Combining Text Semantics and Image Geometry to Identify Relations

Medved, Dennis

2012

Combining Text Semantics and Image Geometry to Identify Relations

Dennis Medved

October 5, 2012

Abstract

Automatically identifying actions and relations between objects in images can be useful for many applications, for example image and video labeling.

In this thesis, the goal is to extract a predefined set of objects from the images and identify the relations linking these objects. Wikipedia is the source of images and texts that are analyzed.

I created a program that takes both geometrical information from images, and semantic information from texts and outputs classification of relations for each object in the images. The baseline of only using geometrical features is improved on by using bag-of-words features based on the articles, and further enhanced by utilizing predicate information.

Acknowledgments

I would like to thank the following people for their involvement in creating this thesis:

Pierre Nugues: thanks for being my supervisor and giving guidance and feedback during the thesis.

Peter Exner: thanks for guidance and help with the semantic parser.

Marcus Stamborg: thanks for help with the coreference interface, and always being a cheerful, positive coworker, contributing to an efficient work environment.

Kalle Åström and **Fangyuan Jiang:** thanks for supplying and annotating the images, and giving valuable feedback.

John David Olovsson: thanks for your shenanigans and tomfoolery, gl hf creating games.

⟨glitter⟩ **Cem Eliyürekli:** ⟨/glitter⟩ thanks for being the victim of high jinks and skulduggery, I hope you will be able to finish your thesis, or at least the 35 x 70 nonogram.

Contents

Chapter 1

Introduction

This chapter introduces my project with some background information and defines the scope and subject of the thesis. The outline of the following chapters is also described.

There exist many services to search images on the Internet, if you want to find an image, for example of a dog. One common way of indexing images is to analyze the text associated with the image, for example the text on the web page where the image is used, or its filename. But what if you are a looking for a more specific image, containing relations between objects in the image? For example: a man throwing a ball to a dog, or a specific celebrity carrying her dog? Combining information derived both from the image, and the text associated with this image, could help answer such queries.

Automatic translation, categorization of images and video, and linking entities between text and images, are other examples of applications, where information about relations can be useful. Utilizing algorithms in cognitive vision, and combining the algorithms in semantic processing, we apply them on images and text in a machine-readable format. The purpose is to analyze and recognize their content, in terms of semantics, objects and relations, and turn their content into dynamic knowledge resources. These kinds of knowledge resources could potentially be applied to many different fields, not just search.

1.1 Subject and scope

Wikipedia is a freely available Internet encyclopedia. It contains a large number of articles in many different languages. The encyclopedia is a substantial resource of images, which are used to illustrate the articles. Wikipedia can therefore be used to find images with relevant text associated with them.

The problem contains two aspects: identifying objects in the images and classifying the relations between the objects. The goal is to improve the identification and classification, with the help of semantic information derived from text associated with the images.

In this thesis, the scope is limited to images taken from Wikipedia containing to the “Horse” or “Pony” category and articles where they are used. The images are gold annotated, i.e. manually annotated by humans, by creating

bounding boxes around horses and humans present in the images. This gives the information about the number of respective humans, and horses, their size and location in the image.

The articles are processed by a semantic parser (Exner and Nugues, 2012) and outputted in the CoNLL format (Surdeanu et al., 2008), a tab separated file format. Each line in the files contains information about a specific word in a sentence. The information is for example: lemma, part of speech, and coreferences, as well as the predicate-argument structure of the sentence.

I have created a program, named Centaur, which takes the image annotations and the CoNLL files as input and outputs for each possible human-horse pair in the images the relations between them. The relations, or predicates, are limited to: *Ride*, *Lead*, or *None*. Where *Ride* and *Lead* are when a human is riding or leading a horse and *None* is no action or an action that is not *Ride* or *Lead*.

The problem to identify objects and actions is greatly reduced from the general case, with an infinite number of potential objects and actions, to a set of two objects and three actions. I focused on the semantic processing part and used gold annotated bounding boxes, effectively reducing the problem to only categorizing the predicates for the human-horse pairs. Generation of the bounding boxes could be produced automatically by a program utilizing cognitive vision algorithms (Felzenszwalb et al., 2010), using the gold annotated only as a key.

1.2 Outline

This thesis is divided into the following chapters:

2. **Background** discusses the source material derived from Wikipedia, and previous work.

3. **Classification** describes linear classification, how features are encoded, and how classification results are evaluated.

4. **Implementation** gives an overview of the architecture of the system, describe the features used, how the classification is done, and how the result is evaluated.

5. **Results** presents the results for different feature combinations, and end with an overview of the results.

6. **Discussion** discusses the results of the thesis, a conclusion and possible extensions of the work is presented.

Chapter 2

Background

This chapter describes previous work related to this thesis, and the source material derived from Wikipedia.

2.1 Previous work

To the best of my knowledge, no work has been done to identify relations in images using combined analyses of image and text data. There are related works however:

Paek et al. (1999) combined image segmentation with a text-based classifier, using image captions as input. A tf-idf weighting is applied on the text, see section 3.4.2 for more information. The goal is to label the images as either taken indoor or outdoor. The results are improved by using both text and image information together, compared to using only one of the classifiers.

Deschacht et al. (2007) used a set of 100 image-text pairs taken from Yahoo! News (http://news.yahoo.com) and tries to automatically annotate the images, utilizing the associated text. The goal was to annotate the presence of specific humans in the images, but also more general objects. The image captions are analyzed to find named entities, but more advanced information is also derived from discourse segmentation, which is used to determine the saliency of entities.

Moscato et al. (2009) had a large corpus of French news articles, composed of a text, images and image captions. They combined a image detector capable of recognizing human faces, and logos, with named entity detection in the text, which weights the entities in a tf-idf like manner. The goal is to correctly annotate the faces and logos found in the images. The images are not annotated by humans, instead named entities in the captions are used as the ground truth, and the classification is based on the articles.

Tirilly et al. (2010) used a large collection of images taken from Flickr (http://www.flickr.com) that users have annotated by supplying keywords and short descriptions. The goal is to categorize the images, utilizing a combination of features derived from image analysis, together with relevant image labels extracted from the text associated with the images.

2.2 Wikipedia

Wikipedia (http://wikipedia.com) is a large, collaboratively edited encyclopedia, consisting of articles on a broad range of subjects. An article, for example about ponies, is represented by a page made out of one or more titled sections, for example: “Horses and ponies”, “History”, “Uses”, etc. Each section has one or more paragraphs of text, and zero or more images. The images usually have a caption, but do not need to have one. See Figure 2.1 for a screenshot of an excerpt from an article.

The images are stored by Wikimedia Commons (http://commons.wikimedia.org), which is a large database of freely usable media files. It is not unusual for editors to reuse an image for more than one article, and an image can therefore have more than one article or caption associated with it.

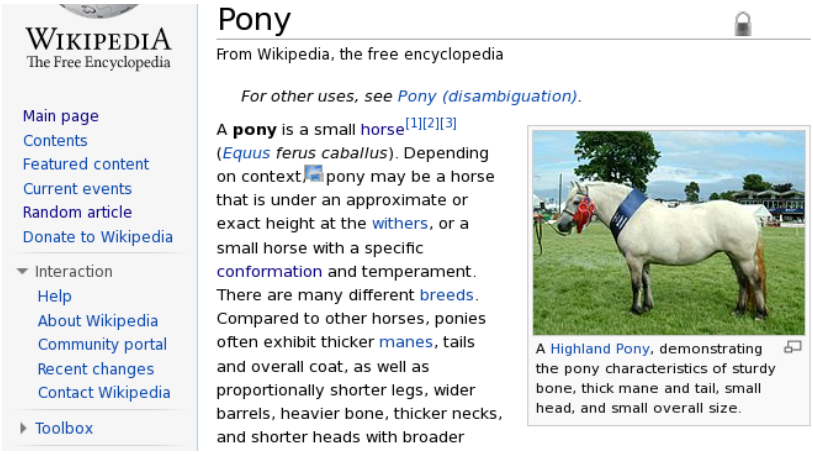

Figure 2.1: The wikipedia article about ponies.

2.3 Building the image data set

The images were selected from the category “Horse” and the category “Pony” from Wikimedia Commons. All images in these categories, with a valid article associated with them, have been included. Wikimedia Commons is collaboratively edited web page. The images are uploaded and placed in categories by the users.

An image placed in the “Horse” or “Pony” category does not need to be of a real horse. It can depict something associated with the words for example: a car, a statue or a painting, see Figure 2.2, 2.3 and 2.4 for examples. Some of the images also include humans, either interacting with the horse or just being part of the background. A image can therefore have zero or more horses in the image, and zero or more humans.

Figure 2.2: A Ford Mustang, an automobile, not a horse. Source: Wikipedia

Figure 2.3: A bottle with depiction of a horse. Source: Wikipedia

Figure 2.4: A military badge, featuring the word “horse”. Source: Wikipedia

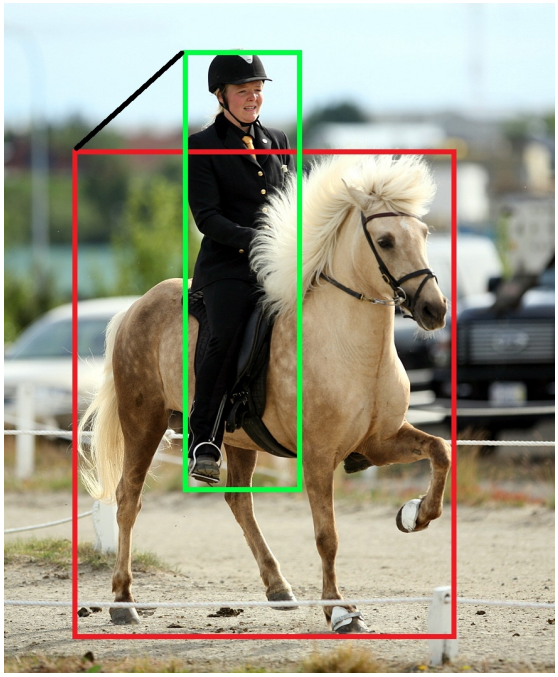

Figure 2.5: A human riding a horse, with bounding boxes displayed. Source: Wikipedia

The selected images were manually annotated by three different annotators, placing a bounding box around each horse and human with the label *Horse* or *Human* respectively, and labeling the interaction between the human-horse pair if the interaction corresponds to *Lead* or *Ride*. The *None*-relationships are left implicit.

Table 2.1 shows some information about the source material, 901 images were extracted from the "Horse" and "Pony" categories at Wikimedia Commons, but only 788 could be used, some images were duplicates and some did not have an valid article associated with them. There exist 2235 possible human-horse pairs in the usable images, but the distribution of relations is quite heavily skewed towards the *None*-relation, and the *Lead*-relation have especially few examples.

Object	Quantity
Images extracted	901
Usable images	788
Human-horse pairs	2235
Relation: *None*	1935
Relation: *Ride*	233
Relation: *Lead*	67

Table 2.1: The quantity of different objects in the source material.

Figure 2.5 shows bounding boxes around a horse and a human, and a line connecting them, indicating a *Ride*-relation. Figure 2.6 shows several humans riding and two persons with no relation to the horses. Figure 2.6 shows a human leading a horse and one with no relation.

The output of this annotation are text-files that are machine-readable, Tables 2.2 and 2.3 shows examples of the formats.

Image ID	Object ID	Type	X Center	Y Center	Width	Height
37	1	1	372.74	421.68	196.24	695.73
37	2	2	420.28	605.76	590.73	795.57

Table 2.2: The text representation of the bounding boxes in Figure 2.5. The type: 1 equals human, and 2 equals horse.

Image ID	Subject ID	Object ID	Type
37	1	2	2

Table 2.3: The text representation of the relations in Figure 2.5. The type: 0 equals *None*, 1 equals *Lead*, and 2 equals *Ride*

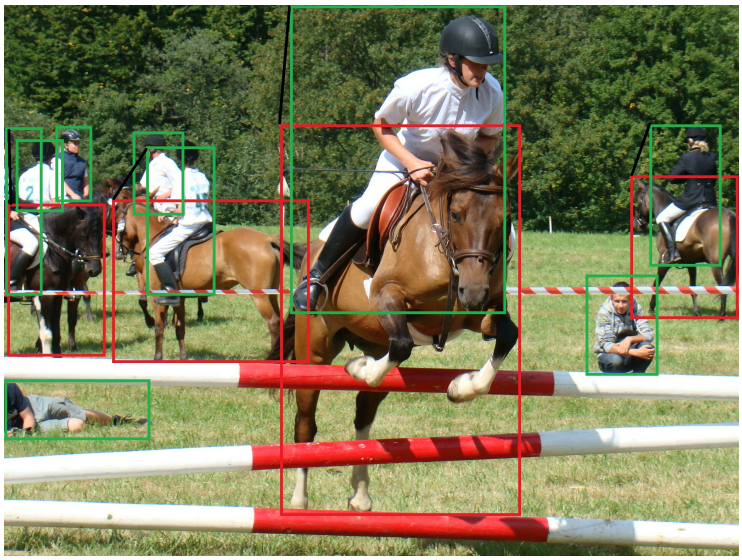

Figure 2.6: Seven humans riding on ponies, two humans as spectators, with bounding boxes displayed. Source: Wikipedia

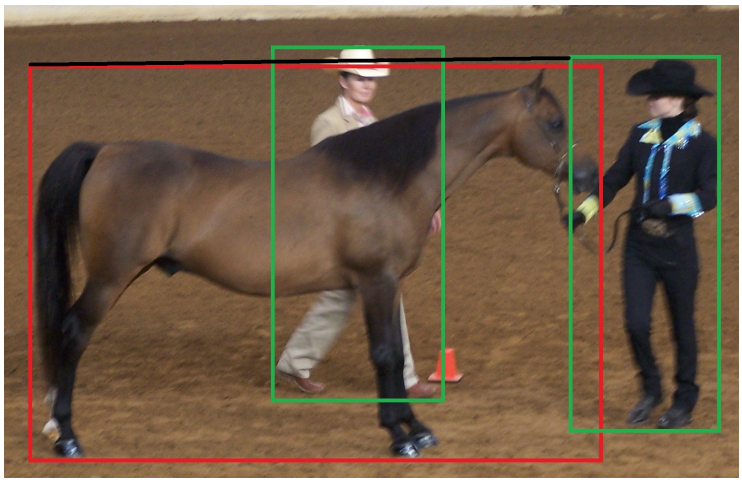

Figure 2.7: One human leading a horse, and one human standing next to it. Source: Wikipedia

2.4 Semantic parsing

Figure 2.8 shows a visual presentation of a sentence parsed from Wikipedia: "Humans provide domesticated horses with food." In the bottom the dependency graph of the sentence is shown, with annotated edges and additional information about each word. A the top a representation of the predicate information is shown, with the predicate sense according to PropBank (http://verbs.colorado.edu/ mpalmer/projects/ace.html) to the left and the arguments to the predicates to the right.

A predicate is an action or relation such as jump or own, but there can exist some ambiguity in the sense of the predicate. The sense is explicitly shown as a number added after the word, in the example there exist two predicates: provide.01 and domesticate.01. A predicate has one or more arguments, but more then three is unusual, provide.01 has three arguments: A0: provider, A1: thing provided, A2: entity provided for, domesticate.01 only has one argument A1: formerly wild beast.

Not shown in this example, but a predicate can also have modifying argument. A modifying argument has the prefix "AM-", and there exist 12 different types of modifiers in the PropBank. Examples of modifiers are AM-DIR: shows motion along some path, AM-LOC: indicates where some action takes place, and AM-TMP: shows when an action took place.

	Humans	provide	domesticated	horses	with	food	.
provide.01	A0		A1				
domesticate.01				A1			

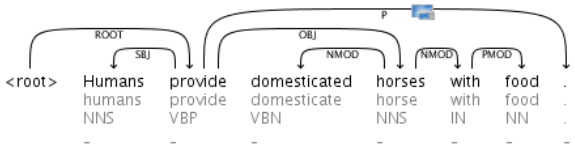

Figure 2.8: Visualization of semantic parsing, showing predicate information and the dependency graph of the sentence. The automatic parser has not recognized the arguments properly "domesticated horses" should be A2.

2.4.1 CoNLL format

I used the parsing framework Athena (Exner and Nugues, 2012) in conjunction with a coreference solver (Stamborg et al., 2012) to parse the Wikipedia articles. It outputs files in the CoNLL format (Surdeanu et al., 2008), which is a tab separated file format, and I added an extra column for coreferences. Table 2.4 shows an example of a parsed article in the file format. Redundant and unused columns are not shown in the example. Explanation of the columns follows:

ID is a counter indicating the word's place in the sentence and is reset for each new sentence.

Form is the word itself in a verbatim form.

Lemma is the word in its dictionary form, for example: sprinting = sprint.

POS is the part of speech, for example: JJ = adjective or NN = Noun, singular or mass.

HEAD indicates which ID is the parent of the word in the dependency graph of the sentence.

DEPREL is the label of the edge in the dependency graph.

FILLPRED indicates if the word is a predicate or not, Y or _ respectively.

PRED is the predicate sense according to PropBank. A predicate is an action such as breed or sprint, but there can exist some ambiguity in the sense of the predicate. The sense is explicitly shown as a number added after the word, for example: breed.01, which has the meaning "to cause to reproduce". A predicate has one or more arguments, breed.01 has two arguments: Arg0: causer, agent, Arg1: entity bred. Predicates can also have modifying arguments such as Argm-tmp that specifies the time (temporal), for example "for six years".

APRED is a variable amount of argument-columns to the predicates. There exists one column for each predicate in the sentence. The leftmost column is associated with the first predicate in the sentence, the second predicate with the next leftmost and so on. An argument is indicated by a *None* empty line, for example A0 which is short for Arg0. In the example "American" is the head of the argument A0, if the argument is longer than one word the HEAD column can be used to find the whole argument.

COREF indicates coreferences between words or phrases in the text, by giving an entity a unique identification number through out the article. Two words or phrases are coreferent if they refer to the same entity. The first entity encountered is given the number (0), the second (1), and so on. A reference can be a single word, or several words in a row. In the latter case the identification number spans several rows.

An example of coreferences: "Seabiscuit is a horse. He has won many races." Here the name *Seabiscuit* and the pronoun *He* are referring to the same entity and are therefore coreferent. Pronouns are (almost) always referring to an entity in implicit way and the reference have to be deduced by the context. Although a small number of pronouns are pleonastic and do not refer to an entity, but are used as a dummy word.

ID	FORM	LEMMA	POS	HEAD	DEPREL	FILLPRED	PRED	APRED: 1	2	3	4	COREF
7	American	american	JJ	8	NMOD	_	_	A0	_	_	_	_
8	breed	breed	NN	5	PRD	Y	breed.01	_	A1	_	_	_
9	of	of	IN	8	NMOD	_	_	A1	_	_	_	_
10	horse	horse	NN	9	PMOD	_	_	_	_	_	_	(1)
11	that	that	WDT	12	SBJ	_	_	_	A0	A0	_	_
12	excels	excel	VBZ	8	NMOD	Y	excel.01	_	_	_	_	_
13	at	at	IN	12	ADV	_	_	_	A2	_	_	_
14	sprinting	sprint	VBG	13	PMOD	Y	sprint.01	_	_	_	_	_
15	short	short	JJ	16	NMOD	_	_	_	_	_	A2	_
16	distances	distance	NNS	14	OBJ	Y	distance.01	_	_	A1	_	_

Table 2.4: Part of a CoNLL file.

Chapter 3

Classification

This chapter describes linear classification, how features are encoded, and how classification results are evaluated.

For each picture that has one human or more, and one horse or more, there exists one or more human-horse pairs, where a possible action can occur. I limited the number of possible predicates for the pairs to only three: *Ride*, *Lead* and *None*. There exist an infinite number of possible predicates, but to have manageable set of actions I restricted to two common predicates: *Lead* and *Ride*, and a predicate *None* that represents every other action or non-action.

The problem is to decide for each human-horse pair in the pictures in which of the three classes they belong, based on the data from the images annotations and the parsed Wikipedia articles.

3.1 Linear classification

The goal of statistical classification is to decide which class an object belongs to, based on the characteristics of the object, also called features. If the decision is based on a linear combination of the feature values, then it is a linear classifier. The feature values can be represented by a feature vector, Table 3.1 shows an example.

Linear classification is s relatively fast classification process, especially for sparse data sets, for example produced by bag-of-words features on documents, see section 3.4.1 for more information. Examples of different algorithms implementing linear classification are: perceptron, logistic regression and support vector machines.

In the case of two classes, a binary classification problem, the problem can be visualized as separating the two different classes with a straight line, dividing them into two separate areas. Figure 3.1 show three possible lines H_1, H_2, H_3 dividing the solid and empty points. H_1, H_2 correctly classifies the dots, but H_3 does not. H_2 also maximizes the distance to both groups, which is preferable.

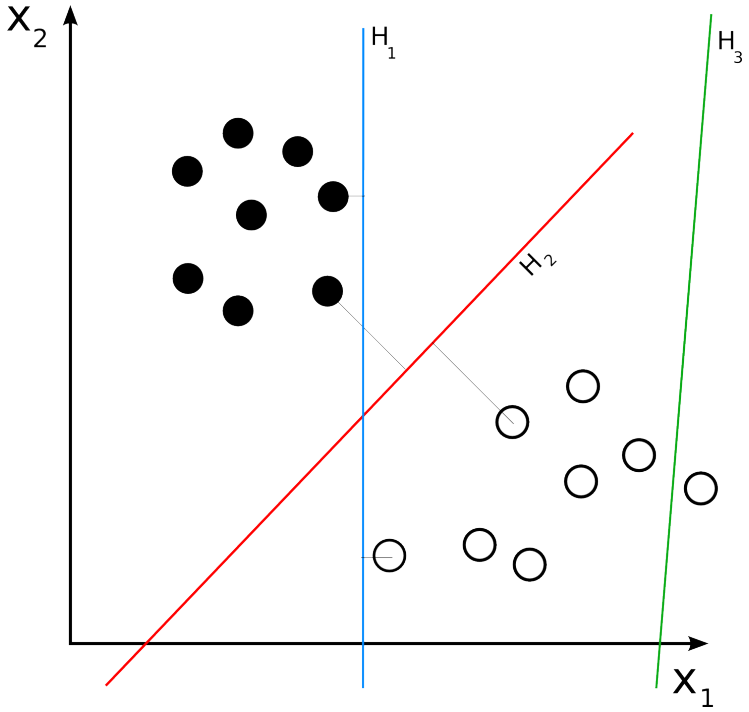

Figure 3.1: Lines H_1, H_2, H_3 dividing the solid and empty points. Source: Wikipedia

3.1.1 LIBLINEAR

For classification purposes I decided to use LIBLINEAR (Fan et al., 2008), which is an open source library for large linear classification. It supports logistic regression and linear support vector machines. The following solvers are available in the library:

0. L2-regularized logistic regression (primal)
1. L2-regularized L2-loss support vector classification (dual)
2. L2-regularized L2-loss support vector classification (primal)
3. L2-regularized L1-loss support vector classification (dual)
4. Multi-class support vector classification by Crammer and Singer
5. L1-regularized L2-loss support vector classification
6. L1-regularized logistic regression
7. L2-regularized logistic regression (dual)

I chose to utilize logistic regression, because of its speed and its ability to output probabilities, not just classes. The choice limited the available solvers to 0, 6 and 7.

3.2 Generating training examples

The first step is to generate training examples that can be used by LIBLINEAR to create a model. A training example is a vector consisting of numeric values. The first value represents the true class of the example, based on the gold annotations, see Table 3.2 for the values of the different predicates. The following values represent features based on either the bounding box data or the parsed Wikipedia texts.

I used features with three different types of output, Boolean: with only true or false as values, or nominal: with a finite set of values, or numeric: with a infinite set of values. The values should be normalized to the unit interval [0, 1] for optimal use with LIBLINEAR.

Examples of each type of feature, and how they can be used to encode training examples follow:

F_Overlap is a Boolean feature that indicates that bounding boxes of the human and the horse has an overlapping area, this is represented by one element: 1 if true, 0 if false.

F_Direction is a nominal feature that tells which direction the human is relative to the horse, with 4 values: North, East, South, West. This is represented by four elements in the vector corresponding to each of the directions, where one of the elements is 1 and the rest are 0.

F_Distance is a numeric feature that shows the distance between center of the human and the horse, this is represented by one element that is a real number. Ideally this should be normalized, but is skipped for this example.

If we have a picture with one horse and two humans, one riding on the horse and one just standing a couple of meters to the right, then there exist two human-horse pairs, see Table 3.1 for the corresponding vectors.

The vector for the rider starts with a 2 corresponding to *Ride*, the first feature value is 1 indicating overlap, followed by 1 0 0 0 which equals North, and is ended by 50.0 the distance between the human and the horse. The vector for the human standing is: 0 = *None*, 0 = No overlap, 0 1 0 0 = East, 200.0 = The distance.

	Class	F_Overlap	F_Direction				F_Distance
Human riding on the horse	2	1	1	0	0	0	50.0
Human standing next to it	0	0	0	1	0	0	200.0

Table 3.1: Training example vectors.

Predicate	Number
None	0
Lead	1
Ride	2

Table 3.2: The predicate classes and their representation as numbers.

3.3 Creating and using a model

The training examples are extracted from the set of images and articles. After generating a large number of examples, the vectors are used as input to LIBLINEAR. After deciding the solver type, a model is created based on the training data. LIBLINEAR tries to create a model that predicts the class of unseen examples, that is, which of the predicates in Table 3.2 the examples belong to.

We can apply the model to classify examples of unknown class. The vectors used as input are of the same type as in Table 3.1, but without the first element corresponding to the gold annotated class. The output is a vector of probabilities for the different classes.

If we assume that a good model was created and reuse the example vectors in Table 3.1, the output from the LIBLINEAR-model could be something like in Table 3.3.

	None (%)	*Lead* (%)	*Ride* (%)
Human riding on the horse	10	14	76
Human standing next to it	82	12	6

Table 3.3: Probability vectors for the examples. We would select the class with the highest probability, for the first line the most probable class is *Ride*, and for the second it is *None*.

3.4 Encoding text features

3.4.1 Bag-of-words

The bag-of-words (BoW) model is a way to simplify the representation of a text, modeling it as an unordered collection of words. This representation ignores most of the semantic information available, even the order of the words. The technique have been used for the purpose of classifying text, often together with some kind of weighting, for example tf-idf (Joachims, 1996).

A dictionary is created with all words that are believed to be encountered, possibly including a special word representing all unknown words, or just ignoring unknown words. The dictionary maps each word to an index, making it easy to represent text as vectors. A text is transformed into a BoW vector by writing down the frequencies of the words, in the text, to the corresponding indices in the vector. An example follows:

In Listing 3.1 there are two lines containing simple sentences.

Listing 3.1: Example sentences.

```
The Phantom has a horse. He also has a dog.  (1)
The horse is named Hero.                     (2)
```

Using these two lines a dictionary is created, see Table 3.4

Word	Index
the	1
phantom	2
has	3
a	4
horse	5
he	6
also	7
dog	8
is	9
named	10
hero	11

Table 3.4: Example dictionary.

Two vectors, 11 element long, are created to represent the two lines, see Table 3.5

	1	2	3	4	5	6	7	8	9	10	11
Line (1)	1	1	2	2	1	1	1	1	0	0	0
Line (2)	1	0	0	0	1	0	0	0	1	1	1

Table 3.5: Example BoW vectors.

3.4.2 tf-idf weighting

A problem with using raw frequencies in the BoW vectors is that all words are treated equally important. Say you have the sentence "The horse is named Hero" the terms "the" and "is" are probably going to be very frequent in other sentences, but the words do not carry much information that can be used for classification.

One solution to the problem is to weight the term frequencies (tf) with the inverse document frequency (idf). The document frequency df_t is defined as the number of documents that contain the term t. Let N be the total number of documents in the collection, then the definition of the idf for the term t is as follows:

$$idf_t = \log \frac{N}{df_t}$$

Idf for a rare terms is then going to be high, and conversely low for a frequent term. Let $tf_{t,d}$ be the term frequency for t in a document d, then the tf-idf weighting scheme is given by:

$$\text{tf-idf}_{t,d} = tf_{t,d} \cdot tf_t$$

Then tf-idf is the highest when a word occurs many times in a small number of documents, and the lowest when a word is present in almost all documents, for example "the".

For more information about term weighting read Salton and Buckley (1988).

3.5 Evaluation metrics

If you have a binary classification problem with Boolean values, there exist four different categories of examples that have been predicted. The actual class of the example can have the values true or false, and the prediction likewise, see Table 3.6.

- True positive, an entity that is true and was correctly predicted
- False negative, an entity that is true and was incorrectly predicted
- True negative, an entity that is false and was correctly predicted
- False positive, an entity that is false and was incorrectly predicted

		Predicted class	
		true	false
Actual class	true	true positive (tp)	false negative (fn)
	false	false positive (fp)	true negative (tn)

Table 3.6: The four different categories of predicted entities.

Precision and recall are common ways of evaluating the performance of classification systems. Recall is defined as the number of correctly predicted and true entities divided by the number of true entities. Recall is the fraction of relevant entities that are retrieved.

For example to classify if a picture includes a horse or not, and in the set of images there are 100 depicting horses. If 75 of the images are correctly classified as horses, then $tp = 75$, $fn = 25$ and recall is 75 / (75 + 25) = 75%.

$$recall = \frac{tp}{tp + fn}$$

Precision is defined as the number of correctly predicted and true entities divided by the number of entities that were predicted as true. Precision is the fraction of retrieved entities that are relevant.

Continuing with the previous example: if 50 images were also misclassified as horses, then $fp = 50$ and precision is 75 / (75 + 50) = 60%.

$$precision = \frac{tp}{tp + fp}$$

There is a trade-off between recall and precision, it is easy to get 100% recall by classifying all examples as true, but then precision is really low. And vice

versa if you only classify one example correctly, then precision is 100% and recall is low. The weighted harmonic mean of precision and recall is called the F-measure. If both precision and recall are deemed equally important, then the weights are equal for both and you get the special case of F_1.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The F_1-measure tends to go towards the lower of the two values, which means that both recall and precision need to be relatively high to get a high F_1-measure.

3.5.1 Confusion matrix

A confusion matrix is a table layout that allows for a visual evaluation of classification. Table 3.6 show the layout for two classes, Table 3.7 shows for three classes. The table shows the distribution of the predicted examples.

The diagonal of the table corresponds to the amount of correctly predicted examples. In Table 3.7, this is for *None* = 250, *Ride* = 150, and *Lead* = 40. It is easy to see how the examples have been misclassified, for example *Lead*: 50 is misclassified as *None*, and 10 as *Ride*.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	250	40	10
	Ride	30	150	20
	Lead	50	10	40

Table 3.7: A confusion matrix for three classes.

3.5.2 Cross-validation

Cross-validation is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. Using the technique lowers the probability of overfitting the model to the training data. If the model is overfitted, then it performs well if you test it on the training data itself, but poorly on an independent data set.

In k-fold cross-validation, the data set is partitioned in k subsets, one subset is left out and training is done on the other $k - 1$ subsets, the model is then evaluated on the subset that was left out. This process is repeated k times, an iteration is called a fold, hence the name "k-fold cross-validation". In each fold a different subset is chosen as the evaluation set, and the results from all k iterations are then averaged to create an estimation.

Chapter 4

Implementation

This chapter gives an overview of the architecture of the system, describes the features used, how the classification is done, and how the result is evaluated.

4.1 Overview

4.1.1 Overview of the whole system

Figure 4.1 shows the architecture of the whole system.

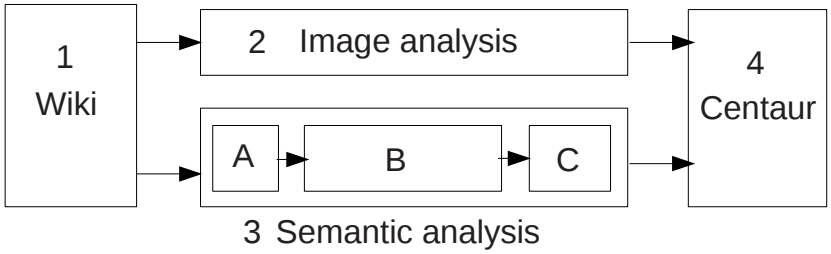

Figure 4.1: An overview of the whole system.

1. Wikipedia is the source of the images, and the articles in the form of the Wiki markup language[1].
2. The images are analyzed: placement of bounding boxes, classification of objects and actions. This was done manually, but could be replaced by an automatic system. The output is text files.
3. The articles are analyzed:
   A. A filter that can choose: whole articles, paragraphs that are the closest to the images, or captions.
   B. A pipeline that takes wiki markup as input and produces CoNLL files as output. A column in the files roughly translate to a step in the pipeline.

[1] http://en.wikipedia.org/wiki/Help:Wiki_markup

C. A coreference solver, takes the CoNLL files and adds an extra column for coreferences.

4. Centaur takes the text files and CoNLL files and outputs classifications of the actions between the human-horse pairs.

4.1.2 Overview of Centaur

Figure 4.2 shows the architecture of the Centaur program.

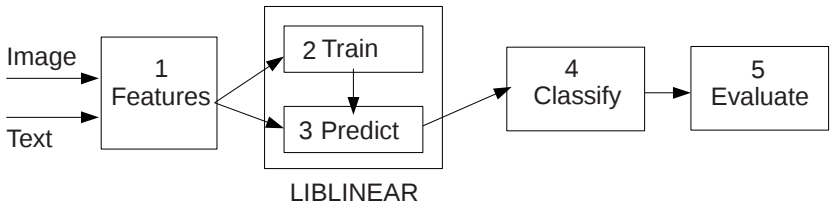

Figure 4.2: An overview of Centaur.

1. Centaur extracts features from the image and article data, and examples are created. Most of the examples are used for training, and some for evaluation.

2. The training examples are used as input to LIBLINEAR which produces a model.

3. The model is queried with test examples and outputs probabilities for the different classes.

4. The probabilities are used to classify the examples.

5. The classifications are evaluated.

4.2 Features

4.2.1 Geometrical features

Geometric features are based on the information gathered from the image analysis, which is the bounding boxes for human and horses. The bounding boxes have a width and a height, and coordinates for the center of the box. See Table 2.2 for a representation of the boxes.

A baseline, with a set of geometric features, was created that produced satisfactory results. The baseline set was then used untouched while developing and testing lexical features. The set contains the following features:

The first three features are described in section 3.2, but F_Direction has four more directions.

F_Overlap is a Boolean feature, if the two bounding boxes overlap or not.

F_Distance is a numeric feature, the length between the centers of the bounding boxes.

F_Direction(8) is a nominal feature, the direction of the human relative the horse, with eight possible directions.

F_Angle is a numeric feature, the angle between the centers of the boxes.

F_OverlapArea is a numeric feature, the size of the overlapping area of the boxes.

F_MinDistanceSide is a numeric feature, the minimum distance between the sides of the boxes.

F_AreaDifference is a numeric feature, the quotient of the areas.

The three following features are combinations of a numeric and a Boolean feature, creating a numeric feature. The Boolean feature is used as a step function: if it is false the output is a constant, if it is true the output is the value of the numeric feature.

F_Distance + F_LowAngle(7) is a numeric feature, F_LowAngle is true if the difference in angle is less then 7°.

F_Angle + F_LowAngle(7) is a numeric feature.

F_Angle + F_BelowDistace(100) is a numeric feature, F_BelowDistace(100) is true if the distance is less than 100.

4.2.2 Lexical features

In Figure 4.1, Module 3.A. is a filter that makes it possible to produce CoNLL files of whole articles, partial articles (the paragraphs that are the closest to the pictures) or captions. The different lexical features have three versions depending on which of the files that are used an input.

The filenames of the images are also used as input to some features, because it is common to have a long descriptive names of the images on Wikipedia. The filenames are not semantically parsed, but there is a heuristic algorithm to break down the names down to individual words.

Bag-of-words features

There exist bag-of-words features for: articles, partial articles, captions, and filenames. There is separate settings and dictionaries for the different versions. The dictionary is created by the words present in the articles of the training set. There is a filter that can exclude words that are either too common, or not common enough, based on their frequency, controlled by a variable threshold.

A BoW vector corresponds to a single file. The values in the vector are normalized. There exist several normalization methods, and the method can be chosen as a setting for the different versions.

Normalization methods

Normalization and term weighting are used for two purposes: The first is to normalize the values to the unit interval [0, 1], the second is to redistribute the weight from less relevant words to more relevant ones. Depending on how you normalise the BoW vectors the result can vary greatly. The following different normalization methods were used:

***None*:** no normalization is used, the values are the original frequencies.

Binary: if a value is greater than 0 then the value is mapped to 1.

Total_max: all values are divided by the global maximal value.

IDF: values are weighted with tf-idf scheme, see section 3.4.2 for more information.

Instance_unit: the vector is normalized to have unit length.

Instance_max: the values are divided by the maximum value in the vector, it also has a fine-tuning parameter.

Instance_IDF: the values are divided by the maximum value in the vector and weighted by the tf-idf scheme, it also has a fine-tuning parameter.

Predicate

A predicate is an action or relation, involving one or more objects as arguments. For example "Cem hugs Marcus" where *hugs* corresponds to the predicate hug.01, the arguments A0 (hugger) and A1 (hugged) corresponds to *Cem* and *Marcus* respectively.

Instead of using all the words in a document to create BoW features, the predicate information can be utilized to filter out more relevant terms. Only using the predicate names and arguments as input, BoW features were created. The words that are not predicates, or arguments to the predicates, are removed as input to the feature.

There exist predicate features for: articles, partial articles, and captions. No version for filenames were created, because no semantic analysis were made on them. The arguments can be filtered firstly depending on their type, for example A0, A1, or AM-TMP, and secondly if only the head or the whole argument is included.

Reusing the example in section 3.4.1, but filtering out everything except the predicate name and the head of argument A1, then the remaining words can be seen in Listing 4.1

Listing 4.1: Example sentences filtered using predicate information.

```
have horse. have dog.   (1)
horse name.             (2)
```

The two predicate names are added to the dictionary because they are missing, see Table 4.1.

Word	Index
the	1
phantom	2
has	3
a	4
horse	5
he	6
also	7
dog	8
is	9
named	10
hero	11
have	12
name	13

Table 4.1: Example dictionary with predicate names added.

Table 4.2 shows the corresponding sparse vectors.

	1	2	3	4	5	6	7	8	9	10	11	12	13
Line (1)	0	0	0	0	1	0	0	1	0	0	0	2	0
Line (2)	0	0	0	0	1	0	0	0	0	0	0	0	1

Table 4.2: Example BoW vectors with predicate filtering.

Specific word filtering

I created binary features for articles, partial articles, legends, filenames and their predicate versions, that were true if any of a set of words were present in the document. The set could be defined for each feature and could consist of one word or more, and is specified using regular expressions.

Versions of the predicate BoW features were also created that could filter on basis of the predicate names, only predicates present in a set of predicates were added. These sets were also specified using regular expressions.

4.3 Coreference resolution

If two words or phrases refer to the same entity, then they are coreferent. For example "Marcus likes horses, he has got many of them. He also likes Cem." *Marcus* and the two *he* refer to the same entity, and *Marcus* is the first mention in the coreference chain. There exist another chain in the example, *horses* and *them* are also coreferent. Figure 4.3 shows a visualization of the sentence.

The coreference information, together with part of speech information, can be used to substitute words in the documents that are coreferent. The first

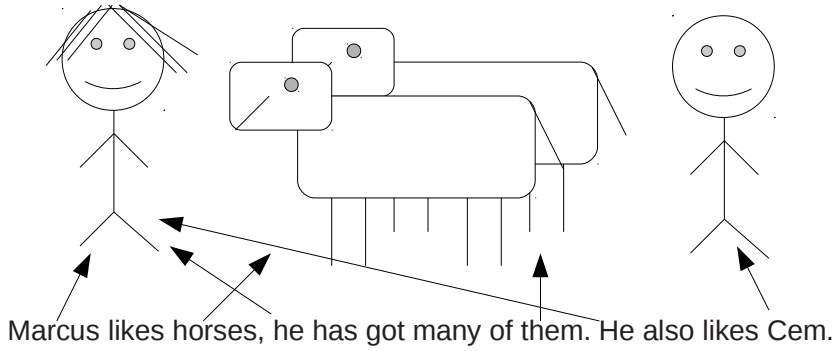

Figure 4.3: A visualization of the entities in the sentence.

mention in a coreference chain, that is the first word or phrase linked to a certain entity in the document, usually contains the most information. The reason behind this is that an entity is usually explicitly mentioned first and then implicitly referenced afterwards.

Words that are coreferent can be substituted with the first mention in the chain, although this is mostly useful with pronouns. The modified documents can thereafter be used with the different lexical features.

4.4 Classification

The easiest way to classify a horse-human pair is to take the corresponding probability vector and pick the class with the highest probability. But sometimes the probabilities are quite equal and there is no clear class to chose. A probability threshold was added, if the maximum probability in the vector is not higher then the threshold, the pair is classified as *None*. Because *None* represent a collection of actions and non-action, it is more likely to be the true class when *Ride* and *Lead* have low probabilities.

Even with the threshold, following this scheme can have the consequences that several humans are classified as riding or leading the same horse. Sometimes that can be the case, but it is much more likely that only one person is riding or leading the horse at one time. Therefore additional constraints were added to the classification: a horse can only have zero or one rider, and zero or one leader. For each class only the most probable human is chosen, and only if it is higher then the threshold.

4.5 Evaluation

For each human-horse pair the predicted class is compared to the actual class. The information derived from this can be used to calculate precision, recall and F_1 for each class. The arithmetic mean of the three F_1 values is calculated, and can be used as a general comparison value. Number of correct classifications is also calculated and a confusion matrix is created. For an example of the evaluation output from Centaur see Table 4.3 and Table 4.4, for more information about evaluation metrics see section 3.5.

	Precision	Recall	F1
None	0.9472	0.9648	0.9559
Ride	0.7685	0.7553	0.7619
Lead	0.4285	0.7553	0.2941
Mean			0.6706

Table 4.3: Precision, recall and F1.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1867	49	19
	Ride	56	176	1
	Lead	48	4	15

Table 4.4: A confusion matrix.

Chapter 5

Results

This chapter presents classification results for different feature combinations, and end with an overview of the results.

There exist four different global settings for classification, and a few local parameters for each feature, for example: normalization method and filter threshold. Furthermore coreference resolution can be added, and there exist dependencies between features. With all these variables, optimization of feature set and parameters is a difficult problem.

The results presented in this chapter are, if not otherwise noted, the baseline feature set with a specific lexical feature added, and heuristically optimized parameters. The evaluations were made using 5-fold cross-validation, see chapter 3.5.2, using solver type: 0. L2-regularized logistic regression (primal).

I experimented with many permutations of features and settings, and I present one set of BoW features that yielded a good result. For predicate features using only one lexical feature gave better results than combining them.

5.1 Geometrical (Baseline)

The set of geometrical features is defined in section 4.2.1, and is used as a baseline for the other results. See Tables 5.1 and 5.2 for results.

	Precision	Recall	F_1
None	0.9472	0.9648	0.9559
Ride	0.7685	0.7553	0.7619
Lead	0.4285	0.7553	0.2941
Mean			0.6706

Table 5.1: Precision, recall and F_1 for geometrical features.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1867	49	19
	Ride	56	176	1
	Lead	48	4	15

Table 5.2: The confusion matrix for geometrical features.

5.2 Bag of words

5.2.1 Articles

See Tables 5.3 and 5.4 for results using BoW on whole articles. IDF was used as normalization method.

	Precision	Recall	F_1
None	0.9510	0.9638	0.9573
Ride	0.7634	0.7896	0.7763
Lead	0.4545	0.2238	0.3000
Mean			0.6779

Table 5.3: Precision, recall and F_1 for BoW on articles.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1865	53	17
	Ride	48	184	1
	Lead	48	4	15

Table 5.4: The confusion matrix for BoW on articles.

5.2.2 Partial articles

See Tables 5.5 and 5.6 for results using BoW on partial articles. Instance_max was used as normalization method.

	Precision	Recall	F_1
None	0.9505	0.9643	0.9574
Ride	0.7679	0.7811	0.7744
Lead	0.4571	0.2388	0.3137
Mean			0.6818

Table 5.5: Precision, recall and F_1 for BoW on partial articles.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1866	51	18
	Ride	50	182	1
	Lead	47	4	16

Table 5.6: The confusion matrix for BoW on partial articles.

5.2.3 Captions

See Tables 5.7 and 5.8 for results using BoW on captions. Instance_max was used as normalization method.

	Precision	Recall	F_1
None	0.9506	0.9648	0.9576
Ride	0.7679	0.7811	0.7744
Lead	0.4705	0.2388	0.3168
Mean			0.6829

Table 5.7: Precision, recall and F_1 for BoW on captions.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1867	51	17
	Ride	50	182	1
	Lead	47	4	16

Table 5.8: The confusion matrix for BoW on caption.

5.2.4 Filenames

See Tables 5.9 and 5.10 for results using BoW on filenames. Instance_max was used as normalization method.

	Precision	Recall	F_1
None	0.9534	0.9622	0.9578
Ride	0.7620	0.8111	0.7858
Lead	0.4411	0.2238	0.2970
Mean			0.6802

Table 5.9: Precision, recall and F_1 for BoW on filenames.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1862	55	18
	Ride	43	189	1
	Lead	48	4	15

Table 5.10: The confusion matrix for BoW on filenames.

5.2.5 Best combination

Combining the BoW features on articles, captions, and filenames, and optimizing the parameters gave the results in tables 5.11 and 5.12

	Precision	Recall	F_1
None	0.9638	0.9638	0.9638
Ride	0.7642	0.8626	0.8104
Lead	0.5135	0.2835	0.3653
Mean			0.7132

Table 5.11: Precision, recall and F_1 for a combination of BoW features.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1865	57	13
	Ride	27	201	5
	Lead	43	5	19

Table 5.12: The confusion matrix for BoW for a combination of BoW features.

5.3 Predicate

5.3.1 Articles

Predicate feature on articles, using only head of arguments, and only arguments of the type A1, A2, and AM-MOD. Instance_IDF was used as normalization method, see Table 5.13 and 5.14 for results.

	Precision	Recall	F_1
None	0.9745	0.9498	0.9620
Ride	0.7301	0.9055	0.8084
Lead	0.4500	0.4029	0.4251
Mean			0.7318

Table 5.13: Precision, recall and F_1 for predicate feature on articles.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1838	70	27
	Ride	16	211	6
	Lead	32	8	27

Table 5.14: The confusion matrix for predicate feature on articles.

5.3.2 Partial articles

Predicate feature on partial articles, using only predicate names. IDF was used as normalization method, see Tables 5.15 and 5.16 for results.

	Precision	Recall	F_1
None	0.9557	0.9715	0.9636
Ride	0.8049	0.8326	0.8185
Lead	0.5185	0.2089	0.2978
Mean			0.6933

Table 5.15: Precision, recall and F_1 for predicate feature on partial articles.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1880	43	12
	Ride	38	194	1
	Lead	49	4	14

Table 5.16: The confusion matrix for predicate feature on partial articles.

5.3.3 Captions

Predicate feature on caption, using only head of arguments. IDF was used as normalization method, see Tables 5.17 and 5.18 for results.

	Precision	Recall	F_1
None	0.9505	0.9726	0.9614
Ride	0.8034	0.7896	0.7965
Lead	0.5000	0.1940	0.2795
Mean			0.6791

Table 5.17: Precision, recall and F_1 for predicate feature on partial articles.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1882	41	12
	Ride	48	184	1
	Lead	50	4	13

Table 5.18: The confusion matrix for predicate feature on partial articles.

5.4 Specific words

The Boolean features for the presence of certain words gave results that were lower than the baseline. Using only certain predicate names gave a positive result, and the best found were on articles with the predicates: *Ride*, *Lead*, Pull and Race, with only the head of arguments A0 and A1. IDF was used as normalization method, see Tables 5.19 and 5.20 for results.

	Precision	Recall	F_1
None	0.9518	0.9700	0.9608
Ride	0.7948	0.7982	0.7965
Lead	0.4827	0.2089	0.2916
Mean			0.6830

Table 5.19: Precision, recall and F_1 for specific word feature on articles.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1877	44	14
	Ride	46	186	1
	Lead	49	4	14

Table 5.20: The confusion matrix for specific word feature on articles.

5.5 Coreferences

Utilizing coreference resolution lowered results for the features tested. For a comparison, coreference resolution was applied on the feature with the best result: predicate feature on articles (section 5.3.1, see Tables 5.21 and 5.22 for results.

	Precision	Recall	F_1
None	0.9735	0.9509	0.9620
Ride	0.7342	0.9012	0.8092
Lead	0.4406	0.3880	0.4126
Mean			0.7280

Table 5.21: Precision, recall and F_1 for predicate feature on articles with coreference resolution.

		Predicted class		
		None	*Ride*	*Lead*
Actual class	*None*	1840	68	27
	Ride	17	210	6
	Lead	33	8	26

Table 5.22: The confusion matrix for predicate feature on articles with coreference resolution.

5.6 Overview of results

	Mean of F_1	Difference (pp)	Error reduction (%)
Baseline	0.6706	0.00	0.00
BoW: Articles	0.6779	0.73	2.22
BoW: Partial articles	0.6818	1.12	3.40
BoW: Captions	0.6829	1.23	3.73
BoW: Filenames	0.6802	0.96	2.91
BoW: Combination	0.7132	4.26	12.9
Predicate: Articles	**0.7318**	**6.12**	**18.6**
Predicate: Partial articles	0.6933	2.27	6.89
Predicate: Captions	0.6791	0.85	2.58
Predicate: Articles + Words	0.6830	1.24	3.76
Predicate: Articles + Coref	0.7280	5.74	17.4

Table 5.23: An overview of the results, with their mean F_1-value, difference and error reduction from the baseline mean F_1-value.

Chapter 6

Discussion

In this chapter the results of the thesis are discussed, and a conclusion and some possible extensions of the work are presented.

6.1 Discussion

Wikipedia was chosen as the resource of the images and text because of its size and because the images are associated with relatively large and relevant texts. The objects, humans and horses, were selected because the relatively high precision of automatic detection in images (Felzenszwalb et al., 2010).

Table 2.1 shows that there are not an equal number of the different predicates. The skewness of the distribution was not unexpected, *None* includes no relation, as well as every other relation not being *Ride* or *Lead*, but it also means there is relatively little training material for the other two relations.

Gold annotated images were used as input to the program, but I originally intended it to be extended to automatically classified images. An image classifier based on Felzenszwalb et al. (2010) was planned to be used, but the time limits of the project did not allow it. The program would have produced the same type of bounding box information as the manual annotators.

The human annotators would probably fare much better than the program though. Automatically annotated images could have created more noise for the input to the relation classification, not recognizing objects or finding non existent objects, or creating non optimal sizes and placement of bounding boxes. But still, a fully automated system architecture would have been preferable.

The baseline with the geometric features were created by experimenting with the available information from the bounding boxes. A large number of feature sets were tested, with different permutations of the geometric features. The search space for possible combinations is very large, even for a small number of features, so the search was carried out in a heuristic manner until the results were sufficiently good.

Classifying the *Lead*-relation was quite hard with only bounding boxes as the input, because there is very little difference between just standing next to a horse and leading it. I was not able to classify any *Lead* correctly until I cre-

ated features combining both Boolean and numeric features, see the last three features in section 4.2.1. With more advanced image analysis, better features could probably be created, for example with better segmentation of the objects than just bounding boxes. Information about the contours of the objects would have helped, but I think the results from only using geometrical features, based on bounding boxes, produced good results.

The semantic features use Wikipedia articles as input, but articles can have large amounts of text, describing different things, and they can have several images depicting different objects. More specific information pertaining to images would help with classification, therefore I created filters to extract partial articles (the paragraph closest to the image) and captions. Filenames were also a source of specific information, but the names were less uniform than captions, some images have very descriptive titles for example: "A_horse_in_a_landscape_with _castlemartin_in_the_background. _The_property_was_occupied_by_the_Carter_ family_from_1730_to_1850.png", others were less informative: "DMZ1.jpg".

Table 5.23 shows an overview of the results, for single BoW features captions gave the best result, followed by partial articles, filenames, and lastly articles. The order of the results were what I excepted, based on how specific information the features had about the images. But for the predicate features, the order is reversed, articles produced the best result, followed by partial articles and captions. I do not have good theory why the predicate features behave the opposite way, compared to the BoW features.

Using specific word filtering did not produce good results, although the selection of key words do heavily affect the feature. It is possible to find a better set of words, but it would probably be quite time consuming, if the process is not automated in some way.

At first I used some obvious words such as "ride" and "lead", and after that I manually read through the predicates in captions and partial articles, looking for other interesting words to try. The negative results could possibly be explained that it is not common to explicitly describe the relations in the images, and only utilizing keywords such as "ride" is of little help.

Applying coreference resolution on the documents lowered the results, Table 5.23 shows a drop of 0.38 percentage points if applied on the predicate feature based on articles. Despite the negative results, I still believe that solving coreferences could improve the results.

The coreference solver that I utilized, was designed to use another version of the CoNLL format, which had some differing columns. To be able to use the solver, some hastily changes were made to its source code, but that also introduced problems to the solver. I manually checked some coreference chains that the program created, and I encountered many strange examples, leading me to believe that the output was of low quality.

6.2 Conclusion

The results show that semantic information, in combination with geometrical features, can be useful to improve classification of relations in images. Table 5.23 shows that the error reduction is 12.9 percent by utilizing a combination of bag-of-words features, an even greater improvement is made by using predicate information, with an error reduction of 18.6 percent compared to baseline.

Utilizing corefence resolution resulted in negative results, but the interface between the semantic parser and the coreference solver was less than optimal.

6.3 Future work

There is room for improvement regarding the coreference solver, either improving the interface to the semantic parser, or changing to another solver. It could also be interesting to try other types of classifiers, not just logistic regression, and see how they fare.

Using automatically annotated images, as input to the program, could be relatively easily implemented and would make all steps in the system automated. This would be preferable, if a version of the system should be able to categorize relations in unseen images, without the help of humans.

A natural continuation of the work is to expand the number of objects and relations, for example in Felzenszwalb et al. (2010) there exist 20 different classifiers for common objects, for example: cars, bottles and birds. All, or a subset, could be chosen as the objects, together with some common predicates between the objects as the relations.

It would also be interesting to try out other sources of images and text than Wikipedia, either other resources available online, or creating a new database by annotating images with text descriptions.

Another interesting expansion of the work, would be to map entities found in the text with objects found in the image. For example if a captions says: "George W. Bush shakes hands with Göran Persson" one could create links between the image and information about the persons.

Bibliography

Deschacht, K., Moens, M., et al. (2007). Text analysis for automatic image annotation. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 1000.

Exner, P. and Nugues, P. (2012). Constructing large proposition databases. In Chair), N. C. C., Choukri, K., Declerck, T., Doan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Joachims, T. (1996). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, DTIC Document.

Moscato, V., Picariello, A., Persia, F., and Penta, A. (2009). A system for automatic image categorization. In *Semantic Computing, 2009. ICSC'09. IEEE International Conference on*, pages 624–629. IEEE.

Paek, S., Sable, C., Hatzivassiloglou, V., Jaimes, A., Schiffman, B., Chang, S., and Mckeown, K. (1999). Integration of visual and text-based approaches for the content labeling and classification of photographs. In *ACM SIGIR*, volume 99.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Stamborg, M., Medved, D., Exner, P., and Nugues, P. (2012). Using syntactic dependencies to solve coreferences. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 64–70, Jeju Island, Korea. Association for Computational Linguistics.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Tirilly, P., Claveau, V., Gros, P., et al. (2010). News image annotation on a large parallel text-image corpus. In *7th Language Resources and Evaluation Conference, LREC'10*.