Automatic estimation of semantic relatedness for sentences using machine learning.

Master's Thesis

Rob van der Goot
s1915770
R.M.van.der.Goot@student.rug.nl

Rijksuniversiteit Groningen
Faculty of Arts
Supervisor: Johan Bos
Second Reader: Gosse Bouma

23-07-2014

Abstract

Because of the growth of interest in natural language processing, semantic relatedness between words already has gotten a lot of attention in research. Since the ability to cope with natural language automatically is getting better, many natural language processing task can already process sentences. There has also been an increasing interest for semantic relatedness between sentences.

The main research question of this paper is: Which features can be automatically extracted from a natural language sentence that represents a good overview of the semantic meaning of a sentence? This can be tested by trying to predict the amount of semantic relatedness between sentences.

For training and evaluation the SICK (Sentences Involving Compositional Knowledge) corpus is used. This corpus consists of 9,927 sentence pairs and an average human judgement score on relatedness between 1 and 5. The sentences in the corpus avoid the use of named entities and encyclopedic information, as well as usage of the past tense. Therefore we can avoid a lot of preprocessing, and we can focus on the task at hand.

The barebone of the system is a random forest regressor. This regressor builds a model based on a lot of features that are extracted from many different models and representations of the original sentence pairs. Most of the features used are based on models built by the C&C tools and Boxer. From these tools the lemmatization, POS tags, a logical model and a Discourse Representation Structure are used. Other data used is WordNet data, the paraphrases database and two large text corpora (Wikipedia and the English Gigaword corpus). The models that are generated are analyzed by a python script, and properties that might contain semantic information are extracted. After this extraction the overlap of a sentence pair for this property is calculated, the result of this is the value of this feature that is inserted in our regression model.

The final system uses 18 features in total. Ranging from very simple features as sentence length and word overlap, to much more complex features as synset overlap and a Compositional Distributional Semantic Model (CDSM). The combination of all features results in a pearson correlation of 0.820, and a mean square error of 0.334. Paraphrasing can improve the precision of almost all features. The result of the total model with all features can improve to a pearson correlation of 0.837 and a mean square error of 0.322. This is a state of the art result compared to other systems built for the same task. For many possible applications this is already a usable result.

Contents

Acknowledgements

You are now reading the final result of my information science study, my Master's Thesis. I would like to thank everyone who supported me during my study years. Starting with my supervisor Johan Bos, but also the rest of the information science department at the rug for all the educational support and advice. And for this thesis in particular Johannes Bjerva for the cooperation.

Finally I want to thank the organisators of shared task 1 of Semeval 2014 [1], for putting forward the task and also for creating a very well structured and good corpus.

Further, I want to thank my great family. Not for the academic/research contribution of course, but just for the support and inspiration.

ps. Appendix A contains a list of all commands that are needed to reproduce my results.

[1] `http://alt.qcri.org/semeval2014/task1/`

Chapter 1

Introduction

The revolution in computer science that has happened roughly the last 50 years, has already solved many problems. Because of increasing computing power and smarter software, many real world application work almost flawlessly. One of the harder areas of interest in computer science is natural language processing. Natural language processing is the study of communication between humans and computers through natural language. There are still many problems in natural language processing because computers can not understand natural language in the way humans can.

One huge part of this problem is to extract and represent the meaning of natural language. The main goal of this paper is to show how to automatically extract the semantic meaning of a natural language sentence. This can be done by trying to predict the similarity between two sentences. If it is possible to extract features semantically meaningful information from both sentences and the overlap of these features give us a good prediction of their relatedness Then we can safely assume that the collection of these features represent the meaning of a sentence.

1.1 Problems

At first glance it might seem that this estimation of the similarity of semantic meaning between two sentences is only useful for usage in another task. But if we manage to get a good working system, this is also very helpful for many different problems.

The first problem that can be solved is that the meaning of a sentence can be saved in a generalized manner. This meaning can of course be used for many additional interesting research tasks and applications.

A more direct use for the relatedness prediction is evaluation. Think for example about the popular field of natural language generation. This is still a very hard task, even the evaluation of it is a difficult part. The results of a natural language generator could be tested against a gold standard using the similarity scores found in the system developed in this paper.

Another task that natural language processing could use these relatedness scores for, is trying to find alternative sentences. Because many systems that generate natural language use predictable linguistic forms, using alternative sentences between the normal sentences can make the language use more diverse. This is also a good property if you want to replicate human language behaviour as good as possible.

Search engines can also greatly benefit from semantic similarity scores. Using these scores it can quickly be guessed if a document has any semantic meaning in common with the search query. If this is the case, it is probably a relevant document, and it should be high in the results.

There are also many other uses for an automatic semantic similarity score, where this score is less important. So I will not go into great detail about them here. Think for example about translation, question answering or summarisation. A more comprehensive discussion about possible applications of semantic relatedness can be found in Budanitsky (1999).

1.2 Semantic relatedness

There are many definitions of semantics, in this paper we will make use of the definition of Saeed (2011), he defines semantics as:

> The study of meaning communicated through language.

So instead of looking for semantic relatedness between two sentences, we can also say we are looking for relatedness between the communicated meaning of two sentences. We are not really looking for the communicated meaning though, instead the focus is on the perceived meaning of the sentences of the human annotators. So the definition can be further simplified, this results in “the similarity between the meaning of sentence A and the meaning of sentence B”.

The problem in this definition is that there is no model to represent the meaning of a sentence in a complete and correct manner. There have been many efforts to create such a model (for example logical models, event-based models, DRS, thesauri etc.). All these models are representations that try to represent a real world situation in a formalized structure. Even though these models do not correspond completely with how human visualize sentences, these models are definitely useful. Because they do represent a great deal of the meaning of the sentence.

In addition there is the problem of the interpretation of a sentence, sentence meaning is subjective. If two people read the same sentence they will both interpret it differently. Unquestionably we can say that there is no such thing as the correct semantic similarity between two sentences.

Because there is no correct similarity value between two sentences, human annotated judgments should be used as gold data. These human ratings are not given by the sender of the message (who knows the intended meaning), but are given by the receiver of the message (that has his/her own interpretation of the sentence). Consequently it is impossible to estimate the degree of the semantic similarity 100% similarly to humans.

1.3 Semantic similarity vs semantic relatedness

A clear distinction that is made in most literature is between semantic similarity and semantic relatedness. Semantic similarity is a sub-category of semantic relatedness. This means that relatedness includes more relations between words than similarity. Two word relations that are good examples of fitting only in the former category (word-pairs that are related) but not in the latter category (word-pairs that are similar), are meronyms and antonyms. Meronyms represent a whole-part relation, think about building-door, laptop-screen or bus-wheel. These words are not semantically similar, but they are definitely related. Words that are opposites of each other are called antonyms. They can be related but they are very often not similar at all, they even have an opposite meaning. Think for example about colors, 'black' and 'white' are antonyms. They are related to each other, and a sentence meaning is usually not altered much after changing one color. But these colors are not similar at all, instead they are meant to be completely different. For sentences this distinction is less clear then it is for words, thus semantic relatedness and semantic similarity are used interchangeably in the rest of this paper.

The distinction between relatedness and similarity can already expose a hard case for predicting a similarity value, namely contradictory sentences. If two sentences contradict each other, they are definitely related in a way. But are they similar? If you would ask a logician the answer would be 'no', because the meanings are opposites. Some linguists might answer 'yes' though, because the only difference might be a negation. Thus there is no consensus on the answer of this question, and there might be a lot of variance in the human judgments of contradictory sentences.

1.4 Research question

It is impossible for a computer to try to guess the degree of semantic relatedness between two sentences right away. That is why the focus is more on calculating the overlap of pieces of semantic meaning between two sentences. The main thing needed for this, are different kinds of semantically informative information that can be extracted from sentences and then compared to each other. So the focus will be on the overlap of semantic features that can automatically be derived from sentences in natural language. Consequently the main research question is: Which features can be automatically extracted from a natural language sentence that represents a good overview of the semantic meaning of a sentence?

This main research question can be splitted down into many other questions. Starting with: which representations of a sentence can be used best to represent the semantic meaning of a sentence? But also within these representations there are many different aspects that might have a very different contribution to the semantic meaning. So sub-questions here will be: Is model X a good model for representing the semantic meaning of a sentence or a real world situation. These sub-questions can then be further divided in questions about the features separately, thus: Is feature A of model X an informational feature regarding semantic meaning?

Furthermore large corpora can tell us a lot about co-occurrences and thus relatedness, and can thus also provide us with features. Finally the use of paraphrases will also be exploited to see if it helps to generate semantically similar sentences, and see if these new sentences give a better score on similarity. This raises two more questions, firstly: To what extent can co-occurrences of words help us to estimate the semantic similarity between sentences? And secondly, can paraphrasing be used to improve the effects of the features?

Chapter 2

Related work

The last couple of years there has been a growing interest in semantic similarity. Most of the research done before is based on semantic similarity between words. This paper will focus on semantic similarity between sentences instead. But since sentences consist of words, the work done on word similarity can also be very useful for sentence level comparisons. Since semantic relatedness and semantic similarity for words can both be interesting for the task at hand, both will be discussed.

2.1 Semantic relatedness between words

In previous research about semantic relatedness between words three different approaches can be distinguished based on the data source used. The first type of data that is used often is a large text corpus. In these large text corpora can be searched for co-occurrences and similar language constructions to find related words. See for examples of this approach Miller and Charles (1991) or for an adaption to larger snippets of text Islam and Inkpen (2008). A nice detailed overview of how this correlation can work for particular examples can be found in Lund and Burgess (1996). The main advantage of using co occurences of a large corpus is robustness, and even if language changes the system only needs a new corpus that consists of raw text. So there is no need for annotation, which makes this an attractive data source.

The second data source that is very popular to use is Wikipedia. Wikipedia can firstly be used as large text corpora to search for co-occurrences and other language phenomena as described in before This is not what it is normally used for in the research about semantic relatedness though. In research in this area the link system and categorical system of wikipedia are very popular information sources. The link system can be used to measure the number of clicks needed to get from one page to another, and the distances between categories is also a meaningful measure. These systems perform well on rather small and simple test sets.

Results with a correlation from 0.55 (Strube and Ponzetto, 2006) to 0.78 (Witten and Milne, 2008) are reported, using just the wikipedia links. But these results are not comparable to the results of the sentence comparison task at all, since no user generated language is used to evaluate. Most of the previous research uses the WordSimilarity-353 collection (Finkelstein et al., 2001) for evaluation. This collection consists mostly of words that have an entry in wikipedia.

Another very popular data source for this task is WordNet (Fellbaum, 1998). WordNet is a large thesaurus which includes a lot of information about words. WordNet is used in a variety of methods to estimate relatedness between words. The correlation scores retrieved by WordNet based systems are usually higher then the correlation scores obtained by Wikipedia based systems. A nice overview of different methods of extracting word similarity from WordNet can be found in (Budanitsky and Hirst, 2006). This shows that the correlations are in the range of 0.74 (Hirst and St-Onge, 1998) up to 0.85 (Jiang and Conrath, 1997). These results are generally better then the results that are achieved by using wikipedia.

2.2 Semantic relatedness between sentences

Sentence relatedness is what we are after, and there has already been some research on this subject. A highly similar task was introduced for Semeval 2012 (Agirre et al., 2012). There is a large variety of methods that are used in this shared task. The main data sources that were used at that time are WordNet and raw text corpora (mainly Wikipedia), but also dictionaries, stopword lists and paraphrases were popular resources. The teams participating in this shared task also made use of existing NLP tools, just as we will. The most popular tools were lemmatizers and POS taggers, followed by other parsers, word sense disambiguation and semantic role labelers.

The standard evaluation metric for semantic similarity scores is the Pearson correlation. This has become the standard because it represents how strong the relation between the gold-standard and the systems estimation is. Another good way to compare the results is the mean square error (mse). The mse tells us how large what the mean difference between our own estimates and gold value is.

The Pearson correlations in Semeval 2012 task range from negative up until 0.8239 (no mse's were reported). While human annotators themselves are reported to score approximately a 0.9 correlation on the human judgements. So the results are promising, but the humans still do a better job at predicting semantic relatedness. This is not surprising since the annotators are also human. Taking all this into consideration the scores are very high, and it will be hard to improve these results.

Chapter 3

Data and resources

Because a supervised learning approach is used, the first thing needed is a labeled training corpus. This corpus is described in the first section of this chapter. To gain more information from the original corpus the raw sentences of the corpus are converted to different models/representations, this preprocessing is discussed in the second part of this chapter. To get even more information about the words used in the corpus additional data from external sources is needed. This external data is discussed in the final part of this chapter.

3.1 Corpus

The only corpus that is made with this exact task in mind is the SICK (Sentences Involving Compositional Knowledge) (Marelli et al., 2014) corpus. This corpus is made available by SemEval 2014, and is used for the SemEval shared task (Marelli et al., 2014). The SICK corpus is the result of merging two other corpora. The first also originates from SemEval, but from another task and year[1]. This corpus consists of descriptions of videos by human annotators. The second corpus used is a corpus with descriptions of images instead of videos (Rashtchian et al., 2010). Some of these sentences are paired with descriptions of the same video/image (they are often highly similar), and some sentences are paired with sentences of other videos/images. Also some of these sentences were negated to obtain some sentence pairs that are a contradiction.

The SICK corpus consists of 9,927 sentence pairs annotated with human judgments of semantic similarity, and semantic relation. Below are some random sentence pairs of the corpus to give an idea what the data looks like, and what kind of sentences are there to be found.

[1]`http://www.cs.york.ac.uk/semeval-2012/task6/index.php?id=data`

Listing 3.1: Some example sentence pairs, taken straight out the corpus.

```
1105
A woman is riding a horse
A man is opening a package that contains headphones
1.1
NEUTRAL

634
A kid swimming in the ocean is tossing a coin into the pool, near the
    man
A father is launching the daughter in a swimming pool
1.6
NEUTRAL

8400
There is no man playing guitar next to a drummer
A man is playing guitar next to a drummer
3.785
CONTRADICTION

2869
Two bikes are being ridden by two people
Two people are riding a motorcycle
3.8
NEUTRAL

44
Two young women are sparring in a kickboxing fight
Two women are sparring in a kickboxing match
4.9
ENTAILMENT
```

To make the examples more clear the tabs in the raw data are replaced with newlines. The first thing each sentence pair has, is an id. This id is unique and ranges from 1-10,000 (some are missing, hence the total of 9,927 sentence pairs). After this the two sentences that form the sentence pair are stated. Finally we are provided with the gold score for both parts of the SemEval shared task. The first score is the gold relatedness score. This score is the average of the ratings of 10 different human annotators. The annotators are instructed to give an estimation of the similarity in meaning between the two sentences within the range of 1 (not related) up to 5 (very related). This score is the score I am trying to predict in this paper.

The second gold score given here needs a little more explanation, it represents a category. It represents the relation between both sentences, as judged by humans. The categories that are used in the corpus are: contradiction, entailment and neutral. A contradictory relation means that it is not possible that both sentences are true at the same time. An entailment relation however, means that if one sentence is true the other one must also be true. Additionally there is the neutral relation. This relation is used when none of the former relations describe the sentence. This basically means that there is no relation between the sentences, and the similarity score will often be low.

In most of the examples in listing 3.1 is easy to see for humans which sentences are descriptions of the same video/image (id's 44 and 2869). The also applies to descriptions of totally different video's (id's 634 and 1105), as well as finding added negations (id 8400).

Table 3.1: Descriptive statistics for the SICK corpus

	N	Mean	Standard deviation
Contradiction	1,459	3.60	0.450
Entailment	2,857	4.57	0.341
Neutral	5,611	2.98	0.927
All	9,927	3.53	1.016

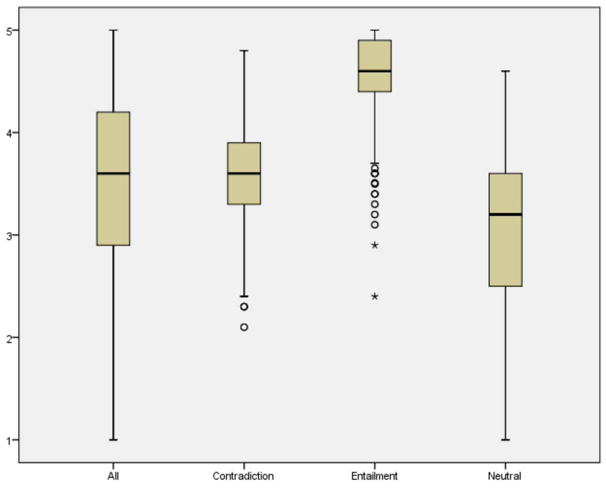


Figure 3.1: Boxplots for each sentence relation category and all combined.

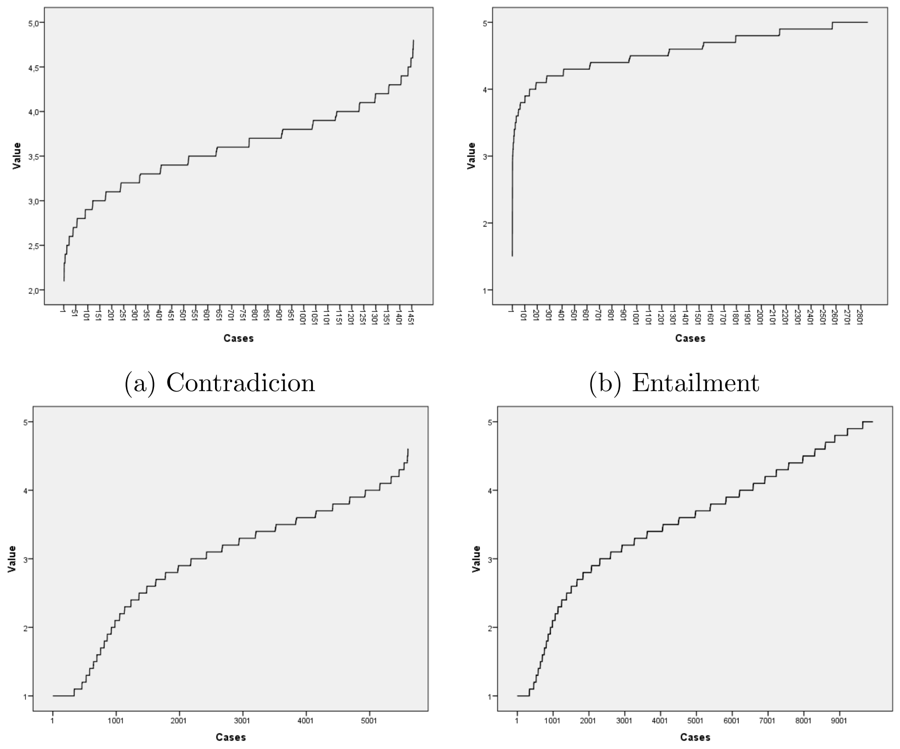
The SICK corpus consists of basic sentences. Almost all sentences are grammatically correct, none of the sentences have encyclopaedic knowledge or proper nouns in them, and they are all in present tense. The sentences are also already tokenized. The combination of these factors make the usage of these sentences as training and test data very convenient. It is not a good reflection of language use in the real world though. The main reason to have a corpus this way is that many problems are already taken care of, so the focus can be completely on the task itself.

3.2 A look at the training data

Before starting to dig into the task, it is good to get an overview of how the value of relatedness is distributed. Table 3.1 shows the summary of the data found in the corpus. The most interesting part is the row for 'All', because all sort of sentence relations need a similarity score. But the other entries in the table are also interesting, because sentence relation is one of the features.

The data in the table shows that entailed sentences have a very high score without much diffusion, while neutral sentences have the lowest scores. In the row of the table that shows the information about all sentences we can see that the data has a pretty central mean, and a standard deviation of 1.016. This means that most of the data (68.2%) can be found in the range of 2.514 - 4.546. For a better comparison of how the data is distributed, see the boxplot in figure 3.1.

This boxplot shows that the data for all the sentences are even distributed. Only the lower values (1-3) are a little less common. There also seems to be a relation between sentence similarity and sentence relation. Neutral sentences have the lowest



(a) Contradicion (b) Entailment

(c) Neutral (d) All

Figure 3.2: The distributions of the semantic similarity scores as annotated by humans.

values, followed by contradiction and entailment.

Figure 3.2 confirms our findings and shows in graphs how the similarity scores are distributed.

3.3 Preprocessing of the SICK corpus

Only basic features can be extracted directly from the raw sentence pairs. To get more semantic information, the sentences needs to be parsed and converted to different representations/models. The tools used for this purpose are the C&C tools (Curran et al., 2007) and Boxer (Bos, 2008). Using the output of these tools gives the ability to look deeper into the semantic meaning of the sentences and leads hopefully to more useful features.

To demonstrate and explain the output files, I will use the first sentence of sentence pair with id 2869 from the sick corpus as example sentence. This sentence is: "Two bikes are being ridden by two people".

3.3.1 Syntax tree

The CCG parser of C&C tools (Steedman, 2000) outputs syntax trees as .ccg files. An example of the main part of the ccg file can be found in listing 3.2.

The syntax information is not used very extensively in this work. Thus a full description of this structure goes beyond the scope of this work. Instead I will only discuss the parts that are used in our own script, with line #10 as example. The first

Listing 3.2: The main part of the .ccg file of our example sentence.

```
ccg(1,
 ba(s:dcl,
  lx(np, n,
   fa(n,
    t(n/n, 'Two', 'two', 'CD', 'I-NP', 'O'),
    t(n, 'people', 'people', 'NNS', 'I-NP', 'O'))),
  fa(s:dcl\np,
   t((s:dcl\np)/(s:ng\np), 'are', 'be', 'VBP', 'I-VP', 'O'),
   fa(s:ng\np,
    t((s:ng\np)/np, 'riding', 'ride', 'VBG', 'I-VP', 'O'),
    fa(np:nb,
     t(np:nb/n, 'a', 'a', 'DT', 'I-NP', 'O'),
     t(n, 'motorcycle', 'motorcycle', 'NN', 'I-NP', 'O')))))).
```

argument ((s:ng\np)/np)is the rule the parser used to determine the POS-tag of this particular word, so this part includes no direct semantic information. After this, the raw word follows (riding). The third argument is the lemma of the word (ride), this can come in very useful if the words need to be compared. The last and most important thing included is the POS-Tag, for our example the POS-Tag is VBG. The POS-Tags used by this parser are the same ones used by the Penn-treebank (Santorini, 1990).

3.3.2 Logical model

Boxer can use a variety of model builders. The best results can be achieved by using a combination of two of those model builders. First Paradox (Claessen and Sörensson, 2003) will try to build a logical model, Paradox is an efficient state of the art model builder. The model built by Paradox is not a minimal model though, only the domain size is minimal. This domain size can be used to rebuild the model in Mace-2 (McCune, 2001) to generate a minimal logical model. Boxer can also try to use world knowledge to create a larger, more informational model. This world knowledge consists of is-a relations from WordNet, these are used to give more information about the instances that are identified. So this can broaden the concepts, and is basically used to describe that if 'a man' is doing something, 'a person' is also doing something. Since this model is giving us more information about the instances of the model, this larger model is used for the features. The logical model for our example sentence is shown in listing 3.3.

Listing 3.3: The .mod file for our example sentence

```
model([d1],
  [f(1,c2number,[d1]),
   f(1,n1abstract_entity,[d1]),
   f(1,n1animate_thing,[d1]),
   f(1,n1artefact,[d1]),
   f(1,n1automotive_vehicle,[d1]),
   f(1,n1bike,[d1]),
   f(1,n1entity,[d1]),
   f(1,n1event,[d1]),
   f(1,n1numeral,[d1]),
   f(1,n1object,[d1]),
   f(1,n1person,[d1]),
   f(1,n1physical_entity,[d1]),
   f(1,n1psychological_feature,[d1]),
   f(1,n1selfC45propelled_vehicle,[d1]),
   f(1,n1symbol,[d1]),
   f(1,n1vehicle,[d1]),
   f(1,n1wheeled_vehicle,[d1]),
   f(1,n2being,[d1]),
   f(1,n2communication,[d1]),
   f(1,n3conveyance,[d1]),nltk
   f(1,n3instrumentality,[d1]),
   f(1,n3sign,[d1]),
   f(1,n6unit,[d1]),
   f(1,v1go,[d1]),
   f(1,v1ride,[d1]),
   f(2,card,[ (d1,d1)]),
   f(2,r1agent,[ (d1,d1)]),
   f(2,r1patient,[ (d1,d1)])]).
```

This file uses the prolog notation, each property ‘f’ represents a fact in the knowledge-base for this sentence. The second argument of the ‘f’ represents the name of the predicate, the third argument is the list of variables used for the unification in prolog, while the first argument of ‘f’ is actually just the size of this list. All the predicates with two instances are thus predicates with the number ‘2 as first argument, and these represents relations between two instances. All properties of instances are represented here as predicates with the number ‘1.

The most interesting aspect of the output by C&C is that it does not only output the model of both sentences separately, but it also tries to generate a combined model for both sentences. This model tells us a lot about how the relation between the two sentences. If it is impossible to create such a model the sentences are contradictions and their meanings will probably not differ much (they will just be opposites). If it is possible to create a model but the combined model is not much larger then the seperate models, the sentences are probably very related. Finally there is the case that there is a combined model that is substantially bigger then the separate models. This means that the sentences both contribute a lot of different information to the combined model, thus the similarity score for this sentence pair is probably low.

3.3.3 Discourse Representation Structure

The Discourse Representation Theory (DRT) (Kamp et al., 2011) is a framework used to express meaning by abstract mental representations. These representations are often shown as boxes (or in a bracket structure, but that is harder to read for humans).

Shortly said, a Discourse Representation Structure (DRS) is a nested structure containing boxes that contain lists of formal logic variables and predicates, these boxes can also be combined using logical connectives. Boxer (Bos, 2008) is used to obtain Discourse Representation Structures automatically, these are outputted as .drs files. The drs file based on our example sentence is found in listing 3.4.

Listing 3.4: The .drs file for our example sentence

```
sem(1,[1001:[tok:'Two',pos:'CD',lemma:two,namex:'O'],1002:[tok:bikes,
    pos:'NNS',lemma:bike,namex:'O'],1003:[tok:are,pos:'VBP',lemma:be,
    namex:'O'],1004:[tok:being,pos:'VBG',lemma:be,namex:'O'],1005:[tok
    :ridden,pos:'VBN',lemma:ride,namex:'O'],1006:[tok:by,pos:'IN',
    lemma:by,namex:'O'],1007:[tok:two,pos:'CD',lemma:two,namex:'O
    '],1008:[tok:people,pos:'NNS',lemma:person,namex:'O']],drs([[]:B
    ,[]:C,[]:D],[[1006]:rel(C,B,agent,0),[1008]:pred(B,person,n,0)
    ,[1007]:card(B,2,eq),[]:rel(C,D,patient,0),[1005]:pred(C,ride,v,0)
    ,[1002]:pred(D,bike,n,0),[1001]:card(D,2,eq)])).
%%%  ______________
%%% |x1 x2 x3      |
%%% |..............|
%%% |agent(x2,x1)  |
%%% |person(x1)    |
%%% ||x1| = 2      |
%%% |patient(x2,x3)|
%%% |ride(x2)      |
%%% |bike(x3)      |
%%% ||x3| = 2      |
%%% |______________|
```

The output is in a prolog notation, the first line is thus a bracket structure and is hard to read. This is the reason that boxer included the box seen at the bottom of the file. This box is commented out and thus only used for humans to understand the DRS easier.

A DRS consists of a list of discourse referents and a list of discourse conditions. In our example the discourse referents are x1, x2 and x3. These are the instances to be found in the DRS, and things can be said about these using discourse conditions. The discourse conditions in our example are agent, person, patient, ride and bike. By inserting the discourse referents in the discourse conditions, it is possible to unify and use the information in the DRS.

The first line of this listing is a prolog list that contains two things. First there is a list with information about every word in the sentence, namely the tokenized word, the POS-tag and the lemma. Secondly there is a list representation of a DRS. Because this is the interesting part, I took out this part and cleared the code with newlines and indentation.

As better visible now in listing 3.5 it can be seen that the DRS is separated in 2 parts. The first part consists of the variables, called x1, x2, x3 in the box of listing 3.4 before. They are now represented as prolog variables with alphabetical characters used as names. The second part contains the information about the variables. Each variable has it's own predicate (pred()), which defines what the variable represents. Besides the predicates there are relations (rel()), they represent the relations the variables have to each other.

Listing 3.5: The prolog representation of the DRS indented.

```
drs(nltk
    [
        []:B,
        []:C,
        []:D
    ],
        [1006]:rel(C,B,agent,0),
        [1008]:pred(B,person,n,0),nltk
        [1007]:card(B,2,eq),
        []:rel(C,D,patient,0),
        [1005]:pred(C,ride,v,0),
        [1002]:pred(D,bike,n,0),
        [1001]:card(D,2,eq)
    ]
)
```

3.4 WordNet

To cope with conceptual differences in sentence pairs information about the relations . An example where this is useful is when one annotator describes a person as a man, while another annotator might describe the same entity as a boy or a person. This information is available in the WordNet thesaurus (Fellbaum, 1998). WordNet groups words in sets of synonyms called synsets. These synsets are then connected in one large thesaurus using hypernyms, hyponyms, meronyms and holonyms.

Hypernyms represent an "is a relation". So if word 1 is a hypernym of word 2, this means that word 1 also fits in the larger category of word 2 (for example, car is a hypernym of vehicle). A hyponym is the same relation as a hypernym but then reversed. If word 1 is a hypernym of word 2, then word 2 is a hyponym of word 1.

A meronym is a "part of" relation. So word 1 is a meronym of word 2 means that word 1 is a part of word 2 (for example, screen is a meronym of laptop). A holonym is the reverse relation of a meronym.

To gather the information of the synsets easily in a python script, the nltk python package (Bird, 2006) is used. This package makes it easy to directly access the data and use pre-defined functions of the package that already does certain calculations.

3.5 Skip-Gram model

A Skip-Gram is almost the same as the well-known N-Gram, it has one modification though. Instead of consisting of N consecutive words, a constant number of words are skipped (usually 1) to get the next word for our N-Gram. The knowledge of words that co-occur with a specific word is information that has a lot of value about the meaning of that word. This information can be used to calculate the similarity between word meanings, which can be combined to get a similarity value for a sentence.

To get a good model I have used the open-source word2vec tool (Mikolov, 2013). As input the first billion characters of wikipedia are used as suggested on the word2vec website. In addition to this training data I added the English Gigaword corpus (Parker et al., 2009), for the reason that data taken from one source might be to homegenic.

All this data is first preprocessed, including tokenization and conversion to lower

Listing 3.6: Some example paraphrases.

```
| woman ||| chick |
| woman ||| dame |
| woman ||| daughter |
| woman ||| donna |
| woman ||| female |
| woman ||| femme |
| woman ||| girl |
| bike ||| bicycle |
| bike ||| bicycling |
| bike ||| bikes |
| bike ||| biking |
| bike ||| cycling |
```

case. The word2vec tool provides a Perl script that already does this. The vectors are trained using the parameter settings suggested by the creators of the word2vec tool[2]. These are a vector dimensionality of 1600 with a context window (N) of 10 words.

3.6 Paraphrases

Paraphrasing means replacing a part of a sentence in order to obtain a new sentence that does not alter the original meaning of the sentence, even though the syntactic structure may be different. Using synonyms of words is in fact a simple form of paraphrasing, but paraphrasing also takes into account multiple word expressions or sub-sentences. The state of the art paraphrases collection is the PPDB(ParaPhrase DataBase) (Ganitkevitch et al., 2013).

This database contains a lot more information then just the paraphrases, but most of it is of little interest for our purposes. Many information is about the POS-tags of the word, how the word is retrieved and overall statistics as how frequent it is. If information like this is needed it is better to retrieve it from one of our models. Because they take the context into account, the values will be more precise. So the database is preprocessed, and all the information that is not used is removed for the sake of efficiency. The result of this is a text file with a list of phrases and their corresponding paraphrases. To get an idea of what this data looks like, there are some examples in listing 3.6

[2]https://code.google.com/p/word2vec/

Chapter 4

Methodology

Because the goal is to use many different features to predict a numerical target value and there is a very good training corpus available, the choice for using a regression model to estimate the similarity of sentence pairs is obvious. From every model/representation as described in chapter 3 the aspects that distribute to the semantic meaning of the sentences have to be extracted. After this is done for both sentences the overlap of these aspects can be calculated and this will be an estimate of the similarity of the sentences focusing only on that particular aspect. This value will be the value inserted into our regression model.

The programming language that is best suited for this task is Python. The main reason to choose Python is that text processing scripts can be written relatively quick and easy. It also offers many additional packages that already provide functionality that is necessary (Especially the Scikit-learn regression model, and the NLTK WordNet acces). The source of the script that calculates the values for the features can be found at our github repository [1]. Unfortunately there is much more needed to gather the necessary data, all the steps that have to be taken to reproduce the results of this paper can be found in appendix A.

This chapter will first explain the choice of the regression model that is used. The remainder of this chapter is divided by the different types of the data that is used for the features, and for each type of data each feature is discussed in it's own section. To conclude this chapter the use of paraphrases to improve the precision of each feature separately is discussed. The discussion of each feature will end with an example calculation for the sentence pair with id '2869'. This sentence pair consists of the following two sentences:

```
Two bikes are being ridden by two people
Two people are riding a motorcycle
```

4.1 Regression model

There are many different regression models available that all have their own advantages and disadvantages. The choice of a regression model is based on three demands. Firstly our regressor should be robust. This is important when using human annotated data that can easily have some outliers, this for example very probable with the judgements for contradictory sentence pairs. Another property of the regressor that is needed for

[1] `https://github.com/bjerva/semeval-relatedness`

this task is that it does not overfit easily, otherwise we will not be able to train and test properly. The last very important property of the regressor that is needed, is that it can let relative weaker features also contribute in the total model. Because we have a lot of different features there will be some features that contribute relatively less. But the combination of all of these relatively less useful features might be a good a significant contribution to the total model.

Taken all these demands in consideration a random forest regressor is an appropriate choice. This regressor meets all our requirements, and is especially known for taking into account the less useful features. So this regressor gets the most of the entire combination of all the features, without having to worry about which feature works better and which has a worse predictive power. Because python is used for the rest of the project, the scikit-learn (Pedregosa et al., 2011) implementation of the random forest regressor is used.

4.2 Raw sentences

The following features are taken from the raw data in the corpus, so they can be used as baseline features.

4.2.1 Sentence length difference

The number of words in a sentence does not give us any information about the semantic meaning of the sentence. It does give an estimate of the complexity of a sentence though (Flesch, 1948). So even though this might be a very weak feature in the total model, it can be a contributing feature if it is already known that the meanings of two sentences are somewhat related. This is the case if we combine this featurewith the upcoming features, so this feature might have a small but significant contribution to the model. The sentences are stored as lists of words, so the difference in sentence length can be calculated using the following formula:

$$|\frac{length(sen_a) - length(sen_b)}{min(length(sen_a), length(sen_b))}|$$

Example

The values used here are the length of the first sentence and the length of the second sentence. This is easy to get in our implementation because the sentences are stored as lists of words. So the example sentences are stored like:

```
[Two, bikes, are, being, ridden, by, two, people]
[Two, people, are, riding, a, motorcycle]
```

If the formula is filled in, this results in:

$$|\frac{8-6}{min(8,6))}| = \mathbf{0.333}$$

4.2.2 Word overlap

Word overlap is a straightforward feature that is often used as a baseline. There are many small variations possible to calculate the word overlap. A very basic method to calculate the word overlap is used. The sentences are already tokenized, so the words can easily be gathered by just splitting the sentences by whitespaces. After this the sets of these words for each sentence are taken, so that every word only occurs once. This makes sense because most words that are used twice in one sentence are stop words. Hence they do not provide much information about the meaning of the sentence, especially not if the word order is not taken into account anymore. To adjust for the lengths of the sentences the length of both sentences is used separately in the formula. For word overlap the following formula is used:

$$\frac{(length(set(sen_a) \cup set(sen_b)) - length(set(sen_b)))}{length(set(sen_a))}$$

Example

The resulting sets of our two example sentences are found in listing 4.1

Listing 4.1: The sets of words used in the example sentences

```
sentence1: [Two, bikes, are, being, ridden, by, two, people\n]
sentence2: [Two, people, are, riding, motorcycle\n]
combined: [Two, bikes, are, being, ridden, by, two, people\n, people.
    riding, motorcycle\n]
```

As can be seen from our example the capitals and newline characters are kept in the words. This means the assumption is made that the first word and the last word can have a special meaning in comparisons to all the other words.

$$\frac{11-5}{8} = \mathbf{0.75}$$

4.2.3 Weighted word overlap

In normal language usage some words are very common while many words are less commonly. The more common used words usually give less information about the meaning of the sentence. These words are usually stop words, while the less common words give us a more distinguishable idea of what the sentence is about. This effect can be easily demonstrated when looking how common the words of our example sentences are. In listing 4.2 the frequencies of every word in our example are shown (counted in the SICK corpus). For humans it was already obvious that the word 'bikes' is more semantically informative then the word 'are', but using weighting we can also automatically try to estimate the informative value of words.

Listing 4.2: The words frequencies for our example sentence.

```
two     bikes   are     being   ridden  by      two     people
1535    39      3806    1080    86      1413    1535     973
```

The main problem with this data is that it only shows estimates the importance of words instead of entire sentences. To use this data the frequencies of both sentences needs to be combined. The final value of this feature will also be a word overlap, but it uses the weights of the words in the sentences. For overlap only the words that occur in both sentences can be of meaning, since the other words can not be compared to words of the other sentence. For all words that occus in both sentences we will calculate their weight using the following formula:

$$\frac{\#sentences - word\_freq}{\#sentences}$$

The word-freq in this formula is the count of the number of occurrences of a word in the entire corpus. After getting these values there still needs to be a normalisation for the length of the sentences, otherwise long sentences will have more change to get a higher value. So the total sum of all the words is divided by the number of words that are used in the summation.

Example

There are four words in the first sentence that also occur in the second sentence, so the weight of all these words should be combined. Note that the word ‘two’ is double, since this occurs two times in the sentence. If a word occurs multiple times it is more important, so it is desirable to have it in the formula twice too.

$$\text{two} \quad \frac{19854 - 1535}{19854} = 0.923$$

$$\text{are} \quad \frac{19854 - 3806}{19854} = 0.808$$

$$\text{two} \quad \frac{19854 - 1535}{19854} = 0.923$$

$$\text{people} \quad \frac{19854 - 973}{19854} = 0.951$$

This results in:

$$\frac{0.923 + 0.808 + 0.923 + 0.951}{4} = \mathbf{0,901}$$

4.3 Syntax tree

For a description of the syntax tree used here, see section 3.3.1. Usually the POS-tags of verbs and nouns have the highest semantic informational value. So these 2 POS-tags have their own overlap feature.

4.3.1 Noun overlap

A collection of all nouns that occur in a sentence can give a good summary of the theme of a sentence. The nouns are thus a good source for recognizing the semantic meaning of a sentence. To calculate the overlap lemma's can be used in favor of the real word. Then it does not matter if a sentence is about a person or about persons.

$$\frac{length(nouns\_sen_a \cap nouns\_sen_b)}{length(nouns\_sen_a \cup nouns\_sen_b)}$$

Example

The nouns that are found in respectively the first sentence and the second sentence are:

```
[person, bike]
[person, motorcycle]
```

If these lists are inputed in the formula, the outcome will be as follows:

$$\frac{1}{3} = \mathbf{0.333}$$

4.3.2 Verb overlap

A verb is a word that describes an action, consequently verbs can have a great contribution to semantic meaning. The calculation of verb overlap uses the same method as the noun overlap. So the formula will be:

$$\frac{length(verbs\_sen_a \cap verbs\_sen_b)}{length(verbssen_a \cup verbs\_sen_b)}$$

Example

The verbs that are found in respectively the first and the second sentence are:

```
[be]
[be, ride]
```

This results in:

$$\frac{1}{2} = \mathbf{0.5}$$

4.4 Discourse Representation Structure

In the box in listing 3.5 it is directly visible that the discourse conditions gives a great deal of semantic information of a sentence. The first DRS feature will thus only use the discourse conditions. Additionally the second and third feature will also take the discourse referents into account, by coupling them with their meanings.

4.4.1 Discourse condition overlap

Since the theme of the DRS is already visible by looking only at the discourse conditions, the amount of overlap between these discourse conditions can be a good estimator for the relatedness of the sentences. So our first feature created for DRS, is just an overlap of these discourse conditions. The formula used for this overlap is:

$$\frac{length(set\_a \cap set\_b)}{length(set\_a \cup set\_b)}$$

Example

As can be seen in section 3.3.3, the discourse conditions of the first sentences are:

```
['person', 'ride', 'bike', 'patient', 'agent']
```

The discourse conditions of the second sentence are not very different, because many of the same properties can be given to the discourse references. Our second sentence has the following list of discourse conditions:

```
['person', 'ride', 'patient', 'motorcycle', 'agent']
```

If these sets are inserted in our formula that was given before, the result will be:

$$\frac{4}{6} = \mathbf{0.667}$$

4.4.2 Agent overlap

An agent in a drs structure is the discourse reference (or entity) that initiates the main action of the sentence. This is very useful information if we speak about the meaning of a sentence. But since most of the sentences only have one agent, this feature can not give a very precise overlap score.

Example

The only agent found for our example is ‘person’. The formula used here is again the overlap formula for sets:

$$\frac{length(set\_a \cap set\_b)}{length(set\_a \cup set\_b)}$$

The second sentence also has only one agent which is also ‘person’ so the agent overlap is perfect, as shown below:

$$\frac{1}{1} = \mathbf{1.0}$$

4.4.3 Patient overlap

In a normal DRS it is not uncommon to have multiple agents or patients. In the SICK corpus this is very rare though. There are only a couple DRS’s with two agents. On the other hand, there is no sentence in the corpus that has multiple patients. So this feature is binary, the patient can either be the same (score = 1.0) or it can be different (score = 0.0).

Example

The patients found in the first sentence is 'bike', but the second sentence has 'motorcycle' as patient. So there is no overlap at all, and the value of this feature will be:

$$\frac{0}{2} = \mathbf{0.0}$$

4.5 Logical model

A logical model consists of 3 kinds of information: instances, predicates about these instances and the relation between the instances. So it makes sense to create features for the overlap of instances, predicates and relations.

4.5.1 Instance overlap

The instances are the 'variables' in the logical model. They can be initiated and then used to state facts about themselves. If the models of both sentences and the combined model is used, it is possible to see if the second sentence includes any complementary instances with respect to the first sentence. For doing this, the following formula is used:

$$1 - \frac{\#instances\_sen1\&2 - \#instances\_sen1}{\#instances\_sen2}$$

Example

In a logical model each entity that takes part in that model is represented as an instance. The logical model of our example (section 3.3.2) contains only one instance, 'd1'. The logical model of the other sentence of this sentence pair contains also 1 instance. Since these 2 instances can be unified by each other, the combined model also contains 1 instance, hence the score is:

$$1 - \frac{1-1}{1} = \mathbf{1.0}$$

This means that both sentences can use the same instance if the logical models are combined. Thus the overlap is perfect.

4.5.2 Predicate overlap

In logical models predicates are used to state facts about the instances. As explained in section 3.3.2, these instances can be recognized by having a '1' as first argument of the fact in the model file outputted by C&C. It comes in very handy here that we are not only provided with the logical models for both sentences separately, but we also have a model file that represents a logical model for both sentences combined. These three models can be used to get the overlap, using the formula below:

$$1 - \frac{\#predicates\_sen1\&2 - \#predicates\_sen1}{\#predicates\_sen2}$$

For contradictory sentences there is no combined logical model available, hence the value 0 is used for this feature then. Because there is no overlap to be found at all. This particular issue is resolved later when the negations are removed, see section 4.7.

Example

Because the model used also makes use of world knowledge to generate more general predicates, the lists of predicates are quite large. As can be counted in our example model in section 3.3.2 our example contains 25 different predicates. All of these represent facts about the instances of the first sentences. The model of the second sentence consists of the exact same predicates except for one predicate. The predicate that is not in the second sentence is 'bike', instead of bike this model has the predicate 'motorcycle'. This means that both models have 25 predicates, and when they are combined into one model it will have one additional predicate. The filled in formula will then be:

$$1 - \frac{26 - 25}{25} = \mathbf{0.96}$$

4.5.3 Relation overlap

The relations in a logical model are predicates that take 2 variables. These relations give us much semantic information since they tell us how the different instances are related to each other. The overlap of the relations is calculated in the same way as the instance overlap.

$$1 - \frac{\#relations\_sen1\&2 - \#relations\_sen1}{\#relations\_sen2}$$

Example

All the relations in the three models are the same, because the objects in both sentences have the same relation to each other. This can be shown by looking in the model files, where all three model files have the same relations:

```
    f(2,card,[ (d1,d1)]),
    f(2,r1agent,[ (d1,d1)]),
    f(2,r1patient,[ (d1,d1)])]).
```

So all the variables have the value '3', and thus the result is 1.0 which tells us this is a perfect overlap.

$$1 - \frac{3 - 3}{3} = \mathbf{1.0}$$

4.6 Other features

4.6.1 WordNet overlap

For the concepts we use the hypernym relations ("is a" relation) as described in section 3.4. First the thesaurus with all the concepts of the first sentence is built, the program starts out with all the concepts of the sentence, and then recursively search for hypernyms. Until the top of the thesaurus is found(the top entity is called 'object'). After this, the same procedure is taken for the combination of all concepts of both sentences. The sizes of these two trees can be compared to see what the second sentence contributes over the first sentence. This is the formula to calculate the value of this feature:

$$1 - \frac{\#concepts\_both\_sentences - \#concepts\_sen1}{\#concepts\_sen1}$$

Example

Our example sentence pair has only one difference in the occurrences of concepts. This is the difference between 'bike' and 'motorcycle'. Because motorcycle is a hypernym of bike, the size of the tree for both sentences did not change (the concept motorcycle was already in the tree before we added the second sentence). Thus the number of concepts are equal, and the overlap is:

$$1 - \frac{27 - 27}{27}$$

4.6.2 Synset overlap

The WordNet synsets used here are described in section 3.4. For each word the first 10 cognitive synonyms are collected. All these words are then collected in a set per sentence. The end value of this feature will be the overlap between the two sets of words.

$$\frac{words\_sen1 \cap words\_sen2}{words\_sen1 \cup words\_sen2}$$

Example

The set of synonyms that can be extracted from WordNet for the first sentence of our example is:

```
['bike', 'beingness', 'being', 'organism', 'two', 'II', '2', '
    motorcycle', 'existence', 'deuce', 'ar', 'are']
```

The second sentence has a very large set for this feature. So the result of the intersection will be relatively small in comparison to the size of the union. And the resulting value of the formula will thus be very low. After inserting the sets, we get:

$$\frac{6}{33} = \mathbf{0.182}.$$

4.6.3 Synset distance

The synsets in wordnet do not only contain a list of synonyms for words. An even more interesting feature is that all the data together is stored as a thesaurus. A thesaurus is a large net that connects and groups words based on their meanings. So WordNet is basically a large web of synsets that are connected by links, these links represents is-a relations. The minimum number of connections that are needed to travel from one synset to another synset is used to calculate the path similarity. Because a score between 0 and 1 is useful for many calculations the path similarity also uses these restrictions, it divides 1 by this amount of connections that are needed to travel trough. The resulting formula is:

$$path_sim(1)\frac{1}{length(word_a \mapsto word_b)}$$

This path similarity is a good representation of the differences in meaning for two words. It is impossible to use the synset path similarity as a feature for comparing sentences though. So there are two steps that are still need to be taken: Firstly the words in the first sentence that occur in the WordNet synsets are collected. These are then coupled to the words of the second sentence with the highest path similarity. After this coupling is done we know what words most have a probable relation.

The scores gotten for the word couples still needs to be combined to one score for the feature used in the model. They can be combined by simply adding together all the path similarity scores. This will not be a fair comparison if we have a long sentence pair and a short sentence pair. Because a long sentence usually have more words found in the thesaurus and if these words all have a low path similarity they can still get a higher score together. So the length of the sentences will have to be taken into account, or even better the amount of words used for the total score (the amount of synsets used for one sentence). So the total score needs to be divided by the number of path similarities that are used to get to that score. If the number of synsets found per sentence is 'n', the formula will be:

$$\frac{path\_sim(1) + ... + path\_sim(n)}{n}$$

Example

First we will take a look at the maximum path similarities found for each word in the first sentence, they are shown in table 4.1.

Table 4.1: A list of the found maximum path similarities.

Word sentence 1	Word sentence 2	Path similarity
two	two	1.0
are	are	1.0
being	people	0.167777
ridden	riding	1.0

The summed value of the path similarities is 3.1667, this is based on 4 word pairs, so the value of this feature is:

$$\frac{3.167}{3} = \mathbf{0.792}$$

4.6.4 Compositional Distributional Semantic Model

For this feature the Skip-Gram model described in section 3.5 is used. Each word in this model has it's own vector with a length of 1600. So first all the vectors for the words in a sentence are collected. These vectors are then summed up using element-wise addition, so that we end up with 1 vector that has 1600 values that represent this sentence in respect to our skip-gram model. This is done for both sentences of a sentence pair. The result is then two different vectors that each represent a sentence. The cosine distance can be used to calculate the distance between the two vectors.

Example

Because of the huge amount of values used here per sentence pair, it is impossible to show the entire calculation in this paper. Instead of showing the entire calculation, below a part of the resulting vectors for both sentences are shown.

```
[ 0.194736 -0.256398  0.235763 .., 0.001883  0.247598  0.164759]
[-0.034507  0.019716  0.367023 ..,-0.083645 -0.006491  0.213231]
```

The cosine distance between these two vectors is **0.300**.

4.6.5 Relations between sentences

Boxer does not only provides models for us, it also tries to identify the relation within a sentence pair. It distinguishes between neutral, entailment and contradictory relations. This is also the other part of shared task 1 of SemEval 2014 (Marelli et al., 2014). Neutral relations are relations where both sentences have nothing to do with each other. Sentence 1 entails sentence 2 if stating that sentence 1 is true also means that sentence 2 is automatically true. While a contradictory relation states that both sentences can never be true at the same time.

The gold data of this task is available in our corpus. But since this data is not available in a real world situation, we will not use it. Instead the predictions of Boxer are used, since they are based on the raw sentences. But this is categorical data, which cannot be inputted directly into our regression model. So each category is treated as it is a separate binary feature, which has the value 1 if the sentence pair belongs to the relation-category and 0 if it does not.

Even though there is no direct mapping from sentence relation to sentence similarity, it is a safe assumption that entailed and contradictory sentences will have a higher similarity then sentences with a neutral relation. See for a closer look at the contributions of similarity scores among the different relations 3.2. So this will probably be a feature that has a valuable contribution to the final model. More information on how boxer tries to categorize the sentence pairs can be found in the Semeval paper of our team: Bjerva et al. (2014).

Example

Boxer tags this sentence pair as having an entailment relation. Even though this might seem reasonable, in the corpus this sentence pair is annotated as a neutral relation. Both relations can be explained by taking a closer look at the sentences. There is an ambiguity in meaning, and the human annotators have been favoring another meaning as the one that boxer chooses. Because the annotated data is not available in a real world situation it can not be used here either, and only our feature for the entailment relation will have the value **1.0**, the features for the other two relations will have the value **0.0**.

4.6.6 ID

Our corpus seemed to be a little biased when looking at the gold values. In the beginning of the corpus the gold similarity scores are higher then on the end of the corpus. To test this I have included two features only based on the id of the sentence pair. One feature is just the id of the sentence pair, for our example **2869**. But what if our corpus is just a little biased? Then it would make more sense to give an approximate value to the feature. This is why our second feature divides the id by 1000, and then rounds it downwards. So the value for our second ID feature is **2**.

Even though these features might contribute to a better similarity prediction, they should not be used in the final model. Because these id's are of course not available in a real world situation. All the other features that are explained in this chapter are, or can be made, available for real world sentences.

4.7 Paraphrases

To increase the predictive value of each feature it might be a good idea to not only look at the two sentences at stake. But instead also try to calculate each separate feature for semantically almost equivalent sentences. The first step for this is to generate sentences with (almost) the same semantic meaning. This can be done by replacing parts of the sentences by their paraphrases. See section 3.6 to see how these paraphrases were obtained.

The process to generate the new sentences is rather straightforward. For each word in sentence 1 the corresponding paraphrases are found. If the replacement text is present in sentence 2 a new sentence pair can be generated using this paraphrase. This last restriction is included because otherwise this method will result in an exploding amount of sentence pairs, while many of them would not retrieve higher scores per feature and thus will not be used at all. Since parsing all the different model/representations takes a long time, it is better to leave those paraphrases out.

Another way to generate semantically similar sentences is to remove negations from the sentences. After this is done, there will be less sentences with a contradiction relation. This means that it will be possible to generate a logical model for contradictory sentence pairs, which was impossible before. So the features that use the logical model will probably have the most benefit from this. To do this I first manually searched in the corpus for all the occurring negations. These negations can be removed by simply adding these words to our paraphrases database, and replace them with their positive equivalent. So the following paraphrases are added to our database:

Listing 4.3: Negation words and their replacements.

```
| not |||  |
| n't |||  |
| no ||| a |
| none ||| some |
| nobody ||| somebody |
```

To get the most out of all the new sentences, also different combinations of paraphrases should be considered. If one sentence pair has 2 different possible paraphrases, this results in 3 new sentence pairs. One sentence pair uses both paraphrases, but a sentence pair that only uses one paraphrase might be better. So for each paraphrase itself a new sentence pair is also generated.

This results in 10,515 new sentence pairs. These new sentences are generated from 4,977 sentence pairs of the original corpus, because the other sentence pair do not have appropriate paraphrases or negations.

After this process all the same tools that were used to process the original sentence pairs are used once again on all new sentence pairs. Now that all the new data is available, a new function is made for every feature we already had. In algorithm 1 is shown how the new feature values are calculated.

```
featureValue = getFeatureValue(senA, senB);
**forall** *sentenceCombination in alternativeSentences* **do**
    sentenceCombinationValue = getFeatureValue(sentenceCombination(senA),
    sentenceCombination(senB));
    **if** *sentenceCombinationValue >featureValue* **then**
        featureValue = sentenceCombinationValue;
    **end**
**end**
**return** featureValue
```

Algorithm 1: Psuedo code that shows how the paraphrases are used with all already existing features.

Using this code all different sentence pair combinations are treated equally. And the highest value that is found for each feature is the one that is used in the final model. So here two assumptions are made. The first assumption is that the meaning of the alternative sentences are (almost) equal to the meaning of the original sentences. If this is true, it is correct to use the values of one the alternative sentences instead of the original value.

The other assumption is that if a sentence pair finds a higher value, this means that it discovered an equality that was not recognized in the original sentence pair. So it found something that always was there, this is known because we already assumed that the meanings of the sentences are equal. If these two assumptions are true it is sensible to use the highest value of all sentence combinations. In the results chapter (chapter 5) it will become clear if these assumptions are in fact true.

Example

For our example sentence pair only one replacement is found. Listing 4.4 shows the differences between the original sentences and the new sentences created by paraphrasing.

Listing 4.4: On top the two original sentences, beneath them the sentences after paraphrasing.

```
Two bikes are being ridden by two people
Two people are riding a motorcycle

Two motorcycle are being ridden by two people
Two people are riding a motorcycle
```

The replacement made here is bikes $\mapsto$ motorcycle. The result of replacing bikes (plural) with motorcycle (singular) is an ungrammatical sentence, but the parsers used will try to parse the sentences anyway and they will not give useless results over a small error like this.

For our example sentence pair the score of several features increase after paraphrasing. All the original values and the new values of the features are shown in table 4.2.

Table 4.2: A table showing how the values of the different features change after the paraphrasing (bold means the value has changed).

Feature	Old score	New score
Sentence length	0.333	0.333
Word overlap	0.75	0.75
Weighted word overlap	0.901	0.901
Noun overlap	**0.333**	**1.0**
Verb overlap	0.5	0.5
Instance overlap	**0.96**	**1.0**
Predicate overlap	**0.96**	**1.0**
Relation overlap	1.0	1.0
Discourse condition overlap	0.667	0.667
Agent overlap	1.0	1.0
Patient overlap	0.0	1.0
WordNet overlap	1.0	1.0
Synset overlap	0.182	0.182
Synset distance	0.792	0.792
Contradiction	0.0	0.0
Neutral	0.0	0.0
Entailment	1.0	1.0
CDSM	0.301	0.301
ID	2869	2869
ID 1-10	2.0	2.0

So not many values change for this example, some of them do get higher though. Especially the noun overlap has an remarkable increasement, this is due to the fact that there are not many nouns in these sentences. Due to this the value of the original sentence pair was probably too low, and the new value comes closer to a trough meaning overlap between the nouns. Another thing that might be surprising is the fact that the word overlap does not change, even though there are more similar words in the sentence. But as described in section 4.2.2, first and last words are treated separately.

Chapter 5

Evaluation

Unlike some other regression models, the random forest regressor does not output a formula. The regressor uses decision trees, which are not possible to visualise here. So there is need for other methods to evaluate the different features. First I will build a model for each feature separately, and the results of these models can be evaluated. Then I group the features based on the data source they use, and also build models for these groups of features. This can expose how semantically informative the different data sources are. Besides this it is also possible to look at the relative importance of each feature in the complete model.

5.1 Features separate

First the model is trained on the different features separately to show how well the different features and models can predict the semantic similarity by themselves. This is thus not a direct mapping for the importance of a feature in the entire model, it tells a great deal about the the prediction power of a feature (group) though. The results are shown in table 5.1. For this evaluation 5,000 sentence pairs are used for training and 4,927 sentence pairs are used for evaluation. The P represents the Pearson correlation, and mse stands for the mean square error. There is also a distinction between the scores that uses paraphrases(+pp) and the scores for the same features without using paraphrases(-pp).

Most correlations get higher if the features use paraphrases, two features get slightly worse results though. These are both WordNet features, WordNet overlap and synset distance. This is probably because WordNet already knows about the relations of the words so these features do not need the paraphrases. The other WordNet feature (synset overlap) works much better with the paraphrasing though, this can be explained because this feature does not use the relations that are available in WordNet. The paraphrasing do seem to have a low impact on the results of the total model, but it must be taken into account that higher scores are very hard to improve. So an increase of 0.07 in correlation and a decrease of 0.012 for the mse is a great improvement if the results are already as high as they are here.

Another surprising feature is the id feature. This feature has no meaning though, it only uses the id given by the corpora. So for a real world application this number would not make sense, therefore we should focus on the combined model that does not use the id feature. The final results are then shown in the last row of the table, they will be discussed in the conclusion (chapter 6).

Table 5.1: The results of using the features separately/grouped with and without using the paraphrasing. (+pp & -pp)

Datasource	P(-pp)	P(+pp)	mse(-pp)	mse(+pp)
Raw Sentences	**0.541**	**0.598**	**0.746**	**0.666**
Word overlap	0.271	0.340	0.944	0.902
Sentence length	0.227	0.229	0.971	0.970
Weighted word overlap	0.501	0.504	0.795	0.793
POS	**0.647**	**0.676**	**0.592**	**0.553**
Noun overlap	0.574	0.624	0.682	0.621
Verb overlap	0.372	0.381	0.877	0.870
Logical model	**0.698**	**0.696**	**0.535**	**0.525**
Instance overlap	0.618	0.619	0.629	0.628
Predicate overlap	0.619	0.651	0.628	0.586
Relation overlap	0.359	0.361	0.886	0.885
DRS	**0.634**	**0.667**	**0.610**	**0.569**
Discourse condition overlap	0.619	0.620	0.628	0.628
Agent overlap	0.317	0.348	0.915	0.894
Patient overlap	0.124	0.124	1.002	1.002
Wordnet	**0.666**	**0.688**	**0.575**	**0.544**
WordNet overlap	0.652	0.651	0.590	0.591
Synset overlap	0.386	0.409	0.917	0.888
Synset distance	0.355	0.349	0.964	0.962
CDSM	**0.608**	**0.607**	**0.681**	**0.681**
Relation prediction	**0.621**	**0.620**	**0.626**	**0.623**
ID	**0.493**	**0.493**	**0.807**	**0.807**
Combined model (with ID)	0.837	0.842	0.305	0.297
Combined model(without ID)	**0.820**	**0.827**	**0.334**	**0.322**

Table 5.2: The order of the top 16 features for each evaluation method.

	Pearson correlation	Relative importance
1	WordNet overlap	CDSM
2	Predicate overlap	WordNet overlap
3	Noun overlap	Discourse condition overlap
4	Discourse condition overlap	Predicate overlap
5	Relation prediction	Noun overlap
6	Instance overlap	Weighted word overlap
7	CDSM	Instance overlap
8	Weighted word overlap	Neutral judgement
9	Synset overlap	Synset overlap
10	Verb overlap	Synset distance
11	Relation overlap	Sentence length
12	Synset distance	Word overlap
13	Agent overlap	Relation overlap
14	Word overlap	Contradiction judgement
15	Sentence length	Verb overlap
16	Patient overlap	Patient overlap

The strongest features are predicate overlap and WordNet overlap, these both use the is-a relation (predicate overlap uses it indirectly). Thus this shows that the is-a relation can come in very useful for extracting semantic meaning.

The data source that scores the highest correlation is the logical model. And the raw sentences yield the lowest correlation score. This results tells us that the interpretation of relatedness is a lot more then just the visible text read by a human. In addition it follows from these results that a logical model is a very effective way of representing semantic meaning.

5.2 Relative importance

Another way to evaluate our results is to use the built-in function in sklearn that estimates the importance of our features. This produces a graph that show the relative importance for each feature. The resulting score for each feature is based on the position of this feature on the decision trees that our regressor uses. So the importance score of a feature does not perfectly correlate with the prediction power the feature contributes to the entire model. For evaluating with this graph, it is obvious to use paraphrases since they give a better score. It is also convenient to leave out the id's because these are impossible to get for real world applications. See for the resulting graph figure 5.1.

If these results are compared with the previous method for evaluation in table 5.1 there are some differences in the order of features. For a comparison of both evaluations, see table 5.2.

Note: There are some different features in this list, since the three relation predictions are only scored for a grouped correlation while they are judged seperately by the relative importance graph. It is preferable to use them as group, but this is impossible for relative importance.

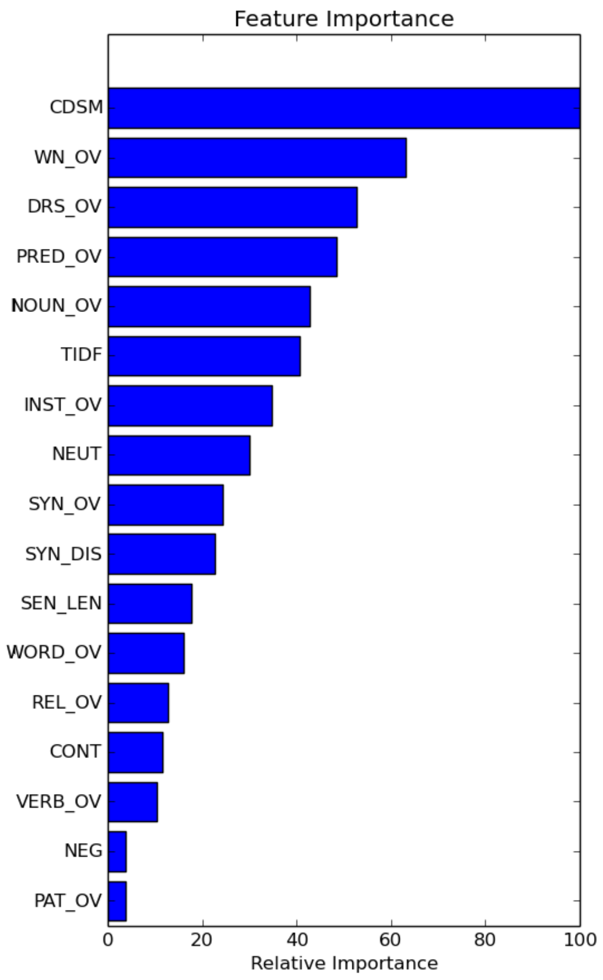


Figure 5.1: Graph that shows the relative importance of the features.

The main difference is found in the relation prediction because the result of those three features is grouped in the table, this is not possible for calculating the relative importance though. The other big difference is the CDSM feature, this feature is on the top of the decision tree in the regression model, and thus is on top of the relative importance list. When using this feature isolated in a regression model the score is not the highest at all.

5.3 Superfluous data sources

It is good to avoid superfluous features, because they only slow the program down while not improving the results. And there might even be features that make the final correlation of the regression model slightly worse. To be really sure if a data source used that does not contribute to the entire model, the total model can be compared to a model without the features of the data source. So I built seven other models, each excluding one feature group. This is done by group because the differences will be very minimal when only leaving out one feature.

Table 5.3: Pearson correlation when excluding one feature group.

Excluded feature group	Pearson correlation	mse
None	0.827	0.322
Relation prediction	0.826	0.323
WordNet	0.826	0.323
DRS	0.825	0.326
POS	0.821	0.332
CDSM	0.820	0.334
Logical model	0.819	0.335
Raw sentences	0.814	0.344

The results are ordered by pearson correlation and can be found in table 5.3. The differences are not very big, since there are always a lot of remaining features which also achieve a high score. This table shows us that Relation prediction, the WordNet features and the DRS features can be excluded from the model without much precision loss. But the combination of deleting these feature groups might gives us a more significant decrease. So I trained one more model on the remaining 4 feature groups, this resulted in a Pearson correlation of 0.819 and a mean square error of 0.334. For efficiency reasons these three features can thus be left out at the cost of a small decrease in correlation.

Another surprising result is that the raw sentences is the worst data source to miss. In our previous evaluation when testing the feature groups seperate (table 5.1) this feature group had the lowest score, but in the final model this feature group has a big contribution.

Chapter 6

Conclusion

The main question of this paper was: Which features can be automatically extracted from a natural language sentence that represents a good overview of the semantic meaning of a sentence? There are numerous features that work good for this task, but the best results can be achieved by combining these features. The best result that is possible to get with the features that are used in this paper is a correlation of 0.827 and a mean square error of 0.322. This is a very good result compared to other systems that do the same task (see [1] for a comparison).

When comparing the features separately we can say that the features that use WordNet data are overall the best features (this includes the logical model features). This is not surprising as WordNet is also a very popular data source for computing word similarity. The features that only use the raw sentences are very weak features. While this data is the only data that is directly visible for a human, it does not represent the semantic meaning very well.

There are some features that use the entire sentence, but some features use scores of word similarity and try to combine the word similarity of different words in a sentence pair to one score. The first category seems to work better because these features have more information about the semantic meaning of the word in context. This also reflects in the evaluation table (see table 5.1).

Paraphrasing is a usefull improvement to increase the precision of most of the features. Almost all the features profit from using the paraphrases, and the few features that get lower scores when trying alternative sentences are only getting a little bit worse.

In final analysis it is good to look at the usability of these results. The correlation score of 0.827 means that our regression model can account for 82,7 % of the similarity score as given by human annotators. So already a lot of the score is taken care of in our model, and the rest of the score is very hard to predict. This has become clear when the last features where added, the score did not increase much while the last features did have a good prediction power by themselves. It would be interesting to get the cross-annotator agreement to get an upper bound of how good humans are at predicting these values, but for this the individual scores are needed which are not publicly available.

The lowest mean square error to achieve with all the features is 0.322. This value means that on average our system is only 0.322 wrong in comparison to a human

[1]`http://alt.qcri.org/semeval2014/task1/index.php?id=results/`

comparison on the scale of 1-5. The results are very promising, and depending on the task, they can be already good enough to use. Many things still needs to be taken care of to use this system in a real world situation tough, the corpus has a very limited set of words and consists of simple sentences. So a lot of preprocessing still needs to be done, and encyclopedic information should be added for usage in a real world application. And the system also needs an annotated training corpus that is similar to the test data (or real world data), which depending on the use can be very hard to come by.

Chapter 7

Future work

This exact task is also performed by 16 other teams[1]. The highest score of our combined model using paraphrases is among the best three systems on this task with a very small difference. This suggests that this might almost be the upper bound of what can be achieved with the technologies available right now.

So even though little improvement can be made on this corpus, there is still a lot of research that still can be done. Starting with making the system usable for real natural language. The SICK corpus used for evaluation in this paper uses a simplified version of natural language. If this model is used for real world language use many pre-processing steps needs to be taken, and even after this the results will probably be worse.

An addition to improve this system could be to make a different regressor for each rte category (neutral, contradiction and entailment) that boxer predicts. As can be seen in figure 3.1 and figure 3.2, the different entailment categories all have a very different contribution. It would be interesting to see if the creation of different regression models based on smaller training sets but on more specific data will work better than the approach taken before.

Another very practical improvement to this approach could be to not use a supervised learning method anymore. Because of our limited evaluation set we could use a supervised learning method here. But it is very hard to get an annotated corpus for real language use. And even if such a corpus is created somehow the system will function worse because of the increased complexity compared to the SICK corpus. If the real world corpus has the same size as the corpus used here, it can never include all the words or language phenomena that exists. In the SICK corpus we could be pretty sure that most phenomena and words that were in the test set, were already trained on in the training set.

Kandola et al. (2002) already made a good start on a supervised learning model. But this method can probably be improved using the deep semantic features that are described in this paper, because the existing systems that use supervised learning are only based on word corpora. With all the different models and representations used in this paper much more and different internal semantic information is available.

[1]`http://alt.qcri.org/semeval2014/task1/index.php?id=results/`

Appendix A

Commands for reproducing

```
All commands that are needed to reproduce the results for Ubuntu
    13.10 64bit. Paradox and vampire needs to be compiled, this is not
     included in these steps. 8gb ram is required and a fast cpu and a
     ssd are recommended.

sudo apt-get install subversion
svn co http://svn.ask.it.usyd.edu.au/candc/trunk candc
first enter wrong password, then authenticate
cd candc
wget http://svn.ask.it.usyd.edu.au/download/candc/models-1.02.tgz --
    http-user=[user] --http-password=[password]
tar -xvzf models-1.02.tgz
ln -s Makefile.unix Makefile
sudo apt-get install swi-prolog
make bin/nc
make bin/boxer
make bin/tokkie
make ext/bin/bliksem
make ext/bin/mace
cd ext/otter-3.3f
make all
cd ../
wget http://downloads.sourceforge.net/project/gsoap2/gSOAP/gsoap_2
    .8.17.zip?r=http%3A%2F%2Fsourceforge.net%2Fprojects%2Fgsoap2%2
    Ffiles%2FgSOAP%2F&ts=1394902772&use_mirror=garr -O gsoap_2.8.17.
    zip
unzip gsoap_2.8.17.zip
cd gsoap-2.8/
./configure --prefix=/home/rob/candc/ext/
sudo apt-get install byacc
sudo apt-get install flex
make
make install
cp gsoap/bin/linux386/soapcpp2 ../bin
cd ../../
make soap
cp ext/otter-3.3f/bin/mace2 ext/bin/mace
cp ext/otter-3.3f/bin/otter ext/bin/otter

----paradox
cp ~/Downloads/paradox3 ext/bin/paradox
http://packages.ubuntu.com/quantal/libgmp3c2
wget http://nl.archive.ubuntu.com/ubuntu/pool/universe/g/gmp4/
```

```
    libgmp3c2_4.3.2+dfsg-2ubuntu1_amd64.deb
sudo dpkg -i libgmp3c2_4.3.2+dfsg-2ubuntu1_amd64.deb
----

----vampire
cp ~/Downloads/vampire ext/bin/vampire
----

chmod 777 ext/bin/*
make -p working
cd working
wget http://alt.qcri.org/semeval2014/task1/data/uploads/sick_trial.
    zip http://alt.qcri.org/semeval2014/task1/data/uploads/sick_train.
    zip http://alt.qcri.org/semeval2014/task1/data/uploads/
    sick_test_annotated.zip http://alt.qcri.org/semeval2014/task1/data
    /uploads/sick_test.zip
unzip '*.zip'
cp SICK_train.txt SICK_all.txt
tail -500 SICK_trial.txt >> SICK_all.txt

----_prepareSICK
change value of infile to sick_all.txt
----
src/scripts/boxer/sick/_prepareSICK
src/scripts/boxer/sick/_preparePAR
src/scripts/boxer/sick/_runSICK
src/scripts/boxer/sick/_evalSICK
cd ../
sudo apt-get install git
git clone git@github.com:bjerva/semeval-relatedness.git
cd semeval-relatedness
sudo apt-get install python-nltk
sudo apt-get install python-requests
sudo apt-get install python-numpy
sudo apt-get install python-scipy
git clone git@github.com:scikit-learn/scikit-learn.git
cd scikit-learn
python setup.py build
sudo python setup.py install
make
----Add to ~/.bashrc
export PYTHONPATH="/home/rob/semeval-relatedness/scikit-learn";
----
sudo apt-get install r-base
python src/_prepareSICK2.py
python src/_runSICK2
python src/semeval_task1.py
```

Bibliography

Agirre, E., M. Diab, D. Cer, and A. Gonzalez-Agirre (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pp. 385–393. Association for Computational Linguistics.

Bird, S. (2006). Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pp. 69–72. Association for Computational Linguistics.

Bjerva, J., J. Bos, R. van der Goot, and M. Nissim (2014). The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *SemEval 2014*. To appear.

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pp. 277–286. Association for Computational Linguistics.

Budanitsky, A. (1999). Lexical semantic relatedness and its application in natural language processing.

Budanitsky, A. and G. Hirst (2006). Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics 32*(1), 13–47.

Claessen, K. and N. Sörensson (2003). New techniques that improve mace-style finite model finding. In *Proceedings of the CADE-19 Workshop: Model Computation-Principles, Algorithms, Applications*, pp. 11–27.

Curran, J., S. Clark, and J. Bos (2007, June). Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic, pp. 33–36. Association for Computational Linguistics.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Finkelstein, L., E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin (2001). Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pp. 406–414. ACM.

Flesch, R. (1948). A new readability yardstick. *Journal of applied psychology 32*(3), 221.

Ganitkevitch, J., B. Van Durme, and C. Callison-Burch (2013). Ppdb: The paraphrase database. In *Proceedings of NAACL-HLT*, pp. 758–764.

Hirst, G. and D. St-Onge (1998). Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database 305*, 305–332.

Islam, A. and D. Inkpen (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 2*(2), 10.

Jiang, J. J. and D. W. Conrath (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

Kamp, H., J. Van Genabith, and U. Reyle (2011). Discourse representation theory. In *Handbook of philosophical logic*, pp. 125–394. Springer.

Kandola, J., N. Cristianini, and J. S. Shawe-taylor (2002). Learning semantic similarity. In *Advances in neural information processing systems*, pp. 657–664.

Lund, K. and C. Burgess (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers 28*(2), 203–208.

Marelli, M., L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli (2014). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Proceedings of SemEval 2014: International Workshop on Semantic Evaluation.*.

Marelli, M., S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli (2014). A sick cure for the evaluation of compositional distributional semantic models. *Proceedings of LREC 2014*.

McCune, W. (2001). Mace 2.0 reference manual and guide. *arXiv preprint cs/0106042*.

Mikolov, T. (2013). word2vec - tool for computing continuous distributed representations of words. - google project hosting,. `https://code.google.com/p/word2vec/`.

Miller, G. A. and W. G. Charles (1991). Contextual correlates of semantic similarity. *Language and cognitive processes 6*(1), 1–28.

Parker, R., L. D. Consortium, et al. (2009). *English gigaword fourth edition.* Linguistic Data Consortium.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Rashtchian, C., P. Young, M. Hodosh, and J. Hockenmaier (2010). Collecting image annotations using amazon's mechanical turk. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Saeed, J. (2011). *Semantics.* Introducing Linguistics. Wiley.

Santorini, B. (1990). Part-of-speech tagging guidelines for the penn treebank project (3rd revision).

Steedman, M. (2000). *The Syntactic Process*. A Bradford book. MIT Press.

Strube, M. and S. P. Ponzetto (2006). Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, Volume 6, pp. 1419–1424.

Witten, I. and D. Milne (2008). An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pp. 25–30.