

Query Suggestion Using a Transfeme Markov Model

Jonatan Wulcan

Master's thesis
2010:E12



LUND INSTITUTE OF TECHNOLOGY
Lund University

Centre for Mathematical Sciences
Mathematical Statistics

TYP AV DOKUMENT <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> Delrapport	<input type="checkbox"/> Kompendium <input type="checkbox"/> Rapport	DOKUMENTBETECKNING LUTFMS-3189-2012
--	---	---

INSTITUTION Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Box 118, 221 00 Lund
--

FÖRFATTARE Jonatan Wulcan

DOKUMENTTITEL OCH UNDERTITEL Query suggestion using a transfeme Markov model
--

SAMMANFATTNING <p>Given a query from a user, the query suggestion problem aims to suggest another query better suited for the users search intent. In this thesis a theoretical framework for the query suggestion problem is presented. In the context of this framework a transfeme Markov model is presented and tested. The transfeme Markov models is a Markov chain where similar transitions are related to each other. The transfeme Markov model was tested against two baselines, an edit distance model and the query suggestion implementation used in eSales, a commercial e-commerce platform. The results show that the transfeme Markov model performs better than the edit distance model but that further work is necessary on the language model.</p>
--

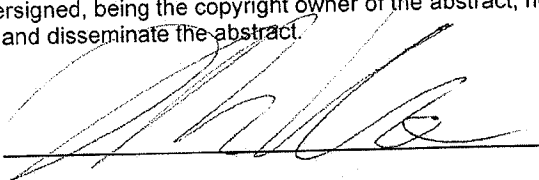
NYCKELORD

DOKUMENTTITEL OCH UNDERTITEL - SVENSK ÖVERSÄTTNING AV UTLÄNDSK ORIGINALTITEL Frågeförslag med transfeme Markov modell

UTGIVNINGSDATUM år 12 mån 05	ANTAL SID 48	SPRÅK <input type="checkbox"/> svenska <input checked="" type="checkbox"/> engelska <input type="checkbox"/> annat
--	------------------------	--

ÖVRIGA BIBLIOGRAFISKA UPPGIFTER	ISSN 2012:E12
--	-----------------------------

I, the undersigned, being the copyright owner of the abstract, hereby grant to all reference source permission to publish and disseminate the abstract.

Signature 

Date 2012-05-10

Abstract

Given a query from a user, the query suggestion problem aims to suggest another query better suited for the users search intent. In this thesis a theoretical framework for the query suggestion problem is presented. In the context of this framework a transfeme Markov model is presented and tested. The transfeme Markov model is a Markov chain where similar transitions are related to each other. The transfeme Markov model was tested against two baselines, an edit distance model and the query suggestion implementation used in eSales, a commercial e-commerce platform. The results show that the transfeme Markov model performs better than the edit distance model but that further work is necessary on the language model.

Populärvetenskaplig sammanfattning

Sökfunktioner spelar en viktig roll i det moderna samhället. Det första man tänker på är kanske de stora internetsökmotorerna så som Google, Yahoo eller Bing men sökfunktioner är också essentiella för många andra tjänster som till exempel internethandel.

Ett vanligt förekommande problem när man använder sökmotorer är att man stavat fel på, eller annat sätt har förvrängt, sin sökfråga. Det finns undersökningar som visar att så mycket som 10%-15% av alla sökfrågor är felstavade. Detta kan bero på att sökfrågor ofta innehåller ovanliga namn eller ord.

För att hjälpa användare att hitta det de söker efter kan sökmotorn visa ett frågeförslag om det anses sannolikt att sökfrågan som användaren angivit är felstavad. Ett modernt tillvägagångssättet för att automatiskt generera frågeförslag är att på olika sätt använda frågeloggar. En frågelogg innehåller historisk information om vilka sökfrågor som användare sökt på.

Detta examensarbete presenterar en metod för automatisk generering av frågeförslag, transfeme modellen. Transfeme modellen bygger på att alla felstavningar kan förklaras med ett antal substitutioner och att sannolikheten för sådana substitutioner är samma oavsett i vilket ord dom förekommer. Ett exempel skulle kunna vara att någon vill söka på **liza marklund** men istället skriver **lisa marklund**. En sådan felstavning skulle kunna förklaras med substitutionen z till s. Transfeme modellen säger då att denna felstavning är lika sannolikt som att man skriver **godsilla** istället för **godzilla** eftersom båda frågeförvrängningarna kan förklaras av samma substitution.

För att uppskatta sannolikheten för olika substitutioner kan man använda en frågelogg. Tanken är att välja sannolikheter för olika substitutioner på så sätt att den frågelogg man har blir så sannolik som möjligt.

För att undersöka hur bra transfeme modellen presterar jämförs transfeme modellen med två andra metoder, en klassisk metod som kallas Levenshtein avstånd och en kommersiell frågeförslagsmetod som kallas eSales. Resultaten visar att transfeme modellen har potential men att man behöver utveckla vissa specifika delar mer för att den ska vara riktigt användbar.

Acknowledgement

This thesis has been produced with the help of many people. My supervisors Björn Brodén, Tomas Malmer, Thomas Raneland and Andreas Jakobsson has given me invaluable insights and provided me with lots of feedback. I also like to send a special thanks to Stefan Ingi Adalbjörnsson for taking an interest in my work and providing me with feedback and insights.

Contents

1	Introduction	9
1.1	Background	9
1.2	Common Query Alterations	10
1.3	Related Work	12
1.4	Thesis Structure	13
1.5	Thesis Contribution	13
2	Theory	15
2.1	Sets and Sequences	15
2.2	Graphs	15
2.3	Probability	16
3	Query Logs	19
3.1	Query Logs	19
4	Corrections	23
4.1	Finding Corrections for Testing	23
4.2	Finding Corrections for Training	24
5	Problem Formulation	27
5.1	The Problem of Query Suggestion	27
6	Models	29
6.1	Transfemes	29
6.2	Transfeme Markov Model	30
7	Inference	33
7.1	Language Model - Click Transitions	33
7.2	Error Model - Transfeme Transitions	33
7.3	Error Model - Noise Considerations	34
8	Computing query suggestions	35
8.1	Time Reversal	35
8.2	Query Suggestion	35
8.3	Preprocessing of Graph Before A* Search	36
8.4	A* Heuristics	36

8.5	A* Node Limit	36
9	Results	37
9.1	Testing and Training Data	37
9.2	Edit Distance	37
9.3	eSales did you mean	38
9.4	Cheating in Language Model	38
9.5	Results	38
10	Discussion	43
10.1	Results	43
10.2	Future Research and Open Questions	43
10.3	Conclusion	45

Chapter 1

Introduction

1.1 Background

In many e-commerce systems the search function has a central role in helping users find the products they desire. Past studies have shown that users often misspell search queries. For instance Cucerzan and Brill [3] has presented figures indicating that 10%-15% of all search queries are misspelled.

To mitigate this problem many modern search engines implement some kind of query suggestion system. A query suggestion system presents an alternative query to the user when it's probable that the alternative query better suits the users search intent. The user then has the choice to see search results for the suggested query instead of the original query.

An example of a query suggestion system can be seen in figure 1.1.

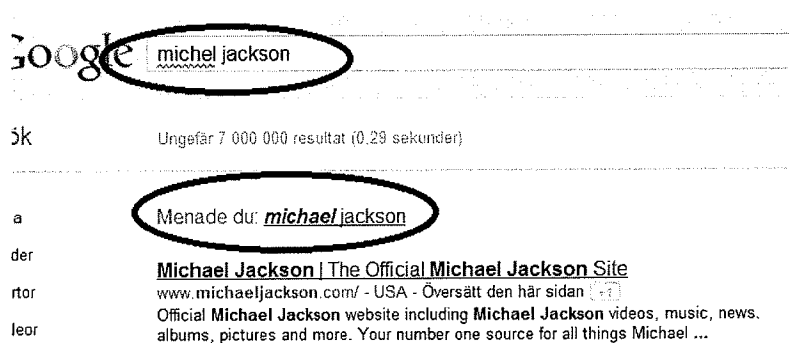


Figure 1.1: Screenshot of Googles query suggestion system. If a user thinks that the suggested query better suits her search intent she may click on the suggested query.

The problem studied in this thesis is the offline query suggestion problem. When studying the offline query suggestion problem one only suggests query after the user has completed her query. In contrast when studying the online query suggestion problem one suggests queries as the user types her query. The

two problems are related and many methods are applicable to both problems.

1.2 Common Query Alterations

When considering the problem of query suggestion it's important to take into consideration some real world query alterations. These query alterations has been observed in one of Sweden's largest e-commerce sites.

Spelling

A common type of alteration occurs when the user simply don't know how the query is spelled. This is common for instance with infrequent words, new words and names.

johrdalen -> jodalen
en shopoholic i new york -> en shopaholic i new york
beurolsconi -> berlusconi
dyskalkulie -> dyscalculia
güttler trädets -> güttler trädets
papillion -> papilion

Key slip

Another type of query alteration occurs when the user slips on the keyboard. That is the user means to write one thing but writes another.

tgner -> tegner
an ders wahlgren -> anders wahlgren
imperuim -> imperium
folosofi mumun -> filosofi mumun
exporing gypsiness -> exploring gypsiness
ordbok spansak -> ordbok spanska
science fiction -> science fiction
fiksar malawi -> fiskar malawi
a beginnersguide the world -> a beginners guide the world
katemosse -> kate mosse

Missing suffix

A query alteration where query suffixes has been removed is also quite common. This might occur when the user tries to use an auto complete feature or she might just expect the search engine to understand the query without the query suffix.

sofie hex -> sofie hexenberg
för matlagning ny -> för matlagning nybörjare

Conjugation

It's common that users choose a conjugation of a word which yields bad results.

trafikskolan turkiska -> trafikskola turkiska
muscle orthopedic -> muscle orthopedics
samtala -> samtal

Synonyms/Language

Another query alteration occurs when users use synonyms of words or change the language of a word.

tuttar -> breast

Word splits/concatenations

Sometimes a user splits two word or concatenates two words incorrectly. Note that the last query also contains two key slips.

int14 -> int 14
priceguide -> price guide
rebell ledaren -> rebelledaren
swaptjing -> swamp thing

Guessing

When a user doesn't remember the title of a book or a name of an author she might guess parts of the query. This yields another set of query alterations.

friluftspedagogik -> friluftslivspedagogik
håkan östergren -> håkan östlund
svinänglarna -> svinänglorna

Search Strategy

Sometimes when a user receives bad result from a search engine she might change her search strategy. For instance a user searches for

pizza rasist

and receives no result. The user then instead searches for

pizza äter

and then receives the desired result.

Jag är inte rabiät. Jag äter pizza : en bok om Sverigedemokraterna
by Niklas Orrenius

Another user might search for

dark series

and receive too many results; the query is underspecified. She may then change search strategy and instead search for the authors name

christine feehan

which yields the desired result.

Dark Secret

by Christine Feehan

Dark Curse - A Carpathian Novel

by Christine Feehan

Dark Prince

by Christine Feehan

Conclusions

In many of the query alterations presented above one can explain the query alteration with substitutions such as 'll' to 'l', 'k' to 'c' or 'ui' to 'iu'. Such substitutions are in this thesis named transfemes and are introduced in chapter 6.

There are also more complicated query alterations such as the search strategy and missing suffix alterations that cannot easily be explained by substitutions. Those query alteration will considered by this thesis from here on.

1.3 Related Work

As far as the author knows the companies behind the major search engines has not made public the query suggestion methods they use. Ma et al writes that due to commercial reasons few public papers have been released that unveils the methods Google and other big search engines adopts [9].

However there exists other papers on the subject. For instance Ma et al proposes a query suggestion model based on bipartite graphs and matrix factorization [9]. Mei et al also use a bipartite graph and use hitting times to generate query suggestions [10].

A more classical concept, initially proposed by Damerau [4] and Levenshtein [8], is the edit distance. The edit distance has been widely used in generic spelling correction [5]. The edit distance introduces a set of valid string operations, deletion of a character, the substitution of a character and the insertion of a character. The edit distance between two strings is the minimum number of such operations needed to transform one string to the other.

There are many ways of utilizing the edit distance when doing query suggestions. A simple one is to find all queries within a certain edit distance radius (e.g 3) of the misspelled query and then somehow rank those queries. The top ranked query would then be presented to the user as a query suggestion.

1.4 Thesis Structure

This thesis starts by introducing notation for some well known mathematical objects in chapter 2. In the next chapter, chapter 3, the query logs are defined and discussed. Chapter 4 presents two methods to extract query corrections from query logs. Once the query corrections are defined one can formally define the problem of query suggestion, which is done in chapter 5. Further on, in chapter 6, a mathematical model to describe query alterations is presented. The following chapter, chapter 7, describes how to choose parameters for the model introduced in chapter 6. Chapter 8 describes a method to do query suggestion using the model presented in chapter 6. In chapter 9 the method is evaluated and results are presented. Finally in chapter 10 the results are discussed and some ideas for further work is presented.

1.5 Thesis Contribution

In chapter 3 a mathematical description of query logs is presented. Chapter 4 introduces two methods of correction extraction from such query logs. Further on chapter 5 presents two novel evaluation metrics in relation to the established recall@1 metric.

In chapter 6 the transference Markov model is presented. The model builds on earlier work by Duan et al [5], presenting a slightly adjusted form of this model in a Markov framework.

Then, in chapter 7, a method of inference is presented, which may also be viewed as an extension of the method in [5], incorporating also a noise consideration.

In chapter 8, a method of using A* search to find corrections is presented. The method was inspired by [5], however the A* heuristics had to be adapted according to the adaptations made in the model and the graph preprocessing step was added.

Chapter 2

Theory

The following chapter is meant to establish notation for some well known mathematical objects. It's not meant to be a complete in any way and it's doubtful if the reader from these definition would be able to understand a concept that is not already known to the reader.

2.1 Sets and Sequences

Finite sequences are used to define the strings the query logs in chapter 3.

Definition 2.1. Let $S(\mathbb{X})$ denote all finite sequences with elements in an arbitrary set \mathbb{X} . Also if $a, b \in S(\mathbb{X})$ let $a + b$ denote the concatenation of a and b . Finally if $a \in S(\mathbb{X})$ let $|a|$ denote the number of elements in a .

Example 2.2. The triple (x_0, x_1, x_2) is an example of an element in $S(\mathbb{X})$ if $x_i \in \mathbb{X}$ for all $i \in \{0, 1, 2\}$.

The power set is introduced to describe probability spaces.

Definition 2.3. The power set of a set, \mathbb{X} , is the set of all subsets of \mathbb{X} . Let $\mathcal{P}(\mathbb{X})$ denote the power set of \mathbb{X} [6].

The indicator function a useful tool to compactly describe relations. It's used in chapter 4 and chapter 5.

Definition 2.4. Let $I(x)$ denote the indicator function. The indicator function is 1 when the statement x is true and 0 otherwise.

There are some confusion whether to include zero in the natural numbers or not. In this thesis zero is included.

Definition 2.5. Let \mathbb{N} denote the natural numbers $\{0, 1, 2, 3, \dots\}$.

2.2 Graphs

Graph theory is a well established branch of mathematics. In this thesis graph theory is used in chapter 8 to solve the most probable path problem.

Definition 2.6. A weighted directed graph is a triple (V, A, w) where V is an arbitrary set called the nodes, A is an subset of $V \times V$ called the arcs and w is a function $w : A \rightarrow \mathbb{R}^+ \cup \{\infty\}$ called the weights. An arc $a \in A$ is said to start in a_0 and end in a_1 .

Remark 2.7. Sometimes the weighted directed graphs are defined to allow negative weights. However in this thesis all weights are always positive.

In chapter 8 the most probable path problem is transformed into a shortest path problem. The following definitions define a shortest path.

Definition 2.8. A path in a weighted directed graph, (V, A, w) , is a sequence of arcs, $(a_0, a_1, a_2, \dots, a_n)$ such that if a_i ends at $v \in V$ then a_{i+1} starts in v . A path is said to start in the node that a_0 starts in and a path is said to end in the node that a_n ends in. The path length of a path is defined as $\sum_{i=0}^n w(a_i)$.

Definition 2.9. Given a weighted directed graph, (V, A, w) , a shortest path between to nodes $a, b \in V$ is a path starting in a and ending in b with minimum length.

2.3 Probability

It's recognized that the fundamental datum in probability theory is a probability space [6].

Definition 2.10. A probability space is a triple (Ω, \mathcal{F}, P) where the following holds.

- Ω is a non-empty set called the sample space.
- $\mathcal{F} \subset \mathcal{P}(\Omega)$ and \mathcal{F} is a σ -algebra. \mathcal{F} is called the set of events.
- P is a function from \mathcal{F} to $[0, 1]$ and P is a probability measure.

Using the definition of probability space one may define the random variables.

Definition 2.11. Given a probability space (Ω, \mathcal{F}, P) , a random variable X is a triple $(\Omega_X, \mathcal{F}_X, f_X)$ where the following holds.

- Ω_X is a non-empty set called the state space.
- $\mathcal{F}_X \subset \mathcal{P}(\Omega_X)$ and \mathcal{F}_X is a σ -algebra.
- f_X is a measurable function from Ω to Ω_X .

If $A \in \mathcal{F}_X$ then $X = A$ is defined as the set $f_X^{-1}(A)$.

Using these notations one may define a Markov chain [1].

Definition 2.12. A probability space (Ω, \mathcal{F}, P) is a (time-homogeneous) Markov chain if the following hold.

- There exist a countable state space S such that $\Omega = S \times S \times S \dots$

- The triples $X_i = (S, \mathcal{P}(S), f_i)$ where $f_i(x) = x_i$ are random variables (f_i are measurable).
- For every $x \in \Omega$ and $i \in \mathbb{N}$ the following holds $P(X_{i+1} = x_{i+1} | X_i = x_i, X_{i-1} = x_{i-1}, \dots, X_0 = x_0) = P(X_{i+1} = x_{i+1} | X_i = x_i)$.
- For all $x, y \in S$, $P(X_{i+1} = x | X_i = y) = P(X_i = x | X_{i-1} = y)$.

If $x, y \in S$ the probability $P(X_{i+1} = y | X_i = x)$ is denoted p_{xy} and called a transition probability.

Denote the Markov chain with state space S and transition probabilities p , $\text{Markov}(S, p)$. For a particular markov chain some states can absorbing.

Definition 2.13. Given a Markov chain $X = \text{Markov}(S, p)$ a state is $s \in S$ is absorbing if $P(X_2 = s | X_1 = s) = 1$.

Intuitively a state is absorbing if once you enter it you can never leave. Each Markov chain can also be described by a matrix, the transition matrix.

Definition 2.14. Given a Markov chain $X = \text{Markov}(S, p)$ and a bijection $\phi : S \rightarrow \{0, 1, 2, \dots, |S| - 1\}$, define the corresponding transition matrix as,

$$\begin{pmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,|S|-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,|S|-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{|S|-1,0} & p_{|S|-1,1} & \cdots & p_{|S|-1,|S|-1} \end{pmatrix},$$

where $p_{\phi(a), \phi(b)} = p_{a,b}$ for all $a, b \in S$.

Each Markov chain can also be described by a weighted directed graph.

Definition 2.15. Given a Markov chain $X = \text{Markov}(S, p)$ the corresponding weighted graph (S, A, w) is defined by the following conditions.

- An arc $a \in S \times S$ is in A iff $p_{a_0, a_1} > 0$.
- $w(a) = p_{a_0, a_1}$ for all $a \in A$.

In chapter 8 the most probable path problem is discussed. The following definitions define a most probable path [1].

Definition 2.16. A path in a Markov chain, $X = \text{Markov}(S, p)$, is a sequence of states $s_0, s_1, s_2, \dots, s_n$. The probability of a path is defined as

$$P(X_0 = s_0, X_1 = s_1 \dots X_n = s_n | X_0 = s_0).$$

Definition 2.17. Given a Markov chain $X = \text{Markov}(S, p)$ and two states $a, b \in S$. A most probable path between a and b is a path where $s_0 = a$ and $s_n = b$ with maximal probability.

Chapter 3

Query Logs

3.1 Query Logs

To test and train query suggestion models one can make use of query logs. The following section defines what is meant by a weblog.

A basic component in the query logs are characters and strings.

Definition 3.1. Let the set of characters, \mathbb{C} , be a set of finitely many elements. Each element represent a character.

Definition 3.2. Let \mathbb{C} be a set of characters. The set of strings \mathbb{Q} is defined as $S(\mathbb{C})$.

Another component of the user logs are the actions. One action describes either that the user enters a search query or that a user clicks on a search result. A special start action is also added to describe the start of a new search intent.

Definition 3.3. Let \mathbb{Q} be the set of strings. The set of actions, \mathbb{A} , is defined as $\mathbb{Q} \cup \{[\text{CLICK}], [\text{START}]\}$. The extra element $[\text{CLICK}]$ represent that a select or purchase action has occurred and the extra element $[\text{START}]$ represent that a new search intent has started.

A user may perform more than one action each time she visits an e-commerce site. An element in the set of sessions describes one such visit.

Definition 3.4. Let \mathbb{A} be a set of actions. The set of sessions, \mathbb{H} , is defined as $S(\mathbb{A})$. Each session describe one visit from a user.

Finally a query log consist of a long sequence of sessions from different users. These logs are often collected during several months.

Definition 3.5. Let \mathbb{H} be the set of sessions. The set of query logs, \mathbb{L} , are defined as $S(\mathbb{H})$.

An example might help clarify what an element in \mathbb{L} might consist of.

Example 3.6. This example of a query log consist of 8 sessions. A typical query log would consist of at least 100.000 such sessions.

```

SESSION
  [START]
  john allen paulos
SESSION
  [START]
  trä ute
  [CLICK]
  [START]
SESSION
  [START]
  personliga val som skapar mirakler
  the secret : kraften
  [CLICK]
  [START]
SESSION
  [START]
  yarden
  [CLICK]
  [START]
  hund
  [CLICK]
  [START]
SESSION
  [START]
  charlesbukowski
  charles bukowski
  [CLICK]
  [START]
SESSION
  [START]
  hund
  [CLICK]
  [START]

```

In the first session the user searches for john allen paulos and then abandons the search. The second session contains a single query and then a select or purchase event. The third session contains two unrelated searches and then a click. The fourth session contains two search intents, one for yarden and one for hund. The fifth session contains the only query correction in the query log, charlesbukowski to charles bukowski.

To make use of a query log one can extract query corrections from it, the process of extracting query corrections is described in chapter 4.

A quite common phenomenon in query logs are the duplication of queries [11]. In this thesis this phenomenon was unwanted and thus all such duplications were removed. Also all query were converted to lower case.

Unfortunately the query logs contains noise. The most important source of noise is that a user may click on a product that isn't in the search results, for instance for product displayed in a top list. Another source of noise is that all sessions may not originate from human user, different computer programs regularly crawl the internet to harvest information [11]. The methods this thesis use to handle noise is described in 4.2 and 7.3.

Chapter 4

Corrections

In this chapter two novel methods to find corrections in query logs are presented. A correction is defined in the following manner.

Definition 4.1. Let \mathbb{Q} be the set of strings. The set of all corrections, \mathbb{K} , is defined as $\mathbb{Q} \times \mathbb{Q}$. The first string represent a potentially misspelled query and the second string represents the corresponding correctly spelled query.

The set of all weighted correction lists is defined in the following manner.

Definition 4.2. Let \mathbb{K} be the set of all corrections. The set of all weighted correction lists, \mathbb{T} , is defined as all functions $t : \mathcal{P}(\mathbb{K}) \rightarrow [0, 1]$ such that $(\mathbb{K}, \mathcal{P}(\mathbb{K}), t)$ form a probability space.

4.1 Finding Corrections for Testing

This method of finding corrections has the advantage that it's simple to understand and therefore fitting for testing purposes. The downside is that this method yields a very noisy result. From manual inspection of the result the author estimates that there may be as high as 80% noise in a correction list obtained using this method, making this method unsuitable for model training.

Definition 4.3. Let $h \in \mathbb{H}$ be a session and $i, j \in \mathbb{N}$ such that $0 \leq i \leq j < |h| - 1$. Then h_j is a test correction for h_i if the following hold.

- There is no k , $i \leq k \leq j$ such that $h_k \in \{[\text{CLICK}], [\text{START}]\}$.
- $h_{j+1} = [\text{CLICK}]$.

Note that given a $h \in \mathbb{H}$ and an i such that $0 \leq i < |h|$ then if h_i has a test correction it must be unique. Given a query log one may find a weighted list of corrections with the weight proportional to the number of occurrences in the query log.

Definition 4.4. Let \mathbb{L} be the set of query logs and \mathbb{T} be the set of all weighted correction lists. Define the test corrections in a query log, $t = T_{\text{test}}(l) : \mathbb{L} \rightarrow \mathbb{T}$,

such that $t(\{k\})$ is proportional to the number of times k_1 is observed as a test correction for k_0 in l .

4.2 Finding Corrections for Training

When training models one may use a slightly more complex method to obtain a weighted correction list with a smaller amount of noise.

Let \mathbb{L} be the set of query logs and \mathbb{A} the set of actions. Given $l \in \mathbb{L}$ define a Markov chain $\text{Markov}(\mathbb{A}, p)$ where p_{xy} is defined by the following conditions.

- If $x = y$ and $\sum_{h \in l} \sum_{i=1}^{|h|-1} I(h_i = x) = 0$ then $p_{xy} = 1$.
- If $x \neq y$ and $\sum_{h \in l} \sum_{i=1}^{|h|-1} I(h_i = x) = 0$ then $p_{xy} = 0$.
- If $x = y$ and $x = [\text{CLICK}]$ then $p_{xy} = 1$.
- If $x \neq y$ and $x = [\text{CLICK}]$ then $p_{xy} = 0$.
- Otherwise

$$p_{xy} = \frac{\sum_{h \in l} \sum_{i=1}^{|h|-1} I(h_i = x) I(h_{i+1} = y)}{\sum_{h \in l} \sum_{i=1}^{|h|-1} I(h_i = x)}.$$

The states in this Markov chain is the actions defined in 3.3. In this Markov chain two kinds of states are absorbing. The first kind is if the state $x \in \mathbb{A}$ satisfy $\sum_{h \in l} \sum_{i=1}^{|h|-1} I(h_i = x) = 0$. These are states that one never observes a next state for. The other kind of state that is absorbing is the $[\text{CLICK}]$ state.

One may define the probability that x is a training correction for y as $\varphi(x, y) = \sum_{i=1}^{\infty} P(X_i = x, X_{i+1} = [\text{CLICK}] | X_1 = y)$.

To compute these probabilities for all state pairs x and y one can write φ in the following way,

$$\varphi(x, y) = \sum_{i=1}^{\infty} P(X_i = x | X_1 = y) P(X_{i+1} = [\text{CLICK}] | X_i = x)$$

Using that $P(X_{i+1} = [\text{CLICK}] | X_i = x)$ is the same for all i one may write this as,

$$\varphi(x, y) = P(X_2 = [\text{CLICK}] | X_1 = x) \sum_{i=1}^{\infty} P(X_i = x | X_1 = y).$$

To compute the infinite sum for all pairs of x and y one may write the sum in matrix form using the transition matrix, A , corresponding to the Markov chain presented above. The sum $\sum_{i=1}^{\infty} P(X_i = x | X_1 = y)$ equals to the (y, x) element in the matrix $\sum_{i=0}^{\infty} A^i$. One may approximate this sum using the first k terms $\sum_{i=0}^{\infty} A^i \approx \sum_{i=0}^k A^i$. In this thesis k was set to 20 and it's unclear if this provides a good approximation.

Define the training corrections in the following manner.

Definition 4.5. Define the training corrections in a query logs, $t = T_{\text{training}}(l) : \mathbb{L} \rightarrow \mathbb{T}$, such that $t(\{k\})$ is proportional to $\varphi(k_1, k_0)$ times the number of occurrences of k_0 in the query log if $\varphi(k_1, k_0)$ is above some threshold (for instance 0.5) and zero otherwise. Also if $k_0 = k_1$ let $t(\{k\})$ be zero.

Here follows an example of the process.

Example 4.6. Given the query log in example 3.6 one would have the following Markov transitions.

```
[CLICK] -> [CLICK]: 1.0
[START] -> john allen paulos: 0.14
[START] -> trä ute: 0.14
[START] -> personliga val som skapar mirakler: 0.14
[START] -> yarden: 0.14
[START] -> hund: 0.28
[START] -> charlesbukowski: 0.14
john allen paulos -> john allen paulos: 1.0
trä ute -> [CLICK]: 1.0
personliga val som skapar mirakler -> the secret : kraften: 1.0
the secret : kraften -> [CLICK]: 1.0
yarden -> [CLICK]: 1.0
hund -> [CLICK]: 1.0
charlesbukowski -> charles bukowski: 1.0
charles bukowski -> [CLICK]: 1.0
```

If one enumerates the states as in the order they appear above one would get the following transition matrix.

$$A = \begin{pmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.14 & 0.14 & 0.14 & 0 & 0.14 & 0.28 & 0.14 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Calculating $\sum_{i=0}^{20} A^i$ yields the following result.

$$\sum_{i=0}^{20} A^i = \begin{pmatrix} 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 15.68 & 1 & 2.80 & 0.14 & 0.14 & 0.14 & 0.14 & 0.28 & 0.14 & 0.14 & \\ 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 19 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The only non zero elements that is not on the diagonal and is not related to the [CLICK] or [START] states are.

```
personliga val som skapar mirakler -> the secret : kraften: 1.0
charlesbukowski -> charles bukowski: 1.0
```

Since both queries are observed once in the query log and both target queries has 100% clickthrough rate the induced weighted correction list is as follows.

```
personliga val som skapar mirakler -> the secret : kraften: 0.5
charlesbukowski -> charles bukowski: 0.5
```

Chapter 5

Problem Formulation

5.1 The Problem of Query Suggestion

To define the problem of query suggestion one need to consider how one should evaluate a query suggestion function. Lets start with defining the query suggestion function.

Definition 5.1. Let \mathbb{Q} be the set of strings and \mathbb{L} be the set of query logs. The set of all query suggestion functions, \mathbb{D} , is the set of all function $d : \mathbb{Q} \times \mathbb{L} \rightarrow \mathbb{Q}$.

The first input argument to a query suggestion function is the potentially misspelled query from a user. The second argument is a training query log to train the query suggestion function. The query suggestion function should output the corrected query if the query was likely misspelled or the same query if the query was likely not misspelled.

One can define different evaluation metrics on the query suggestion functions. The recall@1 (R@1) metric [5] is defined in the following way. One may also define two variants of the recall@1 function.

Definition 5.2. Given two query logs $l_{\text{training}}, l_{\text{test}} \in \mathbb{L}$ and a query suggestion function $d \in \mathbb{D}$. Let P_{test} denote the test corrections for l_{test} , that is $T_{\text{test}}(l_{\text{test}})$. Define the functions $\text{R@1}, \text{R@1}_{\text{same}}, \text{R@1}_{\text{diff}} : \mathbb{L} \times \mathbb{L} \times \mathbb{D} \rightarrow [0, 1]$ as,

$$\begin{aligned} \text{R@1} \quad (l_{\text{training}}, l_{\text{test}}, d) &= \sum_{k \in K} I(d(k_0, l_{\text{training}}) = k_1) P_{\text{test}}(k), \\ \text{R@1}_{\text{same}} \quad (l_{\text{training}}, l_{\text{test}}, d) &= \sum_{k \in K} I(d(k_0, l_{\text{training}}) = k_1) P_{\text{test}}(k | k_0 = k_1), \\ \text{R@1}_{\text{diff}} \quad (l_{\text{training}}, l_{\text{test}}, d) &= \sum_{k \in K} I(d(k_0, l_{\text{training}}) = k_1) P_{\text{test}}(k | k_0 \neq k_1). \end{aligned}$$

Here $k_0 = k_1$ denotes the subset of \mathbb{K} where $k_0 = k_1$ and $k_0 \neq k_1$ denotes the subset of \mathbb{K} where $k_0 \neq k_1$

To formally define the query suggestion problem in this context one needs to somehow summarize these three evaluation metrics.

Definition 5.3. A function $g : [0, 1] \times [0, 1] \times [0, 1] \rightarrow [0, 1]$ is an evaluation summary if its increasing in all its arguments.

The choice of evaluation summary function should depend on many things outside the scope of this thesis. No evaluation summary will be proposed in this thesis.

Using the evaluation summary function one can define the problem of query suggestion.

Definition 5.4. Given two query logs $l_{\text{training}}, l_{\text{test}} \in \mathbb{L}$ and an evaluation summary g . Find a query suggestion function $d \in \mathbb{D}$ such that

$$g(\text{R@1}(l_{\text{training}}, l_{\text{test}}, d), \text{R@1}_{\text{same}}(l_{\text{training}}, l_{\text{test}}, d), \text{R@1}_{\text{diff}}(l_{\text{training}}, l_{\text{test}}, d))$$

is maximized.

Remark 5.5. In the above definition, it is implicitly assumed that the query suggestion function has been selected independently of the test data.

To clarify the definition of the query suggestion function, an example might be helpful.

Example 5.6. A trivial examples of a query suggestion function would be the function $d(q, l) = q$. This query suggestion function would have perfect score on R@1_{same} , $\text{R@1}_{\text{same}} = 1$, but since it never makes any guesses it would score zero on R@1_{diff} , $\text{R@1}_{\text{diff}} = 0$. Since the correctly spelled query is much more usual than the incorrect spelled ones the result of R@1 might be good.

Chapter 6

Models

The model described in this chapter was inspired by and is similar to the model used by Duan, Huizhong and Hsu, Bo-June (Paul) [5], which in turn was inspired by work done in grapheme to phoneme transformation, especially a joint sequence model by Bisani and Ney [2].

6.1 Transfemes

Transfemes[5] are used to describe how a user might transform a correctly spelled string to a misspelled string. The definition presented here differs on minor details from the definition used by Duan et al.

Definition 6.1. Let \mathbb{Q} be the set of strings. A transfeme u is an element in $\mathbb{Q} \times \mathbb{Q}$ such that u_0 and u_1 has no common prefix or suffix. Define \mathbb{U} as the set of all transfemes. Each transfeme, u , is also a relation where a string $q_0 \in \mathbb{Q}$ is u -related to $q_1 \in \mathbb{Q}$ iff u_0 is a substring of q_0 and such a substring can be replaced by u_1 so the resulting string is equal to q_1 .

Example 6.2. The transfeme (' ', ') $\in \mathbb{U}$ can explain the following transitions.

```
int 14 -> int14
price guide -> priceguide
kate mosse -> katemosse
```

Example 6.3. The transfeme ('ph', 'f') $\in \mathbb{U}$ can explain the following transition.

```
philosophy -> filosophy
philanthropy -> filanthropy
```

Each a transition can be explained by exactly one transfeme but a transfeme can explain an infinite number of transitions.

6.2 Transfeme Markov Model

The general framework in which this thesis approach the query suggestion problem is a Markov model. The states in this Markov model is defined in the following way.

Definition 6.4. Let \mathbb{Q} be the set of strings. Define the set of states, \mathbb{W} , as $\mathbb{Q} \times \mathbb{Q} \cup \{\text{[START]}, \text{[CLICK]}\}$. Given an element $w \in \mathbb{Q} \times \mathbb{Q} \subset \mathbb{W}$ let w_{fixed} denote the first string and w_{variable} denote the second string.

The string w_{fixed} represents the part of a query that has already been corrected and the string w_{variable} represent the part of a query that is still potentially misspelled. The query is always corrected from left to right so the full query could be acquired by concatenation, $w_{\text{fixed}} + w_{\text{variable}}$.

Example 6.5. These are some examples of elements in \mathbb{W} . The string before | is w_{fixed} and the string after is w_{variable} .

```
[START]
[CLICK]
Liz|a markland
Jo |Nesbo
Jo Nesbø|
Jonat|han Wulcan
Jonat|an Wulcan
```

In this model we imagine that a user starts of at the start state and then chooses a possibly misspelled query to transition to. At this point the whole query would be in w_{variable} and w_{fixed} would be empty. Then the user would progressively correct her query from left to right. Each correction is represented by a transition where the correction and everything to the left of the correction is moved from the w_{variable} string to the w_{fixed} string. Finally, when the user has completed her corrections she will find her result and transition to the click state. From the click state the user will start again from the start state and repeat the process indefinitely. An example might help to clarify the process.

Example 6.6. This is an example of what a random walk in the transfeme Markov model might look like. The string before | is w_{fixed} and the string after is w_{variable} .

```
[START]
|Lisa markland
Liz|a markland
Liza marklu|nd
[CLICK]
[START]
|JoNesbo
Jo |Nesbo
Jo Nesbø|
```

[CLICK]

...

The transfeme Markov model is defined in the following way.

Definition 6.7. Let \mathbb{W} be the set states, \mathbb{Q} the set of strings and \mathbb{U} the set of transfemes. A Markov chain $X = \text{Markov}(\mathbb{W}, p)$ is a transfeme Markov model with parameter $\alpha \in \mathbb{R}^+$ iff

- $P([\text{START}]|[\text{CLICK}]) = 1$.
- Given $x, y \in \mathbb{Q} \times \mathbb{Q} \subset \mathbb{W}$. If there is no transfeme such that x is related to y then $P(X_1 = y|X_2 = x) = 0$.
- Given $x_a, x_b, y_a, y_b \in \mathbb{W}$ and $u \in \mathbb{U}$ such that x_a u -related to x_b and y_a u -related to y_b then the following must hold $P(X_1 = x_b|X_2 = x_a) = P(X_1 = y_b|X_2 = y_a)$. Denote this probability $P(u)$.
- $\sum_{u \in \mathbb{U}} P(u)^\alpha = 1$.

Remark 6.8. Note that $P(X_1 = x_b|X_2 = x_a) = P(X_1 = y_b|X_2 = y_a)$ makes a statement about where we came from and not where we are going to. This is important since the forward probability must somehow take into account the popularity of different queries but the backward probabilities only need to consider the probabilities of misspellings.

Remark 6.9. The statement about $P(X_1 = x_b|X_2 = x_a) = P(X_1 = y_b|X_2 = y_a)$ makes it difficult to choose these probabilities in such a way that the sum of all transitions from a states is 1. This is because for any transfeme u with $P(u) > 0$ one can always find a state $w \in \mathbb{W}$ such that w has arbitrarily many transitions explained by u , thus there is always a state where the sum of transitions is greater than 1. Therefore from here on after I will disregard this restriction. Since the shortest path algorithm used in 8.2 works fine on general graphs, this is not a big issue. To prevent the transfeme probabilities from being unbounded the last statement in the definition was added. The α parameter controls in general how low/high the transfeme probabilities should be. The most extreme example would be to set $\alpha = \infty$, then you could set $P(u) = 1$ for all $u \in \mathbb{U}$. The α parameter is equivalent to the fudge factor used by Duan et al [5]. The model has been tested with different α values in chapter 9.

Chapter 7

Inference

This chapter explains how given a query log $l_{\text{training}} \in \mathbb{L}$ one can choose transition probabilities for the transfeme Markov model. The training method described is essentially the same as was used by Duan et al [5].

7.1 Language Model - Click Transitions

The language model is the probabilities $P(X_i = x | X_{i+1} = [\text{CLICK}])$. These probabilities represent the popularity of different search queries. Given a query log one can estimate these probabilities by simply creating a histogram of all queries that immediately precedes a [CLICK].

7.2 Error Model - Transfeme Transitions

To estimate the transfemes probabilities $P(u)$ one can make use of the training corrections, $T_{\text{training}}(l)$ of a query log l . The tricky part about this is that these corrections only correspond to the first state after start and the final state before click. Everything in between is hidden.

Example 7.1. Given a correction (lisamarklund, liza marklund) the following can be assumed about the random walk in the transfeme Markov model.

```
[START]
|lisamarklund
???
```

...

```
liza |marklund
[CLICK]
...
```

The question marks represent unknown states.

However if one already had the transfeme probabilities one could calculate the probabilities of every transition given that one starts in a given state and ends up in a given state. Inspired by this idea one may define the evidence of a transfeme. To do that one must first define the probability $P(u|k)$.

Definition 7.2. Given a transfeme u , a correction $k \in \mathbb{K}$ and transfeme probabilities P . Define the probability $P(u|k)$ as the sum of all u -transition probabilities given that the random walk starts at k_0 and ends up in k_1 .

The probabilities $P(u|k)$ can be calculated efficiently using a forward-backward algorithm [2]. The forward-backward algorithm use dynamic programming to calculate forward probabilities and backward probabilities in two separate phases. The probabilities are then combined to calculate $P(u|k)$.

Definition 7.3. Given a query log, $l_{\text{training}} \in \mathbb{L}$, a transfeme u and transfeme probabilities P . Let $t = T_{\text{training}}(l_{\text{training}})$. One may define the evidence of u as

$$e(u) = \sum_{k \in \mathbb{K}} P(u|k)t(k).$$

By normalizing the evidence, one may obtain updated transfeme probabilities $p^*(u)$,

$$p^*(u) = \frac{e(u)}{\sum_{u \in U} e(u)}.$$

By initiating the transfeme probabilities with an uniform distribution over all transfemes and then iterating this process one may obtains the transfeme probabilities. Finally, to control the α norm mentioned in definition 6.7, the final probabilities were calculated using the following formula,

$$p(u) = p^*(u)^{\frac{1}{\alpha}}.$$

Since $\sum_{u \in U} p^*(u) = 1$ this will make sure that $\sum_{u \in U} p(u)^\alpha = 1$.

7.3 Error Model - Noise Considerations

The method described for training the error model doesn't seem to be very robust with respect to outliers. Since the training corrections often tends to be quite noisy this is an issue. To mitigate this, one may discard all the transfeme for which one only finds evidence in a single correction.

Further on one may discard the corrections where the shortest path between the start and the click states is longer than 5 states.

After these pruning steps has been applied one may redo the training of error model to obtain better transfeme probabilities.

Chapter 8

Computing query suggestions

8.1 Time Reversal

Until now I have only discussed and modeled the probabilities backwards in time. For the transition probabilities to be useful when computing query suggestions one would need transition probabilities forward in time.

Fortunately there exist a theorem regarding time reversals of Markov chains.

Theorem 8.1. *Let X be an irreducible Markov chain and π be a stationary distribution on this Markov chain. Then*

$$P(X_{i+1} = a | X_i = b) = \frac{\pi(a)}{\pi(b)} P(X_i = b | X_{i+1} = a).$$

So to time reverse a Markov chain one needs a stationary distribution. Unfortunately, since there are an infinite number of states in the Markov chain this stationary distribution cannot be computed. Instead, one may crudely approximate the stationary distribution with a uniform distribution.

8.2 Query Suggestion

Given a query q one would like to find the best suggestion, c , to present to the user. Using the transfeme Markov model this would be analogous to finding a c such that $P(X_i = c | X_{i+1} = [\text{CLICK}], X_2 = q)$ is maximized.

However since the number of states in the Markov model is infinite it would be infeasible to calculate this probability. To approximate this probability, one may instead calculate the most probable path between q and $[\text{CLICK}]$ and let the query suggestion be the state before $[\text{CLICK}]$ in this path.

Using the logarithmic number system, one can transform the most probable path problem to a shortest path problem. The shortest path problem may be efficiently solved using e.g the A* algorithm [7].

8.3 Preprocessing of Graph Before A* Search

When the A* algorithm traverses a node it evaluates all the edges from the node. This procedure can be ineffective since there may be many high cost edges that are unlikely to be part of the shortest path.

To mitigate this issue, one can split nodes in the graph such that each node has edges with similar sizes. Zero edges has to be added between the newly created nodes to preserve the shortest path in the graph. The edges in a node were split according to the rank of the transfeme corresponding to the edge.

A transfeme, u , is larger than another transfeme, v iff $u_1 = v_1$ and $P(u) > P(v)$. The rank of a transfeme, u , is defined as the number of transfemes that is larger than u .

8.4 A* Heuristics

When using the A* algorithm it's important to provide a A* heuristic function $h : \mathbb{W} \rightarrow \mathbb{R}^+$ that approximates the shortest path between a node and the [CLICK] node. If h always underestimates the shortest path the result of the A* search is guaranteed to be optimal, otherwise it is not. Such function is called admissible heuristics. It's important that the evaluation of the heuristic function is fast as it's called for each node that the A* algorithm traverses.

The heuristic function used in this thesis can be divided into three terms. The first term is the prefix heuristic, h_{prefix} . Denote the set of all states where w_{fixed} has prefix q as $\text{Prefix}(q)$. The prefix heuristic is,

$$h_{\text{prefix}}(w) = \min_{q \in \text{Prefix}(w_{\text{fixed}})} -\log(P(X_2 = [\text{CLICK}] | X_1 = q)).$$

The second term is the next edge heuristic, h_{nextedge} and its defined as the minimum of all the edges from a state w .

The third term is the variable length heuristic, h_{varlen} . It penalizes states with a long w_{variable} . It's defined as a constant β times the length of w_{variable} . This heuristic is not admissible. The β parameter was set to 2.1 using manual tuning on a small dataset.

8.5 A* Node Limit

The A* algorithm has unbounded time and memory complexity when operating on an infinite graph. To mitigate this issue one may simply give up after a fixed number of nodes. The number of nodes before giving up was set to 10000. When this happens the query suggestion function returns the same query as the input query as that is a likely a good guess.

Chapter 9

Results

9.1 Testing and Training Data

The transfeme Markov model was tested on 100 days of data from logged in user of a popular web shop in Sweden. The data was splitted into a training query log, **dataset1**, and a testing query log, **dataset2**. Both sets are equal in the number of sessions and the session were splitted with an odd even rule so that both query logs contains data from the same time period.

9.2 Edit Distance

To establish a baseline an edit distance (Levenshtein distance) [8] model was used. The transfemes that corresponds to edit distance transformations were given a uniform distribution and all other transfemes were given zero probabilities.

Example 9.1. This list contains some examples of transfemes corresponding to edit distance transformations and their corresponding probabilities if α is set to 2.5.

```
'' -> 'a' 0.02055323216760362
'' -> 'b' 0.02055323216760362
'' -> 'c' 0.02055323216760362
...
'a' -> '' 0.02055323216760362
'b' -> '' 0.02055323216760362
'c' -> '' 0.02055323216760362
...
'a' -> 'b' 0.02055323216760362
'a' -> 'c' 0.02055323216760362
...
'b' -> 'a' 0.02055323216760362
'b' -> 'c' 0.02055323216760362
```

...

9.3 eSales did you mean

Another baseline that was used is the eSales did you mean engine [13]. The eSales did you mean engine uses edit distance as an error model and slightly more advanced language model involving list of all product names / product descriptions.

9.4 Cheating in Language Model

All test were also done with the language model trained on both **dataset1** and **dataset2**. These tests can give hints how well the error model performs without interference from a bad language model.

9.5 Results

The performance of the transfeme Markov model and the edit distance model are presented in figures, 9.1, 9.2, 9.3 and 9.4.

The performance of the transfeme Markov model and the edit distance model using the language model cheat are presented in figures, 9.5, 9.6, 9.7 and 9.8.

The eSales query suggestion model was due to technical limitation only tested with the language cheat. However, it's the author's belief that the result wouldn't not change significantly if it was trained on only **dataset1**. The results were as following.

```
total_queries: 305768
num_correct: 169336
total_same: 190441
num_correct_same: 163460
total_diff: 115327
num_correct_diff: 5876
```

The metric results were $R@1_{diff} = 0.051$ and $R@1_{same} = 0.858$.

A discussion about the results can be found in section 10.1.

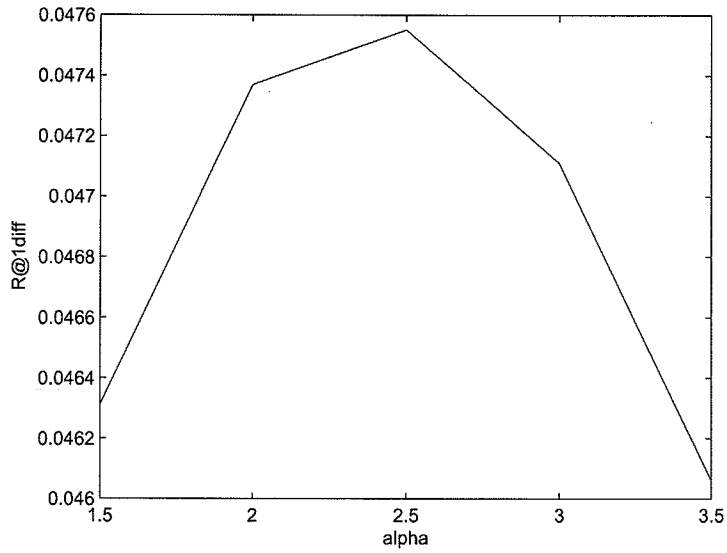


Figure 9.1: The figure show results for transfeme Markov model using different values of α . The metric used was $R@1_{diff}$. The language model was trained with **dataset1**.

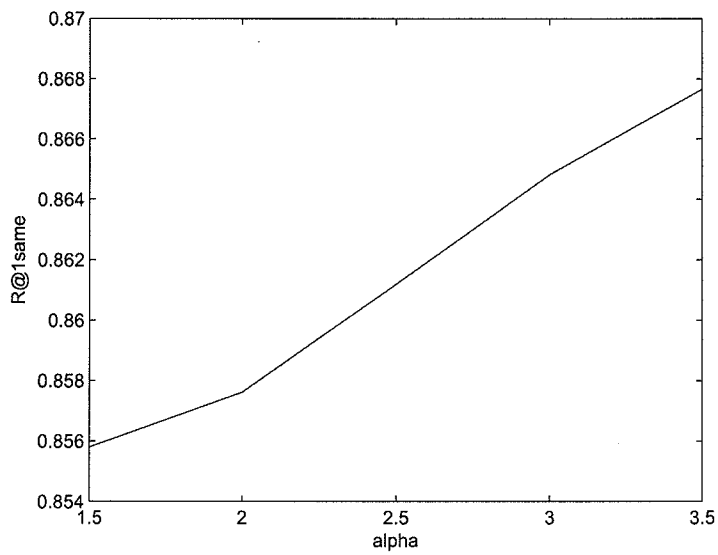


Figure 9.2: The figure show results for transfeme Markov model using different values of α . The metric used was $R@1_{same}$. The language model was trained with **dataset1**.

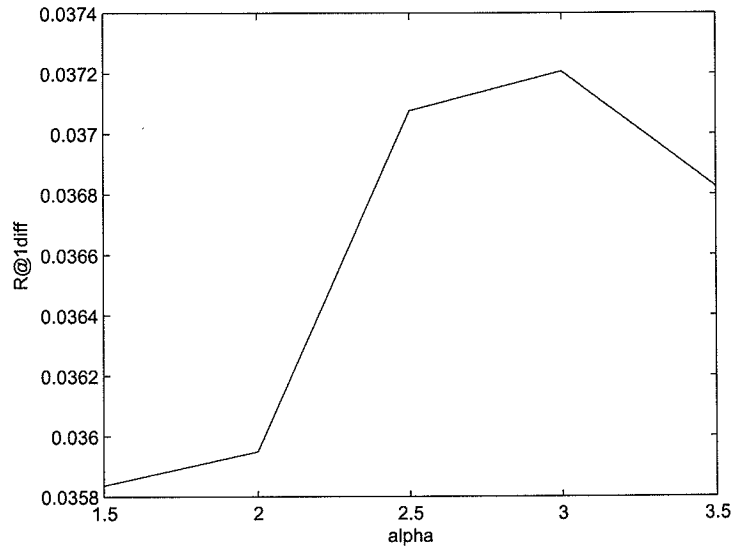


Figure 9.3: The figure show results for edit distance model using different values of α . The metric used was $R@1_{diff}$. The language model was trained with **dataset1**.

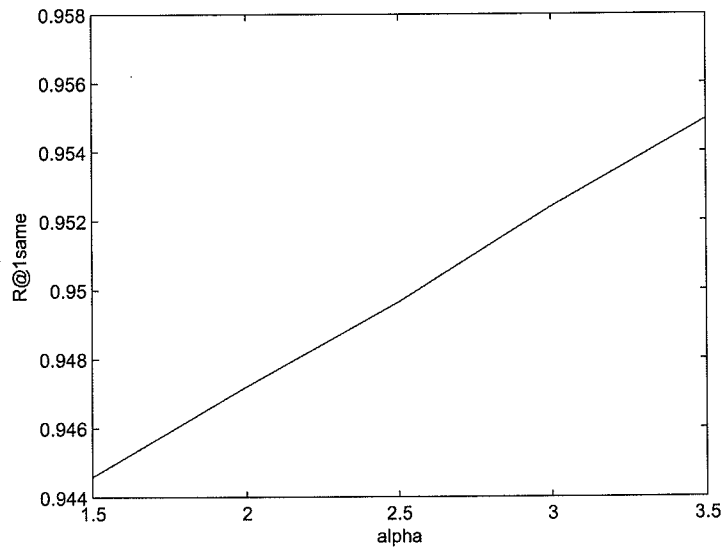


Figure 9.4: The figure show results for edit distance model using different values of α . The metric used was $R@1_{same}$. The language model was trained with **dataset1**.

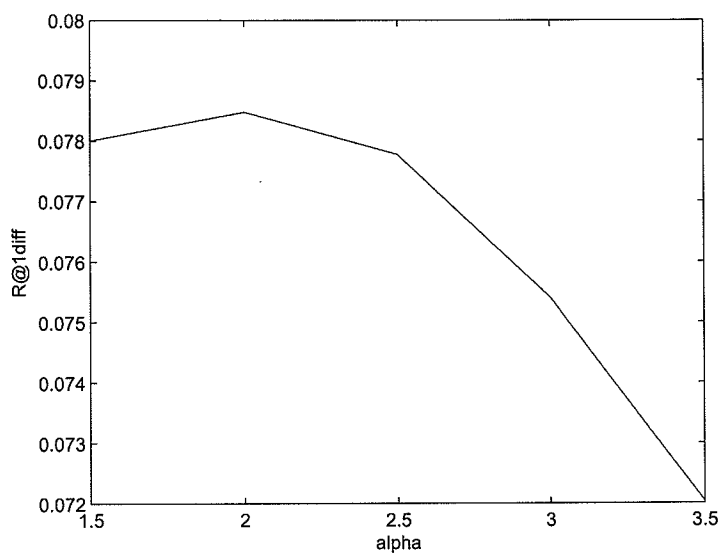


Figure 9.5: The figure show results for transfeme Markov model using different values of α . The metric used was $R@1_{\text{diff}}$. The language model was trained with both **dataset1** and **dataset2**.

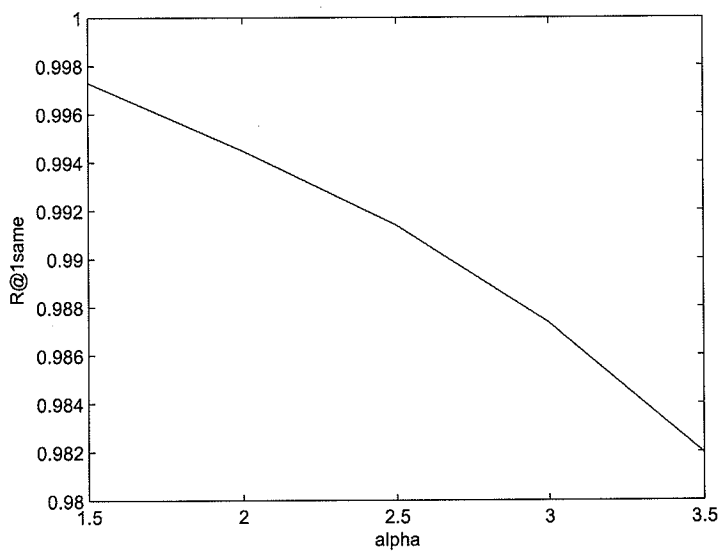


Figure 9.6: The figure show results for transfeme Markov model using different values of α . The metric used was $R@1_{\text{same}}$. The language model was trained with both **dataset1** and **dataset2**.

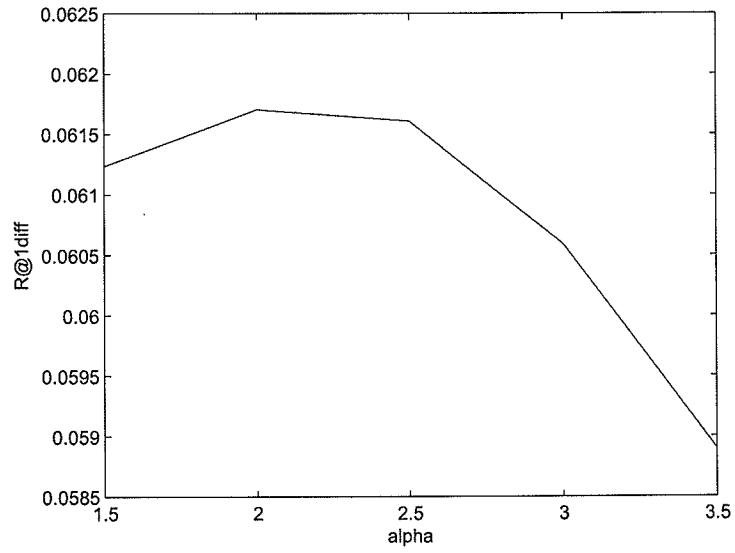


Figure 9.7: The figure show results for edit distance model using different values of α . The metric used was $R@1_{diff}$. The language model was trained with both **dataset1** and **dataset2**.

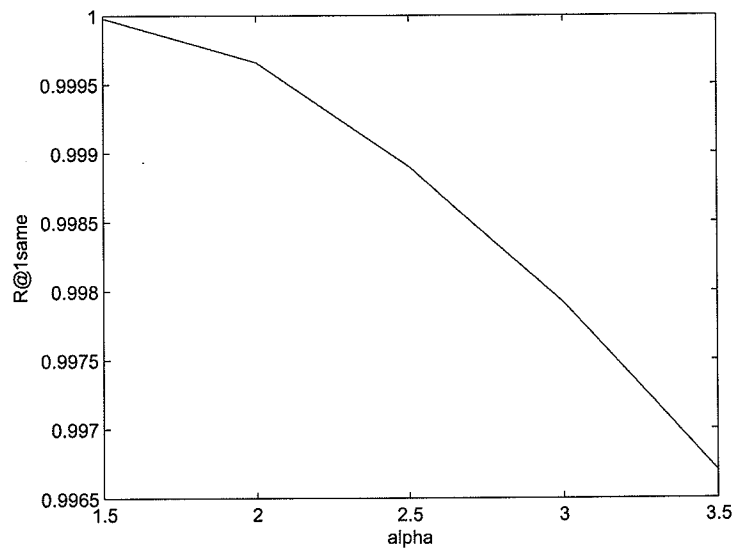


Figure 9.8: The figure show results for edit distance model using different values of α . The metric used was $R@1_{same}$. The language model was trained with both **dataset1** and **dataset2**.

Chapter 10

Discussion

10.1 Results

Figure 9.1 and 9.3 show that the transfeme model outperforms the edit distance model on the $R@1_{\text{diff}}$ metric. However the transfeme model did not outperform the eSales did you mean engine. Since the eSales did you mean engine uses the edit distance error model one can draw the conclusion that eSales language model outperforms the simple language model used in the transfeme model. XOR comparisons between the results of the transfeme Markov model and the eSales engine strengthen this conclusion. Figure 9.5, where the language model was trained on both `dataset1` and `dataset2`, also shows that with a better language model the transfeme Markov model has the potential to outperform the eSales did you mean engine.

10.2 Future Research and Open Questions

Language Model

The results showed that a better language model is necessary. A big issue with the current language model is that it will always predict probability zero for query that have never been observed before. Using an n-gram model this issue could be somewhat mitigated as only the words in the query would have to be previously observed instead of the whole query. Another approach to solve this issue is to use product lists / product descriptions to initiate the language model.

Another important aspect of the language model is that in reality the probability of a query change over time. For instance when an author win the Nobel price in literature the search query related to that author would immediately rise significantly.

It could be important to use different models depending on the popularity of the query. For popular queries a simple ML method may be appropriate but for more unpopular queries a n-gram model may be used. Smoothing could be

used to combine these models.

Another important aspect to consider when designing a language model is that it must be possible to compute the A* heuristic from the language model.

Stationary Distribution

The stationary distribution that is necessary in section 8.1 may be impossible to compute exactly but it's probably possible to find a better approximation than the uniform distribution.

Transition Probabilities Not Summing to 1

The issue described in remark 6.9 is a major theoretical flaw. It's unclear to what extent mitigating this issue may yield better results.

Shortest path versus sum of all paths

In section 8.2 the most probable path was used instead of the sum of all paths. It would be interesting to somehow compare the most probable path to the sum of all paths and see if they are approximately proportional to each other.

More Baselines

Other papers on the query suggestion problem have used a slightly different entry point. Instead of using a query log as training data they use an assumingly noiseless weighted correction list. This makes it difficult to relate this thesis result to other papers and other algorithms.

It would be interesting to test other algorithms using the same testing method as in this thesis. It would also be interesting to test the transfeme Markov model with the same weighted correction list that was used in other papers and compare results.

Proof of Inference Method

In chapter 7 a method for inference was presented. The same method had been successfully used by Duan et al [5]. However Duan et al did not present an rigorous proof for the method. The issue with transition probabilities not summing to 1 makes it difficult to use standard probability methods.

Better Heuristics in A* Search / Other Search Algorithms

The performance of the A* search is essential for the real world uses of this model and the performance of the A* search is very much dependent on the heuristics function used. It's probable that there exists better heuristic functions than the one presented in section 8.4.

It could also be interesting to explore other search algorithms such as the beam search.

Reinforcement Learning

A phenomenon that is not captured by the transfeme Markov model is that the query a user enter depends on what the query suggestion function suggested. If a query is suggested to the user, the query is more likely to be selected by the user since the user just have to click the suggested query instead of entering it manually.

Further on one may design a query suggestion function so that it explores different query suggestions for a misspelled query. One would have to balance the need to exploit (suggested the most probable query suggestion) and explore (try new query suggestions).

This phenomenon could be studied in the framework of Reinforcement learning [12].

10.3 Conclusion

The transfeme Markov model shows potential but must be combined with a better language model before it's ready for industrial use.

Bibliography

- [1] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.
- [2] Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Commun.*, 50:434–451, May 2008.
- [3] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP 2004*, pages 293–300, 2004.
- [4] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7:171–176, March 1964.
- [5] Huizhong Duan and Bo-June (Paul) Hsu. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web, WWW '11*, pages 117–126, New York, NY, USA, 2011. ACM.
- [6] G. B. Folland. *Real Analysis. Modern Techniques and Their Applications. Pure and Applied Mathematics*, 2, 1999.
- [7] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, February 1968.
- [8] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.
- [9] Hao Ma, Haixuan Yang, Irwin King, and Michael R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *Proceedings of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 709–718, New York, NY, USA, 2008. ACM.
- [10] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 469–478, New York, NY, USA, 2008. ACM.

- [11] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33:6–12, September 1999.
- [12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.
- [13] Apptus website. <http://www.apptus.com>, March 2012.

