *Proceedings of the 21st Nordic conference of computational linguistics.* Linköping University Electronic Press.

Klang, M., & Nugues, P. (2018a, November 7). Comparing LSTM and

FOFE-based architectures for named entity recognition.. Retrieved from http://sltc2018.su.se/ (Seventh Swedish Language Technology Conference, SLTC 2018 ; Conference date: 07-11-2018 Through 09-11-2018)

Klang, M., & Nugues, P. (2018b). Linking, searching, and visualizing entities in wikipedia. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)* (pp. 3426–3432).

Klang, M., & Nugues, P. (2019a, October 1-2). Docria: Processing and storing linguistic data with wikipedia. In *Proceedings of the 22nd nordic conference on computational linguistics.* Turku.

Klang, M., & Nugues, P. (2019b). *Hedwig: A named entity linker.* (To be submitted.)

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*(2), 83–97.

Lipczak, M., Koushkestani, A., & Milios, E. (2014). Tulip: Lightweight entity recognition and disambiguation using wikipedia-based topic centroids. In *Proceedings of the first international workshop on entity recognition & disambiguation* (pp. 31–36).

Luo, L., Yang, Z., Yang, P., Zhang, Y., Wang, L., Lin, H., & Wang, J. (2017, 11). An attention-based BiLSTM-CRF approach to document-level chemical named entity recognition, journal = Bioinformatics. , *34*(8), 1381-1388. Retrieved from https://doi.org/10.1093/bioinformatics/btx761 doi: 10.1093/bioinformatics/btx761

Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of the conference on human language technology and empirical methods in natural language processing* (pp. 25–32). Vancouver, British Columbia: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/1220575.1220579 doi: 10.3115/1220575.1220579

Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th annual meeting of the association for computational linguistics* (Vol. 1 (Long Papers), pp. 1064–1074).

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for computational linguistics (ACL) system demonstrations* (pp. 55–60). Retrieved from http://www.aclweb.org/anthology/P/P14/P14-5010

Medelyan, O., Witten, I. H., & Milne, D. (2008). Topic indexing with wikipedia. In *Proceedings of the aaai wikiai workshop* (Vol. 1, pp. 19–24).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems* (Vol. 2, pp. 3111–3119). Lake Tahoe, Nevada: Curran Associates Inc. Retrieved from `http://dl.acm.org/citation.cfm?id=2999792.2999959`

Milne, D., & Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the 17th acm conference on information and knowledge management* (pp. 509–518).

Moosavi, N. S., & Strube, M. (2016). Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th annual meeting of the association for computational linguistics* (Vol. 1: Long Papers, pp. 632–642).

Nivre, J., Abrams, M., Agić, Ž., Ahrenberg, L., Antonsen, L., Aplonova, K., ... Zhu, H. (2018). *Universal dependencies 2.3*. Retrieved from `http://hdl.handle.net/11234/1-2895` (LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University)

Osborne, M. (1999). *CoNLL-99. computational natural language learning. proceedings of a workshop sponsored by the association for computational linguistics*. Association for Computational Linguistics (ACL).

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 conference of the North American chapter of the Association for computational linguistics: Human language technologies* (Vol. 1 (Long Papers), pp. 2227–2237).

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R., & Xue, N. (2011). CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proceedings of the fifteenth conference on computational natural language learning: Shared task* (pp. 1–27). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from `http://dl.acm.org/citation.cfm?id=2132936.2132937`

Pustylnikov, O., Mehler, A., & Gleim, R. (2008). A unified database of depen-

dency treebanks: Integrating, quantifying & evaluating dependency data. In *Proceedings of the sixth international conference on language resources and evaluation (LREC'08).* Retrieved from http://www.lrec-conf.org/proceedings/lrec2008/pdf/851_paper.pdf

Qi, P., Dozat, T., Zhang, Y., & Manning, C. D. (2018, October). Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies* (pp. 160–170). Brussels, Belgium: Association for Computational Linguistics. Retrieved from https://nlp.stanford.edu/pubs/qi2018universal.pdf

Ratinov, L., & Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning* (pp. 147–155).

Röder, M., Usbeck, R., & Ngonga Ngomo, A.-C. (2018). Gerbil–benchmarking named entity recognition and linking consistently. *Semantic Web*, *9*(5), 605–625.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Sil, A., Kundu, G., Florian, R., & Hamza, W. (2018). Neural cross-lingual entity linking. In *Thirty-second aaai conference on artificial intelligence.*

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Sundheim, B. M. (1995). Overview of results of the MUC-6 evaluation. In *Proceedings of the 6th conference on message understanding* (pp. 13–31). Columbia, Maryland: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/1072399.1072402 doi: 10.3115/1072399.1072402

TEI Consortium (Ed.). (2019). *TEI P5: Guidelines for electronic text encoding and interchange (version 3.6).* TEI Consortium. Retrieved from http://www.tei-c.org/Guidelines/P5/ (Accessed September 10, 2019)

Thompson, H. S., & McKelvie, D. (1997). Hyperlink semantics for standoff markup of read-only documents. In *SGML'97 conference proceedings* (p. 227-229). Barcelona.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of conll-2002* (pp. 155–158). Taipei, Taiwan.

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003 - volume 4* (pp. 142–147). Edmonton, Canada: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/1119176.1119195 doi: 10.3115/1119176.1119195

van Gompel, M., & Reynaert, M. (2013, Dec.). Folia: A practical XML format for linguistic annotation a descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, *3*, 63-81. Retrieved from https://clinjournal.org/clinj/article/view/26

Vilain, M., Burger, J., Aberdeen, J., Connolly, D., & Hirschman, L. (1995, November 6-8). A model-theoretic coreference scoring scheme. In *Sixth message understanding conference (MUC-6): Proceedings of a conference held in Columbia, Maryland* (p. 45-52).

Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, *1*(1–2), 1–305.

Wirth, R., & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (pp. 29–39).

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the Association for computational linguistics: human language technologies* (pp. 1480–1489).

Zhang, S., Jiang, H., Xu, M., Hou, J., & Dai, L. (2015). The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing* (Vol. 2: Short Papers, pp. 495–500).

# Named Entity Disambiguation in a Question Answering System

PAPER I

# Named Entity Disambiguation in a Question Answering System

**Marcus Klang, Pierre Nugues**

Lund University, Department of Computer Science
S-221 00 Lund, Sweden
marcus.klang@cs.lth.se, pierre.nugues@cs.lth.se

**Abstract**

In this paper, we describe how we use a named entity disambiguation module to merge entities in a question answering system. The question answering system uses a baseline passage retrieval component that extracts paragraphs from the Swedish version of Wikipedia. The passages are indexed and ranked using the Lucene platform. Prior to the indexing, we carried out a recognition and disambiguation of the named entities. We used the Stagger part-of-speech tagger to tag the documents and we implemented a disambiguation module. We extracted and merged the answer candidates using their wikidata Q-number. This enables the question-answering system to benefit from the synonymy facility of Wikipedia as well as an extended set of properties.

## 1. Introduction

Factoid question answering systems aim at answering short questions, where answers often consist of a single concept or a named entity. The typical architecture of most such systems features a question analysis step, a passage retrieval step, where documents containing the answer are extracted from a collection of texts, and an extraction and ranking step of the candidate answers. See IBM Watson (Ferrucci, 2012) for an example of such an architecture applied to the *Jeopardy!* quiz show.

When the answers correspond to named entities, their identification in the texts that serve as knowledge source and their disambiguation enables the answer extraction step to merge more accurately the candidates coming from different passages. In addition, it makes it possible to carry out inferencing over external structured data sources that can be associated to these entities. While there exist available named entity disambiguators targeted to English, such as AIDA (Hoffart et al., 2011) or SMAPH (Cornolti et al., 2014), to the best of knowledge, there is nothing equivalent for Swedish. In this paper, we describe a named entity disambiguator for Swedish and its integration to the Hajen question-answering system.

## 2. Named Entity Disambiguation

Named entity disambiguation or named entity linking consists in associating a sequence of words, typically a proper noun, to a unique identifier. As source of identifiers, we can use entity repositories, such as Freebase (Bollacker et al., 2008) or Wikidata[1], that define popular nomenclatures.

In the Hajen system, we use the wikidata identifiers that gather properties collected from the Wikipedia online encyclopedia and the infobox tabulated information that is associated to some of its articles. The city of Lund, Sweden, for example, has Q2167 as wikidata identifier, while Lund University has number Q218506. The associated properties are then accessible from the URL: `http://www.wikidata.org/wiki/Qxxx`.

We carry out the named entity linking in three steps: We first use a part-of-speech tagger, Stagger (Östling, 2013), to recognize the named entities; we link the strings to possible wikidata identifiers; and we finally disambiguate them using their popularity, commonness, and a Boolean context method. In a sentence like:

> Göran Persson var statsminister mellan åren 1996 och 2006 samt var partiledare för Socialdemokraterna
> 'Göran Persson was prime minister between 1996 and 2006 and was leader of the Social Democrats',

we identify *Göran Persson* and *Socialdemokraterna* as proper nouns. We then identify the entities matching these strings. The Swedish Wikipedia lists four *Göran Persson* with four different wikidata Q-numbers:

1. Göran Persson (född 1949), socialdemokratisk partiledare och svensk statsminister 1996–2006 (The Swedish Prime Minister): Q53747;
2. Göran Persson (född 1960), socialdemokratisk politiker från Skåne (A Swedish Member of Parliament): Q5626648;
3. Göran Persson (musiker), svensk proggmusiker (A Swedish composer): Q6042900;
4. Jöran Persson, svensk ämbetsman på 1500-talet (A Swedish state councillor from the 16th century, whose first name can be spelled either Jöran or Göran): Q2625684.

We finally rank these candidates using their popularity and context (Klang and Nugues, 2014). Figure 1 shows the output produced by the disambiguation module.

## 3. Entity Linking and Question Answering

We integrated our named entity linker in a baseline question answering system. Following IBM Watson (Chu-Carroll et al., 2012), we used the Swedish version of Wikipedia as textual knowledge source. We processed the whole corpus with the linker so that we associated all the entities we could recognize to their Q-number. For the named entities

[1] `http://www.wikidata.org/`

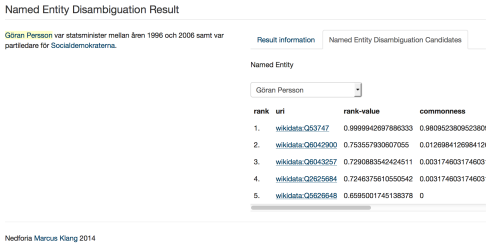



Figure 1: The output of the entity disambiguation module

we identified with the POS tagger that had no Q-number, we used the Wikipedia page name as identifier instead, e.g. "Göran_Persson" for the Prime Minister. We segmented the articles into paragraphs and we indexed them using Lucene (Apache Software Foundation, 2014).

We used a corpus of questions and answers transcribed from the SVT *Kvitt eller Dubbelt – Tiotusenkronorsfrågan* game (Thorsvad and Thorsvad, 2005). Given a question, we retrieve the set of Wikipedia passages having the highest similarity using Lucene's $TF.IDF$ implementation.

### 3.1 Merging the Candidates.

We applied a POS tagger to the passages and we extracted the common nouns, proper nouns, and named entities. We merged all the strings that could be linked to a unique identifier and we created a list of synonyms with the resulting set. When the strings have no identifier, we merge them either by lemma or surface forms. These strings usually consist of a single token: a noun. However, as the POS tagger identifies multiword named entities, a candidate may consist of multiple tokens. In such a case, it is merged in the same way as single tokens.

### 3.2 Ranking the Candidates.

Ranking the candidate answers to a question is done by frequency, i.e. the number of candidate occurrences after merging. Figure 2 shows the output of the system to the question:

> Vem vann melodifestivalen 2004?
> 'who won the Swedish Melody Festival in 2004?'.

## 4. Results and Evaluation

We evaluated the question answering system using four metrics: the median rank, mean rank, number of answered questions (recall), and the mean reciprocal rank (MRR), where MRR $= \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}$, $Q$ is the set of answered questions, and $\text{rank}_i$ is the rank of question $i$. The rank metrics only consider answered questions and we set the limit of retrieved paragraphs to 100 for all the systems.

We considered a question answered if the correct answer could be found in the list of candidate answers. As comparison criterion, we used a lowercase exact string match between the corpus answer and the answers provided by the system in the form of lemma and surface form.

We tested three system configurations:

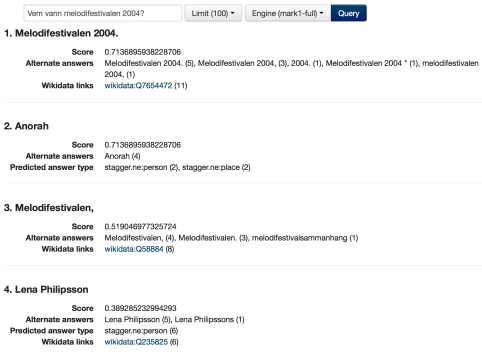



Figure 2: The output of the question answering system

| System | MRR | Median | Mean | Recall |
|---|---|---|---|---|
| Nouns only | 0.138 | 29 | 109.0 | 0.52 |
| Disambig. only | 0.381 | 4.5 | 22.8 | 0.25 |
| Full | 0.172 | 27.0 | 121.2 | 0.58 |

Table 1: The results

**Noun only:** Only nouns and proper nouns, always single tokens;

**Disambiguated only:** Named entities that were successfully connected to a wikidata or Wikipedia entity; and

**Full:** Nouns, named entity candidates, disambiguations, paragraph titles, and numerals.

Table 1 shows the results we obtained. The disambiguation step provide a much better precision but a poorer recall. The opposite applies for nouns, where the recall is higher but precision is lower. The full system with disambiguation increases MRR by nearly 25% compared to only nouns, and the recall by 5.6%.

## 5. Conclusion

A candidate merging step using a named entity linking module produces high precision results, although due to the entity coverage of Wikidata, it misses a significant part of the answers. A baseline merging method has a much lower precision, but a better recall. When combining both methods, we can observe an increase in both precision and recall over the baseline. More importantly, named entity disambiguation provides entry points to structured data, which would allow questions to be answered using a deeper analysis such as inferencing over structured data.

## Acknowledgements

This research was supported by Vetenskapsrådet under grant 621-2010-4800 and the *Det digitaliserade samhället* program.

## References


Apache Software Foundation. 2014. Apache Lucene Core. `http://lucene.apache.org/core/`.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, Vancouver, Canada.

Jennifer Chu-Carroll, James Fan, Nico Schlaefer, and Wlodek Zadrozny. 2012. Textual resource acquisition and engineering. *IBM Journal of Research and Development*, 56(3-4):4:1–4:11.

Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Hinrich Schütze, and Stefan Rüd. 2014. The SMAPH system for query entity recognition and disambiguation. In *Proceedings of Entity Recognition and Disambiguation, ERD'14*, Gold Coast.

David Angelo Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1:1 –1:15, May-June.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh.

Marcus Klang and Pierre Nugues. 2014. A platform for named entity disambiguation. In *Proceedings of the workshop on semantic technologies for research in the humanities and social sciences (STRiX)*, Gothenburg. To appear.

Robert Östling. 2013. Stagger: an open-source part of speech tagger for Swedish. *Northern European Journal of Language Technology*, 3:1–18.

Karin Thorsvad and Hasse Thorsvad. 2005. *Kvitt eller Dubbelt*. SVT Tactic, Stockholm.

# WIKIPARQ: A Tabulated Wikipedia Resource Using the Parquet Format

# WikiParq: A Tabulated Wikipedia Resource Using the Parquet Format

**Marcus Klang, Pierre Nugues**
Lund University, Department of Computer science,
Lund, Sweden
marcus.klang@cs.lth.se, pierre.nugues@cs.lth.se

**Abstract**

Wikipedia has become one of the most popular resources in natural language processing and it is used in quantities of applications. However, Wikipedia requires a substantial pre-processing step before it can be used. For instance, its set of nonstandardized annotations, referred to as the wiki markup, is language-dependent and needs specific parsers from language to language, for English, French, Italian, etc. In addition, the intricacies of the different Wikipedia resources: main article text, categories, wikidata, infoboxes, scattered into the article document or in different files make it difficult to have global view of this outstanding resource. In this paper, we describe WikiParq, a unified format based on the Parquet standard to tabulate and package the Wikipedia corpora. In combination with Spark, a map-reduce computing framework, and the SQL query language, WikiParq makes it much easier to write database queries to extract specific information or subcorpora from Wikipedia, such as all the first paragraphs of the articles in French, or all the articles on persons in Spanish, or all the articles on persons that have versions in French, English, and Spanish. WikiParq is available in six language versions and is potentially extendible to all the languages of Wikipedia. The WikiParq files are downloadable as tarball archives from this location: http://semantica.cs.lth.se/wikiparq/.

**Keywords:** Wikipedia, Parquet, query language.

## 1. Introduction

### 1.1. Wikipedia

Wikipedia is a collaborative, multilingual encyclopedia with more than 35 million articles across nearly 300 languages. Anyone can edit it and its modification rate is of about 10 million edits per month[1]; many of its articles are categorized; and its cross-references to other articles (links) are extremely useful to help name disambiguation.

Wikipedia is freely available in the form of dump archives[2] and its wealth of information has made it a major resource in natural language processing applications that range from word counting to question answering (Ferrucci, 2012).

### 1.2. Wikipedia Annotation

Wikipedia uses the so-called wiki markup to annotate the documents. Parsing this markup is then a compulsory step to any further processing. As for the Wikipedia articles, parts of this markup are language-dependent and can be created and changed by anyone. For example, the {{Citation needed}} template in the English Wikipedia is rendered by {{Citation nécessaire}} in French and {{citazione necessaria}} in Italian. Many articles in French use a date template in the form of {{Year|Month|Day}}, which is not used in other Wikipedias.

Moreover, the wiki markup is difficult to master for the millions of Wikipedia editors and, as a consequence, the articles contain scores of malformed expressions. While it is relatively easy to create a quick-and-dirty parser, an accurate tool, functional across all the language versions is a major challenge.

In this paper, we describe WikiParq, a set of Wikipedia archives with an easy tabular access. To create WikiParq, we reformatted Wikipedia dumps from their HTML rendering and converted them in the Parquet format. In addition to the Wikipedia original content, WikiParq makes it easy to add any number of linguistic layers such as the parts of speech of the words or dependency relations.

The WikiParq files are available for download as tarball archives in six languages: English, French, Spanish, German, Russian, and Swedish, as well as the Wikidata content relevant to these languages, from this location: http://semantica.cs.lth.se/wikiparq/.

## 2. Related Work

There are many tools to parse and/or package Wikipedia. The most notable ones include WikiExtractor (Attardi and Fuschetto, 2015), Sweble (Dohrn and Riehle, 2013), and XOWA (Gnosygnu, 2015). In addition, the Wikimedia foundation also provides HTML dumps in an efficient compression format called ZIM (Wikimedia CH, 2015).

WikiExtractor is designed to extract the text content, or other kinds information from the Wikipedia articles, while Sweble is a real parser that produces abstract syntactic trees out of the articles. However, both WikiExtractor and Sweble are either limited or complex as users must adapt the output to the type of information they want to extract. In addition, they do not support all the features of MediaWiki. A major challenge for such parsers is the template expansion. Dealing with these templates is a nontrivial issue as they can, via the Scribunto extension[3], embed scripting languages such as Lua.

XOWA and ZIM dumps are, or can be converted into, HTML documents, where one can subsequently use HTML parsers to extract information. The category, for instance, is relatively easy to extract using HTML CSS class information. However, neither XOWA nor ZIM dumps make the extraction of specific information from Wikipedia as easy as database querying. In addition, they cannot be easily extended with external information.

[1] https://stats.wikimedia.org/
[2] http://dumps.wikimedia.org/
[3] https://www.mediawiki.org/wiki/Extension:Scribunto

## 3. The Wikipedia Annotation Pipeline

The Wikipedia annotation pipeline consists of five different steps:

1. The first step converts the Wikipedia dumps into HTML documents that we subsequently parse into DOM abstract syntactic trees using jsoup.

2. The second step consists of a HTML parser that traverses the DOM trees and outputs a flattened unformatted text with multiple layers of structural information such as anchors, typeface properties, paragraphs, etc. At the end of this step, we have the text and easily parseable structural information.

3. The third step consists of a linking stage that associates the documents and anchors with unique identifiers, either Wikidata identifiers (Q numbers) or, as a fallback, Wikipedia page titles.

4. The fourth step annotates the resulting text with grammatical layers that are entirely parametrizable and that can range from tokenization to semantic role labelling and beyond. The linguistic annotation is provided by external language processing tools. This step is optional.

5. The fifth and final stage links mentions of proper nouns and concepts, possibly ambiguous, to unique entity identifiers. As in the third step, we use the Wikidata nomenclature to identify the entities. This fifth step is also optional.

### 3.1. Wiki Markup and HTML Parsing

The first and second steps of the annotation pipeline parse and convert the dumps into an intermediate document model. As input data, the markup parser uses either HTML documents from ZIM archives or XOWA outputs. It then uses the jsoup HTML parser[4] to build the document object model (DOM) of every page, where we extract the text, sections, paragraphs, infoboxes, anchors (the wiki links), tables, and lists.

The parser recursively traverses the HTML DOM and uses heuristic hints based on the CSS classes and HTML tags such as `<table>`, `<p>`, `<ol>`, and `<ul>` to carry out the information extraction. It outputs the flattened text and nine independent sequences of ranges, where the sequences describe respectively the tokens, sections, paragraphs, list items, list sections, anchors, headings, italic and bold characters. This structured data is an intermediate format that we call the Multilayer Document Model (MLDM), which is similar to a property graph model. Figure 1 shows the conversion pipeline from the Wikimedia dumps to the abstract syntactic trees (AST) and MLDM layers.

### 3.2. Anchor Resolution

The third step links the documents and anchors to their entity *id*, a unique entity identifier across Wikipedia language editions available from Wikidata.

[4] `http://jsoup.org/`

Prior to the linking step, we collected a set of entities from Wikidata, a freely available graph database that connects the Wikipedia pages across languages. Each Wikidata entity has a unique identifier, a Q-number, that is shared by all the Wikipedia language versions on this entity.

The Wikipedia pages on *Beijing* in English, *Pékin* in French, *Pequim* in Portuguese, and 北京 in Chinese, are all linked to the Q956 Wikidata number, for instance, as well as 190 others languages. In total, Wikidata covers a set of more than 16 million items that defines the search space of entity linking. In the few cases where a Wikipedia page has no Q-number, we replace it by the page name. In addition to the Q-numbers, Wikidata structures its content in the form of a graph, where each node is assigned a set of properties and values. For instance, Beijing (Q956) is an instance of a city (Q515) and has a coordinate location of 39°54'18"N, 116°23'29"E (P625).

We implemented the anchor resolver through a lookup dictionary that uses all the Wikipedia page titles, redirects, and Wikidata identifiers. The page labels, *i.e.* the page titles and all their synonyms, form the entries of this dictionary, while the Wikidata identifiers are the outputs. In case a page label has no Wikidata number, the dictionary uses its normalized page title.

### 3.3. Grammatical Annotation

Once we have extracted and structured the text, we can apply a grammatical annotation that is language specific. Depending on the language, and the components or resources available for it, the annotation can range from a simple tokenization to semantic-role labels or coreference chains.

**Multilinguality.** We implemented the grammatical annotation so that it has a multilingual support at the core. All the language-dependent algorithms are stored in separate packages and tied to the system through abstractions that operate on the Multilayer Document Model. More specifically, the token annotations include a dictionary loosely inspired by the CoNLL format (Buchholz and Marsi, 2006) extended with naming conventions from Exner and Nugues (2014).

**Dependency Injection.** The annotators apply the *dependency injection* (Fowler, 2004) pattern that solves the problem of hard-coded dependencies by making the code only depend on a dependency injector. The dependency injector is constructed once and reused multiple times. The injector can be configured to provide different concrete implementations which allow a high-level way of switching the implementation of an abstraction. The role of the injector is to provide instances of requested abstractions as well as concrete classes. The injector also injects the dependencies needed to construct these instances. We used Guice (Google, 2011) as base library on top of which we developed thread-safe constructions to be able to process indices and storage.

**Tool chains.** The tool chains are instances of annotators and are specific to the languages we process. For English, Spanish, and German, we use CoreNLP (Manning et al., 2014). For French, we use CoreNLP for tokenizing the text and MATE for parsing (Björkelund et al., 2010). For Swedish, we use Stagger (Östling, 2013) and MaltParser

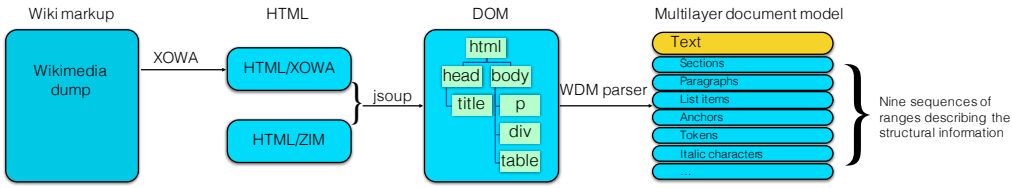



Figure 1: Conversion of Wikipedia dumps into abstract syntactic trees and the Multilayer Document Model (MLDM)

(Nivre et al., 2006). For Russian, there is no linguistic annotation and the tool chain is reduced to nothing as of today.

### 3.4. Entity Linking

The final step links mentions of named entities and concepts to unique Wikidata identifiers. This last operation is an optional step. While Wikidata defines unambiguous mappings between a Q-number and language-dependent strings, strings may have ambiguous Q-numbers. For example, the mention *Göran Persson* in the Swedish Wikipedia refers to at least four different entities with three different Q-numbers: A former Swedish prime minister (Q53747), a progressive musician (Q6042900), a politician (Q5626648), and a Swedish statesman from the 16th century (Q2625684). The latter is also being spelled *Jöran Persson*.

To carry out the mention disambiguation, we reimplemented a variant of TagMe (Ferragina and Scaiella, 2010). TagMe requires minimal grammatical information as it only needs a dictionary of mention-entity pairs and of incoming links. We can then apply it to any of our language versions.

### 3.5. Visualizing the Annotations

We created a tool to visualize the annotations of the Multilayer Document Model. The layers are selectable from a dropdown menu and their visualization uses brat components[5]. Figures 2, 3, and 4 show screenshots with different layers. They all refer to the Wikipedia article *Carl von Linné* in English.

Figure 2 shows the first paragraph of the article with the token and named entity layers, while Fig. 3 shows the entity links with those manually marked as anchors in the text superimposed to the automatically assigned entity links. The numbers refer to the Wikidata nomenclature. Finally, Fig. 4 shows the dependency relations. This layer is optional and depends on the availability of a parser for the language version. This visualization tool is available at: `http://vilde.cs.lth.se:8080/`.

## 4. Parquet: A Storage and Extraction Format

Parquet[6] is a popular column-oriented storage format, i.e. a columnar format, where instead of storing a file by row, the data is structured by column. Table 1 shows a simple example of data tabulated in two columns, where the first one consists of words and the second one, of their parts of speech, and Table 2 shows how the first table is stored using a row-oriented format and a columnar one, where the columns are stored sequentially.

| Word | POS |
|---|---|
| The | dt |
| boy | nn |
| fell | vb |

Table 1: An example of tabulated data.

| Row-oriented | Column-oriented | |
|---|---|---|
| The | | The |
| dt | First column | boy |
| boy | | fell |
| nn | | dt |
| fell | Second column | nn |
| vb | | vb |

Table 2: Storage organization in row- and column-oriented formats.

In files with large numbers of columns, the Parquet format enables a program to read the columns as needed and skip the others. In addition to providing a faster access to the selected columns, such a format is also very efficient for compressing redundant data; something extremely useful in our case.

## 5. The WikiParq Format

We created a program to tabulate and store all the annotation layers we described in Sect. 3. using the Parquet format.

### 5.1. The WikiParq Columns

In its current version, the WikiParq format consists of ten main columns. Some columns borrow concepts from graph database structures in the form of source nodes, target nodes (values), and predicates (or properties) between these nodes:

**uri:** The wikidata identifier of the document, for instance `urn:wikidata:Q34`;

**lang:** The language of the document, for instance `de` for German, `fr` for French;

**doc:** The part of the document. The values can be `article` (the text), `category`, or `disambiguation`;

[5] `http://brat.nlplab.org/`
[6] `https://parquet.apache.org/`

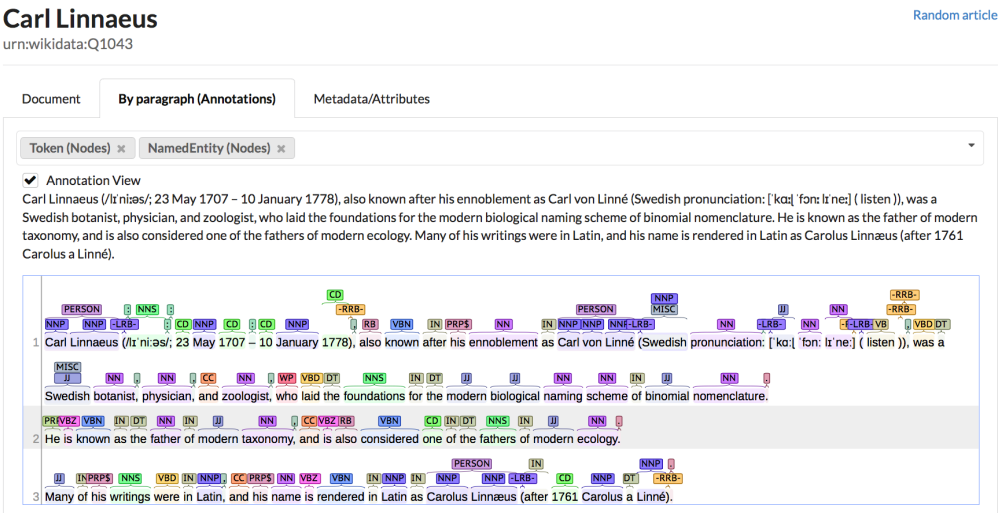



Figure 2: Visualizing tokens and named entities

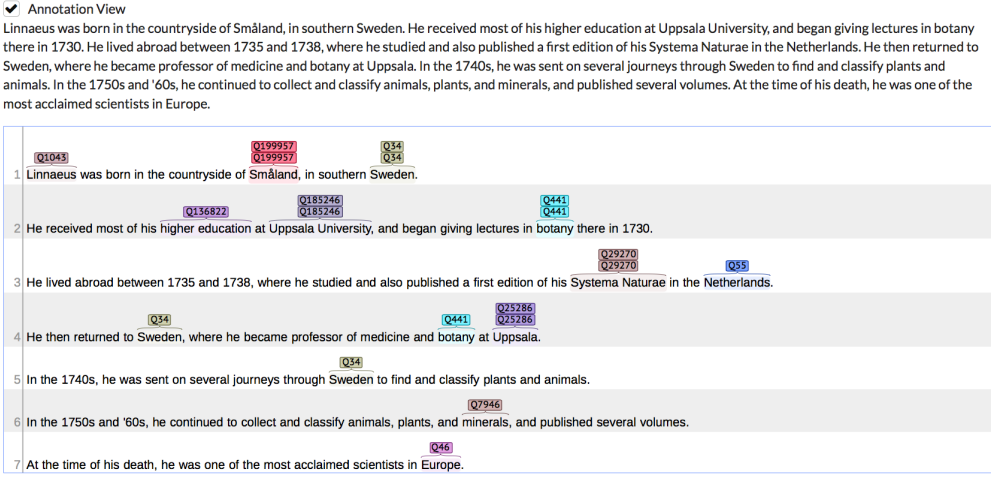



Figure 3: Visualizing the links: Anchors and disambiguated mentions of entities and concepts

**source:** The layer we are annotating. There are two main types of layers consisting of either nodes or edges. The token layer contains nodes: `node/token`. It is equivalent to a node in a graph, while the dependency layer uses edges;

**sourceId:** The identifier of an element in the layer, for instance 1 for the first token in a layer;

**predicate:** The relation annotating the node, for instance the part of speech: `token:pos`. We use similar properties to annotate a token with its lemma, `token:lemma`, a head in a dependency graph, `token:head`, or a resolved Wikipedia link, `link:resolved_target`;

**value[1-9]:** The values of the target node, normally one: `value1`. For instance, for a token and a part-of-speech property, the value can be a common noun: `NN`. Some relations, such as `link:resolved_target`, have more than one value. In this case, we have `value1` for the target, `value2` for the text, etc.

**type:** The format of the value, which can either be `string`, `range`, or `reference`, `doclink`, `weblink`, `wikilink` (anchor),

**valuei1:** Start of the range of the node, if this is relevant, *i.e.* the type is `range` or `reference`. An example of reference that refers to the first token of a document is:

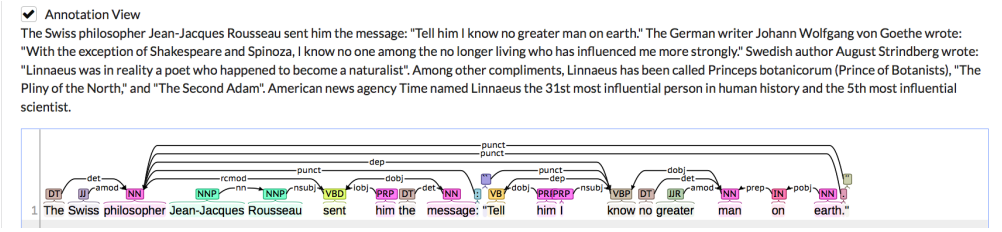



Figure 4: Visualizing the grammatical dependencies

`valuei1 = 1` and `value1 = node/token`;

**valuei2:** End of the range of the node, if this is relevant (`range`).

Table 3 shows examples of Wikiparq annotations for a whole document, a token, its range and part of speech, as well as an anchor to the *Finland* entity.

### 5.2. The WikiParq Predicates

We used a set of predicates to describe the relations linking a node to its values. This allows us to represent data in the form of triples consisting of a node, a predicate, and a value. For instance, the token in Table 3 has a text, a range, and a part of speech that conceptually correspond to:

```
wd:Q34/sv/article/node/3614, token:text, "officiellt"
wd:Q34/sv/article/node/3614, range:token, "18-28"
wd:Q34/sv/article/node/3614, token:cpostag, "ADJ"
```

This representation is very flexible. For instance, we designate and annotate token sequences or sentences through the creation of nodes with ranges corresponding to the spans of the groups or the sentences. We can also easily merge multiple editions of Wikipedia by just considering the lang prefix. Table 4 shows the list of all the predicates we are using so far.

We converted the Wikipedia versions of six languages: English, French, Spanish, German, Russian, and Swedish, from archives in the Multilayer Document Model into WikiParq. We also created a WikiParq version of the Wikidata content relevant to these languages. We used Wikipedia dumps from February 3rd or 4th, 2016 and a Wikidata dump from February 22nd, 2016.

The English WikiParq tar file gives an idea of the resulting data volume: It is 15 Gbytes large, has 4.8 million articles, 49.8 million paragraphs, and 175 million resolved links. For Swedish, the total number of triples is of 7.8 billions for the language-annotated version.

## 6. Querying WikiParq

The Parquet format has been integrated in a number of processing frameworks including map-reduce based databases. It is the default storage format of Spark SQL (Armbrust et al., 2015), a module to query structured data using Spark (Zaharia et al., 2012). Spark is a memory-based implementation of the map-reduce framework (Dean and Ghemawat, 2008) that has become a very popular tool in cluster computing.

Spark SQL makes it very easy to extract information from WikiParq as it follows the familiar SQL syntax and, at the same time, is very fast. In addition, processing can transparently be distributed on a cluster. Spark uses the concept of dataframe, similar to that of R or Python pandas, for loading and saving Parquet files. We can consider dataframes as wrappers to Spark resilient datasets (RDD) with structured schemas.

In the next sections, we provide query examples to show how to extract information from Wikipedia using WikiParq and the Spark (Scala) API.

### 6.1. Loading a File

The API is straightforward and loading a file, here the English Wikipedia (`en_wiki`), only needs two instructions:

```
val en_wiki = sqlContext.read.parquet(filenames)
en_wiki.registerTempTable("enwiki")
```

which result in a table.

### 6.2. Extracting Text

To extract all the articles from a Wikipedia version, here Swedish, we use this simple query:

```
SELECT uri, lang, value1 AS text
FROM svwiki
WHERE predicate = 'document:text'
```

inside a `sqlContext.sql()`.

To count all the nouns in a collection, we use:

```
SELECT COUNT(*)
FROM svwiki
WHERE predicate = 'token:cpostag'
  AND value1 = 'NOUN'
```

which results in about 116 millions for Swedish. The corresponding number for verbs is 34 millions.

In the subsequent examples, we used the Parquet files of six languages: de, en, es, fr, ru, and sv, as well as the Wikidata parquet file.

### 6.3. Counting the Articles per Language

Once the files are loaded, we can extract data or information using SQL queries as for instance for this request:

Extract all the articles on persons in Wikipedia?

| Ann. target | uri | doc | source | sourceId | predicate | value1 | valuei1 | valuei2 | type | lang |
|---|---|---|---|---|---|---|---|---|---|---|
| Document | wd:Q34 | article | null | null | document:text | *Sverige*... | null | null | string | sv |
| Token text | wd:Q34 | article | node/token | 3614 | token:text | officiellt | null | null | string | sv |
| Token range | wd:Q34 | article | node/token | 3614 | range:token | null | 18 | 28 | range | sv |
| Token POS | wd:Q34 | article | node/token | 3614 | token:cpostag | ADJ | null | null | string | sv |
| Link | wd:Q34 | article | node/anchor | 2329 | link:resolved_target | wd:Q33 | null | null | wikilink | sv |

Table 3: Examples of the Wikiparq annotations for a whole document, here the article *Sverige* 'Sweden' in the Swedish Wikipedia, a word in it, here *officiellt*, its range, and part of speech, as well as an anchor (wiki link) to *Finland*. We abridged urn:wikidata in wd

| document | link | category | token | range | edge | ne | paragraph | section |
|---|---|---|---|---|---|---|---|---|
| alt_title | resolved_target | member-of | cpostag | clean | deprel:label | label | source | title |
| title | unresolved_target | title | deprel | heading | head | | | |
| wiki_page_id | | | feats | italic | tail | | | |
| text | | | head | link | | | | |
| category | | | idx | list_item | | | | |
| | | | lemma | list_section | | | | |
| | | | norm | named_entity | | | | |
| | | | pos | paragraph | | | | |
| | | | text | section | | | | |
| | | | | sentence | | | | |
| | | | | strong | | | | |
| | | | | token | | | | |

Table 4: List of available predicates organized by prefix. To have the full name, the prefix is concatenated to the relation name, i.e. document:title or link:resolved_target

To carry this out, we first extract the persons from the Wikidata ontology. We use the *instance of* property (P31) and we keep the entities having the property of being an instance of a human (Q5). This is translated in SQL as:

```
sqlContext.sql("""
   SELECT uri
   FROM wikidata
   WHERE predicate = 'wd:P31'
   AND value1 = 'urn:wikidata:Q5'
""").cache().registerTempTable("persons")
```

and results in a table called `persons`.
We then extract the language versions associated with each of these entities (persons). We run the extraction using this query:

```
sqlContext.sql("""
   SELECT wikidata.uri AS uri, lang
   FROM wikidata
   JOIN persons
   ON persons.uri = wikidata.uri
   WHERE predicate = 'wd:sitelink'
""").registerTempTable("person_sitelinks")
```

that produces a table of entity identifiers (Q-numbers) and languages. The `wd:sitelink` predicate enables us to find the language versions of an entity according to Wikidata.
Finally, we count the persons per language using this query:

```
sqlContext.sql("""
   SELECT lang, COUNT(lang) as count
   FROM person_sitelinks
   GROUP BY lang
   ORDER BY lang
""").show()
```

This results in a table with the number of persons per language version:

```
+----+-------+
|lang|  count|
+----+-------+
|  de| 597515|
|  en|1339313|
|  es| 274878|
|  fr| 462839|
|  ru| 313566|
|  sv| 183926|
+----+-------+
```

### 6.4. Counting the Language Versions of an Article

Going on with this dataset, a second question we may pose is:

> For a given article, how many language versions are there?

which is translated in SQL as:

```
sqlContext.sql("""
   SELECT uri, COUNT(lang) AS numLangs
   FROM person_sitelinks
```

```
   GROUP BY uri
""").registerTempTable("lang_person")
```

The excerpt below shows a subset of the first answers to this question:

```
+--------------------+--------+
|                 uri|numLangs|
+--------------------+--------+
|urn:wikidata:Q100250|       2|
|urn:wikidata:Q100412|       3|
|urn:wikidata:Q100863|       1|
|urn:wikidata:Q101097|       1|
|urn:wikidata:Q101141|       2|
|urn:wikidata:Q101259|       1|
|urn:wikidata:Q101303|       2|
|urn:wikidata:Q101754|       2|
|urn:wikidata:Q101916|       2|
|urn:wikidata:Q102032|       2|
|urn:wikidata:Q102483|       6|
|urn:wikidata:Q102645|       3|
|urn:wikidata:Q10287 |       6|
+--------------------+--------+
```

The Q-number is the Wikidata identifier of a Wikipedia entity. For instance, Q101916 is the identifier of Friedrich von Weech, a German regional historian and archivist, who has German and Swedish versions.

### 6.5. Counting Articles with a Constraint on the Language Versions

At this point, we may wonder:

> How many articles are available in the six versions?

This question is rendered by the following SQL query:

```
sqlContext.sql("""
  SELECT COUNT(uri)
    FROM lang_person
    WHERE numLangs = 6""")
```

and the answer is: 41,509 articles.

### 6.6. Extracting the Text of the Articles

So far, we carried out extractions that could have also been done from Wikidata using the SPARQL language. WikiParq merges Wikidata and Wikipedia and extends the query possibilities to span both resources seamlessly as for instance with this request whose goal could be to build loosely parallel corpora in six languages:

> Extract the text of all the articles with six language versions

Such a request is translated by these two following SQL queries, where the first one selects the articles with six language versions:

```
sqlContext.sql("""
    SELECT * FROM lang_person
    WHERE numLangs = 6
""").cache().registerTempTable("lang6_person")
```

and the second one outputs the text, here in Swedish:

```
sqlContext.sql("""
    SELECT lang6_person.uri, value1 AS text
    FROM svwiki
    JOIN lang6_person
    ON svwiki.uri = lang6_person.uri
    WHERE svwiki.predicate = 'document:text'
""").show()
```

### 6.7. Extracting all the Mentions of an Entity

A last example shows how to build dictionaries of the words or phrases used in Wikipedia to name an entity: A dictionary of mentions. To carry this out, we need to extract all the labels of a entity in Wikipedia. This operation is easy to carry out, for instance for Barack Obama in the English Wikipedia. Barack Obama has Q76 as Wikidata identifier. This leads to this query:

```
sqlContext.sql("""
   SELECT value1 AS target, value2 AS mention,
   COUNT(*) AS freq
   FROM enwiki
    WHERE enwiki.predicate =
      'link:resolved_target'
   AND value1 = 'urn:wikidata:Q76'
   GROUP BY value1, value2
   ORDER BY value1, freq DESC
""").show()
```

that results in a table, where we show the first lines below:

```
+----------------+--------------------+-----+
|          target|             mention| freq|
+----------------+--------------------+-----+
|urn:wikidata:Q76|        Barack Obama|14960|
|urn:wikidata:Q76|               Obama|  887|
|urn:wikidata:Q76|     President Obama|  546|
|urn:wikidata:Q76|President Barack ...|  142|
|urn:wikidata:Q76|     Barack H. Obama|   64|
|urn:wikidata:Q76|       Obama, Barack|   45|
|urn:wikidata:Q76|      Barack Obama's|   40|
|urn:wikidata:Q76|   President Obama's|   21|
|urn:wikidata:Q76|              Barack|   21|
|urn:wikidata:Q76|President Barack ...|   13|
|urn:wikidata:Q76|Obama administration|    9|
|urn:wikidata:Q76|             Obama's|    8|
|urn:wikidata:Q76|President-elect O...|    8|
|urn:wikidata:Q76|           President|    7|
|urn:wikidata:Q76|   Sen. Barack Obama|    7|
|urn:wikidata:Q76|Barack Hussein Obama|    7|
|urn:wikidata:Q76|Barack Hussein Ob...|    6|
|urn:wikidata:Q76|U.S. President Ba...|    6|
|urn:wikidata:Q76|Senator Barack Obama|    6|
|urn:wikidata:Q76|     Barack Obama' s|    5|
+----------------+--------------------+-----+
```

## 7. Conclusion

We have described WikiParq, a unified tabulated format that uses the Parquet standard to package the Wikipedia corpora. In combination with Spark, a map-reduce computing

framework, and the SQL query language, this format makes it easy to write concise database queries to extract specific information from Wikipedia and have the answer in a few minutes.

Currently, six versions of Wikipedia are available as tarball archives in the WikiParq format from this location: `http://semantica.cs.lth.se/wikiparq/`. We also provide a Parquet version of Wikidata, as well as a Scala program and a Jupyter notebook to run the examples described in this paper. We ran and tested all the examples on a laptop with an Intel i7 processor and 16 Gbytes of memory.

## Acknowledgments

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

## 8. Bibliographical References

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., and Zaharia, M. (2015). Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1383–1394.

Attardi, G. and Fuschetto, A. (2015). Wikiextractor. https://github.com/attardi/wikiextractor/.

Björkelund, A., Bohnet, B., Hafdell, L., and Nugues, P. (2010). A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstration Volume*, pages 33–36, Beijing, August 23-27. Coling 2010 Organizing Committee.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Dohrn, H. and Riehle, D. (2013). Design and implementation of wiki content transformations and refactorings. In *Proceedings of the 9th International Symposium on Open Collaboration*, WikiSym '13, pages 2:1–2:10.

Exner, P. and Nugues, P. (2014). KOSHIK: A large-scale distributed computing framework for NLP. In *Proceedings of ICPRAM 2014 – The 3rd International Conference on Pattern Recognition Applications and Methods*, pages 464–470, Angers, March 6-8.

Ferragina, P. and Scaiella, U. (2010). Fast and accurate annotation of short texts with wikipedia pages. In *Proceedings of CIKM'10*, Toronto.

Ferrucci, D. A. (2012). Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1:1–1:15, May-June.

Fowler, M. (2004). Inversion of control containers and the dependency injection pattern. Last accessed: 2013-12-20.

Gnosygnu. (2015). Xowa. https://github.com/gnosygnu/xowa.

Google. (2011). Guice (version 3.0). Last accessed: 2013-11-10.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Nivre, J., Hall, J., and Nilsson, J. (2006). MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219.

Östling, R. (2013). Stagger: an open-source part of speech tagger for Swedish. *Northern European Journal of Language Technology*, 3:1–18.

Wikimedia CH. (2015). Openzim. http://www.openzim.org.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*.

# Docforia: A Multilayer Document Model

PAPER III

# Docforia: A Multilayer Document Model

**Marcus Klang**
Department of computer science
Lund University, Lund
marcus.klang@cs.lth.se

**Pierre Nugues**
Department of computer science
Lund University, Lund
pierre.nugues@cs.lth.se

## Abstract

In this paper, we describe **Docforia**, a multilayer document model and application programming interface (API) to store formatting, lexical, syntactic, and semantic annotations on Wikipedia and other kinds of text and visualize them. While Wikipedia has become a major NLP resource, its scale and heterogeneity makes it relatively difficult to do experimentations on the whole corpus. These experimentations are rendered even more complex as, to the best of our knowledge, there is no available tool to visualize easily the results of a processing pipeline.

We designed Docforia so that it can store millions of documents and billions of tokens, annotated using different processing tools, that themselves use multiple formats, and compatible with cluster computing frameworks such as Hadoop or Spark. The annotation output, either partial or complete, can then be shared more easily. To validate Docforia, we processed six language versions of Wikipedia: English, French, German, Spanish, Russian, and Swedish, up to semantic role labeling, depending on the NLP tools available for a given language. We stored the results in our document model and we created a visualization tool to inspect the annotation results.

## 1 Introduction

Wikipedia is one of the largest freely available encyclopedic sources: It is comprehensive, multilingual, and continuously expanding. These unique properties make it a popular resource now used in scores of NLP projects such as translation (Smith et al., 2010), semantic networks (Navigli and Ponzetto, 2010), named entity linking (Mihalcea and Csomai, 2007), information extraction, or question answering (Ferrucci, 2012).

Nonetheless, the Wikipedia size, where many language versions have now more that one million of articles makes it more difficult to handle than "classic" and older corpora such as the Penn treebank (Marcus et al., 1993). Processing the complete collection of Wikipedia articles, or a part of it, is a nontrivial task that requires dealing with multiple markup variants across the language versions, multiple tools and storage models. In addition, the application of a NLP pipeline to carry out the annotation (tokenization, POS tagging, dependency parsing, and so on) is a relatively costly operation that can take weeks on a single computer.

Docforia is a multilayer document model to store formatting, lexical, syntactic, and semantic annotations on Wikipedia and other kinds of text and visualize them. To deliver results in a reasonable time, Docforia is compatible with cluster programming frameworks such as Spark or Hadoop. Using the Langforia language processing pipelines (Klang and Nugues, 2016a), we processed six language versions of Wikipedia: English, French, German, Spanish, Russian, and Swedish, up to semantic role labeling, depending on the NLP tools available for a given language. We stored the results in the document model. We designed an interactive visualization tool, part of Langforia, so that a user can select languages, documents, and linguistic layers and examine the annotation output.

## 2 The Document Model

We created the Docforia multilayer document model library to store, query, and extract hypertextual information common to many NLP tasks such as part-of-speech tagging, coreference resolution, named entity recognition and linking, dependency parsing, semantic role labeling, etc., in



*Proceedings of the 21st Nordic Conference of Computational Linguistics*, pages 226–230,
Gothenburg, Sweden, 23-24 May 2017. 

a standalone package.

This model is intended for large and heterogenous collection of text, including Wikipedia. We designed it so that we could store the original markup, as well as the the results of the subsequent linguistic processing. The model consists of multiple layers, where each layer is dedicated to a specific type of annotation. It is nondestructive and preserves the original white spaces.

The annotations are encoded in the form of graph nodes, where a node represents a piece of data: A token, a sentence, a named entity, etc., delimited by ranges. These nodes are possibly connected by edges as in dependency graphs. The data structure used is similar to a property graph and Fig. 1 shows the conversion pipeline from the Wikimedia dumps to the abstract syntactic trees (AST) and Docforia layers.

## 3 Previous Work

A few graph-based linguistic data models and serializations have structures that are similar to Docforia. They include HyGraphDB (Gleim et al., 2007), the Linguistic Framework Annotation (Ide and Suderman, 2014), Off-Road LAF (Lapponi et al., 2014), the D-SPIN Text Corpus Format (Heid et al., 2010), and the oft cited UIMA project (Ferrucci and Lally, 2004). Some tools also extend UIMA such as DKPro Core (Eckart de Castilho and Gurevych, 2014).

In contrast to the UIMA project (Ferrucci and Lally, 2004), which also provides an infrastructure to represent unstructured documents, the Docforia library by itself does not define an equivalent analysis infrastructure or rich type system. Docforia's main focus is data extraction and storage of informal heterogenous data, where the schema can change many times during a project.

The primary motivation of Docforia was a faster adaptability in research projects, where rigidity can adversely affect productivity. Docforia is semi-structured, contains a simplified static-type system for common types of layers and has support for a dynamic-type system. The static types are defined by convention, can be overridden, and are by no means enforced.

## 4 Use-case: Wikipedia

We convert Wikipedia from HTML dumps into Docforia records using an annotation pipeline. The first step converts the HTML documents into DOM trees using jsoup[1]. The second step extracts the original page structure, text styles, links, lists, and tables. We then resolve the links to unique Wikidata identifiers. These steps are common to all the language editions we process.

Wikidata is central to the multilingual nature of Docforia. Wikidata is an entity database, which assigns unique identifiers across all the language editions of Wikipedia. The University of Gothenburg, for instance, has the unique id: Q371522 that enables to retrieve the article pages in English, French, Swedish, or Russian.

In addition to the common processing steps and depending on the available tools, we can apply linguistic annotations that are language specific using Langforia. These annotations can range from a simple tokenization to semantic-role labels or coreference chains. We save all the results of the intermediate and final steps as files in the Parquet format; each record being a Docforia document as binary blob in addition to metadata such as Wikidata Q-number, title, and page-type. We selected this format because of its portability, efficiency, and ease of use with the Apache Spark data processing engine.

## 5 Application Programming Interface

The Docforia API builds on the concepts of document storage and document engine. The document storage consists of properties, layers (node or edge layers) to store typed annotations, token, sentence, relationship, where the nodes can have a range, and finally sublayer variants: gold, predicted, coreference chains. The document engine defines query primitives such as covers, for instance the tokens in a anchor, transactions, and partial lightweight documents called views.

The Docforia data structure is similar to a typed property graph. It consists of nodes (tokens, sentences, paragraphs, anchors, ...), edges (connections between e.g tokens to form a dependency tree), and properties per node and edge (Token: pos, lemma, ...).

The piece of code below shows how to create tokens from a string and assign a property to a range of tokens, here a named entity with the *Location* label:

```
Document doc = new MemoryDocument(
  "Greetings from Lund, Sweden!");
```

[1] http://jsoup.org/

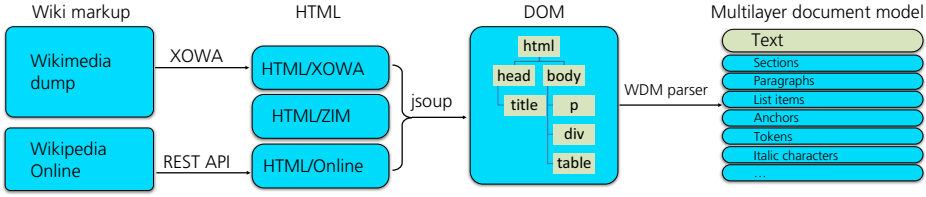



Figure 1: Conversion of Wikipedia dumps into abstract syntactic trees and the Docforia multilayer document model (Klang and Nugues, 2016b).

```
// 01234567890123456789012345678

Token Greetings   =
  new Token(doc).setRange(0,  9);
Token from        =
  new Token(doc).setRange(10, 14);
Token Lund        =
  new Token(doc).setRange(15, 19);
Token comma       =
  new Token(doc).setRange(19, 20);
Token Sweden      =
  new Token(doc).setRange(21, 27);
Token exclamation =
  new Token(doc).setRange(27, 28);

Sentence greetingsSentence =
  new Sentence(doc).setRange(0, 28);

NamedEntity lundSwedenEntity =
  new NamedEntity(doc)
    .setRange(Lund.getStart(),
      Sweden.getEnd())
        .setLabel("Location");
```

The API provides SQL-like query capabilities and the code below shows how to find the named entities in a document:

```
NodeTVar<Token> T = Token.var();
NodeTVar<NamedEntity> NE =
  NamedEntity.var();

List<Token> lundLocation =
  doc.select(T, NE)
    .where(T).coveredBy(NE)
    .stream()
    .sorted(StreamUtils.orderBy(T))
    .map(StreamUtils.toNode(T))
    .collect(Collectors.toList());
```

## 6 Visualization

We built a front-end application, part of Langforia, to enable the users to visualize the content of Docforia-based corpora. This application has the form of a web server that embeds the Docforia library and Lucene to index the documents. We created a Javascript component for the text visualization on the client. This client provides a user interface for searching and visualizing Docforia data in the index. The layers are selectable from a dropdown menu and the supported visualizations are the ranges and relationships between them.

Figure 2 shows the annotations of the parts of speech, named entities, and dependency relations of the sentence:

> Göteborgs universitet är ett svenskt statligt universitet med åtta fakulteter, 37 000 studenter, varav 25 000 helårsstudenter och 6000 anställda.
>
> 'The University of Gothenburg is a Swedish public university with eight faculties, 37,000 students, (25,000 full-time), and 6,000 staff members.'

The visualization tool is similar to the brat[2] components (Stenetorp et al., 2012), but includes a tooltip support and has a faster rendering. If we hover over the words, it shows the properties attached to a word in CoNLL-like format. In Fig. 3, the properties correspond to the word *Vasaparken*.

## 7 Conclusion and Future Work

We described Docforia, a multilayer document model, structured in the form of a graph. It enables a user to represent the results of large-scale multilingual annotations. Using it and the Langforia language processing pipelines, we annotated

[2] http://brat.nlplab.org/

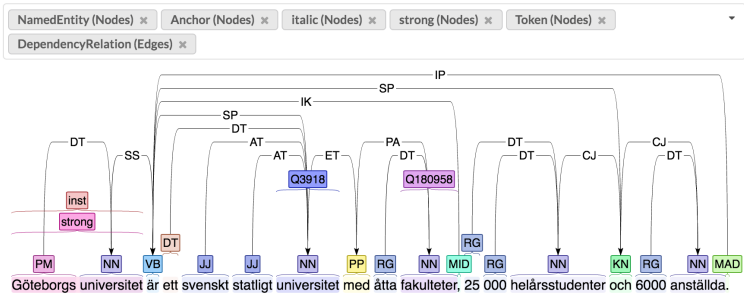



Figure 2: Visualization of six layers including: Tokens, named entities, and dependency relations


Figure 3: Visualization of properties

Wikipedia dump (Klang and Nugues, 2016a). When applied to Wikipedia, MLDM links the different versions through an extensive use of URI indices and Wikidata Q-numbers.

Together with Docforia, we used the Lucene library to index the records. The resulting system can run on a single laptop, even with multiple versions of Wikipedia.

Docforia is written in Java. In the future, we plan to develop a Python API, which will make it possible to combine Python and Java tools. This will enable the programmer to build prototypes more quickly as well as experiment more easily with machine learning algorithms.

Docforia is available from github at `https://github.com/marcusklang/docforia`.

## Acknowledgements

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

## References

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, September.

David Angelo Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1:1 –1:15, May-June.

Rüdiger Gleim, Alexander Mehler, and Hans-Jürgen Eikmeyer. 2007. Representing and maintaining large corpora. In *Proceedings of the Corpus Linguistics 2007 Conference, Birmingham (UK)*.

Ulrich Heid, Helmut Schmid, Kerstin Eckart, and Erhard Hinrichs. 2010. A corpus representation format for linguistic web services: The D-SPIN text

corpus format and its relationship with ISO standards. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Nancy Ide and Keith Suderman. 2014. The Linguistic Annotation Framework: a standard for annotation interchange and merging. *Language Resources and Evaluation*, 48(3):395–418.

Marcus Klang and Pierre Nugues. 2016a. Langforia: Language pipelines for annotating large collections of documents. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 74–78, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Marcus Klang and Pierre Nugues. 2016b. WIKIPARQ: A tabulated Wikipedia resource using the Parquet format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4141–4148, Portorož, Slovenia, may.

Emanuele Lapponi, Erik Velldal, Stephan Oepen, and Rune Lain Knudsen. 2014. Off-Road LAF: Encoding and processing annotations in NLP workflows. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Mitchell Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on CIKM*, CIKM '07, pages 233–242, Lisbon, Portugal.

Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the ACL*, pages 216–225, Uppsala.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 403–411.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April. Association for Computational Linguistics.

# Multilingual Supervision of Semantic Annotation

PAPER IV

# Multilingual Supervision of Semantic Annotation

**Peter Exner** **Marcus Klang** **Pierre Nugues**

Lund University
Department of Computer Science
Lund, Sweden
{Peter.Exner, Marcus.Klang, Pierre.Nugues}@cs.lth.se

## Abstract

In this paper, we investigate the annotation projection of semantic units in a practical setting. Previous approaches have focused on using parallel corpora for semantic transfer. We evaluate an alternative approach using loosely parallel corpora that does not require the corpora to be exact translations of each other. We developed a method that transfers semantic annotations from one language to another using sentences aligned by entities, and we extended it to include alignments by entity-like linguistic units. We conducted our experiments on a large scale using the English, Swedish, and French language editions of Wikipedia. Our results show that the annotation projection using entities in combination with loosely parallel corpora provides a viable approach to extending previous attempts. In addition, it allows the generation of proposition banks upon which semantic parsers can be trained.

## 1 Introduction

Data-driven approaches using natural language processing tackle increasingly complex tasks with ever growing scales and in more varied domains. Semantic role labeling is a type of shallow semantic parsing that is becoming an increasingly important component in information extraction (Christensen et al., 2010), question answering (Shen and Lapata, 2007), and text summarization (Khan et al., 2015).

The development of semantic resources such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005) made the training of models for semantic role labelers using supervised techniques possible. However, as a consequence of the considerable manual efforts needed to build proposition banks, they exist only for a few languages. An alternative approach to using supervision is to transfer knowledge between resources, a form of distant or related supervision. Methods for directly projecting semantic labels from a resource-rich language to a resource-scarce one were introduced in Padó (2007).

In this paper, we describe a method for aligning and projecting semantic annotation in loosely parallel corpora by using entities and entity-like linguistic units. Our goal is to generate multilingual PropBanks for resource-scarce languages. We used multiple language editions of Wikipedia: An English edition annotated up to a semantic level using the PropBank semantic roles, and syntactically annotated editions of Swedish and French Wikipedias. By aligning Wikipedias by entities, we constructed loosely parallel corpora and we used them to generate PropBanks in Swedish and French. We provide an evaluation of the quality of the generated PropBanks, together with an evaluation on two external FrameNets.

## 2 Previous Work

As an alternative to using supervised efforts for relation extraction, distant supervision can be employed to transfer relational knowledge representations from one resource to another. Distant supervision for relation extraction was introduced by Craven and Kumlien (1999) in the context of biomedical information extraction. Mintz et al. (2009) describe a method of using an external knowledge base as an indirect way of annotating text. Hoffmann et al. (2010) introduced the usage of Wikipedia infoboxes in distantly supervised relation extraction.

The concept of transferring linguistic annotation, in the context of part-of-speech tags, across parallel corpora was introduced in Yarowsky et al. (2001). Cross-lingual annotation projection of FrameNet semantics has been described by Padó and Lapata (2009) and Basili et al. (2009). In Van der Plas et al. (2011), the authors describe an automatic method of direct transfer of PropBank semantics requiring no manual effort. Akbik et al. (2015) describe an approach to generate multilingual PropBanks using filtered annotation projection and bootstrap learning in order to handle errors stemming from translation shifts in corpora.

Most previous approaches have used professionally translated parallel corpora, mainly EuroParl (Koehn, 2005) and United Nations Corpora (Rafalovitch and Dale, 2009), to transfer semantic annotation. However, creating these resources requires manual efforts; they are thus limited in size and in the number of languages they cover. In contrast to parallel corpora, loosely parallel corpora describe similar concepts and events, but are not necessarily the result of a focused effort to translate a large corpus.

In Exner et al. (2015), we introduced the concept of using entities as a method for aligning sentences and transferring semantic content in loosely parallel corpora. However, the presented approach has the following limitations: (1) it was evaluated on one language only and (2) the evaluation was performed on the generated PropBank itself.

The contributions of this paper are the following: (1) We extend Exner et al. (2015) by including pronouns and other linguistic units that in a local context exhibit the characteristics of entities. (2) We present and evaluate two methods for aligning sentences by using entities. (3) We demonstrate the effectiveness and generalizability of our approach by projecting semantic annotations to two languages, Swedish and French, and we evaluate it using two external proposition databases, the Swedish SweFN++ (Borin et al., 2010) and French ASFALDA (Candito et al., 2014; Djemaa et al., 2016) that are both semantically-annotated corpora using adaptations of FrameNet frames. (4) We release the source code used in the annotation projection and we provide the generated PropBanks in Swedish and French[1].

## 3 Method

The aim of the method is to generate PropBank-like resources by fully annotating sentences in target languages using semantic content, in whole or partially, from a source language. We start with loosely parallel corpora in two languages: a **source language** (SL) expressing the semantic content that we want to transfer to a **target language** (TL). We then disambiguate and uniquely identify the entities in all the sentences. By using the unique identifier of each entity, we gain the ability to align sentences from two different languages forming sentence pairs $(s_{SL}, s_{TL})$. We annotate the $(s_{SL}, s_{TL})$ pairs, $s_{SL}$ to semantic and syntactic levels and $s_{TL}$ to a syntactic level. From each $(s_{SL}, s_{TL})$ pair, we learn the alignments between predicates ($p_{SL}$) in $s_{SL}$ and verbs ($v_{TL}$) in $s_{TL}$. Finally, using the aligned entities and the predicate-verb alignments in each $(s_{SL}, s_{TL})$ pair, we transfer the semantic annotation in the form of predicate-argument structures. Figure 1 shows an overview of this approach.

### 3.1 Using Loosely Parallel Corpora

A prerequisite to projecting semantic annotation between two sentences is that they share the same semantic structure. To this end, we assumed that entities have a constraining property on the sets of predicate-argument structures they can instantiate. By aligning loosely parallel corpora through entities, pairs of sentences in two different languages that we will extract, although they are not translations of each other, should overall express the same semantic content. Furthermore, we believe that by applying our method on a large scale, the most frequent alignments of entities will elicit valid alignments.

In this context, even partial semantic content from a source sentence, $s_{SL}$, may be useful for annotating a target sentence, $s_{TL}$. As an example, consider the following sentence pair:

$s_{SL}$ It$_{\text{A0}}$ features$_{01}$ Kelsey Grammer$_{\text{A1}}$ in his ninth ...
and is the first time$_{\text{AM-TMP}}$ the Simpsons$_{\text{A0}}$ visit$_{01}$ Italy$_{\text{A1}}$
$s_{TL}$ I avsnittet besöker familjen Simpsons Italien
In the episode visit the family Simpsons Italy

[1] http://semantica.cs.lth.se

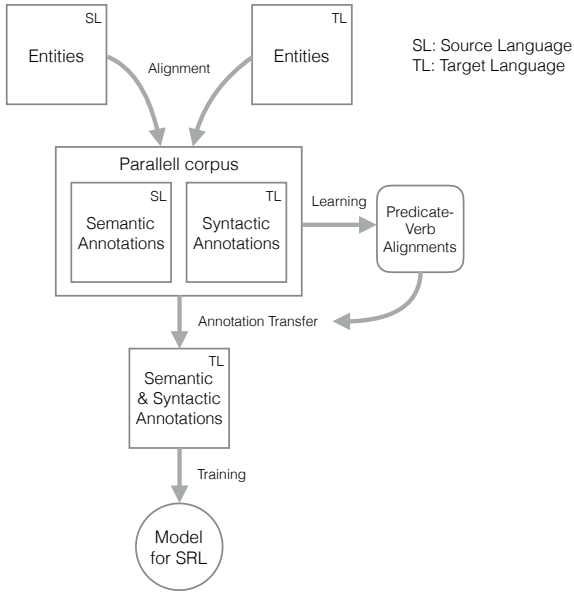



Figure 1: An overview of the approach for transferring semantic annotation from a **source language** (SL) to a **target language** (TL)

in which $s_{SL}$ has been aligned with $s_{TL}$ through the entities $(the\ Simpsons,\ Italy)$. $s_{SL}$ expresses the two predicates $feature.01(it_{A0},\ Kelsey\ Grammer_{A1})$ and $visit.01(the\ first\ time_{AM-TMP},\ the\ Simpsons_{A0},\ Italy_{A1})$. Although $s_{TL}$ is not an exact translation of $s_{SL}$, as it lacks the predicate $features.01$ and the temporal argument $(AM\text{-}TMP)$ of $visit.01$, the partial transfer of the semantic content enables us to annotate $s_{TL}$ with the predicate $besöka.01(familjen\ Simpsons_{A0},\ Italien_{A1})$.

### 3.2 Entity Disambiguation

Entity linking is the process of finding mentions, e.g. persons, cities, organizations, events, concepts, in text, and if available, assign them with a unique identifier provided by a knowledge base. We used Wikidata Q-numbers as identifiers as they provide globally unique identifiers between the different language editions of Wikipedia. As an example, consider the following entities:

*Beijing*, *Pékin*, and *Pequim*

as expressed in English, French, and Portuguese respectively. Although they have differing surface forms, they are all linked to the Q956 Wikidata number, as well as in 190 other languages. In total, Wikidata covers a set of more than 13 million items that defines the entity search space.

To carry out entity linking, we reimplemented a variant of TagME (Ferragina and Scaiella, 2010). The motivating factors behind our reimplementation were:

1. It enabled us to resolve mentions to identifiers in Wikidata, providing us with multilingual and coherent entity identifiers.

2. By using the same entity linker for multiple languages, we obtained a more consistent mention resolution across all the languages.

3. It eased the adaptation to new execution environments, in our case a cluster of computing nodes.

Our implementation of TagME requires minimal grammatical information as it only needs mention statistics derived from anchors and a dictionary of mention-entity pairs and of incoming links.

The entity linking algorithm consists of four steps: detection, candidate voting, selection, and resolution of overlapping mentions.

1. We find all the possible mentions consisting of tokens in sequences up to a maximum length of 6. The mentions found at this stage might be overlapping. We treat overlapped mentions independently and they contribute votes to all the other mentions. As an example, consider the following sentence:

   *Prime Minister of Japan*

   containing the two mentions: *Japan* and *Prime Minister of Japan*. In this case, the overlapped mention *Japan* will contribute a vote to the overlapping mention *Prime Minister of Japan*.

2. We compute the votes for each candidate belonging to a mention. To bound the computation time, we use voting groups consisting of a collection of mentions using a sliding window approach. The vote weight per candidate is the sum of all the inlink relatedness between all the candidates (Ferragina and Scaiella, 2010). In our case, we use all the candidates in a voting group.

3. We rank all the candidates per mention using the computed votes. We then prune the mention list using a coherence criterion and a threshold that we set empirically.

4. In the final step, we resolve the mention overlap using a greedy algorithm. The algorithm selects the overlapping mention, where the entity candidate has the largest global vote, removing all the locally overlapping mentions, until there is no overlap globally.

### 3.3 Syntactic and Semantic Annotation

In our experimental setup, we used the English edition of Wikipedia as our SL, and we annotated it with syntactic and semantic dependencies. For the syntactic-semantic parsing, we used an open-source semantic role labeler (Choi, 2012) trained on OntoNotes 5.0 (Weischedel et al., 2013).

We transferred the semantic annotation to two TLs, the Swedish and French editions of Wikipedia, both annotated with syntactic dependencies. For French syntactic parsing, we applied a transition-based dependency parser (Bohnet and Nivre, 2012; Bohnet and Kuhn, 2012) trained on a French Treebank described in Candito et al. (2010). Correspondingly, to preprocess the Swedish edition of Wikipedia, we applied a pipeline consisting of a POS tagger (Östling, 2013) and a syntactic dependency parser (Nivre et al., 2006).

### 3.4 Extension to Entity-like Tokens

Entities have the property of being uniquely identifiable across languages on a global scope. However, an obvious drawback to using entities as a means of aligning sentences and transferring roles, is that roles are not always instantiated by entities. To reclaim these instances, we extended the entity alignment to include entity-like **linguistic units** (LU). We focused on units that have the property of being uniquely identifiable and limited to the scope of a sentence pair. Units correspond to sequences of tokens the entity disambiguator has either failed to classify as an entity or otherwise lack the ability to be uniquely identified in a global context.

Our algorithm detects entity-like LUs as spans of tokens sharing the same surface form in both $s_{SL}$ and $s_{TL}$. In addition, we set the constraint that they occur at most once in each sentence. As a consequence, this removes any misalignment issue since a LU in $s_{SL}$ can be matched to only one LU in $s_{TL}$. This method enables us to include amounts, dates, and noun phrases that the entity disambiguator fails to detect.

Using similar constraints, we also include pronouns in the detection of entity-like LUs. However, rather than using the surface form of pronouns, which would unlikely match across languages, we instead categorize them by case, gender, and number. For English, Swedish, and French, third person singular pronouns have different surface forms based on gender. Therefore, in order to increase precision, we

limit the detection to only include third person pronouns. Although this constraint certainly limits the recall, this should not significantly impact the training procedure as the pronouns in the first and second persons are in very limited numbers in Wikipedia.

### 3.5 Aligning Sentences

The first challenge in transferring semantic annotation between loosely parallel corpora is to align sentences expressing the same semantic content. Our baseline method for aligning sentences extracts all the entities from a sentence and forms entity-sentence pairs, $(e_1...e_n, s)$. By aligning entities in different entity-sentence pairs, we form new triples containing a source sentence, a target sentence, and the subset of entities by which they are aligned $(s_{SL}, s_{TL}, e_1...e_s)$, where $k_{min} \leq s \leq k_{max}$ and $k_{min}$, $k_{max}$ are prior parameters of our choice.

The baseline method is, in its simplicity, independent of any syntactic or lexical markup. It only requires the annotations from an entity disambiguator. However, one drawback lies in the inclusion of entities ungoverned by any predicate. As a consequence, the alignment of partial semantic content, as described in Sect. 3.1, becomes problematic. We therefore extended this baseline algorithm by using sets of entities projected by either arguments in $s_{SL}$ or a verb in $s_{TL}$. Using this projection method, we then form entity-sentence pairs:

> $(e_1...e_p, s)$, where each entity in $(e_1...e_p)$ is governed by an argument belonging to a predicate in $s_{SL}$

and

> $(e_1...e_v, s)$, where each entity in $(e_1...e_v)$ is governed by a verb in $s_{TL}$.

The method for aligning entities in different entity-sentence pairs remains the same as for the baseline method. In Sect. 4.1, we investigate the effectiveness of the two methods under different settings.

### 3.6 Forming Predicate-Verb Alignments

Although we use entities as a mechanism to align sentences and transfer predicate-argument roles, predicates in $s_{SL}$ and verbs in $s_{TL}$ cannot be aligned by entities alone. In addition, some sentence pairs contain more than one predicate or verb, sharing the same subset of entities. This creates a combinatorial problem, where one predicate in $s_{SL}$ could possibly be aligned to two or more verbs in $s_{TL}$, or vice versa. Furthermore, the application of a semantic parser to each $s_{SL}$ annotates each predicate with a sense. This requires a method to induce new predicates and senses for the verbs in $s_{TL}$.

Most previous work relies on word alignments or uses bilingual dictionaries to transfer the predicate annotation between languages. However, when applied to new languages and domains, these approaches face a scaling problem requiring either training on parallel corpora or otherwise dictionaries which may not be available for every language.

Our approach builds on Exner et al. (2015) and automatically infers new predicate labels while scaling with the size of corpora and domains. A formal description of our alignment is:

1. We determine all the combinations of predicate-verb pairs, $(p_i, v_k)$, extracted from all $(s_{SL}, s_{TL})$ pairs, where $p_i \in s_{SL}$ and $v_k \in s_{TL}$.

2. We assign $count(p_i, v_k)$ as the number of $(p_i, v_k)$ in all $(s_{SL}, s_{TL})$, where $s_{SL} \in SL$ and $s_{TL} \in TL$.

3. For each $p_i \in SL$, we form alignments as $(p_i \rightarrow v_k) = max(count(p_i, v_1), ..., count(p_i, v_n))$.

4. For each $v_k \in (p_i \rightarrow v_k)$, we form a new TL predicate by using the lemma of $v_k$ and an incremental counter based on the number of times $v_k$ has appeared in an alignment.

We select the verb candidates for the alignment using lexical and syntactical rules to filter auxiliary verbs and other non-predicates.

### 3.7 Transferring Propositions

Given a pair of aligned sentences, $(s_{SL}, s_{TL})$, we transfer the semantic annotation from a predicate, $p_{SL} \in s_{SL}$, to a verb, $v_{TL} \in s_{TL}$, if $(p_{SL} \rightarrow v_{TL}) = max(count(p_i \rightarrow v_{TL})), \quad (p_i \rightarrow v_{TL}) \in (s_{SL}, s_{TL}), \quad \forall p_i \in s_{SL}$. If a $s_{TL}$ is supervised by more than one $s_{SL}$, we select the $s_{SL}$ having the larger subset of aligned entities with $s_{TL}$. We restrict the semantic transfer to predicate-argument structures containing at least one numbered argument and a temporal or location modifying argument, or at least two numbered arguments.

We transfer the argument roles by using the aligned entities between $s_{SL}$ and $s_{TL}$. We assign the argument role to the governing token in the token span covered by each entity. However, if the argument token in $s_{SL}$ is dominated by a preposition, we search for a preposition in $s_{TL}$ governing the entity and assign it the argument role. We obtain the complete argument spans by taking the yield from the argument token.

## 4 Evaluation

In this section, we evaluate the approach described in Sect. 3 and we apply it to three language editions of Wikipedia in order to generate PropBanks for two languages: Swedish and French. The evaluation tries to answer the following questions:

1. How do different parameters and methods affect our approach?
2. What is the quality of the generated PropBanks and what level of performance can we expect in a practical setting?
3. Are there any differences between the languages, and if so what causes them?

### 4.1 Experimental Setup

For our experimentations, we chose the English, Swedish, and French editions of Wikipedia. These three Wikipedias are all among the top 6 in terms of article counts. As SL, we selected the English edition, and as TLs we select Swedish and French editions. We preprocessed all the articles to filter infoboxes, lists, diagrams, and to keep only text without any markup. Table 1 summarizes the statistics of the linguistic units in our chosen Wikipedias.

| LANGUAGE | TOKENS | SENTENCES | ENTITIES | PREDICATES | ARGUMENTS |
|---|---|---|---|---|---|
| English | 3825M | 279M | 439M | 186M | 450M |
| Swedish | 481M | 71M | 58M | - | - |
| French | 1269M | 74M | 181M | - | - |

Table 1: Characteristics of Wikipedias used in the experimental setup

### 4.2 Predicate-Verb Alignment

We first evaluated how the predicate→verb alignment method described in Sect. 3.6 performs under different conditions and we examined how the number of entities, the method used, and the frequency affect the quality of the alignments. We grouped the English→Swedish alignments by their frequency into three bands: High, medium, and low. We then randomly sampled alignments from each band, in total 100 alignments and we used them to evaluate their precision. We defined precision as the number of English→Swedish alignments that we evaluate as correct divided by the total number of alignments in a sample. Figure 2 shows the precision and number of alignments using different number of entities and methods.

We observe that the precision increases with the number of entities used in the alignments. However, this increase is followed by a decrease in the number of alignments created. We also note that in all the

alignments, our projection method outperforms our baseline method for aligning sentences in terms of precision. Using three projected entities, we reach a precision of roughly 80% and 1,000 alignments.

We also investigated if the higher frequency of an alignment improved precision. Figure 3 shows the breakdown of precision curves into three frequency bands, formed using projected alignments. We observe that using three projected entities, alignments with high-medium frequencies show little to no error. This provides empirical evidence to our hypothesis in Sect. 3.1, that the most frequent alignments of entities will elicit valid alignments and that precision will scale with the amount of data used by the method.

The combination of aligning sentences with three projected entities gave us the optimal trade off between precision and number of alignments created. Therefore, in the rest of the evaluation, we use these settings.

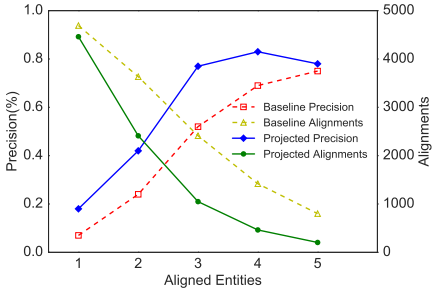



Figure 2: Graph of predicate→verb alignment precision and count under different parameter settings.

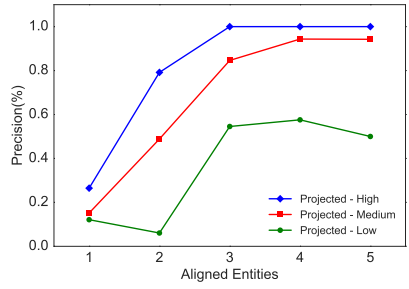



Figure 3: Graph of projected predicate→verb alignment precision, broken down by frequency band: high, medium, and low

### 4.3 Generated PropBanks

Using the annotation projection methods described in Sect. 3, we generated PropBanks in Swedish and French. We limited the PropBanks to only include fully annotated sentences and we removed the sentences exhibiting parsing errors, such as sentences having more than one syntactic root. We used these generated corpora to perform the error analysis in Sect. 4.5.

To evaluate our approach in a practical and automatic setting, we used samples of two linguistic resources: the Swedish FrameNet project (Borin et al., 2010) and the French FrameNet (Candito et al., 2014; Djemaa et al., 2016). We evaluated the generated Swedish and French corpora on a random sample of 100 sentences, from the Swedish FrameNet and the French FrameNet respectively. As PropBank and FrameNet have different annotation styles, we converted the sampled sentences from frame semantics to the semantics used in PropBank.

Table 2 shows the characteristics of the generated PropBanks and the FrameNets used in the evaluations.

| DATASET | TOKENS | SENTENCES | PREDICATES | ARGUMENTS |
|---|---|---|---|---|
| Generated-Swedish | 198,008 | 13,767 | 14,552 | 32,659 |
| Generated-French | 968,417 | 47,795 | 50,091 | 121,641 |
| SweFN++ (TEST) | 1,258 | 101 | 101 | 265 |
| French FrameNet (TEST) | 3,606 | 100 | 107 | 227 |

Table 2: Characteristics of the generated PropBanks used for training the SRL models and the FrameNets used for evaluating the trained models

### 4.4 Experimental Results

We evaluated the quality of the generated PropBanks in a practical setting as well as the effectiveness of using entity-like LUs in addition to entities. To assess the usefulness of the generated corpora, we first trained a semantic role labeler (Björkelund et al., 2010) on them. We split the generated corpora into 60:20:20 training, development, and testing sets, and we ran a selection process using a greedy forward selection and greedy backward elimination procedure to find the optimal set of features (Johansson and Nugues, 2008; Björkelund et al., 2009). We then used the trained models to automatically parse the test sets described in Sect. 4.3. Table 3 shows the evaluation of the semantic role labeler trained on the generated corpora.

The performance of the semantic role labeler, trained on the generated PropBanks, compares favorably with the automatic evaluations on parallel corpora described in Padó and Lapata (2009). For Swedish, using entity-like LUs, we observe an improvement of the labeled F1-measure by 10%. For French, we do not see the same dramatic increase, which we believe is caused by the large differences in pronoun classification and surface forms between English and French. We believe this discrepancy in improvement stems from projecting entity-like LUs across language groups: while English and Swedish belong to the Germanic branch, French belongs to the Romance group. Although more investigation is needed, these early results suggest that the annotation projection using entity-like LUs is most efficient when applied within a language group.

| | | LABELED | | | UNLABELED | | |
|---|---|---|---|---|---|---|---|
| LANGUAGE | LINGUISTIC UNITS | P | R | F1 | P | R | F1 |
| Swedish | Entities (Baseline) | 79.88 | 36.89 | 50.47 | 93.49 | 43.17 | 59.07 |
| | Entities + Unique Tokens | 84.82 | 44.26 | 58.17 | 92.67 | 48.36 | 63.55 |
| | Entities + Unique Tokens + Pronouns | 72.18 | 52.46 | **60.76** | 81.58 | 59.29 | **68.67** |
| French | Entities (Baseline) | 68.64 | 45.21 | 54.51 | 75.45 | 49.70 | 59.93 |
| | Entities + Unique Tokens | 64.03 | 48.50 | 55.20 | 70.36 | 53.29 | **60.65** |
| | Entities + Unique Tokens + Pronouns | 64.31 | 49.10 | **55.69** | 69.41 | 52.99 | 60.10 |

Table 3: Evaluation of semantic role labeling on the SweFN++ and French FrameNet corpora.

### 4.5 Error Analysis

To understand the quality of the generated PropBanks, we conducted an analysis of the predicate and argument errors. We randomly sampled 200 errors, of which 100 errors stemmed from the incorrect projection of argument labels and 100 were incorrect projections of predicates. Tables 4 and 5 show the type of errors for predicates and arguments respectively.

Using loosely parallel corpora, it is no surprise that the largest group of errors in predicate projection stems from sentences expressing differing semantic content. This error comes from sentence pairs, that although they contain the same subset of entities, express differing semantic content. However, as shown in Sect. 4.2, the precision of alignments increases with the number of alignments, leading us to believe that this category of error can be corrected using more data. The second largest error group is formed by different types of parsing errors occurring during the preprocessing stage. Encouragingly, only 6% of predicate projection errors stem from translation shifts, which is a further indication that entities exhibit a constraining property on the types of predicates that can instantiate them, even across languages.

Looking at argument projection errors, we again notice a group of errors stemming from misaligned sentences in loosely parallel corpora, *Differing Semantic Content* and *No Source Equivalent*. Looking beyond, alignment errors due to argument labels being assigned to the wrong token is the single most frequent error. The second largest category of errors is composed of expressions that can not be considered as entities, e.g. *In other words* and *During this time*. Finally, we observed a class of error stemming from entities undergoing a shift in specificity across sentences in two languages. These translation shifts included entities being referred to by their name in one language and by their entity type in the other

language, e.g. *London→the city*.

| ERROR CLASS | NUMBER |
|---|---|
| Differing Semantic Content | 66 |
| Parsing Error: Target Syntax | 8 |
| Translation Shifts: Predicate Mismatch | 6 |
| Parsing Error: Target SRL | 5 |
| Parsing Error: Entity Disambiguation | 5 |
| Auxiliary Verb | 4 |
| Light Verb Constructions | 4 |
| No Source Equivalent | 1 |
| No Target Equivalent | 1 |
| TOTAL | 100 |

Table 4: Error analysis of English→Swedish predicate→verb alignments.

| ERROR CLASS | NUMBER |
|---|---|
| Alignment Error: Non Argument Head | 16 |
| Argument is not Entity-like | 14 |
| No Source Equivalent | 14 |
| Parsing Error: SRL | 14 |
| Differing Semantic Content | 13 |
| Translation Shift: Argument Entity | 12 |
| Parsing Error: Entity Disambiguation | 9 |
| Parsing Error: Target Syntax | 4 |
| Translation Shift: Argument function | 3 |
| Parsing Error: Source Syntax | 1 |
| TOTAL | 100 |

Table 5: Error analysis of English→Swedish argument alignments.

## 5 Conclusion

In this paper, we have described the construction of multilingual PropBanks by aligning loosely parallel corpora using entities. We have trained a semantic role labeler on the generated PropBanks and that we evaluated in a practical setting on frame-annotated corpora. Our results compares favorably to annotation transfer using parallel corpora. In addition, we have extended the entity alignment to include alignment by entity-like linguistic units such as pronouns and dates.

We believe the growing source of loosely parallel corpora and their alignment using entities offers an alternative way to creating multilingual hand-annotated corpora. By performing a semantic projection on loosely parallel corpora, in our case multiple language editions of Wikipedia, we have presented an alternative approach to using parallel corpora. We believe our approach can be extended beyond encyclopedias to similar resources, such as news articles in multiple languages describing the same events.

One future improvement could be to leverage ontologies that categorize entities into types. We believe that such ontologies would prove useful in adjusting the specificity of entities in order to handle some translation shifts across languages. In addition, our current method of forming predicate→verb alignments could be extended by including information about the entity type.

While projecting pronouns from English to Swedish showed an improvement, we did not observe the same improvement when projecting from English to French. Therefore, an additional avenue of investigation could compare the performance of annotation projection within versus across language groups. In addition, a coreference solver could provide an alternative means of resolving pronominal mentions to entities.

## Acknowledgements

This research was supported by Vetenskapsrådet under grant 621-2010-4800, and the *Det digitaliserade samhället* and eSSENCE programs.

## References


Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 397–407.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Roberto Basili, Diego De Cao, Danilo Croce, Bonaventura Coppola, and Alessandro Moschitti. 2009. Cross-language frame semantics transfer in bilingual corpora. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 332–345. Springer.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics.

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds: a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87. Association for Computational Linguistics.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465. Association for Computational Linguistics.

Lars Borin, Dana Dannélls, Markus Forsberg, Maria Toporowska Gronostaj, and Dimitrios Kokkinakis. 2010. The past meets the present in swedish framenet+. In *14th EURALEX international congress*.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010. Statistical french dependency parsing: treebank conversion and first results. In *Seventh International Conference on Language Resources and Evaluation-LREC 2010*, pages 1840–1847. European Language Resources Association (ELRA).

Marie Candito, Pascal Amsili, Lucie Barque, Farah Benamara, Gal De Chalendar, Marianne Djemaa, Pauline Haas, Richard Huyghe, Yvette Yannick Mathieu, Philippe Muller, Benot Sagot, and Laure Vieu. 2014. Developing a french framenet: Methodology and first results. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Jinho D. Choi. 2012. *Optimization of Natural Language Processing Components for Robustness and Scalability*. Ph.D. thesis, University of Colorado Boulder.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 52–60.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 77–86.

Marianne Djemaa, Marie Candito, Philippe Muller, and Laure Vieu. 2016. Corpus annotation within the french framenet: a domain-by-domain methodology. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, may.

Peter Exner, Marcus Klang, and Pierre Nugues. 2015. A distant supervision approach to semantic role labeling. In *Fourth Joint Conference on Lexical and Computational Semantics (* SEM 2015)*.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM.

Raphael Hoffmann, Congle Zhang, and Daniel S Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 286–295.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187. Association for Computational Linguistics.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Applied Soft Computing*, 30:737–747.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Robert Östling. 2013. Stagger: An open-source part of speech tagger for swedish. *Northern European Journal of Language Technology (NEJLT)*, 3:1–18.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36(1):307–340.

Sebastian Padó. 2007. *Cross-lingual annotation projection models for role-semantic information*. Ph.D. thesis, Saarland University.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Alexandre Rafalovitch and Robert Dale. 2009. United nations general assembly resolutions: A Six-Language parallel corpus. In *Proceedings of the MT Summit XII*, pages 292–299. International Association of Machine Translation, August.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 12–21, Prague, June.

Lonneke Van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 299–304. Association for Computational Linguistics.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

# Langforia: Language Pipelines for Annotating Large Collections of Documents

# Langforia: Language Pipelines for Annotating Large Collections of Documents

**Marcus Klang**
Lund University
Department of Computer Science
Lund, Sweden
Marcus.Klang@cs.lth.se

**Pierre Nugues**
Lund University
Department of Computer Science
Lund, Sweden
Pierre.Nugues@cs.lth.se

## Abstract

In this paper, we describe **Langforia**, a multilingual processing pipeline to annotate texts with multiple layers: formatting, parts of speech, named entities, dependencies, semantic roles, and entity links. Langforia works as a web service, where the server hosts the language processing components and the client, the input and result visualization. To annotate a text or a Wikipedia page, the user chooses an NLP pipeline and enters the text or the name of the Wikipedia page in the input field of the interface. Once processed, the results are returned to the client, where the user can select the annotation layers s/he wants to visualize.

We designed Langforia with a specific focus for Wikipedia, although it can process any type of text. Wikipedia has become an essential encyclopedic corpus used in many NLP projects. However, processing articles and visualizing the annotations are nontrivial tasks that require dealing with multiple markup variants, encodings issues, and tool incompatibilities across the language versions. This motivated the development of a new architecture.

A demonstration of Langforia is available for six languages: English, French, German, Spanish, Russian, and Swedish at http://vilde.cs.lth.se:9000/ as well as a web API: http://vilde.cs.lth.se:9000/api. Langforia is also provided as a standalone library and is compatible with cluster computing.

## 1 The Demonstration

Langforia is a multilingual annotation and visualization platform available as a web service and as a standalone library. Figure 1 shows the interface, where the user chooses the language and tool chain s/he wants to use from the drop-down menu to the left. Depending on the language and the availability of components, the annotations can range from tokenization to dependency parsing, semantic role labeling, and entity linking. The user then either enters a text or writes the name of a Wikipedia page and presses the "Annotate" button. If the document to analyze is a raw text, it is sent directly to the server; if it is a Wikipedia page name, the client first fetches the HTML content of this page from https://www.wikipedia.org/ and then sends it to the Langforia server. Figure 2, left part, shows the resulting annotations for the *Osaka* article from the Swedish Wikipedia for three layers, tokens, named entities, and dependency relations, while the right part of the figure shows the entity linking results.

## 2 Motivation and Significance

We designed Langforia with a specific focus for Wikipedia, although the pipeline can process raw text. Wikipedia has become an essential encyclopedic corpus used in many NLP projects. In translation (Smith et al., 2010), semantic networks (Navigli and Ponzetto, 2010), named entity linking (Mihalcea and Csomai, 2007), information extraction, or question answering (Ferrucci, 2012), Wikipedia offers a multilingual coverage and an article diversity that are unequalled. However, processing articles are nontrivial tasks that require dealing with multiple markup variants, encodings issues, tool incompatibilities





*Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 74–78, Osaka, Japan, December 11-17 2016.

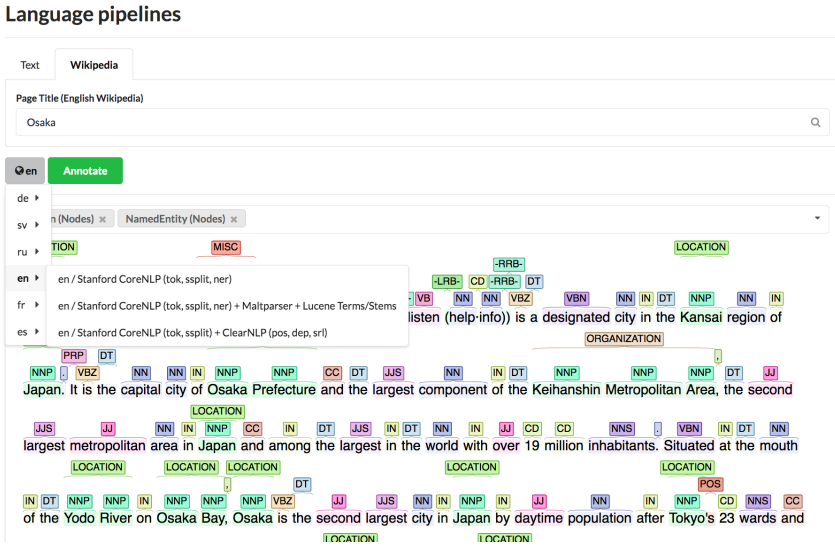



Figure 1: The Langforia interface. The upper part of the figure shows the input box, where the user either selects a Wikipedia page (Wikipedia tab), here the article on Osaka in English, or enters a raw text (Text tab); The center part shows the language selection menu with six languages (de, en, es, fr, ru, sv), here English, and submenus to choose the toolchain (three toolchains for English); and the Annotate button; The lower part shows the annotated text, where the annotation layers are selectable from a drop down menu in the block just above (black triangle to the right), here the tokens and named entities

across the language versions and significant processing capacities. In addition, the scale and heterogeneity of the Wikipedia collection makes it relatively difficult to do experimentations on the whole corpus. These experimentations are rendered even more complex as, to the best of our knowledge, there is no available tool to visualize easily annotation results from different processing pipelines.

Langforia builds on a document model (Klang and Nugues, 2016) that stores the linguistic annotations and enables the pipeline to abstract the components across the languages and tools. This model consists of layers, where each layer is a sequence of ranges describing a specific annotation, for instance the parts of speech or the syntactic dependencies. It provides a format common to all the pipelines that makes them insensitive to the input/output features of a tool.

The list of annotated layers varies depending on the tool availability for a specific language. The layers common to all the versions are compatible with the Wikipedia markup: They include the text, paragraphs, text styles, links, and page sections. Using this document model as input, we created a client visualizer that let users interactively visualize the annotations. Beyond the demonstration, Langforia is available in the form of a library that provides a uniform way to process multilingual Wikipedia dumps and output the results in a universal document model. This could benefit all the projects that use Wikipedia as a corpus.

## 3 System Architecture

Langforia consists of three parts: A set of language processing components assembled as tool chains; a multilayer document model (MLDM) library; and a visualizer.

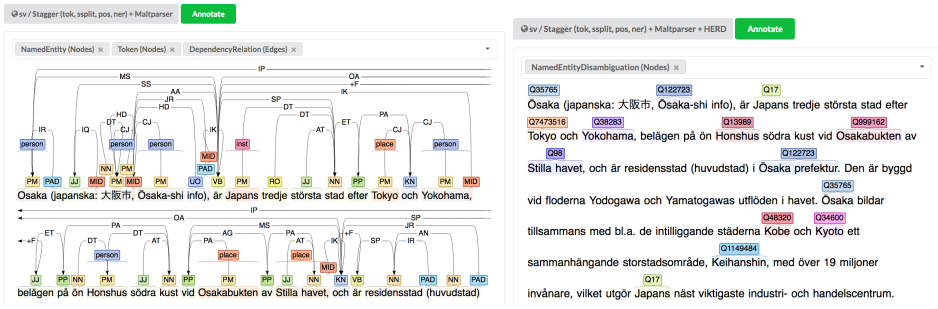



Figure 2: Left part: Visualization of three layers: Tokens, named entities, and dependency relations from the *Osaka* page in Swedish; right part: Visualization of named entity linking with Wikidata identifiers

### 3.1 Tool Chains

We annotate Wikipedia HTML pages into MLDM records using an annotation pipeline: a sequence of processing components. The first step converts the HTML documents into DOM trees using jsoup[1]. The second step extracts the original page structure, text styles, links, lists, and tables. We then resolve the links to unique Wikidata identifiers. Wikidata is an entity database[2], part of Wikimedia, which assigns unique identifiers across all the language editions of Wikipedia. The city of Osaka, for instance, has the unique id: Q35765 that enables the system to retrieve the article pages in English, French, Swedish, or Russian. We keep the original links occurring in the Wikipedia pages and we resolve them using Wikidata identifiers, when they exist, or to normalized page names as a fall back. These steps are common to all the language editions we process. If the input is plain text, we skip these steps.

The annotation tool chains are specific to the languages. We abstracted these chains so that they are instances of a generic annotator. For English, Spanish, and German, we use CoreNLP (Manning et al., 2014) or ClearNLP (Choi, 2012). For French, we use CoreNLP for tokenizing the text and MATE for parsing (Björkelund et al., 2010). For Swedish, we use Stagger (Östling, 2013) and MaltParser (Nivre et al., 2006). For Russian, only the tokenization is available for now. We also link mentions of named entities and concepts to unique Wikidata identifiers. To carry this out, we reimplemented a variant of TAGME (Ferragina and Scaiella, 2010).

### 3.2 The Document Model

The MLDM library[3] (Klang and Nugues, 2016) defines a model for storing, querying, and extracting hypertextual information common to many NLP tasks in a standalone package. We designed this model so that it could store the original Wikipedia markup, as well as the subsequent linguistic annotations: Part-of-speech tagging, coreference resolution, named entity recognition and linking, dependency parsing, semantic role labeling, etc.

The model consists of multiple layers, where each layer is dedicated to a specific type of annotation. The annotations are encoded in the form of graph nodes, where a node represents a piece of data: a token, a sentence, a named entity, etc., delimited by ranges. These nodes are possibly connected by edges as in dependency graphs. This data structure used is similar to a property graph.

### 3.3 Visualization

The interactive visualization tool enables the user to examine the results. We designed it so that it could handle large documents with more than 10,000 tokens with a fast rendering of the annotations and allow cross sentence annotations, such as for paragraphs and sections. The layers are selectable from a dropdown menu and the supported visualizations are the ranges and relationships between them.

[1] http://jsoup.org/
[2] http://www.wikidata.org
[3] https://github.com/marcusklang/docforia

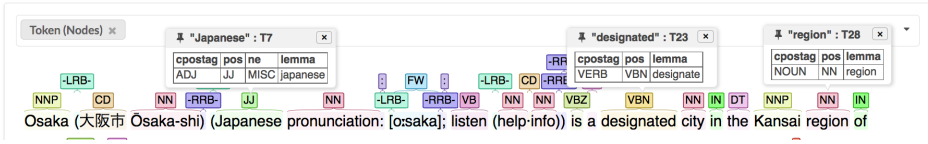



Figure 3: The properties attached to the words *Japanese*, *designated*, and *region*, in the form of tooltips

In Fig. 3, we selected the token layer that by default displays the parts of speech of the words. If we hover over the words, the visualizer shows the properties attached to a word in CoNLL-like format in a tooltip that the user can fix, move, and discard. Figure 3 shows the properties of the words: *Japanese*, *designated*, and *region*. Finally, we estimated the rendering speed (time to interactive use) on 30,000 annotations (tokens) with Intel Core i7, 2.3 GHz, with 16 GB RAM running a Chrome browser and we obtained the figure of 7.7s seconds, i.e. 3,800 annotations per second.

## 4 Related Work

The UIMA project (Ferrucci and Lally, 2004) provides an infrastructure to store unstructured documents. In contrast, the MLDM library and Langforia emphasize on simplicity, portability, ease of integration, minimal dependencies, and efficiency. Other toolchains include CoreNLP (Manning et al., 2014). However, CoreNLP cannot process the Wikipedia markup or easily integrate external tools. In addition, CoreNLP does not provide a storage model and its data structures are primarily meant to extend its functionalities. In contrast to CoreNLP, Langforia builds on Docforia that provides dynamic and typed annotations as well as multiple sublayers such as gold and predicted. Finally, CoreNLP does not provide a query API for its data structures.

The Langforia visualization tool is similar to the brat[4] components (Stenetorp et al., 2012) for the text visualization. Brat produces good visual results and has support for multiple layers of information. However, to the best of our knowledge, it lacks tooltip support in the embeddable version and it does not handle line-wrapped annotations well. In addition, it revealed too slow to render a large number of annotations in the documents we tested.

## 5 Conclusion and Future work

We described Langforia, a multilingual tool for processing text and visualizing annotations. Langforia builds on a multilayer document model (MLDM), structured in the form of a graph and unified tool chains. It enables a user to easily access the results of multilingual annotations and through its API to process large collections of text. Using it, we built a tabulated version of Wikipedia (Klang and Nugues, 2016) that can be queried using a SQL-like language. When applied to Wikipedia, MLDM links the different versions through an extensive use of URI indices and Wikidata Q-numbers.

## 6 Availability

The Langforia demonstration is accessible at: `http://vilde.cs.lth.se:9000/` and the web API at: `http://vilde.cs.lth.se:9000/api`. The source code is available from github at: `https://github.com/marcusklang/`.

## Acknowledgments

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

[4] `http://brat.nlplab.org/`

## References


Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstration Volume*, pages 33–36, Beijing, August 23-27.

Jinho D. Choi. 2012. *Optimization of Natural Language Processing Components for Robustness and Scalability*. Ph.D. thesis, University of Colorado at Boulder, Boulder, CO, USA. AAI3549172.

Paolo Ferragina and Ugo Scaiella. 2010. Fast and accurate annotation of short texts with wikipedia pages. In *Proceedings of CIKM'10*, Toronto.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, September.

David Angelo Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1:1 –1:15, May-June.

Marcus Klang and Pierre Nugues. 2016. Wikiparq: A tabulated Wikipedia resource using the Parquet format. In *Proceedings of LREC 2016*, pages 4141–4148, Portorož, Slovenia.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on CIKM*, CIKM '07, pages 233–242, Lisbon, Portugal.

Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the ACL*, pages 216–225, Uppsala.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219.

Robert Östling. 2013. Stagger: an open-source part of speech tagger for Swedish. *Northern European Journal of Language Technology*, 3:1–18.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 403–411.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April. Association for Computational Linguistics.

# Overview of the Ugglan Entity Discovery and Linking System

# Overview of the Ugglan Entity Discovery and Linking System

**Marcus Klang** **Firas Dib** **Pierre Nugues**
Lund University
Department of Computer Science
S-221 00 Lund, Sweden
marcus.klang@cs.lth.se pierre.nugues@cs.lth.se
firas.dib@gmail.com

## Abstract

Ugglan is a system designed to discover named entities and link them to unique identifiers in a knowledge base. It is based on a combination of a name and nominal dictionary derived from Wikipedia and Wikidata, a named entity recognition module (NER) using fixed ordinally-forgetting encoding (FOFE) trained on the TAC EDL data from 2014-2016, a candidate generation module from the Wikipedia link graph across multiple editions, a PageRank link and cooccurrence graph disambiguator, and finally a reranker trained on the TAC EDL 2015-2016 data.

In our first participation in the TAC trilingual entity discovery and linking task, we obtained a *strong typed mention match* of 0.701 (Ugglan2), a *strong typed all match* of 0.592 (Ugglan4), and *typed mention ceaf* of 0.595 (Ugglan1).

## 1 Introduction

The goal of the trilingual entity discovery and linking task (EDL) in TAC 2017 is to recognize mentions of entities in Chinese, English, and Spanish text and link them to unique identifiers in the Freebase knowledge base. In the TAC datasets, the mentions have a type, either persons (PER), geopolitical entities (GPE), organizations (ORG), locations (LOC), or facilities (FAC), and their syntactic form can consist of proper or common nouns, called respectively named and nominal mentions. Some entities of the annotated corpus are not in Freebase. They are then linked to a NIL tag and clustered across identical entities, each with a unique ID.

In this paper, we describe Ugglan, a generic multilingual EDL platform that required minimal adaptation to the TAC 2017 tasks. We detail the system architecture and its components as well as the experimental results we obtained with it.

## 2 System Overview

Ugglan has a pipeline architecture that consists of three main parts:

- A mention discovery that uses a finite-state automaton derived from Wikipedia and/or a feed-forward neural network trained on the TAC 2014-2016 data (Ji et al., 2014, 2015; Ji and Nothman, 2016);
- An entity linker that uses mention-entity pairs extracted from Wikipedia and ranks them using PageRank;
- A reranker trained on the TAC 2014-2016 data.

The Ugglan architecture is modular and parameterizable, and its parts can use independent algorithms. To build it, we used a set of resources consisting of Wikipedia, Wikidata, and DBpedia.

### 2.1 Mention Discovery

The first part of the processing pipeline is the discovery of mentions of entities in text. It starts with a custom multilingual rule-based tokenization of the text and a sentence segmentation. We then normalize the letter case based on statistics from all the Wikipedia pages.

We discover the mentions using a combination of a finite state transducer (FST) built from mentions extracted from Wikipedia and an optional

named entity recognizer (NER) based on neural networks. As alternative, Ugglan can also use the Stanford NER (Finkel et al., 2005).

The mention discovery results in an overgeneration of mention candidates. For instance, the phrase

United States of America

results into as many as 8 candidates; see Fig. 1. We prune them using parameterized rules.

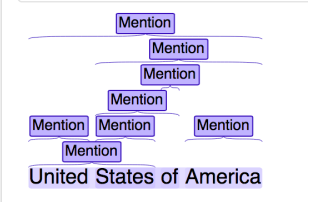



Figure 1: Mention candidates produced by the finite-state transducer for the phrase *United States of America*

Finally, our system does not output overlapping mentions. We resolve this overlap using a statistical estimation of the mention "linkability": The **link density**; see Sect. 4.2. Figure 2 shows the overall mention detection steps.

### 2.2 Entity Linking

Once we have carried out the detection, we associate each mention with entity candidates by querying a mention-entity graph.

Some of the entities are referred by mention variants along a text, for instance starting with "Barack Obama" and then "Obama" or "Barack". We augment the entity recall by sorting the mentions in a document with a partial ordering using the `is_prefix` or `is_suffix` relations so that we have:

$$\begin{array}{lcl} \text{Barack} & \prec & \text{Barack Obama} \quad \text{and} \\ \text{Obama} & \prec & \text{Barack Obama.} \end{array}$$

We then expand the candidates of the substrings by adding the candidates of the including strings. For the mentions: "Barack Obama" and "Obama", we add all the candidate entities of "Barack Obama" to the candidates of "Obama".

We extracted the graph of mentions to candidate entities from Wikipedia as well as the graph of entity–entity cooccurrences. We built this graph from outlinks gathered from the combination of seven Wikipedia editions.

Finally, we disambiguate the text entities using a local graph of candidates, on which we apply the PageRank algorithm. For each mention, PageRank assigns a weight to the candidates that enables us to rank the entities.

### 2.3 Reranking and Classification

After the entity linking step, each mention has a ranked list of entity candidates. We rerank these lists using a multilayer neural network trained on the TAC2015-16 data. This also results in some entities being assigned the NIL identifier.

We assign a type to the entities using a predefined dictionary mapping derived from DBpedia; this type is possibly merged with that obtained from the NER, if no available mapping exists.

At this point, we have discovered and resolved the named expressions. We apply a discovery to the nominal expressions (NOM) using a dictionary collected from Wikidata and a coreference resolution based on exact string matches.

Finally, we discard the classes not relevant to the TAC task.

## 3 Building Ugglan's Knowledge Base

Ugglan relies on a graph of mention-entity and entity-entity for both the discovery and the linking stages. We constructed this knowledge base from a set of resources:

- Seven Wikipedia editions: en, es, zh, de, fr, ru, and sv;
- Wikidata, which binds these editions together;
- DBPedia, which we used for the class mapping.

### 3.1 The Wikipedia Corpus

The Wikipedia corpus is our first resource from which we extract the text, the mentions, and link

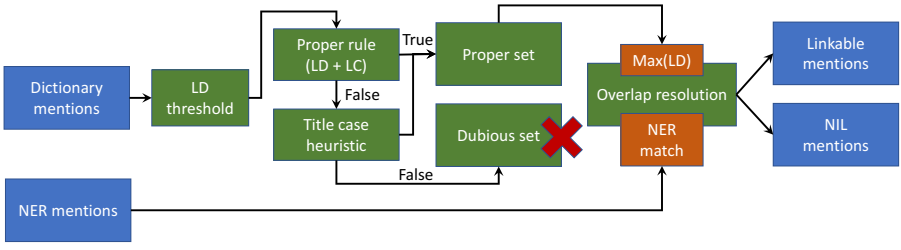



Figure 2: Overview of the mention pipeline

graph. We can access its content in multiple ways and with varying degrees of accuracy.

A simple approach is to download a XML dump, apply some filtering to the raw wiki markup, and then use this output as text. However, Wikipedia features many templates that are language dependent. Using a dump approach would leave the template expansion unsolved, as well as the local context of the links and information about the logical structure.

We opted therefore to use a rendered HTML version which expands the templates and has its Scribuntu scripts executed. This produces a HTML dump of Wikipedia, which is as true to the user online version as possible without actually replicating the full Wikipedia infrastructure. Concretely, we used a combination of Xowa (offline Wikipedia reader) and the public Wikipedia REST API to get the HTML dumps.

#### 3.1.1 HTML Annotation Processing

The raw HTML must be filtered and simplified to be easy to process. We converted the hierarchical structure using a DOM tree produced by Jsoup and applied heuristics to produce a flattened version with multiple layers of annotations using the Docforia structure (Klang and Nugues, 2017). These layers include information on anchors, headers, paragraphs, sections, lists, tables, etc.

The Chinese version was processed in a special way as it can be translated in multiple variants: simplified, traditional, and localizations. To get coherent statistics, we reimplemented the translation mechanism used by the online version to produce a materialized `zh-cn` Wikipedia edition.

We finally resolved all the pages and anchors in the flattened version to Wikidata, which produces multilingual connections for most entities.

### 3.2 Multilingual Resources: Wikidata and DBpedia

Wikidata is Ugglan's repository of multilingual entities as it contains clean mappings between the multiple language editions, as well as detailed structured data. TAC uses Freebase as knowledge base and we created mappings to translate Freebase entities to Wikidata. Wikidata entities are represented by unique identifiers called Q-numbers. We mapped the Wikidata entities missing from Freebase to NIL-xx identifiers, where xx is a number unique for the entity.

We chose DBpedia to map entities to TAC classes as it produced subjectively better mappings than using Wikidata. Wikidata would have required additional rules to carry out the conversion.

### 3.3 Mention–Entity Graph

We associated each mention in Wikipedia with a set of potential entities in text. We used a dictionary, where the entry (or key) is the entity mention from Wikipedia and the value is the Q-number. In Wikipedia, we extracted the mentions from five sources across the languages we consider:

**The title** of the entity's Wikipedia page with and without parentheses. For instance, we have $M_{Q90} = \{\text{Paris}\}$ and $M_{Q167646} = \{\text{Paris, Paris\_(mythology)}\}$.

**The Wikipedia redirects:** i.e. page names that automatically redirect to another

page like the *EU* page that moves the reader to *European Union*. This enables us to collect alternative names or spelling variants so that we expand the mention list for the European Union entity, $M_{Q458}$, from {European Union} to {European Union, EU}.

**The disambiguation pages,** where a page title is associated to two or more entities. The English Wikipedia has a disambiguation page associated with *Paris* that links to about 100 entities, ranging from the capital of France to towns in Canada and Denmark and song and film titles, and finally

**The wikilinks.** A link in Wikipedia text is made of a word or a phrase, called the label, that shows in the text, and the page (entity) it will link to, where the wiki markup syntax uses double brackets: `[[link|label]]`. When the link and the label are different, the label is often a paraphrase of the term. We therefore consider all these labels as candidate mentions of the entity. Examples in Swedish of such labels for the former Swedish Prime Minister Göran Persson, Q53747, include the name itself, *Göran Persson*, 468 times, *Persson*, 14 times, and *Han Som Bestämmer*, 'He Who Decides', one occurrence.

**The first bold-faced string.** We also used the first bold-faced string in the first paragraph of an article as it often corresponds to a synonym of the title (or the title itself).

### 3.4 Statistics

During the mention gathering, we also derive statistics for a given language. Before we compute these statistics, we apply a procedure that we called *autolinking*. In an article, the Wikipedia guidelines advise to link only one instance of an entity mention[1]: Normally the first one in the text. With the autolinking procedure, we link all the remaining mentions provided that we have sequences of exactly matching tokens. The statistics we collect are:

- The frequency of the mention string over the whole Wikipedia collection (restricted to one language);
- The frequency of the pair (mention, entity) that we derive from the links without autolinking (only manually linked mentions);
- The count of (entity1, entity2) pairs in a window corresponding to a paragraph and limited to 20 linked mentions. This is carried out after autolinking;
- Capitalization statistics for all the tokens: We extract token counts for all tokens with a frequency greater than 5 and we break them down by case properties: uppercased, lowercased, titlecased, and camelcase;

## 4 Mention Recognition

Ugglan uses a multilingual rule-based tokenizer and segmenter that we implemented using JFlex. For logographic languages such as Chinese, the tokens are equivalent to characters. The parser was customized to accept a mixture of both logographic and alphabetic text.

The tokenizer was used in conjunction with the Lucene analyzers. Lucene provides an infrastructure consisting of common filters and normalizers for many languages, from which we use case folding, accent stripping, Unicode form normalization, and stemming. These pipelines are configurable and easy to adapt for new languages.

The mention discovery is carried out by two modules: A dictionary-based finite-state transducer and a named entity recognizer (NER) using neural networks that we trained on the TAC data; see Sect. 5. These two modules can work in tandem or independently.

The mention recognition pipeline has three primary steps: discovery, filtering, and overlap resolution. In addition, the discovery pipeline can be configured to use one of three modes: NER-only, dictionary-only, and a hybrid mode. The primary

[1] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking#Overlinking_and_underlinking

difference between these modes is how the filtering and overlap resolution operates.

Figure 2 shows an overview of these steps.

### 4.1 Discovery

Before querying the FST dictionary, we normalize the tokens in uppercase or lowercase characters for English and Spanish using statistics derived from Wikipedia. For instance, we convert *BEIJING* into the title case variant *Beijing*. However, due to ambiguity, we did not apply case normalization to title-cased words.

### 4.2 Filtering

The mentions of named entities are likely to be linked in Wikipedia. Examining the articles, we observed that, given word sequence, the relative frequency of linkage often reveals its ambiguity level. For instance, while the word *It* can refer to a novel by Stephen King, it is rarely an entity and, at the same time, rarely linked.

The link density (LD) is a measure derived from the analysis of text linkage in Wikipedia. It loosely corresponds, as the original text is not fully linked, to the probability of a sequence of tokens being linked in the source edition.

We estimated the linkage probability by applying the FST dictionary to Wikipedia in an offline step. We counted the exact matches with the known ground truth: The anchors created by the Wikipedia editors. In addition, before counting, *autolinked* anchors were added that matched existing ones perfectly.

$$\text{Link density} = \frac{\#\text{Anchor}}{\#\text{Text} + \#\text{Anchor}} \quad (1)$$

In addition to link density, we used the gold standard mention counts as a measure significance: The link count (LC).

Before the overlap resolution, all the mentions from the dictionary are classified into either a proper set or a dubious set. The proper set consists of all the mentions which exceed the LD and LC thresholds; The proper set also includes the mentions which do not exceed these thresholds if at least 75% of the tokens in the mention are title cased.

### 4.3 Overlap Resolution

The mentions placed in the *proper set* and the mentions found by the NER will be merged and resolved in the overlap resolution step. The NER module itself only outputs nonoverlapping mentions. This stage works differently depending on which mode is used:

- *NER-only* accepts only NER mentions and produces linkable mentions, if an exact dictionary match is found.
- *Dictionary-only* ignores the NER mentions and solves overlapping mentions by picking the mention which has the largest LD value until no overlap exists.
- *Hybrid* merges NER mentions with dictionary matches by trusting the NER output where applicable i.e. when multiple candidates exist, it chooses the NER output, otherwise the dictionary output.

## 5 Named Entity Recognizer

We developed a named entity recognizer (NER) based on a feed-forward neural network architecture and a *fixed ordinally-forgetting encoding* (FOFE) (Xu et al., 2017; Zhang et al., 2015). This NER is part of the mention recognition module; see Figure 2 for the dataflow.

The NER operates over sentences of tokens and outputs the highest probability class using a moving focus window with varying width. The focus window represents potential named entity candidates and ranges from one to seven words. A more detailed explanation of this, and why there is an upper limit, is explained in Sect. 5.3

The NER can recognize both named and nominal expressions and predict their class. The named or nominal types are just extensions to the classes. If there were $N$ classes originally, there would be $2N$ outputs if all nominal classes were included. In the TAC2017 Ugglan system, the possible classes are:

- {*PER, GPE, ORG, LOC, FAC*}-*NAM* and
- {*PER, GPE, ORG, LOC, FAC*}-*NOM*.

The NER is identical in its construction for English and Spanish, without any language specific feature engineering. However, we modified this module for Chinese since the word segmentation was not found reliable. In Chinese, we used the individual characters (logograms) as word features and none of the corresponding Latin character features. In any case, the Chinese character features would be a subset of the word features.

### 5.1 The fixed ordinally-forgetting encoding

We applied a *fixed ordinally-forgetting encoding* (FOFE) (Xu et al., 2017; Zhang et al., 2015) as a method of encoding variable-length contexts to fixed-length features. This encoding method can be used to model language in a suitable manner for feed-forward neural networks without compromising on context length.

The FOFE model can be seen as a weighted bag-of-words (BoW). Following the notation of Xu et al. (2017), given a vocabulary $V$, where each word is encoded with a one-hot encoded vector and $S = w_1, w_2, w_3, ..., w_n$, an arbitrary sequence of words, where $e_n$ is the one-hot encoded vector of the nth word in $S$, the encoding of each partial sequence $z_n$ is defined as:

$$z_n = \begin{cases} 0, & \text{if n = 0} \\ \alpha \cdot z_{n-1} + e_n, & \text{otherwise,} \end{cases} \quad (2)$$

where the $\alpha$ constant is a weight/forgetting factor which is picked such as $0 \leq \alpha < 1$. The result of the encoding is a vector of dimension $|V|$, whatever the size of the segment.

Zhang et al. (2015) showed that we can always recover the word sequences $T$ from their FOFE representations if $0 < \alpha \leq 0.5$ and that FOFE is almost unique for $0.5 < \alpha < 1$. Zhang et al. (2015) make the assumption that the representation is (almost) always unique in real texts.

### 5.2 Features

The neural network uses both word and character-level features. The word features extend over parts of the sentence while character features are only applied to the focus words: The candidates for a potential entity.

#### 5.2.1 Word-level Features

The word-level features use bags of words to represent the focus words and FOFE to model the focus words as well as their left and right contexts. As context, we used all the surrounding words up to a maximum distance, defined by the floating point precision limits using the FOFE $\alpha$ value as a guide.

Each word feature is used twice, both in raw text and normalized lower-case text. The FOFE features are used twice, both with and without the focus words. For the FOFE-encoded features, we used $\alpha = 0.5$.

The beginning and end of sentence are explicitly modeled with `BOS` and `EOS` tokens, which have been added to the vocabulary list.

The complete list of features is then the following:

- Bag of words of the focus words;
- FOFE of the sentence:
  - starting from the left, excluding the focus words.
  - starting from the left, including the focus words.
  - starting from the right, excluding the focus words.
  - starting from the right, including the focus words.

This means that, in total, the system input consists of 10 different feature vectors, where five are generated from the raw text, and five generated from the lowercase text.

#### 5.2.2 Character-Level Features

The character-level features only model the focus words from left to right and right to left. We used two different types of character features: One that models each character and one that only models the first character of each word. We applied the FOFE encoding again as it enabled us to weight the characters and model their order. For these features, we used $\alpha = 0.8$.

In order to ensure the characters fall into an appropriate range, we encoded them with a simple

modulo hash. Each characters ASCII value is normalized to be within the range 0 and 128. This limitation is reasonable since most characters of English and Spanish are in the ASCII table. The Spanish characters in the range 128:256 are confused with unaccented ASCII characters, for instance $\tilde{n}$ with $q$.

#### 5.2.3 Projection Layers

**Characters.** The character features are generated first as sparse one-hot encoded vectors of dimension 128 and then projected to a dense representation of lower dimension: 64. To project the character features, we used a randomly initialized weight matrix, which is trained as part of the network. This procedure produced better results than the direct input of one-hot vectors.

**Words.** We projected the word-level features to a 256-dimension dense representation. We initialized the projection layer with two different word2vec (Mikolov et al., 2013) models that we pretrained on the en, es, and zh wikipedias. One model was trained on normalized text, while the other was trained on the raw untouched text. These are incorporated into the rest of the network and consequently trained as a part of it.

When creating the word projection layers from the word2vec models, we used the weights of the top 100,000 words. We disregarded all the other words and instead mapped them to a *unknown word vector*. More specifically, for the case-sensitive projection layer, we return a `UNK` vector when we encounter a word with no embedding; if this word is equal to its normalized lower cased variant, we return a special `unk` vector instead.

### 5.3 Named Entity Candidates

The potential named entity candidates are produced by looping over each word in the sentence with a moving window that expands up to seven words. This exhaustively generates all the possible candidates in the sentence, which in turn produces a lot of noise. In the training process, we sample this noise to build a set of negative examples and instruct the network how to discriminate mention boundaries and invalid mentions.

The upper bound of seven was found by going through the annotations of the TAC 2016 data and seeing if there was any clear cut-off where results would start to diminish. After seven words, we found there was little benefit to go any further. This upper limit value is significant because each candidate which is not a positive sample is considered negative and in turn used in the training process.

A large focus window results in many negative samples, which are not representative of the real world. As the negative candidates are randomly sampled, we would (to some degree) get a skewed distribution of the negative samples. If, for example, the upper bound was set to 12 words, there would be many negative 12-token long samples in comparison to how many positive examples there are. We attempted to weigh the different selections with respect to the positive mention count. We set it aside for the TAC 2017 evaluation due to a lack of time.

In total, we considered three different cases to create the training data:

1. The mention candidate matches exactly an annotated sequence;
2. The candidate is completely disjoint, i.e., contains no annotated words;
3. The candidate partially or completely overlaps with an annotated sequence or is a subset of the sequence,

where an annotated sequence corresponds to all the words annotated with a given class in the data, such as `University of Lund`.

The first case corresponds to the positive examples that we label with the TAC classes, while the two last cases are the negative examples that we label as `NONE`. We will keep this stratification in the training step.

### 5.4 Training

In the data set we collected, the negative samples outnumber the positive ones by an order of magnitude. We used a manually-specified distribution of the samples to mitigate this unbalance and train the network. At the beginning of each epoch, the

data is shuffled and the negative samples are re-selected according to the distribution. This means that we continuously introduce new negative examples and previously unseen data to the network, which helps with regularization.

Table 1 shows the distribution we used for the TAC 2016 EDL task.

| Candidate type | Ratio of sample size |
|---|---|
| Negative: Overlapping | 60% |
| Negative: Disjoint | 30% |
| Positive | 10% |

Table 1: The distribution between positive and negative mention candidates.

We trained the NER system with data from TAC 2014-15 and evaluated it on the 2016 data; see Table 2.

### 5.5 Neural Network Architecture

The network architecture can be conceptually divided into two parts: A first part projects the input features into a dense space and a second one classifies the input and outputs a class (see Figure 3). The classification part of the network consists of three hidden layers, which have batch-normalization layers sliced in-between them, and a final layer that outputs multiclass predictions.

During the development, we tested and evaluated several hyperparameters using a grid search method. Table 3 shows the final hyperparameters used in the TAC2017 EDL task. We started from initial values identical to those in Xu et al. (2017).

Both the learning rate and dropout followed a linear decay schedule in which they would have a final value of 0.0064 and 0.1024, respectively, by the end of training. We also conducted tests that showed that having a constant, lower dropout rate, yielded slightly better results. We did not use them for the EDL task due to time constraints.

### 5.6 Candidate Pruning

We exhaustively generated all the possible mention candidates that we passed to the classification step. The output is a probability distribution for each class (named and nominal), whose sum is 1. We used the highest probability class to tag the mention if it was 0.5 or greater, otherwise we ignored the output and assigned it to the `NONE` class.

No overlapping mentions were output, instead each mention had to have no overlap at all. We evaluated two different algorithms to determine which mentions to keep: The highest probability and the longest match:

- The highest probability algorithm proceeds from left to right and uses the highest probability, nonoverlapping, leftmost mention.
- The longest match instead uses the longest, nonoverlapping, rightmost mention.

During testing, the highest probability algorithm produced the best results, a few points greater than the longest first. The output was also visually cleaner upon manual inspection. We did a grid search for the cutoff value and found that 0.5 produced the best results. Nonetheless, both 0.4 and 0.6 yielded similar results and would be reasonable choices as well.

## 6 Entity Linking

### 6.1 Generation of Entity Candidates

We used the mentions from the recognition step to produce the entity candidates. Each mention found by the FST dictionary has a unique ID that serves as entry point to the mention-entity graph (Sect. 3.3).

### 6.2 Construction of a Local Graph to Disambiguate Entities

We build a local graph to disambiguate the entities in a text. The nodes of the graph consist of the mentions and the candidates. We link these nodes with three types of edges:

1. The mention-to-candidate edges;
2. The entity cooccurrence edges linking entities when they cooccur in the Wikipedia corpus;
3. The entity inlink edges, reflecting links between pages (entities) in Wikipedia;

While the mentions are language-dependent, the entities reside in a multilingual domain and their

| Language | Named | | | Nominal | | | Overall | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 |
| English | 0.734 | 0.816 | 0.773 | 0.580 | 0.805 | 0.674 | 0.801 | 0.676 | 0.733 |
| Chinese | 0.769 | 0.792 | 0.780 | 0.554 | 0.757 | 0.639 | 0.769 | 0.612 | 0.682 |
| Spanish | 0.736 | 0.685 | 0.709 | 0.584 | 0.657 | 0.618 | 0.736 | 0.567 | 0.640 |

Table 2: Results from evaluating on the 2016 data (not including wikipedia dictionary).

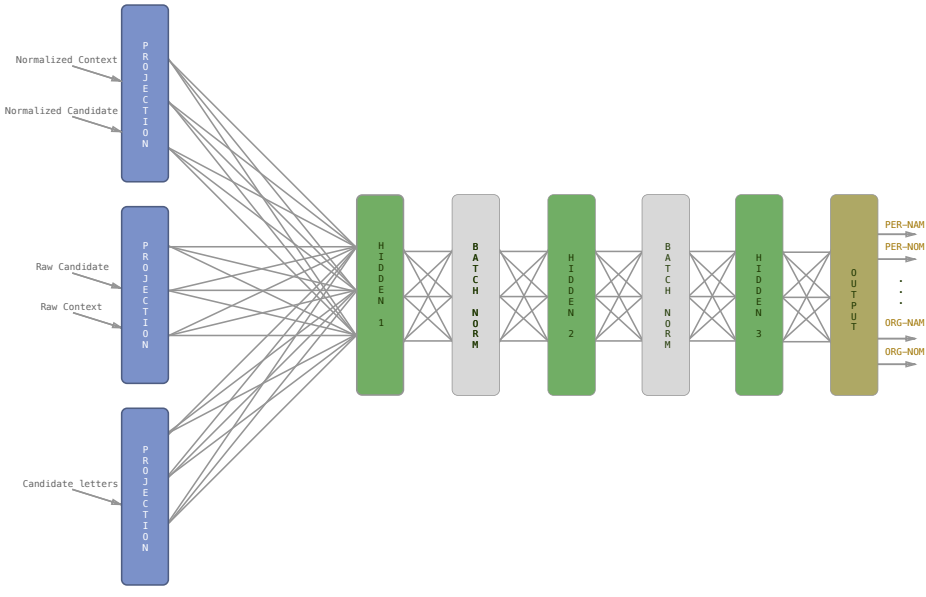



Figure 3: Overview of the full NER network architecture

edges are aggregated from all the editions of Wikipedia we consider.

Fig. 4 shows an overview of the graph.

Following Södergren and Nugues (2017), the graph is weighted using the PageRank algorithm (Brin and Page, 1998). The candidates are ranked per mention and a normalized weight is produced. The top three candidates are reordered if the candidate title exactly matches the mention.

In contrast to Södergren and Nugues (2017), we included the inlinks and we carried out the disambiguation inside a window of 20 mentions. We introduced this window method to reduce the computation and the upper-bound execution time. At the start of the disambiguation, the window is set at the beginning of the text and then shift by 10 mentions. The windows are then partially overlapping and, in case of conflict, we use the rankings from the left one.

### 6.3 Reranker

To reduce errors made by the disambiguator and introduce a NIL candidate, we trained a reranker on the TAC 2015-2016 data. We generated a training set of examples by applying the graph-based disambiguator to all the available annotated text. We limited the set of candidates to the top three entities for each mention or up to the correct one if necessary. We then marked each candidate in

| Name | Value |
|---|---|
| Weight initialization | RELU Uniform |
| Max. window size | 7 |
| Epoch count | 160 |
| Learning rate | 0.1024 |
| Dropout | 0.4096 |
| Optimizer | ADAM |
| L2 regularization | 0.0 |
| Neuron count | 512 |
| Batch size | 512 |
| Activation function | Leaky RELU |

Table 3: The final hyperparameters used in the TAC2017 EDL task

these lists as positive or negative according to the gold standard. We assigned all the detected mentions overlapping gold standard mentions to the NIL entity. We discarded the rest.

We used two sets of features, *candidate* and *context*, resulting in two models:

1. The candidate set contains the Jaccard similarity coefficient between the entity title and the mention, the PageRank weight, and the commonness defined as $P(E_q|M_i)$, where $E$ is the entity and $M$ is the specific mention. All the features in the candidate set are encoded as a quantitized one-hot encoded array.

2. The context set includes the *candidate* features and additional FOFE-encoded left and right contexts surrounding the mention using the same word embeddings as the NER derived from Wikipedia.

We trained the reranker as a binary classifier using a feed-forward neural network with binary cross entropy loss and sigmoid activation. The network consists of 3 dense layers of size 64 for the *candidate* model and 128 for the *context* model.

We incorporated the *candidate* and *context* model into the entity disambiguation using the following equation to produce final ranking score:

$$\text{Final score} = RV \cdot RRS^{\alpha} \quad (3)$$

where $RV$ is rank value and $RRS$ is the rerank score, which is equivalent to the prediction probability.

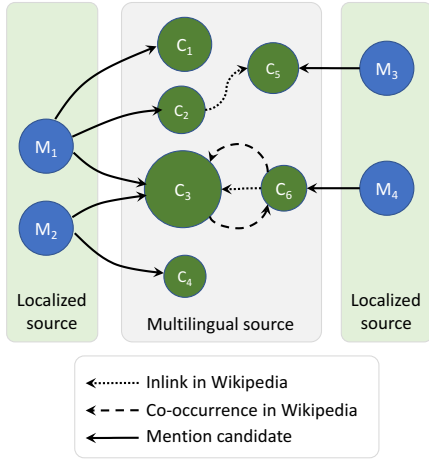



Figure 4: Overview of the entity disambiguation graph

We performed a grid search to find the optimal $\alpha$ for the reranker using the gold standard training set and we selected the best of the two models for each language.

### 6.4 Postprocessing

The postprocessing stage involves consists of the following steps: a baseline coreference resolution, a nominal discovery, and a filtering. The baseline coreferencing method uses the linked mentions as input and tries to find exact matches of the first and last word of each linked mention in the text. When such a match is found, the two mentions are coreferring.

To discover the nominal mentions, we first built a seed word set from the nominal mentions in the TAC data. We then built a dictionary, where the keys were the entities and the values, the nominal phrases compatible with each entity. We extracted these values from Wikidata descriptions that we intersected with the seed word set, the instances-of and occupation relations, with their respective labels and aliases.

Finally, we filtered out the mentions, where we could not find an acceptable class using the entity-to-class mapping dictionary or using the NER pre-

| **Language** | en | es | zh |
|---|---|---|---|
| NERC | $U_2$: 0.833 | $U_2$: 0.804 | $U_1$/$U_3$: 0.760 |
| NEL | $U_4$: 0.751 | $U_2$: 0.751 | $U_4$: 0.718 |
| NELC | $U_4$: 0.726 | $U_2$: 0.733 | $U_1$/$U_3$: 0.760 |
| CEAFm | $U_4$: 0.783 | $U_3$: 0.728 | $U_4$: 0.736 |

Table 4: Best version for the named class. F1 score. Results on the TAC 2017 data

| **Languages** | en | | | | | es | | | | | zh | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $U_1$ | $U_2$ | $U_3$ | $U_4$ | $U_5$ | $U_1$ | $U_2$ | $U_3$ | $U_4$ | $U_5$ | $U_1$ | $U_2$ | $U_3$ | $U_4$ | $U_5$ |
| NERC | 0.813 | **0.833** | 0.825 | 0.813 | 0.797 | 0.802 | **0.804** | 0.788 | 0.749 | 0.763 | **0.760** | 0.750 | **0.760** | 0.750 | 0.741 |
| NEL | 0.732 | 0.738 | 0.730 | **0.751** | 0.691 | 0.747 | **0.751** | 0.746 | 0.701 | 0.687 | 0.716 | 0.691 | 0.716 | **0.718** | 0.690 |
| NELC | 0.711 | 0.717 | 0.710 | **0.726** | 0.668 | 0.784 | **0.788** | 0.788 | 0.745 | 0.765 | **0.757** | 0.755 | **0.757** | 0.746 | 0.741 |
| CEAFm | 0.752 | 0.751 | 0.752 | **0.783** | 0.754 | 0.722 | 0.714 | **0.728** | 0.685 | 0.700 | 0.726 | 0.702 | 0.726 | **0.736** | 0.720 |

Table 5: Breakdown of results for runs $U_1$ to $U_5$, F1 scores, bold indicates the best result per language

dicted class, when available.

## 7 Results

Ugglan was primarily targeting the named entity disambiguation. It was not designed for the nominal and NIL entities, and hence its results on these categories are not at the same level in terms of accuracy. Therefore, we will merely analyze the named results, where the result categories correspond to these acronyms:

**NERC,** Named Entity Recognition and Classification, corresponding to `strong_typed_mention_match` in the evaluation script.

**NEL,** Named Entity Linking (`strong_link_match`);

**NELC,** Named Entity Linking and Classification ( `strong_typed_link_match`);

**CEAFm,** Clustered Mention Identification (`CEAFm`).

We submitted five runs, $U_1$ to $U_5$. Table 4 shows an overview of the best combination per language and type of result taken from the official evaluation data and Table 5 shows the full breakdown.

The pipeline setup for the particular runs were selected using the TAC 2016 evaluation as a guide. The runs $U_1$ to $U_3$ used available training data from TAC 2014-2016, while $U_4$ and $U_5$ only used TAC 2014-2015. Table 6 shows the different configurations.

From the results in Table 4, the typed classification is best using only the FOFE-based NER. The Stanford NER is better when it comes to clustered mentions.

## 8 Discussion

### 8.1 Mention Recognition

Ugglan's ability to find linkable mentions is determined by the recall level of the FST dictionary. The NER only helps in reducing noise, thus increasing precision at the expense of possibly lowering the overall recall. The *hybrid* mode tries to mitigate the recall loss by including mentions which have no overlap with any NER mention. NIL mentions are only found using a NER or if the mention was linked and could not be resolved to a Freebase entity. The FOFE NER was trained to identify NOMs, but these were never used as they could not reliably be linked to existing linkable mentions.

### 8.2 Linking

The windowing approach limits the computation complexity at the expense of a possible lower precision. We did not evaluate the effects of the window size and the values were picked arbitrarily using human insight only. Arguably, the optimal size depends on the impact of topic drift, as the linker performs best with a coherent and compatible context with as many related mentions as possible. The more diverse the context is in terms of mentions and candidates, the noisier the graph

| | Mention Recognition | | | | | | NER | | | Reranker | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NER-only | | | Hybrid | | | | | | Candidate | | | Context | | | None | | |
| | en | es | zh | en | es | zh | en | es | zh | en | es | zh | en | es | zh | en | es | zh |
| $U_1$ | ✓ | ✓ | | | | ✓ | F | F | F | | | | ✓ | ✓ | ✓ | | | |
| $U_2$ | ✓ | ✓ | ✓ | | | | F | F | F | | | | ✓ | ✓ | ✓ | | | |
| $U_3$ | | | | ✓ | ✓ | ✓ | F | F | F | ✓ | | | | ✓ | ✓ | | | |
| $U_4$ | | | | ✓ | ✓ | ✓ | S | S | F* | ✓ | | | | ✓ | ✓ | | | |
| $U_5$ | | | | ✓ | ✓ | ✓ | S | F* | S | | | | | | | ✓ | ✓ | ✓ |

Table 6: System configuration, where F stands for FOFE and S for Stanford NER. F* is an older FOFE model

becomes and the relevant context may shrink as it would require a bigger context to get sufficient supporting candidates to produce a good linkage.

## Acknowledgments

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

## References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks* 30(1–7):107–117. Proceedings of WWW7.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*. Ann Arbor, pages 363–370.

Heng Ji and Joel Nothman. 2016. Overview of TAC-KBP2016 trilingual entity discovery and linking and its impact on end-to-end cold-start kbp. In *Proceedings of the Ninth Text Analysis Conference (TAC2016)*.

Heng Ji, Joel Nothman, and Ben Hachey. 2014. Overview of TAC-KBP2014 trilingual entity discovery and linking. In *Proceedings of the Seventh Text Analysis Conference (TAC2014)*.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP2015 trilingual entity discovery and linking. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Marcus Klang and Pierre Nugues. 2017. Docforia: A multilayer document model. In *Proceedings of the 21st Nordic Conference of Computational Linguistics*. Gothenburg, page 226–230.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 746–751. http://www.aclweb.org/anthology/N13-1090.

Anton Södergren and Pierre Nugues. 2017. A multilingual entity linker using PageRank and semantic graphs. In *Proceedings of the 21st Nordic Conference of Computational Linguistics*. Gothenburg, page 87–95.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1237–1247.

ShiLiang Zhang, Hui Jiang, MingBin Xu, JunFeng Hou, and LiRong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 495–500. http://www.aclweb.org/anthology/P15-2081.

# Linking, Searching, and Visualizing Entities in Wikipedia

PAPER VII

# Linking, Searching, and Visualizing Entities in Wikipedia

**Marcus Klang, Pierre Nugues**
Lund University, Department of Computer science,
Lund, Sweden
marcus.klang@cs.lth.se, pierre.nugues@cs.lth.se

**Abstract**

In this paper, we describe a new system to extract, index, search, and visualize entities in Wikipedia. To carry out the entity extraction, we designed a high-performance, multilingual, entity linker and we used a document model to store the resulting linguistic annotations. The entity linker, HEDWIG, extracts the mentions from text using a string matching engine and links them to entities with a combination of statistical rules and PageRank. The document model, Docforia (Klang and Nugues, 2017), consists of layers, where each layer is a sequence of ranges describing a specific annotation, here the entities. We evaluated HEDWIG with the TAC 2016 data and protocol (Ji and Nothman, 2016) and we reached the CEAFm scores of 70.0 on English, on 64.4 on Chinese, and 66.5 on Spanish.

We applied the entity linker to the whole collection of English and Swedish articles of Wikipedia and we used Lucene to index the layers and a search module to interactively retrieve all the concordances of an entity in Wikipedia. The user can select and visualize the concordances in the articles or paragraphs. Contrary to classic text indexing, this system does not use strings to identify the entities but unique identifiers from Wikidata. A demonstration of the entity search and visualization will be available for English at this address http://vilde.cs.lth.se:9001/en-hedwig/ and for Swedish at: http://vilde.cs.lth.se:9001/sv-hedwig/.

**Keywords:** named entity recognition, entity linker, wikipedia

## 1. Introduction

Wikipedia has become a popular NLP resource used in many projects such as text categorization (Wang et al., 2009), information extraction, question answering (Ferrucci, 2012), or translation (Smith et al., 2010). In addition to its size and diversity, Wikipedia, through its links, also enables to create a graph that associates concepts, entities, and their mentions in text. Wu and Weld (2010), for instance, used the "wikilinks", the Wikipedia hyperlinks, to collect the mentions of an entity and build sets of synonyms for an open information extraction system.

However, according to the edition rules of Wikipedia, only the first mention of an entity should be linked in an article. An automatic wikification is then necessary to associate the subsequent mentions with an entity (Mihalcea and Csomai, 2007). In addition, searching entities using names in the form of strings can be tricky as names are sometimes ambiguous and entities may have more than one name. Finding all the occurrences of an organization like the United Nations would require five or more queries as they can be mentioned not only as *the United Nations*, but also as: *UN*, *U.N.*, *United Nations Organization*, *UNO*, etc.

In this paper, we describe a novel multilingual system to process, index, search, and visualize all the mentions of an entity in Wikipedia. This system consists of an entity linker, HEDWIG, that extracts the mentions from text using a named-entity recognition engine and links them to entities with a combination of statistical rules and PageRank. We applied HEDWIG to the whole collection of English and Swedish articles of Wikipedia. We then used Lucene to index the layers and a search module to interactively retrieve all the concordances of an entity in the articles, paragraphs and metadata. The user can then select a concordance s/he wants to visualize. As opposed to the Wikipedia index, the system uses unique identifiers to index the entities and not their mentions, which enables the users to carry out more easily exhaustive searches.

## 2. Previous Work

Most named entity linkers adopt a two-step procedure, where they first identify the mentions and then link them to a unique identifier.

### 2.1. Mention Detection

The mention detection step, or spotting, has been addressed by a variety of techniques. Mihalcea and Csomai (2007) used a dictionary associating the entities with their surface forms, where the surface forms are simply n-grams. They extracted all the strings in a text that matched any of the surface forms in the dictionary to produce the set of mention candidates. As the candidates may overlap, the authors ranked them using a *keyphraseness* metric defined as the number of documents, where the mention was linked divided by the number of documents, where the mention occurred. They set the number of mentions to keep to 6% of the total number of words in the document following figures they observed in Wikipedia.

Milne and Witten (2008) also used a dictionary of surface forms as well a classifier to decide on the mentions to keep. They trained the classifier on Wikipedia mentions, either linked, the positive examples, or nonlinked, the negative ones. As features, they used the link probability (keyphraseness), relatedness, disambiguation confidence, generality, location, and spread.

Lipczak et al. (2014) used the Lucene's finite state transducers and Solr Text Tagger to detect the mentions. They collected the surface form dictionary from Wikipedia as well as Freebase and Google's wikilinks. The tagging step results in an overdetection that is pruned using lexical filters. The final selection of mentions is carried out in the linking step.

Cucerzan (2014) used a dictionary of surface forms collected from Wikipedia, anchor text, page titles, redirection pages, etc, and a set of rules to identify the mentions in the text. As in Lipczak et al. (2014), the overgeneration is

solved at the linking stage.

Piccinno and Ferragina (2014) used a dictionary of surface forms similar to Cucerzan (2014) to spot the mentions. They also used a pruner to discard unlikely annotations based on a classifier and a coherence metric with the set of neighboring entities. This final selection is done at linking time.

Sil et al. (2015) used classifiers based on neural nets and conditional random fields trained on three languages.

Some annotators also used an external named entity recognition module to carry out this mention detection as AIDA (Hoffart et al., 2011) and Tan et al. (2015) that used Stanford NER (Finkel et al., 2005).

### 2.2. Entity Linking

Bagga and Baldwin (1998) is one of the earliest works that introduced the notion of linkage to unique things through the task of cross-document coreference. The main difference with entity linking is that predefined lists of entities do not exist but have to be found. Bagga and Baldwin (1998) created summary vectors and tried to cluster them to form linkages. These summary vectors were created from noun phrases contained within coreference chains in documents. Using cosine similarity with a predefined threshold, they were able to cluster coreferences that crossed the document boundaries.

Bunescu and Pasca (2006) first explored entity linking using Wikipedia as knowledge base. They used hyperlinks, redirects, disambiguation pages, and the category hierarchy, which would be used by almost every major paper since. Using context article similarity based on 55-word window vector space model (VSM) cosine similarity and a taxonomy kernel, they trained SVM models to recast the disambiguation problem as a classification. They reported accuracies between 55.4% and 84.8% depending on which model they used.

Cucerzan (2007a) introduced clearly defined end-to-end pipelines – starting with text and ending with linked entities – as well as a notion of collective agreement in the disambiguation component. Using a *document vector* comprised of surface form context, entity context, and categories, he could maximize an agreement between the proposed entity candidates. Using the top two stories from 10 MSNBC news categories in January 2, 2007, he reported an accuracy of 91.4% versus 88.3% from 350 random Wikipedia pages.

Milne and Witten (2008) introduced important concepts such as relatedness and commonness which still defines a strong baseline used by many following papers in one form or another.

Hoffart et al. (2011) used an ensemble system to compute a linear combination of entity probabilities, context similarities, and entity coherences, where the *popularity prior* corresponds to the number of in-links to a Wikipedia entity; the *context similarity* compares the context of the input by computing a similarity between all the tokens in the input against a key phrase defined for entities they extracted from YAGO. A key phrase is a phrase that is derived from link texts, category names, citation titles, and other references; finally, *coherence* provides a way of comparing different entity candidates in a text in order to measure how compatible they are.

Lipczak et al. (2014) built a set of all the entity candidates for all the mentions in a document. They started from an entity core corresponding to the default senses. Using this core, they built a topic centroid from Wikipedia categories and discard entities from the core that are outside the topic. They finally refined the core and rank the remaining entities using a cosine similarity.

Eckhardt et al. (2014) built a graph of entity-mention pairs, where they weighted the edges with $P(E|M)$ probabilities. They applied a variant to PageRank to rank the entities.

Sil et al. (2015) described a trilingual system that uses a classifier with features such as the number of mention–entity matches in Wikipedia, acronym match, pointwise mutual information between entities and categories, etc.

Tan et al. (2015) used a graph of entity-mention and entity-entity edges. The edges are weighted by a function of the context similarity between a mention and an entity description in Freebase and functions of relatedness and context similarities. The entity ranking is eventually determined by a random walk in the graph.

Cucerzan (2007b) and Han and Zhao (2009) described other algorithms for NERL. In contrast to most of these previous works, multilingual support is at the core of HEDWIG.

## 3. Extraction of the Wikipedia Structure

Before we apply the linker to Wikipedia, we convert the HTML pages into a multilayer document model; see Sect. 5. This preprocessing step parses the HTML documents into DOM trees and extracts the original page structure, text styles, links, lists, and tables. We then resolve all the Wikipedia links to unique Wikidata identifiers, where Wikidata is an entity database, which assigns unique identifiers across all the language editions of Wikipedia.

The United Nations, for instance, has the unique id: Q1065, which enables to retrieve the article pages in English, French, Swedish, or Russian. Figure 1 shows examples of these ids in the *United Nations* article from the English Wikipedia, where we have replaced the manually set Wikipedia anchors (the wikilinks) with their Wikidata numbers: Q245065 for *intergovernmental organization* and Q60 for *New York City*. Figure 2 shows the first paragraph of the corresponding article from the Swedish Wikipedia, *Förenta nationerna* 'United Nations', where *mellanstatlig organisation*, the Swedish word for intergovernmental organization, has also the Q245065 number.

## 4. Entity Linking

Once we have collected and structured the text, we apply the entity linking module to find all the mentions of an entity in text and link these mentions to a unique identifier.

### 4.1. Set of Entities

We used the wikilinks to build a repository of (mention, entity) pairs and Wikidata as the nomenclature for the unique entity identifiers. We collected all the wikilinks in the Wikipedia articles, where each link consists of a label and the name of the destination page:

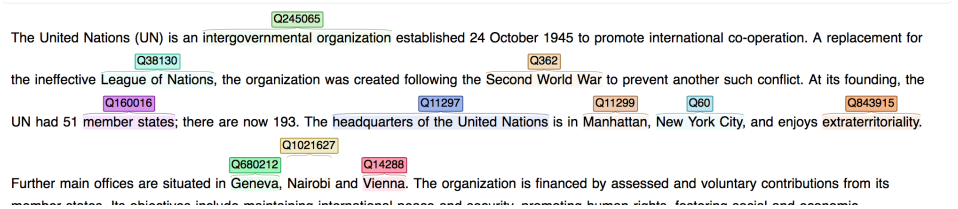



Figure 1: Visualization of anchors with Wikidata Q-numbers. The first lines of the *United Nations* article in the English Wikipedia

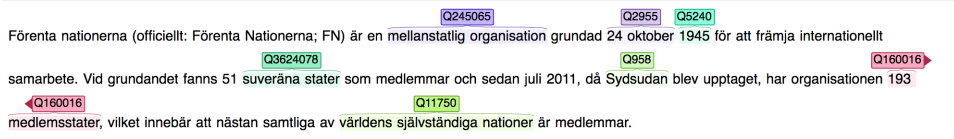



Figure 2: Visualization of anchors with Wikidata Q-numbers. First paragraph of the *Förenta nationerna* 'United Nations' article in the Swedish Wikipedia

`[[destination|label]]`. We parsed these links into (mention, entity page) pairs and we translated the entity pages into Wikidata Q-numbers.

We annotated each mention-entity pair with a set of properties: its frequency, its frequency relative to the mention, $P(E|M)$, if the mention is in a dictionary, if the mention consists of stop words. We then pruned the knowledge base from unique mentions for entities with a high frequency, mentions that are only stop words, etc.

During the mention gathering, we also derived statistics for a given language. Before we computed these statistics, we applied a procedure that we called *autolinking*. In an article, the Wikipedia guidelines advise to link only one instance of an entity mention[1]: Normally the first one in the text. With the autolinking procedure, we link all the remaining mentions provided that we have sequences of exactly matching tokens.

The statistics we collect are:

- The frequency of the mention string over the whole Wikipedia collection (restricted to one language);
- The frequency of the pair (mention, entity) that we derive from the links without autolinking (only manually linked mentions);
- The count of (entity1, entity2) pairs in a window corresponding to a paragraph and limited to 20 linked mentions. This is carried out after autolinking;
- Capitalization statistics for all the tokens: We extract token counts for all tokens with a frequency greater than 100 and we break them down by case properties: uppercased, lowercased, titlecased, and camelcase;

[1] `https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking#Overlinking_and_underlinking`

### 4.2. Mention Recognition

To detect the mentions in an unannotated text, we use a two-step procedure: We first generate the mention candidates using a finite-state transducer; this results in a very large overgeneration. We then apply a mention segmenter that identifies the mentions to keep for the linking phase.

Following Lipczak et al. (2014) and Södergren and Nugues (2017), we used an automaton to spot the mentions. This automaton uses Lucene's finite-state transducers and is efficient in terms of memory usage and execution time. Depending on the language and the availability of manually-annotated data, we can complement this candidate generation with two named-entity recognition systems trained on the annotated data: The first one being based on an extension of the fixed-size ordinally forgetting encoding (FOFE) technique (Xu et al., 2017; Zhang et al., 2015) and the second one being CoreNLP (Manning et al., 2014).

The overgeneration of mention candidates impairs the quality of the downstream linker. To discard the very unlikely ones, we introduced rules based on the frequency of the manual links applied to mention $M$ and its link probability $lp$. We denote $M_{linked}$ a mention with a manual hyperlink; this would correspond to the wiki markup: `[[link|mention]]`, and $M_{autolinked}$, an autolinked mention. We define:

$$\begin{aligned} lp(M) &= P(M_{autolinked}|M) \\ &= \frac{\#M_{autolinked}}{\#M_{autolinked} + \#M_{unlinked}}, \end{aligned}$$

where $\#M_{autolinked}$ is the number of times a mention is linked in the Wikipedia collection and $\#M_{unlinked}$, its frequency when unlinked.

The rules are:

1. Remove the mentions $M$ where $lp(M) < D_{lp}$, for instance with $D_{lp} = 0.01$;

2. Keep the mentions where $lp(M) > K_{lp}$, and $\#M_{linked} > K_f$, with, for instance, $K_{lp} = 0.15$ and $K_f = 25$. All these mentions are candidates for the linking step;

3. Set the rest in a dubious set.

### 4.3. The Linking Step

We applied the JUNG implementation of PageRank (Brin and Page, 1998; O'Madadhain et al., 2003) to the tagged mentions. Following Eckhardt et al. (2014) and Södergren and Nugues (2017), we created a node for every mention-entity pair that is detected in the text and we ran PageRank on this graph; we used the JUNG default settings.

We analyzed the internal links of Wikipedia to determine the entities that appear in the same context. Two entities are linked if the article of Entity $A$ links to the article of Entity $B$ or there exist at least one link to the article of Entity $A$ and another one to the article of Entity $B$ occurring in the same paragraph.

We then re-ranked the PageRank candidates using a feed forward neural network consisting of three layers with RELU activations, a crossentropy loss, and a sigmoid output. We trained the model on the output of the PageRank disambiguator applied to the TAC 2015 dataset. The features we used consist of the mention tokens, candidate title tokens, both as word embeddings on 256 dimensions, the Jaccard distance between the mention and candidate title, the commonness and pagerank weights.

We evaluated the system with the same method as used in the TAC 2016 competition (Ji and Nothman, 2016) and we reached the `CEAFm` scores of 70.0 on English, on 64.4 on Chinese, and 66.5 on Spanish. We applied our linker to Swedish without any language adaptation.

We deployed the entity linker on our cluster and we used HDFS to spread the Wikipedia dump across the nodes as well as to save the final result.

## 5. The Document Model

We represented Wikipedia and the entity annotations using the Docforia document model[2] (Klang and Nugues, 2016b; Klang and Nugues, 2016a; Klang and Nugues, 2017). Docforia is designed it so that we can store the original markup, as well as any subsequent linguistic annotation. It consists of multiple layers, where each layer is dedicated to a specific type of annotation.

The annotations are encoded in the form of graph nodes, where a node represents a piece of data: a token, a sentence, a named entity, etc., delimited by ranges. These nodes are possibly connected by edges as in dependency graphs. The data structure used is similar to a property graph.

## 6. Indexing

We created an indexing tool, Panforia, to retrieve the entities from the annotated documents. As input, Panforia uses the output of the entity annotation in the form of Parquet files. Panforia is based on the Lucene search and indexing library. Each Docforia record is converted into a Lucene document by mapping record properties and documents to Lucene fields. In addition, a binary copy of the Docforia record is embedded with the indexed fields, which provides the ranges and relationships between nodes needed for the visualization.

Building directly on the Lucene library, instead of existing packages such as Solr or ElasticSearch, allowed us to optimize the insertion into an index. One key advantage of the Panforia indexer is that it can read the output from the Wikipedia pipeline, Parquet files, without a conversion step.

## 7. Visualization

The front-end of Panforia is a web server that embeds the Docforia library, Lucene, and a client-side web application. To search an entity, we enter a Wikidata Q-number, for instance, `urn:wikidata:Q168756`, corresponding to the entity identifier, here the University of California, Berkeley. Figure 3 shows the results of this search, where in each row, the entity is listed by its mention together with its left and right contexts. The document that contains the source of the concordance is listed in the leftmost column and the offset from the beginning in the last column.

In the figure, we can see that the entity has many possible mentions: *University of California, Berkeley*, *Berkeley*, *UC Berkeley*, etc. All these mentions and concordances are automatically retrieved through the entity index. We can visualize the document by clicking on a link in the left column.

For each document, the interactive visualization tool also enables the user to examine the annotated layers resulting from the HTML parsing (Sect. 3.). These layers include the manually set anchors, the automatically detected entities, and text enrichment. These layers are selectable from the dropdown menu to the right. Figure 4 shows an example with the automatically linked entities, the text in bold (strong) and in italics.

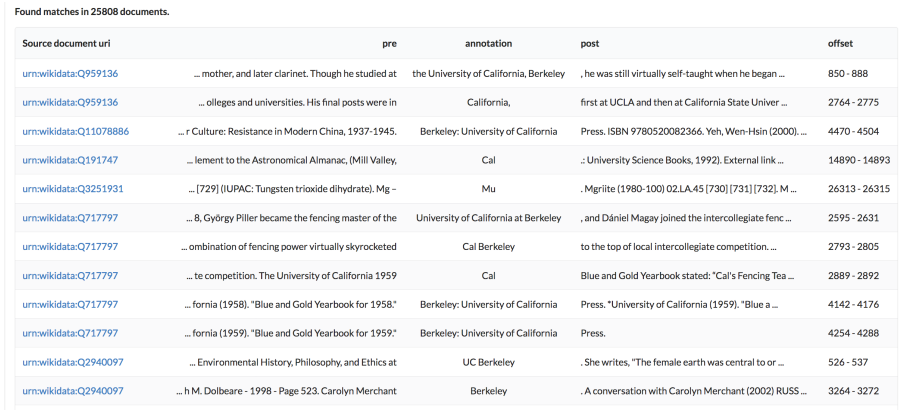
Figure 5 shows an example of results we obtained in the Swedish Wikipedia when we searched the entity Göran Persson, the former Swedish Prime Minister, using his Q-number: Q53747. This mention, *Göran Persson*, is ambiguous and Wikipedia lists as many as four different entities with this name: The former Swedish Prime Minister, a progressive musician (Q6042900), a Swedish social democratic politician, former member of the Riksdag (Q5626648), and a Swedish statesman from the 16th century (Q2625684). The latter is also spelled Jöran Person.

Searching the mention *Göran Persson* would return articles or concordances with any of these entities, while searching the entity through its Q-number only returns the intended person, either with her/his name or with other mentions such as *Persson* or *Göran*. The results are given in the forms of concordances with left and right contexts (Fig. 5).

## 8. Conclusion and Future Work

We have described a system to extract, index, search, and visualize entities on the English and Swedish Wikipedia. Given a Wikidata Q-number, a user can interactively retrieve all the concordances of an entity in the articles, para-

[2] `https://github.com/marcusklang/docforia/`



Found matches in 25808 documents.

| Source document uri | pre | annotation | post | offset |
|---|---|---|---|---|
| urn:wikidata:Q959136 | ... mother, and later clarinet. Though he studied at | the University of California, Berkeley | , he was still virtually self-taught when he began ... | 850 - 888 |
| urn:wikidata:Q959136 | ... olleges and universities. His final posts were in | California, | first at UCLA and then at California State Univer ... | 2764 - 2775 |
| urn:wikidata:Q11078886 | ... r Culture: Resistance in Modern China, 1937-1945. | Berkeley: University of California | Press. ISBN 9780520082366. Yeh, Wen-Hsin (2000). ... | 4470 - 4504 |
| urn:wikidata:Q191747 | ... lement to the Astronomical Almanac, (Mill Valley, | Cal | .: University Science Books, 1992). External link ... | 14890 - 14893 |
| urn:wikidata:Q3251931 | ... [729] (IUPAC: Tungsten trioxide dihydrate). Mg – | Mu | . Mgriite (1980-100) 02.LA.45 [730] [731] [732]. M ... | 26313 - 26315 |
| urn:wikidata:Q717797 | ... 8, György Piller became the fencing master of the | University of California at Berkeley | , and Dániel Magay joined the intercollegiate fenc ... | 2595 - 2631 |
| urn:wikidata:Q717797 | ... ombination of fencing power virtually skyrocketed | Cal Berkeley | to the top of local intercollegiate competition. ... | 2793 - 2805 |
| urn:wikidata:Q717797 | ... te competition. The University of California 1959 | Cal | Blue and Gold Yearbook stated: "Cal's Fencing Tea ... | 2889 - 2892 |
| urn:wikidata:Q717797 | ... fornia (1958). "Blue and Gold Yearbook for 1958." | Berkeley: University of California | Press. "University of California (1959). "Blue a ... | 4142 - 4176 |
| urn:wikidata:Q717797 | ... fornia (1959). "Blue and Gold Yearbook for 1959." | Berkeley: University of California | Press. | 4254 - 4288 |
| urn:wikidata:Q2940097 | ... Environmental History, Philosophy, and Ethics at | UC Berkeley | . She writes, "The female earth was central to or ... | 526 - 537 |
| urn:wikidata:Q2940097 | ... h M. Dolbeare - 1998 - Page 523. Carolyn Merchant | Berkeley | . A conversation with Carolyn Merchant (2002) RUSS ... | 3264 - 3272 |

Figure 3: Searching an entity in the Wikipedia pages, where Q168756 is the Wikidata identifier of the University of California, Berkeley. The entity concordances, where each concordance is listed with its source, mention in the text, left and right contexts, and position in the text

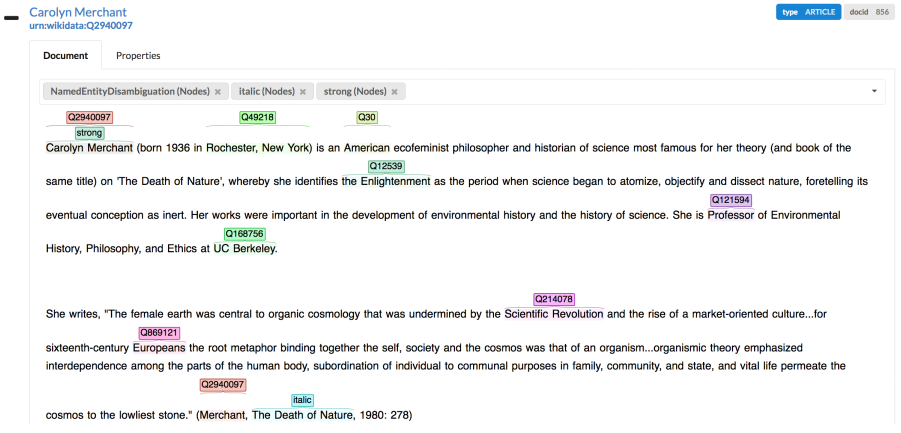
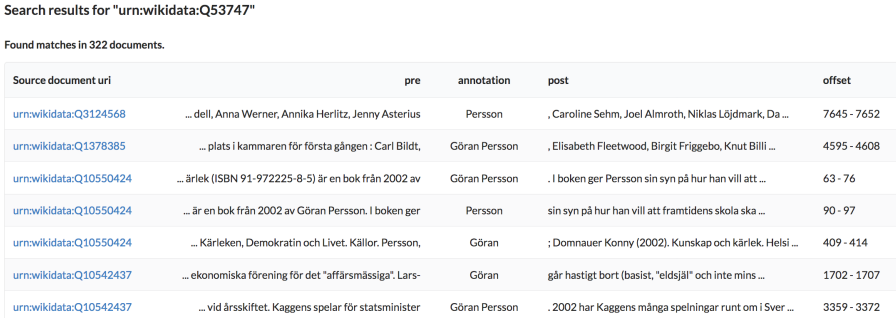



Figure 4: Visualization of annotated layers: The automatically linked entities, text in bold and in italics



**Search results for "urn:wikidata:Q53747"**

Found matches in 322 documents.

| Source document uri | pre | annotation | post | offset |
|---|---|---|---|---|
| urn:wikidata:Q3124568 | ... dell, Anna Werner, Annika Herlitz, Jenny Asterius | Persson | , Caroline Sehm, Joel Almroth, Niklas Löjdmark, Da ... | 7645 - 7652 |
| urn:wikidata:Q1378385 | ... plats i kammaren för första gången : Carl Bildt, | Göran Persson | , Elisabeth Fleetwood, Birgit Friggebo, Knut Billi ... | 4595 - 4608 |
| urn:wikidata:Q10550424 | ... årlek (ISBN 91-972225-8-5) är en bok från 2002 av | Göran Persson | . I boken ger Persson sin syn på hur han vill att ... | 63 - 76 |
| urn:wikidata:Q10550424 | ... är en bok från 2002 av Göran Persson. I boken ger | Persson | sin syn på hur han vill att framtidens skola ska ... | 90 - 97 |
| urn:wikidata:Q10550424 | ... Kärleken, Demokratin och Livet. Källor. Persson, | Göran | ; Domnauer Konny (2002). Kunskap och kärlek. Helsi ... | 409 - 414 |
| urn:wikidata:Q10542437 | ... ekonomiska förening för det "affärsmässiga". Lars- | Göran | går hastigt bort (basist, "eldsjäl" och inte mins ... | 1702 - 1707 |
| urn:wikidata:Q10542437 | ... vid årsskiftet. Kaggens spelar för statsminister | Göran Persson | . 2002 har Kaggens många spelningar runt om i Sver ... | 3359 - 3372 |

Figure 5: Concordances of the entity *Göran Person*, Q53747. The results are given in the form of concordances with a left and right contexts.

graphs, and metadata. The user can then select a concordance and the annotations s/he wants to visualize.

This system could be improved in many ways. The entity linker makes no assumption on the language and could easily be applied to other Wikipedias. We plan to extend this demonstration to four other languages: French, German, Spanish, and Russian and for one entity, show the concordances in the six languages.

Finally, we plan to introduce a coreference resolution for the languages where a coreference-annotated corpus exists or where a solver is available.

The demonstrations will be available at: `http://vilde.cs.lth.se:9001/en-hedwig/` for English and `http://vilde.cs.lth.se:9001/sv-hedwig/` for Swedish.

## 9. Acknowledgements

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

## 10. Bibliographical References

Bagga, A. and Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, April.

Bunescu, R. C. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *European Chapter of the Association for Computational Linguistics*, volume 6, pages 9–16.

Cucerzan, S. (2007a). Large-scale named entity disambiguation based on wikipedia data. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, volume 7, pages 708–716.

Cucerzan, S. (2007b). Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 708–716, Prague, June. Association for Computational Linguistics.

Cucerzan, S. (2014). Name entities made obvious: The participation in the ERD 2014 evaluation. In *Proceedings of Entity Recognition and Disambiguation, ERD'14*, Gold Coast.

Eckhardt, A., Hreško, J., Procházka, J., and Smrž, O. (2014). Entity linking based on the co-occurrence graph and entity probability. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, pages 37–44.

Ferrucci, D. A. (2012). Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1:1 –1:15, May-June.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370, Ann Arbor.

Han, X. and Zhao, J. (2009). Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 215 – 224.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh.

Ji, H. and Nothman, J. (2016). Overview of TAC-KBP2016 Tri-lingual EDL and Its Impact on End-to-End KBP. In *Proceedings of the Ninth Text Analysis Conference (TAC 2016)*, Gaithersburg, Maryland. National Institute of Standards and Technology.

Klang, M. and Nugues, P. (2016a). Docforia: A multilayer document model. In *Proceedings of SLTC 2016*, Umeå, November.

Klang, M. and Nugues, P. (2016b). WIKIPARQ: A tabulated Wikipedia resource using the Parquet format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4141–4148, Portorož, Slovenia, may.

Klang, M. and Nugues, P. (2017). Docforia: A multilayer document model. In *Proceedings of the 21st Nordic Conference of Computational Linguistics*, page 226–230, Gothenburg, May.

Lipczak, M., Koushkestani, A., and Milios, E. (2014). Tulip: Lightweight entity recognition and disambiguation using wikipedia-based topic centroids. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, pages 31–36.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Mihalcea, R. and Csomai, A. (2007). Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on CIKM*, CIKM '07, pages 233–242, Lisbon, Portugal.

Milne, D. and Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518.

O'Madadhain, J., Fisher, D., White, S., and Boey, Y.-B. (2003). The JUNG (Java Universal Network/Graph) framework. Technical Report UCI-ICS 03-17, School of Information and Computer Science, University of California, Irvine.

Piccinno, F. and Ferragina, P. (2014). From tagme to wat: A new entity annotator. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, pages 55–62, New York, NY, USA. ACM.

Sil, A., Dinu, G., and Florian, R. (2015). The IBM system for trilingual entity discovery and linking at TAC 2015. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Smith, J. R., Quirk, C., and Toutanova, K. (2010). Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 403–411.

Södergren, A. and Nugues, P. (2017). A multilingual entity linker using PageRank and semantic graphs. In *Proceedings of the 21st Nordic Conference of Computational Linguistics*, page 87–95, Gothenburg, May.

Tan, Y., Zheng, D., Li, M., and Wang, X. (2015). BUPT-Team participation at TAC 2015 knowledge base population. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Wang, P., Hu, J., Zeng, H.-J., and Chen, Z. (2009). Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 19(3):265–281.

Wu, F. and Weld, D. S. (2010). Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the ACL*, page 118–127, Uppsala, Sweden.

Xu, M., Jiang, H., and Watcharawittayakul, S. (2017). A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247, Vancouver, Canada, July. Association for Computational Linguistics.

Zhang, S., Jiang, H., Xu, M., Hou, J., and Dai, L. (2015). The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 495–500, Beijing, China, July. Association for Computational Linguistics.

# Comparing LSTM and FOFE-based Architectures for Named Entity Recognition

# Comparing LSTM and FOFE-based Architectures for Named Entity Recognition

**Marcus Klang, Pierre Nugues**

Department of Computer Science
Lund University, S-221 00 Lund
Marcus.Klang@cs.lth.se, Pierre.Nugues@cs.lth.se

**Abstract**

LSTM architectures (Hochreiter and Schmidhuber, 1997) have become standard to recognize named entities (NER) in text (Lample et al., 2016; Chiu and Nichols, 2016). Nonetheless, Zhang et al. (2015) recently proposed an approach based on *fixed-size ordinally forgetting encoding* (FOFE) to translate variable-length contexts into fixed-length features. This encoding method can be used with feed-forward neural networks and, despite its simplicity, reach accuracy rates matching those of LTSMs in NER tasks (Xu et al., 2017). However, the figures reported in the NER articles are difficult to compare precisely as the experiments often use external resources such as gazetteers and corpora. In this paper, we describe an experimental setup, where we reimplemented the two core algorithms, to level the differences in initial conditions. This allowed us to measure more precisely the accuracy of both architectures and to report what we believe are unbiased results on English and Swedish datasets.

## 1. Introduction

Named entity recognition (NER) aims at identifying all the names of persons, organizations, geographic locations, as well as numeric expressions in a text. This is a relatively old task of NLP that has applications in multiples fields such as information extraction, knowledge extraction, product recommendation, and question answering. Named entity recognition is also usually the first step of named entity linking, where the mentions of named entities, once recognized, are disambiguated and linked to unique identifiers (Ji and Nothman, 2016; Ji et al., 2017).

Over the time, NER has used scores of techniques starting from hand-written rules, to decision trees, support vector machines, logistic regression, and now deep neural networks. The diversity of applications and datasets makes it difficult to compare the algorithms and systems. Researchers in the field quickly realized it and the committee of the message understanding conferences (MUC) first defined procedures for a systematic evaluation of NER performance (Grishman and Sundheim, 1996). The CoNLL 2002 and 2003 conferences (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) further developed them and provided standardized annotations, multilingual datasets, and evaluation scripts, that are still references today.

In spite of a continuous research, designing a perfect domain-independent NER is still an unmet goal. New ideas and architectures make that the state-of-the-art is improving every year. However, the figures reported in the NER articles are difficult to compare precisely as the experiments often involve external resources such as gazetteers and non-published corpora.

In this paper, we describe an experimental setup, where we reimplemented two of the best reported algorithms and where we defined identical initial conditions. This allowed us to measure more precisely the accuracy of both architectures and to report what we believe are unbiased results on English and Swedish datasets.

## 2. Previous Work

NER has been addressed by many techniques. Participants in the MUC conferences, such as FASTUS, used extensively gazetteers and regular expressions to extract the mentions (Appelt et al., 1993). The CoNLL conferences started to distribute annotated corpora that enabled participants to train classifiers such as logistic regression, decision trees, perceptrons, often organized as ensembles. For a review of early systems from 1991 to 2006, see Nadeau and Sekine (2007).

With the advent of deep learning, long short-term memory architectures (LSTM) (Hochreiter and Schmidhuber, 1997) have become standard to recognize named entities. Out of the 24 teams participating in the trilingual entity disambiguation and linking task (EDL) of TAC 2017, 7 used bidirectional LSTMs – with varying degrees of success (Ji et al., 2017).

Chiu and Nichols (2016) reported a score of 91.62 on the CoNLL 2003 test set with LSTM and convolutional neural networks (CNN) on character embeddings using the development set and the training set to build their model; Lample et al. (2016) used LSTM and conditional random fields (CRF) and reached 90.94 on the same test set; Ma and Hovy (2016) combined LSTM, CNN, and CRF and obtained 91.21.

Parallel to the LSTM achievements, Zhang et al. (2015) recently proposed an approach based on *fixed-size ordinally forgetting encoding* (FOFE) to translate variable-length contexts into fixed-length features. This encoding method can be used with feed-forward neural networks and, despite its simplicity, reach accuracy rates matching those of LTSMs in NER tasks (Xu et al., 2017).

All the reported performance figures are now close and may be subject to initialization conditions of random seeds. See Reimers and Gurevych (2017) for a discussion on their validity. In addition, all the experiments are carried out on the same data sets, again and again, which may, in the long run, entail some data leaks.

In this paper, we report experiments we have done with reimplementations of two of the most accurate NER taggers on English, to be sure we could reproduce the figures and that we applied to the Swedish Stockholm-Umeå corpus (SUC) (Ejerhed et al., 1992).

## 3. Datasets and Annotations

**Annotated datasets.** As datasets, we used the English corpus of CoNLL 2003, OntoNotes, and SUC, that bracket the named entities with semantic categories such as location, person, organization, etc. The corpora use either IOB v1 or v2 as annotation tagsets. We converted the annotation to IOBES, where S is for single-tag named entities, B, for begin, E, for end, I, for inside, and O for outside. For the bracketed example from CoNLL:

> Promising 10th-ranked [$_{MISC}$ American $_{MISC}$] [$_{PER}$ Chanda Rubin $_{PER}$] has pulled out of the [$_{MISC}$ U.S. Open Tennis Championships $_{MISC}$] with a wrist injury, tournament officials announced.

the annotation yields:

> Promising/O 10th-ranked/O American/S-MISC Chanda/B-PER Rubin/E-PER has/O pulled/O out/O of/O the/O U.S./B-MISC Open/I-MISC Tennis/I-MISC Championships/E-MISC with/O a/O wrist/O injury/O ,/O tournament/O officials/O announced/O ./O

The CoNLL 2003 dataset is derived from the Reuters corpus (RCV).

**Word embeddings.** For English, we used the pre-trained Glove 6B embeddings (Pennington et al., 2014) and the lower-cased 100 to 300 dimension variants. In addition, we trained our own cased and lowercased embeddings using the Word2vec algorithm provided by the Gensim library (Řehůřek and Sojka, 2010). For Swedish, we used Swectors (Fallgren et al., 2016) and we trained Swedish embeddings from the Swedish Culturomics Gigaword Corpus (Eide et al., 2016).

## 4. Systems

We implemented two systems: one based on FOFE, which is an extension to that of Klang et al. (2017) and Dib (2018) and the second one on LSTM, taking up the work of Chiu and Nichols (2016).

### 4.1 FOFE

The FOFE model can be seen as a weighted bag-of-words (BoW). Following the notation of Xu et al. (2017), given a vocabulary $V$, where each word is encoded with a one-hot encoded vector and $S = w_1, w_2, w_3, ..., w_n$, an arbitrary sequence of words, where $e_n$ is the one-hot encoded vector of the $n$th word in $S$, the encoding of each partial sequence $z_n$ is defined as:

$$z_n = \begin{cases} 0, & \text{if n = 0} \\ \alpha \cdot z_{n-1} + e_n, & \text{otherwise,} \end{cases} \quad (1)$$

where the $\alpha$ constant is a weight/forgetting factor which is picked such as $0 \leq \alpha < 1$. The result of the encoding is a vector of dimension $|V|$, whatever the size of the segment.

**Features.** The neural network uses both word and character-level features. The word features extend over parts of the sentence, while character features are only applied to the focus words: The candidates for a potential entity.

**Word-level Features.** The word-level features use bags of words to represent the focus words and FOFE to model the focus words as well as their left and right contexts. As context, we used all the surrounding words up to a maximum distance. The beginning and end of sentence are explicitly modeled with `BOS` and `EOS` tokens, which have been added to the vocabulary list.

Each word feature is used twice, both in raw text and normalized lower-case text. The FOFE features are used twice, both with and without the focus words. For the FOFE-encoded features, we used $\alpha = 0.5$. The complete list of features is then the following:

- Bag of words of the focus words;
- FOFE of the sentence: starting from the left, excluding the focus words; starting from the left, including the focus words; starting from the right, excluding the focus words; and starting from the right, including the focus words.

This means that, in total, the system input consists of 10 different feature vectors, where five are generated from the raw text, and five generated from the lowercase text.

**Character-Level Features.** The character-level features only model the focus words from left to right and right to left. We used two different types of character features: One that models each character and one that only models the first character of each word. We applied the FOFE encoding again as it enabled us to weight the characters and model their order. For these features, we used $\alpha = 0.8$. Higher choice of alpha for character features matches the original implementation. Our hypothesis is, using a higher alpha for the FOFE encoded character features increases its likelihood to remain salient during training.

**Training.** NER datasets are traditionally unbalanced with regards to the negative outside class. To produce enough positive examples to fit the model, we balanced every mini-batch, so that it contains a constant and adjustable ratio of positive and negative classes. The size of an epoch is defined by the number of mini-batches we can fill with the smallest class repeated $T$ times.

### 4.2 LSTM

The LSTM model uses the sequential input directly, which does not require any preprocessing. We feed the network with the input sentences. Before training as a performance optimization, we sorted all the sentences by length and we then divided them into mini-batches. This reduces the amount of masking, and thereby wasteful computations as the majority of mini-batches will be of fixed length.

We use the same set of input features as Chiu and Nichols (2016):

- Word-level, the matching word-embedding for the input word or the unknown word embedding if the word is not in our vocabulary.
- Word-character level, all the characters per word are mapped to embeddings trained with the model. We extracted the alphabet manually and the language is specific.
- Word-case feature, per word class mapping such as lower, upper, title, digits etc.

**Architecture.** The word-character level features are passed through a convolution layer with a kernel of size 3 and a max-pooling layer with a window matching the maximum word length, resulting in a fixed-width character feature.

We tested LSTM cell sizes of dimension 100 and 200, our character embedding set at 30, and a maximum word length at 52. Dropout was set to 50% for recurrent LSTM connections, character feature and before the output layer. We observed that the output dropout had the greatest influence on the results.

All the word and character features are then concatenated per word and fed to a single BILSTM layer consisting internally of two independent LSTM cells which represent the forward and backward passes. The BILSTM output is the concatenation of both passes. We computed the tag scores for the BILSTM-CNN model using softmax from a single dense layer. The BILSTM-CNN-CRF model replaces the dense softmax layer with a CRF layer.

We used a negative log likelihood as loss function for the BILSTM-CNN-CRF model and categorical crossentropy for BILSTM-CNN.

## 5. Experimental Setup

We implemented all the models using Keras and Tensorflow as its backend. Early stopping was performed on all the models with a patience ranging from 5 to 10 depending on model; the parameters from the best epoch were selected for the resulting classifier. The word-embeddings were preinitialized without any preprocessing or normalization. In addition, we froze them during training but in a future work we may enable training. All the models used the Nadam optimizer.

**Hyperparameters.** We carried out a minimal hyperparameter search for BILSTM variants as usable parameters could be found in previous work. However, we could not use FOFE parameters as they produced poor results for us. We performed a smaller hyperparameter search on the CoNLL 2003 dataset to find more optimal parameters.

**Evaluation.** All the models produce IOBv2 annotations, IOBES is postprocessed by simple rules into correct IOBv2 tags. The annotated datasets were evaluated using conlleval from the CoNLL 2003 task, using tab delimiter instead of space, this because SUC3 has tokens with spaces in them.

SUC3 is evaluated on the 4 statistically significant classes instead of all 9: PERSON, PLACE, INST and MISC. The MISC is the combination of the remaining 5. Ontonotes 5 is evaluated on PERSON, GPE, ORG, NORP, LOC and MISC using the same principle as SUC. Following (Chiu and Nichols, 2016), we excluded the New Testaments portion from Ontonotes 5 as it lacks goldstandard annotations for NER.

For crossvalidation, we indexed all the sentences of the full dataset and we randomly split the index into 10 folds; this created 10 sets of indices. For each fold, we used one of them as test set and the rest as training set. For the training part, we used a 90/10% split to create a validation part which is used to determine when to stop training. Finally, we combined the predictions of the test part in each fold, 10 of them, into one dataset which we evaluated to produce the final score.

## 6. Results

BILSTM models outperform FOFE-CNN, as can be seen in Table 1. We trained FOFE-CNN models on Ontonotes 5 and SUC 3 with similar settings as the CoNLL 2003 dataset, these parameters produced subpar models which were not comparable without a new hyperparameter search.

Character features are important, as can be seen in Table 3 with more substantial improvements for lowercase embeddings. CRF improves the result for most embeddings and larger networks appear to have mixed results.

## Acknowledgements

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

## References

Douglas Appelt, Jerry Hobbs, John Bear, David Israel, Megumi Kameyama, and Mabry Tyson. 1993. SRI: Description of the JV-FASTUS system used for MUC-5. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore*, pages 221–235, San Francisco, August. Morgan Kaufmann.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the ACL*, 4:357–370.

Firas Dib. 2018. A multilingual named entity recognition system based on fixed ordinally-forgetting encoding. Master's thesis, Lund University, Lund.

Stian Rødven Eide, Nina Tahmasebi, and Lars Borin. 2016. The Swedish culturomics gigaword corpus: A one billion word Swedish reference dataset for NLP. In *From Digitization to Knowledge 2016: Resources and Methods for Semantic Processing of Digital Works/Texts, Proceedings of the Workshop*, volume 126, pages 8–12, Krakow.

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Magnus Åström. 1992. The linguistic annotation system of the Stockholm-Umeå corpus project. Technical Report 33, Department of General Linguistics, University of Umeå.

Per Fallgren, Jesper Segeblad, and Marco Kuhlmann. 2016. Towards a standard dataset of swedish word vectors. In *Sixth Swedish Language Technology Conference (SLTC), Umeå 17-18 nov 2016*.

| CoNLL03 Test | Glove 6B 100d | | | Glove 6B 200d | | | Glove 6B 300d | | | RCV 256d | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| BILSTM 100d | 83.64 | 83.82 | 83.73 | 83.27 | 83.46 | 83.37 | 82.85 | 82.81 | 82.83 | 85.51 | 87.66 | 86.57 |
| BILSTM-CNN 100d | 86.87 | 90.00 | 88.41 | 85.39 | 88.77 | 87.05 | 84.75 | 88.74 | 86.70 | 86.25 | 89.09 | 87.65 |
| BILSTM-CNN-CRF 100d | 89.25 | 90.10 | 89.67 | 88.63 | 89.20 | 88.92 | 88.61 | 88.56 | 88.59 | 89.25 | 89.41 | **89.33** |
| BILSTM-CNN-CRF 200d | 89.76 | 90.44 | **90.10** | 89.08 | 89.82 | **89.45** | 88.80 | 88.83 | **88.81** | 88.92 | 88.92 | 89.07 |
| FOFE-CNN | 79.87 | 84.17 | 81.97 | 82.26 | 83.11 | 82.68 | 83.64 | 83.16 | 83.40 | 87.91 | 87.80 | 87.86 |

Table 1: CoNLL03 Test results

| Ontonotes5 Test | Glove 6B 100d | | | Glove 6B 200d | | | Glove 6B 300d | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 |
| BILSTM 100d | 81.32 | 82.46 | 81.89 | 82.65 | 80.03 | 81.32 | 81.49 | 80.09 | 80.79 |
| BILSTM-CNN 100d | 84.20 | 86.91 | 85.53 | 82.59 | 85.84 | 84.19 | 82.92 | 86.40 | **84.62** |
| BILSTM-CNN-CRF 100d | 82.70 | 86.50 | 84.56 | 83.91 | 84.10 | 84.00 | 83.13 | 84.84 | 83.97 |
| BILSTM-CNN-CRF 200d | 84.62 | 86.79 | **85.69** | 86.43 | 86.00 | **86.22** | 81.86 | 84.50 | 83.16 |

Table 2: Ontonotes 5 Test results.

| SUC3 Test | Swectors 300d | | | Gigawords 256d | | | Gigawords Lcase 256d | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 |
| BILSTM 100d | 81.82 | 64.07 | 71.87 | 81.91 | 77.98 | 79.90 | 79.14 | 75.99 | 77.53 |
| BILSTM-CNN 100d | 79.72 | 74.83 | 77.20 | 82.33 | 81.79 | 82.06 | 82.23 | 80.46 | 81.34 |
| BILSTM-CNN-CRF 100d | 82.55 | 78.31 | 80.37 | 86.13 | 83.28 | **84.68** | 82.53 | 82.12 | **82.32** |
| BILSTM-CNN-CRF 200d | 85.09 | 77.48 | **81.11** | 81.63 | 79.47 | 80.54 | 82.15 | 80.79 | 81.47 |
| SUC3 10-fold crossvalidation | | | | | | | | | |
| BILSTM-CNN-CRF 100d | 82.07 | 80.22 | 81.14 | 85.22 | 83.62 | 84.41 | 84.31 | 84.32 | 84.31 |

Table 3: SUC 3 Test results


Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Heng Ji and Joel Nothman. 2016. Overview of TAC-KBP2016 Tri-lingual EDL and Its Impact on End-to-End KBP. In *Proceedings of the Ninth Text Analysis Conference (TAC 2016)*, Gaithersburg.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. 2017. Overview of TAC-KBP2017 13 Languages Entity Discovery and Linking. In *Proceedings of the Tenth Text Analysis Conference (TAC 2017)*, Gaithersburg.

Marcus Klang, Firas Dib, and Pierre Nugues. 2017. Overview of the ugglan entity discovery and linking system. In *Proceedings of the Tenth Text Analysis Conference (TAC 2017)*, Gaithersburg, Maryland, November.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the ACL (Volume 1)*, pages 1064–1074.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticæ Investigationes*, 30(1):3–26.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on EMNLP*, pages 1532–1543, Doha.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on EMNLP*, pages 338–348, Copenhagen.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147, Edmonton.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158, Taipei.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1)*, pages 1237–1247, Vancouver.

ShiLiang Zhang, Hui Jiang, MingBin Xu, JunFeng Hou, and LiRong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP (Volume 2)*, pages 495–500.

# Docria: Processing and Storing Linguistic Data with Wikipedia

# Docria: Processing and Storing Linguistic Data with Wikipedia

**Marcus Klang**
marcus.klang@cs.lth.se
Lund University
Department of Computer Science
S-221 00 Lund, Sweden

**Pierre Nugues**
pierre.nugues@cs.lth.se
Lund University
Department of Computer Science
S-221 00 Lund, Sweden

## Abstract

The availability of user-generated content has increased significantly over time. Wikipedia is one example of a corpus, which spans a huge range of topics and is freely available. Storing and processing such corpora requires flexible document models as they may contain malicious or incorrect data. Docria is a library which attempts to address this issue with a model using typed property hypergraphs. Docria can be used with small to large corpora, from laptops using Python interactively in a Jupyter notebook to clusters running map-reduce frameworks with optimized compiled code. Docria is available as open-source code at https://github.com/marcusklang/docria.

## 1 Introduction

The availability of user-generated content has increased significantly over time. Wikipedia is one example of a corpus, which spans a huge range of topics and is freely available. User-generated content tests the robustness of most tools as it may contain malicious or incorrect data. In addition, data often comes with valuable metadata, which might be semi-structured and/or incomplete. These kinds of resources require a flexible and robust data model capable of representing a diverse set of generic and domain-specific linguistic structures.

In this paper, we describe a document model which tries to fill the gap between fully structured and verifiable data models and domain-specific data structures. This model, called Docria, aims at finding a tradeoff between the rigidity of the former and the specificity of the latter. To show its merits, we contrast the application of fully structured data models to practical noisy datasets with the simplicity of Docria.

## 2 Related Work

Linguistically annotated data have been stored in many different formats, often developed to solve practical problems. We can group prior work into three categories:

**Formats** – the technical formats which are used to serialize the data;

**Document models** – conceptual descriptions of how the data is connected, often mapped to concrete software implementations;

**Applications and tooling** – user-facing applications for annotation, search, etc.

in this section, we will focus on the low-level formats and libraries to parse and access the data contained within.

Pustylnikov et al. (2008), in their work on unifying 11 treebanks, made a summary of formats typically used, which shows a dominance of XML variants and CoNLL-like formats. We examine some of them here.

**Tabular annotation.** The tabular annotation in plain text is one of the simplest formats: One token per line and white space separation for the data fields connected to the token followed by a double line separation to mark a sentence. This kind of format was used first in the CoNLL99 task on chunking (Osborne, 1999) and then on subsequent tasks. Its main merits are the ease of use with regards to writing parsers and its readability without documentation.

Universal Dependencies (Nivre et al., 2019) is an example of a recent project for multilingual corpora using this format. It defines a variant called CoNLL-U, an adaption of the format used in CoNLL-X shared task on multilingual dependency

parsing (Buchholz and Marsi, 2006). CoNLL-U includes field descriptions at the start of a document using hashtag (#) comments, adds subword support, and a field, if used, would allow for untokenization by including information about spacing between tokens.

CoNLL-* formats are tightly connected to data used in the shared tasks. Variations of these plaintext formats in the wild have no real standard and are mostly ad-hoc development. The field separation is a practical aspect, which may vary: spaces or tabulations. Depending on the corpus, these are not interchangeable as the token field might include ordinary spaces as part of the data field.

**Semi-structured formats.** Semi-structured formats specify stricter rules and a frequent choice is to follow the XML syntax to implement them (Bray et al., 2008). XML is hierarchical and can support higher-order structures such as sections, paragraphs, etc. XML has been used successfully in the development of the TIGER Corpus (TIGER XML) (Brants et al., 2002) and the Prague Dependency Treebank (PML) (Hajič et al., 2018).

The XML annotation relies on a schema defining its content on which programs and users must agree. Aside from TIGER XML and PML, the Text Encoding Initiative (TEI) and FoLiA XML (van Gompel and Reynaert, 2013) are general purpose XML schema definitions focused on linguistic and text annotation. TEI and FoLiA provide extensive documentation and guidelines on how data should be represented in XML.

**Graph formats.** From primarily hierarchical formats, the NLP Interchange Format (NIF) provides a graph-oriented way of connecting information which builds on existing standards such as LAF/GrAF, RFC 5147, and RDF. The main innovation in NIF is a standardized way of referring to text with offsets also known as a stand-off annotation. NIF is similar to WIKIPARQ (Klang and Nugues, 2016).

## 3 Docria

Docria is a document model based on typed property hypergraphs. We designed it to solve scalability and tooling problems we faced with the automatic processing and annotation of Wikipedia. This corresponds notably to:

- The lack of document models and storage solutions that could fit small and large corpora and that could be compatible with research practices;
- The impossibility to use the same document model with potentially costly large-scale extraction algorithms on a cluster with a map-reduce computing framework such as Apache Spark.

**Motivation.** These aspects were dominant in the construction of Docria, for which we set a list of requirements:

**Openness** – release the library as open source[1]; share processed corpora such as Wikipedia in formats used by this library; invite others to use the library for various tasks;

**Scalability** – from small corpora using a few lines of code to show a concept on a laptop to large-scale information extraction running on multiple computers in a cluster with optimized code;

**Low barrier** – progressive learning curve, sensible defaults, no major installations of services or configurations. Specifically, we wanted to reduce barriers when we shared larger corpora with students for use in project courses;

**Flexibility** – capable of representing a diverse set of linguistic structures, adding information and structures progressively, changing structure as needed;

**Storage** – reducing disk-space and bandwidth requirements when distributing larger corpora.

**Design.** To meet these goals, we implemented Docria in both Python and Java with a shared conceptual model and storage format. One of the user groups we had in mind in the design step was students in computer science carrying a course project. As our students have programming skills, we elected a programmer-first approach with a focus on common tasks and algorithms and a tooling through an API.

Python with Jupyter notebooks provides an interactive Read-Evaluate-Print-Loop (REPL) with rich presentation possibilities. We created extensions for it to reduce the need for external tooling and so that with a few lines of code, a programmer can inspect the contents of any Docria

[1] https://www.github.com/marcusklang/docria

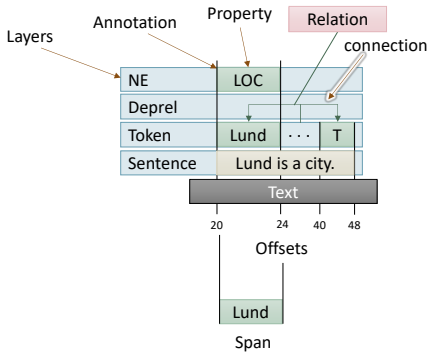



Figure 1: Docria data model

document. Through a matching implementation in Java, Docria provides a path to scale up when needed, as specific tasks can be orders of magnitudes faster than with a CPython implementation.

Docria documents consist of text collections and layers, shown in Figure 1. Text collections allow for multiple representations of a single text. A layer is a collection of nodes. These nodes can have fields which refer to the text collections. One particular restriction we impose is that a user must define a schema per layer. This is essential for introspection and verification of the data contained in documents. The schema defines the available fields and their data type with support for metadata.

**Datatypes.** The datatypes include basic types such as Boolean, integer, float, and string. Advanced types include text spans, node spans, node references, and node array references, which enable a programmer to represent graph structures. Field types, which are node references, must specify a target layer. In addition, this restriction results in well-defined dependencies between layers, which can be used in the future for partial document reconstruction when reading.

Using a relational database analogy, layers correspond to tables; they contain nodes which are equivalent to rows with fields, which are typed columns with specialized support for references to other nodes in other layers.

**Stand-off references.** Docria uses stand-off references in which we separate text from linguistic layers. These layers refer to ranges in the original text. To simplify the implementation and reduce sources of common bugs, the text string is split into pieces according to the offsets and stores text as a list of substrings, which is reconstructed without a loss by a join. Offsets, when serialized, only refer to spans of substrings. Software implementations can reconstruct offsets by computing the actual substring length and creating a lookup table. This will generate correct offsets even if the in-memory representation of a string differs, which is the case with standard strings in Java and Python 3.

**Binary format.** For the binary format, we selected MessagePack. MessagePack is self-describing, has an open well-defined specification, and has multiple open-source implementations in a diverse set of programming languages. The binary format can be used on a per document basis or in an included collection container, which writes multiple binary documents in sequence. This binary format was also designed to allow for a quicker content listing by separating content into compartments which can be read independently: document properties, schema, text, and layer data.

**The Wikipedia corpus.** We used the official REST API provided by Wikimedia and a page listing from the official dump page to collect the Wikipedia corpus. We downloaded all the pages in HTML format from this page listing in October 2018. This HTML format was processed and converted into a DOM using JSoup. Using recursive rules, we transformed the DOM into a flat text representation with structural layers referring to ranges such as section, paragraph, and anchors. Furthermore, we linked anchors to Wikidata by translating page targets to Q-numbers where available. We also retained formatting, such as bold and italics. We stored all this information using Docria.

In this dump, there are 5,405,075 pages excluding redirections.

## 4 Evaluation

We applied the spaCy library[2] to annotate all the English Wikipedia pages with parts of speech, entities, and dependency graphs, and we made the result available at `http://fileadmin.cs.lth.se/papers/nodalida2019/`. On average, each page of the corpus, after annotation,

[2] `https://spacy.io/`

contains 72.2 sentences, 901.8 tokens, 144.8 entities, and 4,383 characters.

We used this annotated corpus to evaluate the technical aspects of Docria and compare them to XML. We chose XML as it is pervasive in the literature and capable of representing all the structures present in Wikipedia.

We selected FoLiA as the XML format. FoLiA is well-defined, has good tooling, defines a diverse set of structural annotations which covers most, if not all, aspects of Wikipedia. FoLiA also has an official Python library, which we used to read documents.

Millions of XML files can be stored uncompressed in a file system. However, this often results in considerable overhead in terms of access times and reading and is therefore not practical for efficient processing. In addition, XML is verbose and contains redundant information. All this makes compression and streaming a necessity when storing and processing millions of documents.

To compare FoLiA XML with Docria, we chose to use a sequential tarball format with a bzip2 compression. We chose this format as it provided the most similar way to store documents in sequence applicable to both FoLiA XML and Docria. We created one XML file per article in-memory and saved them in a sequence using the tarfile API of Python. The structures we included for the comparison were section, paragraph, entities, tokens with their part of speech and lemma, and dependency relations.

## 5 Benchmark

We stored the Wikipedia corpus in 432 parts, containing on average 12,512 pages per part. Due to time constraints, the metrics below are computed using only 64 of the 432 parts.

First, we measured the difference in size when compressed: FoLiA XML files are on average 2.47 times larger than the matching Docria files. The compressed Docria parts have a mean size of 85.0 MB[3] compared to 209.8 MB for the compressed FoLiA XML parts. This translates to a compressed size of 6.8 kB resp. 16.8 kB on average per page.

Secondly, we measured the cost of decompressing the files in memory. Reading a single bzip2 Docria compressed file without any processing and a 1 MB buffer requires, on an Intel Xeon at 3.40 GHz, 16.3 sec $\pm$ 18.9 ms compared to 104 seconds $\pm$ 136 ms to read FoLiA XML, both averaged over 7 runs. Reading compressed FoLiA XML over binary Docria tar-files is on average 6.4 times slower.

Uncompressed Folia XML documents are on average 9.5 times larger per document with a mean size of a page of 314.5 kB vs. 32.1 kB for Docria. For comparison, the mean average size of raw UTF-8 encoded text is of 4.4 kB per page. Put another way, using the plain text as starting point, Docria has an annotation overhead of 7.6 times vs. 69.6 times for XML.

[3] 1 MB = 1,000,000 bytes

## 6 Programming Examples

In this section, we show programs for three basic operations:

1. Create a new document and add a token with part-of-speech annotation.

2. Read a sequential tarball and print all the tokens of all the sentences of the corpus;

3. Read a sequential tarball and extract the entities of type person.

**Create a document and add a part of speech.** We first create a document from a string and we add a token layer. We then add a node to this layer, spanning the 0..4 range and we annotate it with a part of speech using the `add()` method as this:

```
# Initial include
from docria import Document, \
     DataTypes as T

# Create a document
doc = Document()

# Add main text
doc.maintext = "Lund University"

# Create a token layer with two fields
doc.add_layer("token",
      pos=T.string, text=T.span)

# The token layer, when displayed
# in a Jupyter notebook, will be
# rendered as a HTML table.
tokens = doc["token"]

# Adding a token node
# referencing range 0:4
token = tokens.add(
  pos="PROPN",
  text=doc.maintext[0:4]
)
```

**Print the tokens.** We assume we have a tarball of documents segmented into sentences and tokens, and annotated with the parts of speech. We read the tarball with `TarMsgpackReader` and we access and print the sentences, tokens, and parts of speech using the Python dictionary syntax.

```
from docria.storage \
    import TarMsgpackReader

with TarMsgpackReader(
    "enwiki00001.tar.bz2",
    mode="r|bz2") as reader:
  for rawdoc in reader:
    # Materialize document
    doc = rawdoc.document()

    # Lists all layers with field
    # types and metadata
    doc.printschema()

    # Print the original text
    # Equivalent to doc.text["main"]
    print(doc.maintext)

    for sentence in doc["sentence"]:
      # Print the full sentence
      print(sentence["tokens"].text())

      for tok in sent["tokens"]:
         # Form <TAB> part-of-speech
         print("%s\t%s" %
               (tok["text"], tok["pos"])
```

**Extract entities of a certain type.** We assume here that the tarball is annotated with entities stored in an `ENTITY` layer. We read the tarball and access the entities. We then extract all the entities of category `PERSON`:

```
with TarMsgpackReader(
    "enwiki00001.tar.bz2",
    mode="r|bz2") as reader:
  for rawdoc in reader:
    # Materialize document
    doc = rawdoc.document()

    # Get the entity layer
    entities = doc["entity"]

    # Filter out PERSON in entity
    # layer having field label
    # equal to PERSON
    query = (entities["label"]
             == "PERSON")

    for person in entities[query]:
      # Tokens represents potentially
      # many tokens, text()
      # transforming it to a string
      # from the leftmost
      # to the rightmost token.
      print(person["tokens"].text())
```

## 7 Discussion

When converting the Wikipedia corpora to fit the FoLiA XML format, we had issues identifying a suitable span annotation for the Wikipedia anchor link. We decided to associate it with the FoLiA XML entity type.

In addition, when using stand-off annotations, some documents did not pass validation with offset errors, possibly due to normalization issues common to Wikipedia text. This gives an argument that these kinds of formats do not work reliably with noisy datasets. We instead included the sentences as text and used the nospace attribute to allow untokenization, which does increase verbosity slightly.

Initially, we used the official foliapy library, but we were unable to get a decent performance with it, potentially addressed in the future. We resorted to using the LXML DOM matching example documents with Folia. To ensure correctness, we verified samples of our XMLs using foliavalidator.

## Acknowledgments

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program, grant number 340-2012-5738.

## References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Tim Bray, Eve Maler, François Yergeau, Michael Sperberg-McQueen, and Jean Paoli. 2008. Extensible markup language (XML) 1.0 (fifth edition). W3C recommendation, W3C. Http://www.w3.org/TR/2008/REC-xml-20081126/.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Maarten van Gompel and Martin Reynaert. 2013. Folia: A practical xml format for linguistic annotation — a descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, 3:63–81.

Jan Hajič, Eduard Bejček, Alevtina Bémová, Eva Buráňová, Eva Hajičová, Jiří Havelka, Petr Ho-

mola, Jiří Kárník, Václava Kettnerová, Natalia Klyueva, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Petr Pajas, Jarmila Panevová, Lucie Poláková, Magdaléna Rysová, Petr Sgall, Johanka Spoustová, Pavel Straňák, Pavlína Synková, Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Barbora Vidová Hladká, Daniel Zeman, Šárka Zikánová, and Zdeněk Žabokrtský. 2018. Prague dependency treebank 3.5. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Marcus Klang and Pierre Nugues. 2016. WIKIPARQ: A tabulated Wikipedia resource using the Parquet format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4141–4148, Portorož, Slovenia.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Gabrielė Aleksandravičiūtė, et al. 2019. Universal dependencies 2.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

M. Osborne. 1999. *CoNLL-99. Computational Natural Language Learning. Proceedings of a Workshop Sponsored by The Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

Olga Pustylnikov, Alexander Mehler, and Rüdiger Gleim. 2008. A unified database of dependency treebanks: Integrating, quantifying & evaluating dependency data. In *LREC 2008*.

# Hedwig: A Named Entity Linker

# Hedwig: A Named Entity Linker

**Marcus Klang**
Lund University
Department of Computer Science
Lund, Sweden
Marcus.Klang@cs.lth.se

**Pierre Nugues**
Lund University
Department of Computer Science
Lund, Sweden
Pierre.Nugues@cs.lth.se

## Abstract

Named entity linking is the task of identifying named things such as "Barack Obama" or a city such as "New York" in text commonly found in newspapers, discussion forums, etc. We propose Hedwig, an end-to-end named entity linker which uses a combination of word and character BILSTM models for mention detection, Wikidata and Wikipedia derived knowledge-base with global information aggregated over 9 language editions, and a page-rank algorithm for entity linkage. Hedwig was evaluated on the TAC2017 dataset with a final score of 59.9% on CEAFmC+, an improvement over our previous generation linker Ugglan, and a trilingual entity link score of 71.9%.

## 1 Introduction

Named entity linking (NEL) is the task of automatically finding and linking mentions of things to unique identifiers. The word *thing* is too broad for the linkage problem; the more concrete definition used in this paper is linking uniquely separable things which we can identify by a *name*, i.e. named entities. The classes of *named entities* we will try to link are instances of persons, organizations, locations, etc.

Take for instance the named entity of class location: "New York". This mention can refer to the state[1] of New York or the large city[2] situated in that particular state. The matching unique identifier could be the English Wikipedia label: `New_York_City`.

The typical NEL pipeline consists of many phases including a name finding, mention detection (MD) phase (e.g. detecting "New York" in a text), a candidate generation (CD) phase (e.g. state or city), and a entity linking (EL) phase (e.g. assigning the label). In addition, these phases might be defined independently (Cucerzan, 2007), or trained jointly (Ganea and Hofmann, 2017). The MD phase is frequently a named entity recognizer (NER), which finds and classifies spans of strings to a set of predefined classes such as persons, organization, location, etc. The CD phase uses the classified mention as input, possibly with context, and from this information generates a list of entity candidates. Finally, the entity linking phase ranks and selects the most probable or coherent set of candidate entities and assigns each mention a label which corresponds to the given candidates unique identifier.

The unique identifier can be local or global, and its concrete format is determined by the linker method, which can span the spectrum of fully supervised to unsupervised. This paper uses a supervised approach by linking to predetermined identifiers provided by an entity repository which we refer to as the Knowledge-base (KB).

Depending on which specific task and thus a matching benchmark is selected, different requirements will arise. This paper will present a named entity linker for the 2017 edition of Text Analysis Conference (TAC) Entity Discovery and Linking (EDL) task with its provided benchmark (Ji et al., 2017). This task was selected because it provides a multilingual gold standard. This dataset is diverse in its content and is a combination of real-world noisy texts found on the internet. This type of dataset presents challenges applicable to arguable all entity linkers where real-world use is the goal.

### 1.1 TAC EDL 2017

The TAC EDL task consists of linking 2 categories of mentions:

- Named mentions which has 5 classes:
  - `PER`, Persons (not fictional)

[1] https://en.wikipedia.org/wiki/New_York_(state)
[2] https://en.wikipedia.org/wiki/New_York_City

  - ORG, Organizations, (companies, institutions, etc.)
  - LOC, Location (natural locations, such as mountains, oceans, lakes etc.)
  - GPE, Geopolitical entities (cities, administrative areas, countries, states, municipalities, etc.)
  - FAC, Facilities (airports, transportation infrastructure, man-made buildings, hospitals etc.)

- Nominal mention linkage, commonly hyponyms of mentions such as "Barack Obama" would be the named mention and the nominal mention would be "president". Other common relations are `son`, `wife`, `daughter`, `father`, `company`, `area` etc. The nominal mentions are classified and linked in the same manner as named mentions.

The corpus is a mixture of discussion forum (DF) and newswire (NW) text in three languages: English, Spanish, and Mainland Chinese stored in XML and HTML (2014). The gold standard provides links to the Freebase KB and out of KB labels which starts with "NIL" followed by a number which spans all 3 languages. The Freebase identifiers is connected in the BaseKB provided by TAC. The final score is based on the performance on all three languages.

We subdivided the corpus into a training and test-set based on years. The 2017 dataset is the test set, and 2014-2016 is our training-set. It is important to note that the 2014 edition is not available for Spanish and Chinese.

Nominal mentions were added in 2015, but they were not fully annotated until 2016 which means scarce data is available for training nominal detection using deeper models.

### 1.2 Specifics and Limitations to Hedwig

Hedwig uses data and statistics from Wikipedia, almost exclusively, and as such, it is natural for us to use Wikidata as the primary KB. Wikidata provides unique identifiers in the form of Q-numbers, e.g. "Barack Obama" the president has identifier: `Q76`. Wikidata binds the Wikipedia languages editions together, and is updated along and is thus more up to date than other repositories, making it the most logical choice for us. To be compliant with the TAC EDL gold standard, the Q-numbers are converted into Freebase using a mapping[3] provided by Google, produced at the archiving and termination of Freebase as a public open knowledge base. This mapping is not perfect, a low number of Q-numbers was represented by multiple freebase entries and is heuristically resolved using the lowest Q-number.

## 2 Related Work

Named entity recognition has its modern roots with the CoNLL03 task of Language-Independent Named Entity Recognition (Tjong Kim Sang and De Meulder, 2003), where the best model used simple linear classifiers (Florian et al., 2003), which made progress until Ratinov and Roth (2009). Neural models, starting with Collobert et al. (2011), were further developed with an exponential weight encoding method called FOFE (Fixed-Size Ordinally-Forgetting Encoding) using feedforward neural networks (Zhang et al., 2015) as an alternative to more computationally demanding models. These architectures were ultimately surpassed by deeper recurrent neural models using LSTMs (Hochreiter and Schmidhuber, 1997) and CRFs in different combinations with or without word character encoders (Chiu and Nichols, 2016; Ma and Hovy, 2016).

Word embeddings are a key ingredient to NER; the most commonly used word embedding started with Mikolov et al. (2013), followed by Pennington et al. (2014), to more recent developments by Mikolov et al. (2018), and deeper models by Peters et al. (2018).

Modern entity linking uses a variety of methods such as simple classification models (Bunescu and Paşca, 2006; Cucerzan, 2007; Milne and Witten, 2008), end-to-end linkage with a voting scheme for linkage (Ferragina and Scaiella, 2010), graphical models (Hoffart et al., 2011; Guo and Barbosa, 2014), integer linear programming (Cheng and Roth, 2013), fully probabilistic models (Ganea et al., 2016) to deeper neural models (Ganea and Hofmann, 2017). Wainwright et al. (2008)[4] proved that entity linkage which tries to maximize local and global agreement jointly during linkage is NP-hard to solve, which most authors approximate or simplify to reach feasibility.

[3] `https://developers.google.com/freebase/`

[4] cited in Globerson et al. (2016).

## 3 Data

In the making of Hedwig, we used these data sources:

- 9 Wikipedia editions: en, es, fr, de, sv, ru, zh, da, no, scraped using the Wikipedia REST API[5] in October 2018.
- Wikidata JSON dump from October 2018
- TAC EDL Data 2014-2016
- Manually annotated mappings for classes in Wikipedia to a set of predefined classes.

### 3.1 Wikidata data: Our KB

The Wikidata JSON dump is delivered as one large gzip or bzip2. This file when decompressed is one single JSON object; it does however use "one JSON object per line" approach for easier processing. Using standard bash tools, it was split into many parts with 50,000 objects per file to enable efficient cluster processing. The dump was converted by an in-house built Wikidata parser which transforms the JSON dump into Parquet files for further processing and information extraction. Converted information was: Q-number, description, alias, claims also known as properties and sitelinks. A subset of most common claim datatypes are supported, the rest is either ignored or encoded as plain strings.

### 3.2 Wikipedia

We scraped all 9 editions using the REST API by first downloading a list of page names from Wikimedias dump site[6], more specifically the [lang]wiki-[date]-all-titles-in-ns0.gz file which contains all page labels[7]. This file was then processed in sequence and parallel downloaded using a custom Python tool. This page name list is slightly out of sync with the online version, resulting in that some pages does not exist; pages not found were ignored. Pages that failed to download due to server errors or network failure was retried once more at the end of the scraping process which consisted of retrying 5-250 pages, most of these pages were retrieved.

The REST API provided by Wikimedia is using Parsoid[8] internally and produces HTML outputs with added metadata tags and attributes for elements. This parser does not produce an exactly identical structure to the publicly facing rendering. However, when sampling pages, no visible differences to the content or format was significantly evident. For the Chinese version, a translation table was included in the output to convert the character sequences between the different variants: Hong Kong, Singapore, etc.

## 4 Refinement

### 4.1 Wikipedia

Wikipedia is refined in 2 steps:

- Import, converting HTML to Docria layers.
- Link, resolving Wikipedia anchors to Wikidata.

In the Import step, HTML was parsed using JSoup[9] which converts the raw HTML string into a structured Document Object Model (DOM). We used rules applied recursively to the DOM Tree to filter out the markup and produce a flattened document consisting only of readable text. In addition, during the flattening process or tree traversal, enough information was retained to produce multiple layers of spans with added metadata covering paragraphs, sections, anchors, lists, tables, italic, bold etc. These layers were stored using Docria which can represent this type of data and store exact string mappings. With the exception of the Chinese version, all editions were parsed identically, the Chinese version required a preconversion step to produce a Mainland Chinese version using the provided translation table; this conversion might not be perfectly accurate.

The link step used sitelinks in the Wikidata dump and all anchors in the processed Wikipedia dump to fully link all pages. It is necessary to do this link step as many links are placeholder suggestions for future articles.

Ultimately, this produces a docria dump which is fully linked to Wikidata, and contains enough semantic information and structure for all further processing.

### 4.2 Word Embeddings

The word embeddings are trained in-house from a 2016 version of Wikipedia on English, Spanish and Chinese, which was produced using the `word2vec` tool released by Mikolov et al. (2013).

[5] https://[lang].wikipedia.org/api/rest_v1/
[6] http://dumps.wikimedia.org/
[7] Wikipedia name for article titles
[8] https://www.mediawiki.org/wiki/Parsoid
[9] https://jsoup.org/

### 4.3 Entity classification

Many articles in Wikipedia do not conform to acceptable classes in TAC EDL. To improve entity linkage precision, entity candidate filtration based on class is advantageous to reduce noise.

Using the TAC Annotation guidelines as the basis, we sampled a diverse set of entities by writing rules that sampled entities with usable instance-of relations and mapped them to 9 classes:

**PER,** natural persons, humans

**PER_F,** fictional characters or persons

**LOC,** natural locations, continents, lakes, rives, mountains, streams, etc.

**GPE,** geopolitical entities, countries, administrative areas, regions, etc.

**ORG,** organizations, institutions, business entities etc.

**EVT,** events, sport-events, conflicts, wars, etc.

**WRK,** products, newspaper, books, films, tv-series, etc.

**FAC,** man made buildings, transportation infrastructure, airports, hospitals, landmarks, etc.

**NONE,** all that does not fall into any above, this explicitly includes: wikipedia categories, templates, disambiguation pages etc.

The input features for the model were:

- direct relations such as P31 (instance-of) Q5 (human)
- Boolean indicating the existence of a P31 instance. Empirically, most entities without an instance-of are exceptionally hard to disambiguate and should, most of the time, be classified as NONE.
- all path segments from the initial seed entity with source, relation and target, to the depth of 5.

Breadth first search (BFS) search to the depth of 5 was used to find suitable segments by following edges for relations: 31 (instance-of), 279 (subclass-of), 1269 (facet-of) and 1889 (different-from). The training data consists of seed entities mapped to one of these 9 classes, and all features were generated on the fly. To produce negative examples, all unclassified are assumed to have label NONE. To balance positive and negative bias, the negative share was set to 10% per mini-batch. In practice, this makes the classifier default to NONE for all unknown relations; the large positive share should force the classifier to learn given examples, and retain the most significant patterns common to each class. Phrased differently, we favored precision over recall in this setting.

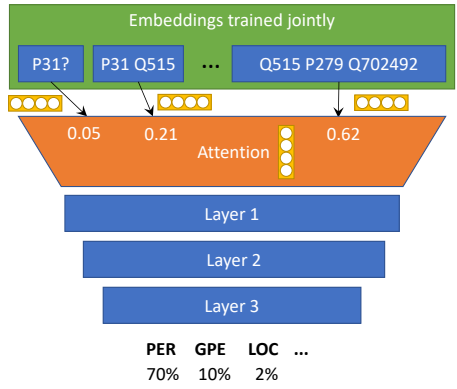



Figure 1: Entity classifier neural network, P31 is instance-of, Q515 is "city" entity and Q702492 is "urban area" which is a subclass of (P279) city.

We trained a three layer neural model with a simplified attention mechanism to classify types as shown in Figure 1. This model treats all input features as a trainable embedding, a simplified attention mechanism is used to produce a single vector of all inputs through a weighted average produced by the attention mechanism and then 3 RELU layers with a final softmax layer to produce output prediction.

### 4.4 Mention Dictionary

To bootstrap candidate generation, all anchor texts, aliases and titles of Wikipedia articles were used to build a mention dictionary. In addition, the finished dictionary is applied to an auto-linked version of each Wikipedia article to count the share of matches in linked vs non-linked text. With auto-linking, we refer to the process of linkage densification by using the existing anchors per document to add missing links with the exact same sequence of tokens. If multiple segmentation is possible, we resolve them using a dominant-right rule which se-

lects the longest to the right sequence when equal length segmentations are encountered.

The gathered statistics are the basis for the link-density measure which gives a measure of if a mention should be linked or not, i.e. a baseline mention filtering method.

$$ld(m_i) = \frac{C(m_i \text{ is linked})}{C(m_i)} \quad (1)$$

Tokenization uses an in-house JFlex parser described in 5.1. To normalize token sequences, we use the Lucene analysis infrastructure to convert token sequence to query terms to find entries in the dictionary which is detailed in 5.2. The dictionary uses the finite-state transducer (FST) implemented in Lucene and was selected for its query performance and lower memory requirements.

### 4.5 Entity, Mentions, and Context

We have two types of statistically extracted mappings:

**Monolingual,** modeling the relation of mentions and words to entities in localized language;

**Multilingual,** modeling the entity to entities in context over all editions.

#### 4.5.1 Multilingual

The goal is to compute the pointwise mutual (PMI) information of an entity $e_i$ being in the same context as entity $e_j$:

$$\text{PMI}(e_i, e_j) = \log \frac{P(e_i, e_j)}{P(e_i)P(e_j)} \quad (2)$$

PMI produces an unbounded value, we therefore uses a slightly different formalization to compute the normalized PMI (NPMI) (Bouma, 2009), in this case using raw counts $C$.

$$\text{total} = \sum_i \sum_j C(e_i, e_j) \quad (3)$$

$$\text{NPMI}(e_i, e_j) = \frac{\log \frac{C(e_i, e_j)}{C(e_i)C(e_j)/\text{total}}}{-\log \frac{C(e_i, e_j)}{\text{total}}} \quad (4)$$

Practically, this is computed by counting the occurrences of two entities in the same context, ultimately resulting in a sorted list of entities based on PMI values. This list might be exhaustive and is filtered to only retain the top-$k$ entries.

The context in this paper is defined as entities in the same paragraph. Only top-$k$ filtering produces sub-optimal results as PMI will favor entities with strong but not statistical significant mutual occurrences. This property is not desirable for frequently occurring entities as semantically relevant entities in this case will have lower PMI values and thus be filtered out.

To improve PMI lists, a cutoff based on counts is dynamically determined for each list. Specifically, all counts are sorted from highest to lowest counts and the count limit is determined by when a total share of 80% has been reached, or minimal of 2 if there are too few entries. Subjectively, this heuristic produces better results for frequently occurring entities and makes little difference for rare entities where we do not have statistically significant mutual occurrences.

#### 4.5.2 Monolingual

In addition to multilingual entity relations, localized word, mention relations are computed using the same method as above. Words have a minimum NPMI value of 0.1 to surpass, which seems to be where most nonsense words start to appear. This loosely results in a statistical basis for linkage based on words, mentions, and other entities.

## 5 Pipeline

### 5.1 Segmentation and Tokenization

Tokenization is implemented using a custom JFlex parser optimizer to find common patterns in many languages, the parser is fully defined in that unknown characters or rules will yield separate tokens. It will split tokens based on whitespace for all relevant languages. The parser also applies rule-based pattern matching to detect acronyms, title cased words, upper cased words, numbers, years, periods in many Unicode variants, citation, parenthesis, etc.

Segmentation is rule-based and uses the tokenization stream and its token classification as input. Concretely, it uses a split rule with a minimum sentence length of 4 to retain data even if the segmentation is incorrect, but also to avoid common rule-based mistakes, which are hard to encode without a context aware model.

### 5.2 Mention detection

The mention detection consists of 2 steps:

- Dictionary based detection, classification and overlap resolution using dominant right rule;
- Neural NER model with dictionary detection as part of input features.

#### 5.2.1 Baseline: Dictionary Detection

Using only the localized mention dictionary and cherry-picked link-density (ld) cutoff values per language[10], mentions are found in input sentences. However, this only finds potential mentions without class information. Named entity classes are determined by using candidate generation and picking the top candidate entity and its class according to our produced entity class mapping from section 4.3.

#### 5.2.2 Named Entity Recognition using ML

The mention detector model is a BILSTM-BILSTM CHAR-CRF model. Concretely, this model contains the following elements:

- Character sequences per word using a BILSTM layer, essentially compressing embeddings into one vector which is concatenated with word embedding;
- Word sequences in sentences using a BILSTM layer, using three inputs: word embedding, char embedding from above and dictionary feature from the dictionary detection;
- Finally all output is downprojected and feed into a linear chain CRF layer to predict the most likely tag sequence.

This model predicts all mentions used for linkage. To generate candidates, we take token spans as marked by the model and use the mention dictionary to find all candidates.

### 5.3 Link stage

We start with all candidate lists consisting of possible Q-numbers. The pagerank method will give weight to each vertex in a directed graph based on the available edges.

In the context of linking: a vertex is either a word, mention or an entity. Each candidate entity has an associated list of words, mentions and other entities it co-occurs with, these lists were built using the NPMI cutoff method described in section 4.5. Edges are created using these lists by searching the context for the existance of these mentions, words and candidate entities, and generating vertices and directed edges towards all relevant candidates. The context is unbounded meaning the entire document. This produces a graph with directed edges: from word, mentions to candidate entities, and entity to entity edges.

The final output is a normalized pagerank value for each candidate per mention. The normalized pagerank values are sorted to produce a ranking list. The highest ranking candidate is selected for linkage. There is a minimal NIL detection, filtering out entities which have the NONE class.

The pagerank implementation is in-house and uses power iteration until convergence or max number of iterations with default damping of 0.15, alpha=0.85.

## 6 Results

Results are presented in Table 1. The baseline is weak, which is to be expected as it only uses the mention dictionary. Compared to the our older benchmark on TAC2017 which used a reranker, the new model is competitive, and improves on almost all metrics except NER on English and linkage on English and Spanish, however small differences.

The model favors precision over recall, which can be seen overall. Chinese is particularly strong on entity linkage and increased the most on final metric.

## 7 Discussion

### 7.1 Mention detection

Mention detectors based on neural models are not perfect and can have recall issues. When trained on a noisy dataset such as TAC, it tends to favor precision heavily, resulting in some mentions not being predicted. One way of mitigating this is to expand mentions by using the found mentions and searching for partials or full matches, e.g. linking all occurrences of “Obama” when “Barack Obama” is found as a mention.

TAC contains a large amount of spelling mistakes, particularly in discussion forum texts, domain specific variations which produces a different surface from is also commonplace such as e.g. “Microsoft” and “Micro$oft” forms which are easily recognizable to humans.

When training neural models such as LSTM, it can be tricky to know when to stop training, as such we employ model checkpointing using the

[10] 0.35 for English, 0.25 for Spanish and 0.45 for Chinese

| | | NER | | | NERC | | | KBIDs | | | CEAFmC+ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| Tri | Baseline | 78.4 | 43.1 | 55.6 | 73.5 | 40.4 | 52.1 | 62.1 | 53.8 | 57.6 | 66.7 | 36.7 | 47.3 |
| | Ugglan | 89.4 | 58.4 | 70.6 | 83.0 | 54.3 | 65.6 | 80.1 | 61.7 | 69.7 | 70.9 | 46.4 | 56.1 |
| | NER-Only | 91.5 | 61.6 | 73.6 | 87.9 | 59.2 | **70.8** | 0.0 | 0.0 | 0.0 | 12.9 | 8.7 | 10.4 |
| | Pagerank | 91.4 | 61.7 | **73.7** | 86.2 | 58.3 | 69.5 | 82.4 | 63.8 | **71.9** | 74.3 | 50.2 | **59.9** |
| Eng | Baseline | 81.1 | 41.3 | 54.7 | 75.4 | 38.4 | 50.9 | 65.3 | 52.1 | 58.0 | 67.3 | 34.3 | 45.4 |
| | Ugglan | 90.6 | 65.0 | **75.7** | 83.8 | 60.1 | 70.0 | 82.3 | 62.1 | **70.7** | 69.4 | 49.8 | 58.0 |
| | NER-Only | 93.2 | 62.8 | 75.0 | 89.2 | 60.2 | **71.9** | 0.0 | 0.0 | 0.0 | 22.1 | 14.9 | 17.8 |
| | Pagerank | 92.5 | 63.4 | 75.3 | 85.6 | 58.7 | 69.6 | 77.4 | 64.6 | 70.4 | 71.6 | 49.0 | **58.2** |
| Spa | Baseline | 71.5 | 47.7 | 57.3 | 64.7 | 43.2 | 51.8 | 52.7 | 55.1 | 53.8 | 55.7 | 37.2 | 44.6 |
| | Ugglan | 88.5 | 59.1 | 70.8 | 84.8 | 56.6 | 67.9 | 83.9 | 59.5 | **69.6** | 70.0 | 46.7 | 56.0 |
| | NER-Only | 92.2 | 58.4 | **71.5** | 88.5 | 56.0 | 68.6 | 0.0 | 0.0 | 0.0 | 21.7 | 13.7 | 16.8 |
| | Pagerank | 92.2 | 58.3 | **71.5** | 88.7 | 56.0 | **68.7** | 84.6 | 58.4 | 69.1 | 74.0 | 46.8 | **57.3** |
| Cmn | Baseline | 83.0 | 41.0 | 54.9 | 80.6 | 39.8 | 53.3 | 71.8 | 53.9 | 61.5 | 76.7 | 37.9 | 50.7 |
| | Ugglan | 89.1 | 57.4 | 69.8 | 82.4 | 53.1 | 64.6 | 85.4 | 64.3 | 73.4 | 71.2 | 45.8 | 55.8 |
| | NER-Only | 90.0 | 63.0 | **74.1** | 86.8 | 60.8 | **71.5** | 0.0 | 0.0 | 0.0 | 13.8 | 9.7 | 11.4 |
| | Pagerank | 90.0 | 63.0 | **74.1** | 85.0 | 59.5 | 70.0 | 84.8 | 68.2 | **75.6** | 76.4 | 53.5 | **62.9** |

Table 1: Final results

F1 score on a validation set during training. Ultimately, picking the final parameters from the best saved model. The validation set in this case is a 10% sample of the original data. Training on all data might yield an improvement. For Spanish and Chinese, improvements might be larger as these are about 50% of English, and in some cases even less when broken down into annotated tags. From the results, the NOM category is the weakest category to train on and to detect as they tend to be highly ambiguous, insufficient data and repetition to learn useful information.

One potential approach not evaluated is to train on TAC 2016 data, predict on TAC 2015 data and then merge and train again as described by Bernier-Colborne et al. (2017).

We used an in-house embedding we know is worse than publicly available state-of-the-art. The motivation was that these can be reproduced in any language that has a Wikipedia edition, enabling us to expand the linker to many more languages. Testing different pre-trained embeddings that are available for languages in TAC might yield improvement in detection.

### 7.2 Linker

The linker in Hedwig uses a plain non-personalized variant of PageRank which is solved using the power iteration algorithm. The fact that it is non-personalized means that important weight information with regards to available computed PMI values might reduce its ability to resolve important entities favoring frequent entities which in this context improves the results.

The linker stage uses the entire document with all possible candidates to form collective agreement, the consequence is that all unique candidates will receive a single weight. This means that the linker is unable to model topic drift or a change of meaning for different mentions, it will pick the global most coherent choice. For TAC this limitation is reasonable but might not hold for longer texts.

Compared to Ugglan (TAC2017), which had a machine learned reranker for the linking component, we can see that for English and Spanish, the results are slightly worse for linkage. A future improvement might be a reranker.

## 8 Conclusion

Hedwig, the next generation entity linker, surpasses our old solution in general, however has marginal mixed results when analyzing the details. The platform is solid and has ample opportunities for further development. Particularly, the linker stage can be improved.

## References


Gabriel Bernier-Colborne, Caroline Barriere, and Pierre André Ménard. 2017. Crims systems for the tri-lingual entity detection and linking task. In *TAC*.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *11th conference of the European Chapter of the Association for Computational Linguistics*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web*, pages 927–938. International World Wide Web Conferences Steering Committee.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 621–631.

Zhaochen Guo and Denilson Barbosa. 2014. Robust entity linking via random walks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 499–508. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, Cash Costello, and Sydney Informatics Hub. 2017. Overview of tac-kbp2017 13 languages entity discovery and linking.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 495–500.