



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

Master Thesis

Comparative Argument Mining

Mirco Franzek

5franzek@informatik.uni-hamburg.de

Studiengang Informatik

Matr.-Nr. 6781911

Erstgutachter: Prof. Dr. Chris Biemann

Zweitgutachter: Dr. Alexander Panchenko

Abgabe: April 2017

I am C3P0, protocol droid, human-cyborg relations.
I am fluent in over 6 million forms of communication. – *C3PO*

Contents

1	Introduction	1
1.1	Motivation: An Open-Domain Comparative Argumentative Machine (CAM)	1
1.2	Related Work	1
1.2.1	Argumentation Theory	1
1.2.2	Argument Mining	1
1.2.3	Domain-Specific Comparative Systems	3
2	Building a data set for Comparative Argument Mining	7
2.1	Common Crawl Text Corpus	7
2.2	Prestudies	8
2.2.1	Sentence Selection	8
2.2.2	Prestudy A	9
2.2.3	Prestudy B	11
2.2.4	Validation of results	13
2.3	Main Study	16
2.3.1	Design changes	16
2.3.2	Sentence Selection	16
2.3.3	Brands	19
2.3.4	Computer Science	20
2.3.5	Random	21
2.3.6	Validation of results	21
3	Classification of Comparative Sentences	24
3.1	Experiments	24
3.2	Choice of Algorithms	24
3.3	Features	25
3.3.1	N-Gram Models	26
3.3.2	Part of Speech	26
3.3.3	Mean Word Embeddings	26
3.3.4	Sentence Embeddings	26
3.3.5	Dependency Features	26
3.3.6	Other Features	26
3.4	Classification with three classes	26
3.4.1	Baseline	26

3.4.2	Results	27
3.5	Binary classification	27
3.5.1	Baseline	27
3.5.2	Results	28
3.6	Final results	28
3.7	Discussion	28
4	Conclusion	29
	Bibliography	31
	Eidesstattliche Versicherung	33

1 Introduction

1.1 Motivation: An Open-Domain Comparative Argumentative Machine (CAM)

1.2 Related Work

1.2.1 Argumentation Theory

[Habernal et al., 2014] presented a comparison between the results of two different annotation studies. One used the Claim/Premise-Model, while the other one used the Toulmin model. They emphasized that there is no "one-size-fits-all" model. (ERWEITERN)
(BEGRÜNDEN: warum kein modell genutzt)

1.2.2 Argument Mining

[Lippi and Torroni, 2016] gave a summary of the research topic "Argument Mining" in general. They introduced five dimensions to describe Argument Mining problems: granularity of input, the genre of input, argument model, the granularity of target and goal of analysis. Furthermore, the typical steps of Argument Mining Systems are defined. First, the input must be divided into argumentative (e.g. claim and premise) and non-argumentative parts. This step is described as a classification problem. Second, the boundaries of the argumentative units are identified; this is understood as a segmentation problem. Third, the relations between argumentative units are identified. For instance, claims and premises are connected with a "support" relation.

A system which is capable of recognising comparative sentences and their components such as the compared entities, the property on which the entities are compared to and the direction of comparison was presented in [Fiszman et al., 2007]. The evaluation showed that the outcome has a high quality (SCORE?). However, the presented system is specific to the domain of studies to drug therapy. The system uses patterns generated from sentences (WHICH SENTENCES), as well as domain knowledge. Therefore, the methods cannot be transferred for the problem of this thesis.

[Park and Blake, 2012a] presented a domain-specific approach on argumentative sentence detection. The problem is formulated as a binary classification task (a sentence is either comparative or not). As in [Fiszman et al., 2007], the features are tailored for medical publications. Lexical features capture the presence of specific words, many of them

bound to the medical domain. The analysis of 274 sentences resulted in syntactic features. (BEISPIEL) Similar to [Fizman et al., 2007], the features cannot be directly transferred to other domains.

A recent publication on Comparative Argument Mining is [Gupta et al., 2017], where a set of rules for the identification of comparative sentences (and the compared entities) is derived from *Syntactic Parse Trees*. With those rules, the authors achieved a F1 score of 0.87 for the identification of comparative sentences. The rules were obtained from 50 abstracts of biomedical papers. Such being the case, they are domain dependent.

Because this thesis deals with user-generated content from the web, publications dealing with similar data are of interest.

The challenges occurring while processing texts from social media are described in [Šnajder, 2017]. In this publication, social media is broadly defined as “less controlled communication environments [...]”. Besides the noisiness of text, missing argument structures and poorly formulated claims are mentioned. It is expected that the text used in this thesis will have the same shortcomings. Additionally, [Šnajder, 2017] emphasized that analyzing social media texts can delivery reasons behind opinions.

In addition to the challenges mentioned above, [Dusmanu et al., 2017] also points to the specialized jargon in user-generated content like hashtags and emoticons. With this in mind, [Dusmanu et al., 2017] classified tweets about the “Brexit” and “Grexit” either as argumentative or as non-argumentative. Besides features used in other mentioned papers, new features covering hashtags and sentiment are added. They achieved a F1 score of 0.78 (using Logistic Regression) for the classification. It must to be said that the data set is small (SIZE) and the domain is rather specific.

Many publications on argument mining are dealing with a classification problem of some kind. Publications dealing with the identification of argument structures are of relevance for this thesis, as they provide valuable insights on the suitability of features and algorithms.

[Aker et al., 2017] summarized and compared features used in other publications for identification of argumentative sentences. In addition, a Convolutional Neural Network (as described in [Kim, 2014]) was tested. Two existing corpora and six different classification algorithms were used. As a result, structural features are most expressive; Random Forest is the best classifier.

[Stab and Gurevych, 2014] described a two-step procedure to identify components of arguments (such as claim and premise) and their relationships (like “premise A supports claim B”). The identification step is formulated as a multi-class classification. For the identification of argumentative components, a F1 score of 0.72 is reported.

How different datasets represent the argumentative unit of a claim is analysed in [Daxenberger et al., 2017]. After an analysis of the datasets and their annotation scheme, [Dax-

enberger et al., 2017] conducted two experiments. In the first one, each learner (Logistic Regression, Convolutional Neural Networks and LSTM) was trained and evaluated (10-fold cross-validation) on each dataset one after another. On average, the macro F1 score for identifying claims was 0.67 (all results ranging from 0.60 to 0.80). No significant difference between the results of Logistic Regression and the neural models was found. In isolation, lexical, structural and word embeddings were the best features. Structural features turned out to be the weakest. The second experiment was conducted in a cross-domain fashion. For each pair of datasets, one was used as the training set and the other one as the test set. The average macro F1 score was 0.54. In this scenario, the best feature combination outperformed all neural models. However, it is assumed that there might not be enough training data for the neural models. As the last point, [Daxenberger et al., 2017] noted that all claims share at least some lexical clues.

The role of discourse markers in the identification of claims and premises are discussed in [Eckle-Kohler et al., 2015]. A discourse marker is a word or a phrase which connects discourse units (citation). For instance, the word “as” can show a relation between claim and premise: “As the students get frustrated, their performance generally does not improve”. A similar function for words like “better”, “worse” or “because” is expected in this thesis. [Eckle-Kohler et al., 2015] showed that discourse markers are good at discriminating claim and premises. If claim and premise are merged into one class “argumentative”, this can be used to identify argumentative sentences. The F1 score is not presented, but the accuracy is between 64.53 and 72.79 percent.

A summary of several features for the identification of argumentative sentences can be found in chapter ??.

1.2.3 Domain-Specific Comparative Systems

The enormous amount of Comparison Portals shows the need for comparisons. Frequently aired television spots empathize the popularity of those portals.

Most of those portals are specific to a few domains and a subset of properties, for example, car insurances and their price. Because of that, those systems have some restrictions. Comparisons are only possible between objects of the domains and predefined properties. Source of the data is usually databases. Humans are involved in gathering, entering and processing the data.

Comparison Portals solely compare and deliver facts. Because of that, they can only give the advice to choose X over Y based on the facts collected. An insurance X might be the best in the comparison (e.g., best price), while the internet is full of complaints about lousy service.

Examples of classical Comparative Portals are *Check24*, *Verivox*, *Idealo*, *GoCompare*, and *Compare*¹, just to name a few.

¹<https://check24.de>, <https://verivox.de>, <https://idealo.de>, <https://gocompare.com>,

As an example, Check24 can compare a wide variety of different objects like several insurances, credit cards, energy providers, internet providers, flights, hotels and car tires. After the user entered some details (based on the object type, see figure 1.2.3), Check24 shows a ranking of different service providers. The user can choose different properties to re-rank the list. For instance, to compare different DSL providers, the user has to enter her address, how fast the internet should be and if she wants telephone and television as well. She can then select price, speed, and grade (rating) to sort the resulting list.

Figure 1.2.1: Check24 DSL Provider

The other mentioned sites work similarly. They provide more of a ranking than a comparison.

Another interesting type of websites are Question Answering Portals like *Quora* or *GuteFrage*². Although comparisons are not their primary goal, a lot of comparative questions are present on those sites. On Quora, more than 2.380.000 questions have the phrase “better than” in their title. If *Ruby* and *Python* are added, 10.100 questions remain.³ Same is true for the German site *GuteFrage*, though, the numbers are smaller than on Quora.⁴

More interestingly are systems which can compare any objects on arbitrary properties. Two examples are *Diffen* and *Versus*⁵.

Versus aggregates freely available data sources like Wikipedia and official statistic reports. For example, the comparison of “Hamburg vs. Berlin” uses Wikipedia for the number of universities, worldstadiums.com for the availability of sport facilities and the

²<https://compare.com> - all last checked: 12.12.2017

³<https://quora.com>, <https://gutefrage.net> - all last checked: 12.12.2017

⁴Checked via Google on 11th of December. Search phrase: “better than” site:quora.com and ruby python “better than” site:quora.com

⁵334.000 for “besser als” site:gutefrage.net and 78 for ruby python “Besser als” site:gutefrage.net

⁶<https://diffen.com>, <https://versus.com> - all last checked: 12.12.2017

Economist for the Big Mac Index. Presumably, some human processing is involved as the possible comparisons are limited. For instance, a comparison of Hamburg and Darmstadt is not possible as Darmstadt is not available on Versus. Likewise, “Ruby vs. Python” is not possible, Versus suggests to compare “Rome vs. Pyongyang” instead. Although Versus shows how many users “liked” the objects, it does not give a clear statement which one is better. For instance, it is not possible to check automatically whether Hamburg or Berlin is better for a short city trip. The user must search manually all valid properties like the number of museums, theaters, the price of public transport tickets and so on.

Similar to Versus, Diffen aggregates different data sources (see figure 1.2.3). All in all, the aggregated information is similar to Versus. The comparison is also tabular. Besides the automatically aggregated data, users can add information on their own. Diffen describes itself as “inspired by Wikipedia”⁶. Diffen does not enforce any restrictions on the objects of comparison, but it faces the same problem as Versus as objects are missing. A comparison between Darmstadt and Hamburg is likewise not possible: all cells for Darmstadt in the table are just empty.

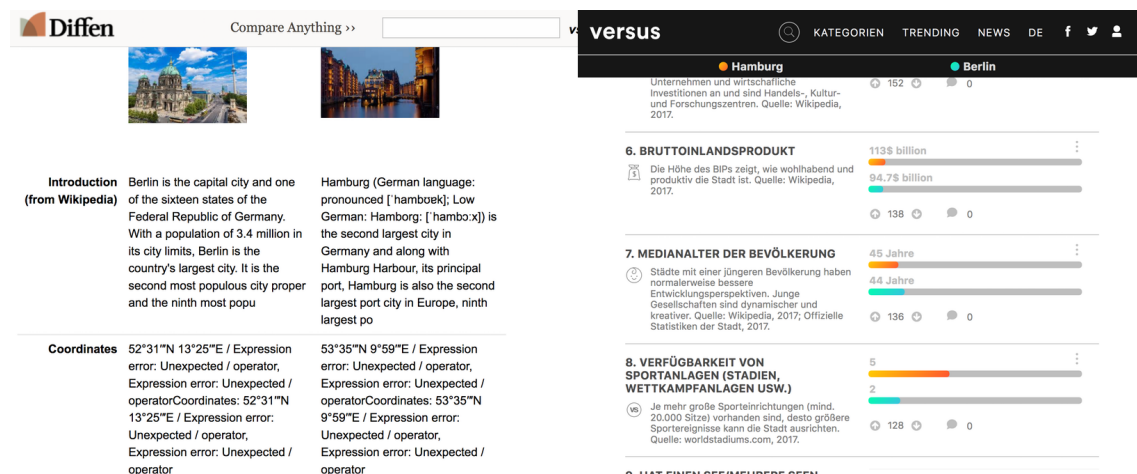


Figure 1.2.2: “Hamburg vs. Berlin” on Diffen and Versus

Neither Versus nor Diffen provides a comprehensible reason why an object is better than another one. They merely aggregate facts and bring them face to face. Despite the aggregation approach of both systems, many meaningful comparisons are not possible or not helpful (like “Hamburg vs. Darmstadt”, “Java vs. C#”, “Dr Pepper vs. Orange Juice”). Also, the user can not define the properties for the comparison. The sites provide every information available for the objects. For instance, Versus shows 42 properties for “Hamburg vs. Berlin” and only 35 for “Hamburg vs. Munich”.

To summarize, a lot of different comparison portals exist and are widely used. Espe-

⁶<https://www.diffen.com/difference/Diffen>About> - Last checked: 11.12.2017

cially the domain-specific portals do a good job, but inflexibility dearly buys the performance. First, the portals can only compare objects on predefined properties. Second, the data acquisition is not fully automatic. Domain-unspecific systems are good at aggregating information but do not provide a reasonable explanation to prefer X over Y.

Adding information like comments and product reviews can enrich the comparison with reasons and opinions, such as “Ruby is easier to learn than C” or “Python is more suitable for scientific applications than Erlang as many libraries exist”.

2 Building a data set for Comparative Argument Mining

Due to the novelty of Argument Mining (and especially Comparative Argument Mining), the supply of data sets is small. Thus, a new data set had to be created.

This data set was designed to answer two questions: if a given sentence compares two known objects, and if it does, if the first-mentioned object is better or worse than the second one. Those questions will be translated to several classification tasks in the later chapters.

The data set was created using the crowdsourcing platform CrowdFlower¹. As described in detail in the following chapters, the annotators were asked to assign one of four (and later three) classes to a sentence in which the objects of interest are highlighted.

The final data set contains 7421 sentences, each containing one of 273 object pairs. Each sentence was at least annotated by three different annotators.

2.1 Common Crawl Text Corpus

The sentences for the crowdsourcing task were obtained from a CommonCrawl² data set. CommonCrawl is a non-profit organisation which crawls the web and releases the crawled data for free use.

The data³ used in this thesis was already preprocessed (see [Panchenko et al., 2017]). First, it contains only English text. Duplicates and near-duplicates were removed, as well as all HTML tags. The texts were then split into sentences.

The preprocessed sentences were used to obtain the sentences for the crowdsourcing task. To make them manageable, an Elasticsearch index (from now on called "the index") was created. The index contains 3,288,963,864 unique sentences.

To get an idea if there are enough comparative sentences in the index, it was queried for all sentences containing one of the words "*better*" or "*worse*", as those words often indicate a comparison. This query returns 32,946,247 matching sentences. Querying for "*is better than*" still returns 428,932 sentences.

Those numbers show that there are enough sentences in the index to create a dataset for the given task. Even if only 1% of the sentences containing "*is better than*" are truly

¹<https://www.crowdfunder.com> (23.02.2018)

²<https://commoncrawl.org> (23.02.2018)

³Download Link

comparative, there would be 4289 training examples for the machine learning algorithm using this query.

2.2 Prestudies

Before the main crowdsourcing task could start, several questions had to be answered:

1. How to extract sentences from the index? (How should the query look like?)
2. How to preprocess those sentences? (How to highlight the objects of interest?)
3. Which classes should be assigned to the sentences?
4. How to phrase the guidelines?

Two prestudies were conducted to answer those questions.

2.2.1 Sentence Selection

The sentences for the crowdsourcing task should have a high probability of being comparative so that enough positive examples for the machine learning part are present. To ensure this, a list of cue words which indicate comparison was compiled by hand. For the prestudy, those words were “*better*”, “*worse*”, “*inferior*”, “*superior*”, and “*because*”. Comparable objects are needed as well. A list of object pairs was selected by hand (see table 2.2.1). The pairs were selected in a way that they span a wide range of different domains, such as programming languages, countries and pets. The idea behind this is that pets are compared differently than programming languages. In this way, there will be different comparison patterns in the data.

Table 2.2.1: Objects of the Annotation Prestudy

First Object	Second Object	# Sentences
Ruby	Python	100
BMW	Mercedes	100
USA	Europe	100
Beef	Chicken	100
Android	iPhone	100
Cat	Dog	100
Football	Baseball	100
Wine	Beer	100
Car	Bicycle	100
Summer	Winter	100

However, not all comparisons will contain one of the cue words mentioned above. Two different queries were used to overcome the coverage problem. Sevenhundred-fifty sentences were obtained using query 2.1 (seventy-five for each pair) and 250 using query 2.2

(twenty-five for each pair). The second query will also match not-anticipated sentences such as “*I like X more than Y since Z.*”.

Listing 2.1: Prestudy Sentence Selection Query A

```

1 {
2   "query":{
3     "bool":{
4       "must":[
5         {
6           "query_string":{
7             "default_field":"text",
8             "query":"(better OR worse OR superior OR inferior OR
           ↪ because) AND \"<OBJECT_A>\" AND \"<OBJECT_B>\" "
9         }
10      ]
11    }
12  }
13 }
14 }
```

Listing 2.2: Prestudy Sentence Selection Query B (shortened)

```

1 [...]
2   "query_string":{
3     "default_field":"text",
4     "query":" \"<OBJECT_A>\" AND \"<OBJECT_B>\" "
5   [...]

```

Table 2.2.2 shows some sentences obtained with this method. The objects of interest are printed in italics.

Table 2.2.2: Extracted Sentences

Sentence	Cue Words Used
He’s the best pet that you can get, Better than a <i>dog</i> or <i>cat</i> .	Yes
<i>Android</i> phones have better processing power than <i>iPhone</i>	Yes
10 Things <i>Android</i> Does Better Than <i>iPhone</i> OS	Yes
<i>Dog</i> scared of <i>cat</i>	No
In fact, many ‘supercars’ will use <i>BMW</i> or <i>Mercedes</i> engines.	No

2.2.2 Prestudy A

The first prestudy had two goals. First, it should assess if the sentence selection method returns enough comparative sentences. Second, the design of the study as described below should be checked. On that account, a crowdsourcing task with one hundred of the 1000 sentences was started.

For each sentence, the annotators should decide to which class a sentence belongs. The classes are described in table 2.2.3. The classes `BETTER`, `WORSE` and `NO_COMP` directly refer to the questions stated at the beginning of chapter 2. The class `UNCLEAR` was added to capture all sentences which are in a way comparative but do not fit into the classes `BETTER` or `WORSE`.

This is potentially useful for OBJECT_A, PHP, JS and OBJECT_B.

Please select how OBJECT_A and OBJECT_B compare in the sentence above? (required)

☐ OBJECT_A is BETTER than OBJECT_B
☐ OBJECT_A is WORSE than OBJECT_B
☐ Comparison of OBJECT_A and OBJECT_B is UNCLEAR
☐ NO COMPARISON of OBJECT_A and OBJECT_B

Figure 2.2.1: Annotator view (Prestudy A)

Table 2.2.3: Classes for Prestudy A and B

Class	Description
<code>BETTER</code>	The first object in the sentence (<code>OBJECT_A</code>) is better than the second one (<code>OBJECT_B</code>)
<code>WORSE</code>	The first object is worse
<code>UNCLEAR</code>	Neither <code>BETTER</code> nor <code>WORSE</code> fits, but the sentence is comparative
<code>NO_COMP</code>	The sentence is not comparative or the sentence is a question

In each sentence, the first object of interest was replaced with `OBJECT_A`, while the second one was replaced with `OBJECT_B`. Table 2.2.5 shows examples of processed sentences. The idea behind this was to enable the annotators to quickly see which objects should be taken into account for assigning a class. For example, in sentence three of table 2.2.5, the annotator might be confused which of the objects are of interest, yet the replacement makes it clear that he should ignore “C” and “VB”. The view of the annotator (for a single sentence) is shown in figure 2.2.2.

Each annotator saw five sentences to annotate per page. He was also able to look into the annotation guidelines anytime he wanted. To filter out low-quality annotators, twelve sentences were selected as test questions. Each participant took a quiz (eight test questions) before the actual annotation process. One of the five sentences on the page was a test question as well. The annotator had to keep an accuracy of 70% on the test questions, otherwise he was removed from the task.

Figure 2.2.2 shows the class distribution of the annotation results. The numbers after the class names show the absolute members of that class. As 45% of the sentences are comparative, the selection procedure works satisfying.

The agreement of the annotators was acceptable (see table 2.2.4).

Some uncertain sentences are shown in table 2.2.5, which displays the sentence and the decision of each annotator. As one can see in sentence two to five, annotators frequently

Figure 2.2.2: Class Distribution (Prestudy A)

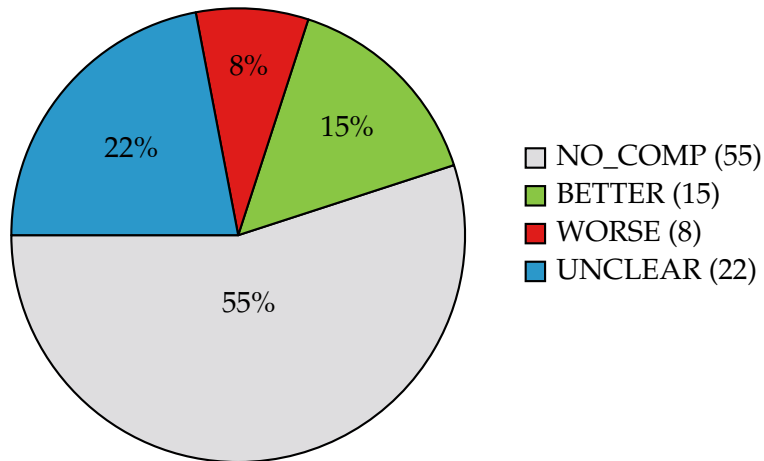


Table 2.2.4: Agreement (Prestudy A)

Distinct classes	Sentences	Percentage
1	37	37.00%
2	58	58.00%
3	5	5.00%

were not able to distinguish between `NO_COMP` and `UNCLEAR`. This is true in fifty-one of the fifty-eight cases (87%) where two different classes were assigned.

Fourteen out of fifty-five participants took part in an exit survey to rate the task. The overall satisfaction was rated with 3.2 out of 5. While the instructions (4.5), difficulty (4.4) and payment (3.8) got acceptable to good ratings, the test questions (2.9) were criticized. Also, 32 potential annotators failed the quiz. A second prestudy was conducted to address the discovered problems.

2.2.3 Prestudy B

Two-hundred sentences were annotated in the second prestudy. To address the shortcomings mentioned above, the task design was changed in several aspects.

However, some aspects were identical to the first study. As the sentence selection process worked fine, the same 1000 base sentences were used in the second prestudy. Each sentence was annotated by three annotators. The annotators saw five sentences per batch, one being a test question. They had to pass a quiz of eight test questions and had to keep an accuracy of 70% on the test questions during the annotation procedure. The classes were the same as in the first prestudy (see table 2.2.3).

As the pair “*Ruby vs. Python*” requires knowledge in computer science, this need was expressed in the title of the task. To address the problem with the confusion between `UNCLEAR` and `NO_COMP`, the wording on these classes in the annotator’s view was

Table 2.2.5: Uncertain Sentences (Prestudy A)

#	Sentence	Ann. 1	Ann. 2	Ann. 3
1	The only reason OBJECT_A is used over OBJECT_B, is because of libraries...	WORSE	UNCLEAR	NO_COMP
2	Agile development is the most popular model at the moment because of architectures like OBJECT_A on Rails and Django (for OBJECT_B)	NO_COMP	NO_COMP	UNCLEAR
3	Your C and VB devs can suddenly easily write web apps and your OBJECT_A and OBJECT_B devs can too - with the added bonus of much better performance	NO_COMP	NO_COMP	UNCLEAR
4	I'm a huge OBJECT_A/Django & OBJECT_B/Rails fan, but I will never stop using PHP because it is so broadly accepted and supported	NO_COMP	UNCLEAR	UNCLEAR
5	It's why I mention OBJECT_A and OBJECT_B, because they've at least heard of them.	NO_COMP	NO_COMP	UNCLEAR

changed. The new view is displayed in figure 2.2.3.

In the first prestudy, some annotators complained that the test questions were not fair. In fact, they contained some special cases in a way that they did not represent the whole data set appropriately. In the second prestudy, more test questions (fifty-one instead of twelve) test questions were used. Those test questions covered easy and harder cases. Explanations for the harder test questions were added. The annotator saw those explanations after he failed the test question.

The sentence preprocessing was altered as well. Instead of replacing the object, `:[OBJECT_A]` or `:[OBJECT_B]` was appended. The colon and square brackets emphasize where the object of interest ends and the suffix begins. The idea behind this was that the removal of the objects also removed some context from the sentences, which might be useful to classify them correctly. In addition, the objects were shown in a different colour than the rest of the text.

The class distribution for the 200 sentences is presented in figure 2.2.4.

As in the first prestudy, nearly half of the sentences are comparative. The agreement (see table ??) was better than in the first prestudy. The confusion between `UNCLEAR` and `NO_COMP` is still the main problem for the sentences where only two annotators could agree. However, in the second prestudy this confusion only makes up forty-five out of the seventy-two cases (62.5% instead of 87.9% in the first prestudy). Compared to the first prestudy, the amount of sentences where all annotators agreed on one class increased from 37% to 62.5%. Table 2.2.7 shows some of the uncertain sentences.

python:[OBJECT_A], and ruby:[OBJECT_B], too) undoubtedly influenced people's votes when it

What describes the comparison in the sentence above best? (required)

☐ OBJECT_A is better than OBJECT_B (BETTER)

☐ OBJECT_A is worse than OBJECT_B (WORSE)

☐ There is a comparison, but not between OBJECT_A and OBJECT_B (UNCLEAR)

☐ There is NO comparison. (NO_COMP)


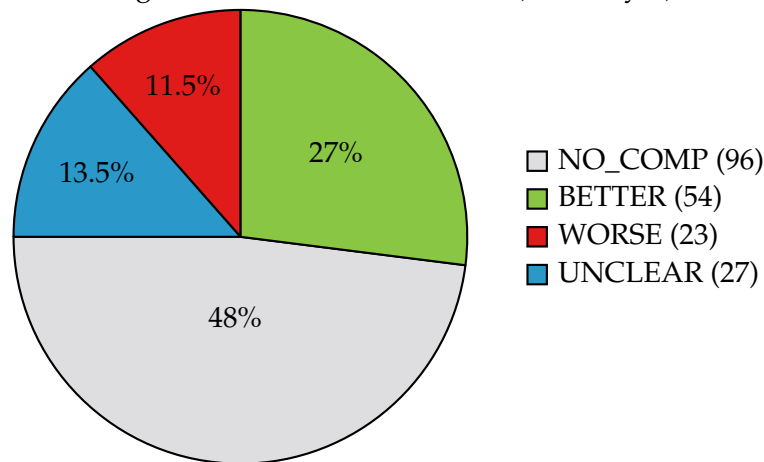
 See the instructions for some examples.

Figure 2.2.3: Annotator view (Prestudy B)

Figure 2.2.4: Class Distribution (Prestudy B)



From 125 candidate annotators, thirty-five failed the initial quiz. Twelve annotators were removed during the annotation process as they answered too many test questions wrong.

Twenty-two annotators took the exit survey. The overall satisfaction increased to 3.7 out of 5. The test question fairness was now rated with 3.7 instead of 2.9. The rating for the payment slightly increased to 3.9, yet the payment was not changed. However, the rating for the instructions decreased to 3.9 and for the difficulty to 3.5. The change in numbers is explained by the increased amount of sentences, which introduce new cases which are not directly reflected in the annotation guidelines.

2.2.4 Validation of results

Only a small fraction of the annotators took the exit surveys in both prestudies which reduces their explanatory power. However, it gave valuable hints to improve the study design. Yet, the agreement of the annotators is the more important signal.

Using two comparable objects and cue words to query the index returns a satisfying amount of comparative sentences. Sentence seven (table 2.2.7) shows that it is also beneficial to query the index without cue words. The changes in the second prestudy were well received by the annotators. In the end, they improved the quality of results as there

Table 2.2.6: Agreement (Prestudy B)

Distinct classes	Sentences	Percentage
1	125	62.50%
2	71	35.50%
3	4	2.00%

Table 2.2.7: Uncertain Sentences (Prestudy B)

#	Sentence	Ann. 1	Ann. 2	Ann. 3
6	Google shouldn't have mandated an inferior map app on the iphone:[OBJECT_A] (as opposed to android:[OBJECT_B]).	BETTER	WORSE	NO_COMP
7	(See android:[OBJECT_A] Dethrones the iphone:[OBJECT_B] .)	BETTER	NO_COMP	NO_COMP
8	android:[OBJECT_A] didn't out pace the iphone:[OBJECT_B] this year, it just sold slightly better in America	WORSE	NO_COMP	WORSE
9	To me it is much better than iphone:[OBJECT_A] and android:[OBJECT_B] .	NO_COMP	UNCLEAR	UNCLEAR
10	(android:[OBJECT_A] , Crush , iPad , iphone:[OBJECT_B])	NO_COMP	BETTER	NO_COMP

are more cases where all three annotators agreed on one class.

The distinction between UNCLEAR and NO_COMP is still a problem. This illustrates that the choice of a class is subjective to some degree. For instance, sentence five (table 2.2.5) can be understood in two ways. First, one can read it as *PHP is better than OBJECT_A and OBJECT_B* because “[...] it is so broadly accepted and supported”. Since those qualities are not mentioned for OBJECT_A and OBJECT_B, they can be seen as absent for those objects which makes them worse than PHP. Then, UNCLEAR would be the correct label, because the sentence is comparative but does not compare the objects of interest against each other.

Second, one can read that PHP is an alternative, but there is no expression that it is better, since the sentence does not say it is *more accepted* or *more supported* than OBJECT_A or OBJECT_B. In this case, NO_COMP is the correct class.

Due to an error in the creation of the crowdsourcing task, the sentences where not shuffled. This means that the first one-hundred sentences of the second prestudy are the same as the one-hundred sentences of the first prestudy. Another problem is the bias: all sentences contained only the pairs *Ruby vs. Python* and *Android vs. iPhone*. Because the goal of the prestudy was mainly to assess the sentence selection method and the guidelines, this does not invalidate the results. Those problems were removed in the main study.

All in all, the prestudy was successful. There are only few cases where no agreement on the class could be achieved. The prestudy showed that the task at hand is not easy, but feasible.

2.3 Main Study

2.3.1 Design changes

The insights from the prestudy influenced the design of the main study.

The class `UNCLEAR` was renamed to `OTHER`, `NO_COMP` to `NONE`. Those names are a better description for the classes. Eventually, after the first 1500 sentences were finished the class `OTHER` was dropped completely (see section 2.3.3). The change was reflected in the annotation guidelines as well.

Instead of one big task, one task per domain was created. All tasks used the same annotation guidelines.

2.3.2 Sentence Selection

The sentence selection process was similar to the prestudy. The pairs and the cue words (see figure 2.3.1) changed. The cue words were generated using JoBimText, a software package for distributional semantics. JoBimText⁴ was queried for the nine words most similar to *better* and *worse*, so that more, different comparisons are captured by the selection process.

Figure 2.3.1: Cue Words

better	decent	harder	nastier
easier	safer	slower	inferior
faster	superior	poorly	mediocre
nicer	solid	uglier	
wiser	terrific	poorer	
cooler	worse	lousy	

Three domains were fixed for the object pairs. The domains were chosen in a way that a majority of people can decide whether a sentence contains a comparison or not. Also, a wide range of comparison patterns should be included in the data.

The most specific domain was “*Computer Science Concepts*”. It contains objects like programming languages, database products and technology standards such as Bluetooth and Ethernet. Many computer science concepts can be compared objectively, for instance, one can compare Bluetooth and Ethernet on their transmission speed. Some basic knowledge of computer science was needed to label sentences correctly. For example, to compare Eclipse and NetBeans, the annotator must know what an Integrated Development Environment (IDE) is and that both objects are Java IDEs. The need for this knowledge was communicated to the prospective annotators. The objects for this domain were manually extracted from “*List of ...*” articles from Wikipedia⁵.

⁴<http://ltmaggie.informatik.uni-hamburg.de/jobimviz/> (28.02.2018)

⁵ADD TO APPENDIX

The second, broader domain was “*Brands*”. It contains objects of different types (e.g. car brands, electronics brands, and food brands). As brands are present in everyday life of people, it is expected that anyone can label the majority of sentences containing well known brands such as “*Coca-Cola*” or “*Mercedes*”. As with computer science, the objects for this domain were extracted from “*List of ...*” articles from Wikipedia⁶.

The last domain is not restricted to any topic. For each one of twenty-four randomly selected seed words, ten similar words were extracted using JoBimText. The seed words (see figure 2.3.2) were created using <https://randomlists.com>⁷. Listing 2.3 shows the result⁸ for the seed word “*Yale*”. Duplicates or too broad terms (like *university*) were removed by manually.

Figure 2.3.2: Seed words for the Random domain

book	coffee	hoover	pencil	tolkien
car	cork	metallica	salad	wine
carpenter	florida	nbc	soccer	wood
cellphone	hamster	netflix	starbucks	xbox
christmas	hiking	ninja	sword	yale

Listing 2.3: Similar words to “Yale”

```

1 [...]
2   "results":
3     [{"score":701.0,"key":"yale#NP"},
4     {"score":245.0,"key":"harvard#NP"},
5     {"score":151.0,"key":"princeton#NP"},
6     {"score":135.0,"key":"mit#NP"},
7     {"score":135.0,"key":"cornell#NP"},
8     {"score":121.0,"key":"stanford#NP"},
9     {"score":116.0,"key":"university#NP"},
10    {"score":111.0,"key":"nyu#NP"},
11    {"score":111.0,"key":"university#NN"},
12    {"score":109.0,"key":"dartmouth#NP"}]
```

In the following, this domain is called *Random*. Some example pairs for all domains are shown in table 2.3.1.

Especially for brands and computer science, the object lists are long (4493 brands and 1339 for computer science). The frequency of each object was checked using a frequency dictionary to reduce the number of possible pairs. All objects with a frequency of zero and ambiguous objects were removed from the list. For instance, the objects “*RAID*”

⁶ADD TO APPENDIX

⁷Last checked: 25.01.2018

⁸<http://ltmaggie.informatik.uni-hamburg.de/jobimviz/ws/api/stanford/jo/similar/harvard%23NP?numberOfEntries=10f> (25.01.2018); Some uninteresting fields were removed for brevity

Table 2.3.1: Example pairs

Brands	Computer Science	Random
Microsoft vs. Apple	Java vs. Python	Baseball vs. Hockey
Nikon vs. Leica	Eclipse vs. Netbeans	Fishing vs. Swimming
Coca-Cola vs. Pepsi	OpenGL vs. Direct3D	SUV vs. Minivan
Nike vs. Adidas	Integer vs. Float	Kennedy vs. Nixon
Ibuprofen vs. Advil	USB vs. Bluetooth	Plastic vs. Wood
Ford vs. Honda	Oracle vs. MySQL	Harvard vs. Princeton

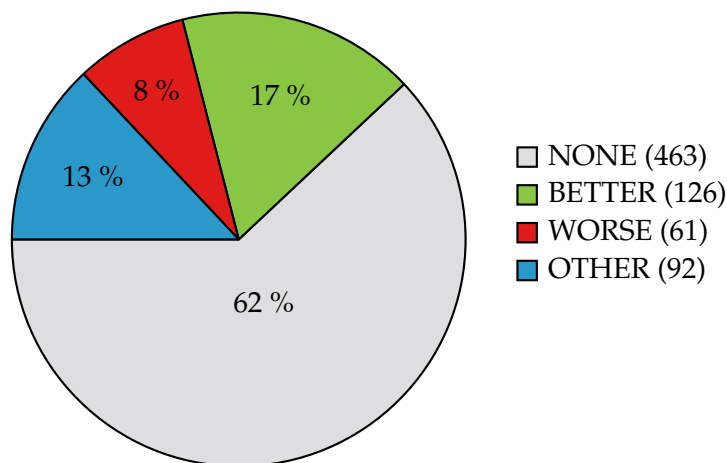
(a hardware concept) and “Unity” (a game engine) were removed from the computer science list as they are also regularly used nouns.

The remaining objects were combined to pairs. For each type, all possible combinations were created. For brands and computer science, the type is the URL of the Wikipedia page. For the random domain, the seed word was used. This procedure guarantees that only meaningful pairs are created.

The index was then queried for entries containing both objects of each pair. For 90% of the queries, the cue words were added to the query. All pairs were the query yielded at least one-hundred sentences were kept.

From all sentences of those pairs, 2500 for each category were randomly sampled as candidates for the crowdsourcing task. To check the sentence selection method once again, a small, random subset of the sentences was labelled by the author prior to the crowdsourcing task. Those labels were discarded for the crowdsourcing task. The label distribution of the 742 sentences is presented in the figure 2.3.3. The numbers are similar to the prestudy which shows that the procedure still works.

Figure 2.3.3: Data precheck



2.3.3 Brands

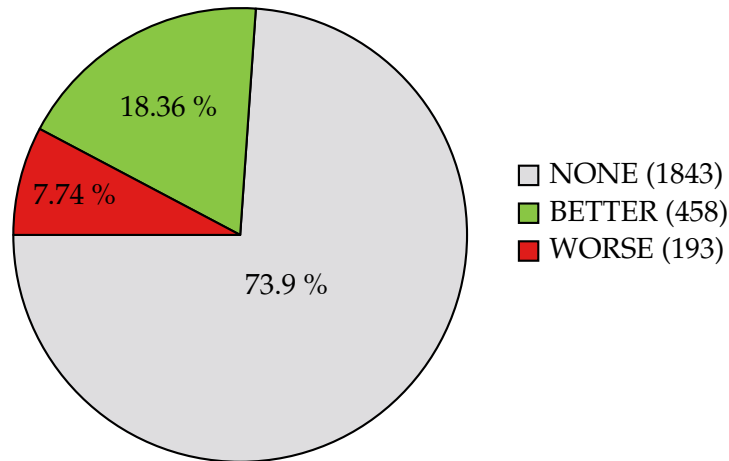
For the Brands domain, 2493 sentences were annotated. The sentences contained objects of sixty-three pairs. As shown in table 2.3.2, the annotators could agree on one class for the majority of sentences.

Table 2.3.2: Agreement for Brands

Distinct classes	Sentences	Percentage
1	1791	71.84%
2	645	25.87%
3	57	2.29%

The class distribution is presented in figure 2.3.4. The amount of comparative sentences is lower (26.1%) than in the prestudy. The reason for this is the abandonment of the OTHER (UNCLEAR) class.

Figure 2.3.4: Class distribution for Brands



Even with the rephrasing of that class, it was too confusing for the annotators. The class was not different enough from NONE (NO_COMP). Eventually, all sentences labelled as OTHER (124 sentences for brands) were merged into NONE. This decision was made after 750 sentences were labelled for each domain. First machine learning experiments also showed that OTHER is not distinguishable from NONE for all tested features and algorithms.

For the first 750 sentences, 398 passed the quiz while 357 failed. During the annotation process, fifty-two were removed as they answered to many test questions wrong. The overall satisfaction was 3.4 out of 5. In detail, the acceptance of the task was okay (instructions: 3.9, test question fairness: 3.4, ease of job: 3.4, payment: 3.8).⁹

⁹The 750 sentences were, due to a mistake, divided into two tasks, one with one-hundred (twenty-six participants in the survey) and one with 650 (fourty-one participants). The presented satisfaction values are the averages weighted by number of participants

For the rest of the sentences, fifty-one participants took the exit survey. The removal of OTHER increased the overall satisfaction to 3.8 (instructions: 4, test question fairness: 3.5, ease of job: 3.7, payment: 3.7).

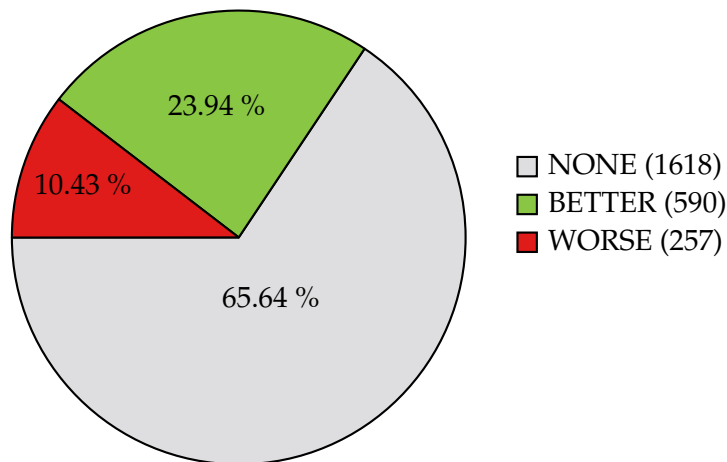
2.3.4 Computer Science

For the Computer Science domain, 2465 sentences (containing one of forty-three pairs) were annotated. As with brands, the agreement of the annotators (table 2.3.3) is satisfactory. The class distribution (figure 2.3.5) is better, as more sentences are comparative.

Table 2.3.3: Agreement for Computer Science

Distinct classes	Sentences	Percentage
1	1757	71.28%
2	643	26.09%
3	65	2.64%

Figure 2.3.5: Class distribution for Computer Science



Again, many people failed the quiz (121 failed, 132 passed) for the first 750 sentences. Thirty-five people dropped out during the annotation process because they missed too many test questions. The twenty-seven participants who took the exit survey rated the overall satisfaction with 3.5 (instructions: 3.5, test question fairness: 3, ease of job: 3.2, payment: 3.6).

The numbers improved for the rest of the sentences (after the removal of OTHER). Only sixty-four failed the quiz (379 passed); twenty-six were removed during the annotation process. Forty-seven participants took the exit survey on the second part. The overall satisfaction was rated with 3.9 (instructions: 4.2, test question fairness: 4, ease of job: 3.8, payment: 3.9).

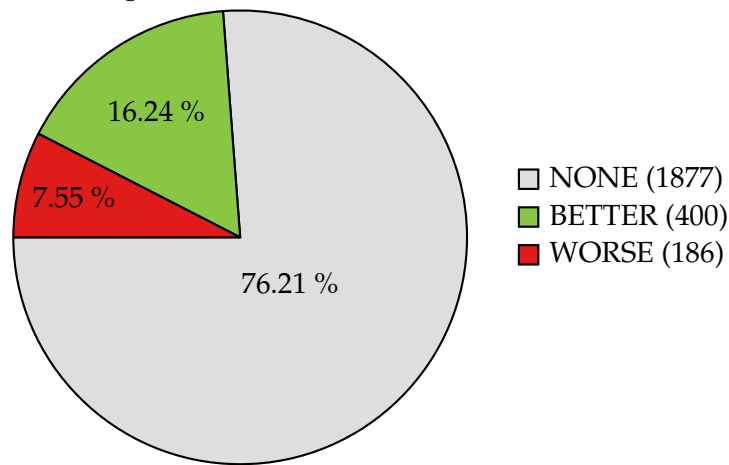
2.3.5 Random

The random domain has 2463 sentences and 167 pairs. The agreement (table 2.3.4) and class distribution (figure 2.3.6) are satisfactory as well.

Table 2.3.4: Agreement for Random

Distinct classes	Sentences	Percentage
1	1766	71.70%
2	627	25.46%
3	70	2.84%

Figure 2.3.6: Class distribution for Random



As with the other domains, the first 750 sentences (with `OTHER`) performed worse than the rest, as 128 annotators failed the quiz (164 passed) and twenty-seven annotators were removed during the task due to bad performance on the test questions. The Random domain got the worst satisfaction rating (twenty-nine participants) of all domains. The overall satisfaction was rated with 3.1 out of 5 (instructions: 3.6, test question fairness: 3.4, ease of job: 3, payment: 3.4). Once more, the numbers improve after `OTHER` was removed. Only seventy-five failed the quiz (423 passed). The ratings (thirty-seven participants) increased as well. The overall satisfaction was now rated with 4.0 (instructions: 4.1, test question fairness: 3.9, ease of job: 4, payment: 3.7).

2.3.6 Validation of results

Table 2.3.5 summarises the agreement of the annotators on all three domains. The annotators could agree on one class for the majority of the sentences. However, for 28.92% of the sentences, no clear decision could be made. For 2.67%, all annotators choose a different class.

This shows that the task at hand is difficult even for humans. For instance, sentence one in table 2.3.6 requires knowledge on mobile phones to be classified correctly. It also

Table 2.3.5: Agreement for all

Distinct classes	Sentences	Percentage
1	5275	71.08%
2	1948	26.25%
3	198	2.67%

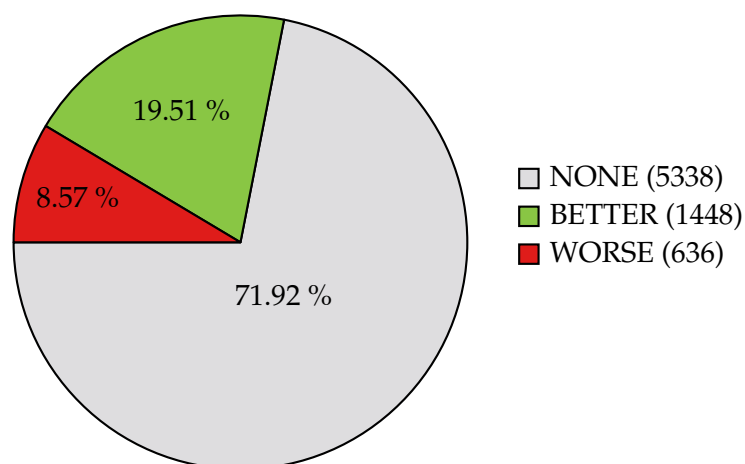
lacks a cue word. Sentence five includes a cue word (*harder*) and *than* which make it look like a comparison. However, the sentence is not a comparison in a way that one object wins against the other. In sentence two, the annotators overlooked the negation (“*wouldn’t*”).

Table 2.3.6: Uncertain Sentences (All)

#	Sentence	Ann. 1	Ann. 2	Ann. 3
1	Samsung:[OBJECT_A] voids warranty for rooting, Motorola:[OBJECT_B] doesn’t.	NONE	WORSE	BETTER
2	I wouldn’t say Microsoft:[OBJECT_A] is worse than Apple:[OBJECT_B] .	NONE	WORSE	WORSE
3	If you use NetBeans:[OBJECT_A] instead of Eclipse:[OBJECT_B]	OTHER	BETTER	NONE
4	Visibly faster than Python:[OBJECT_A] ... faster than Java:[OBJECT_B] I’d bet.	NONE	BETTER	BETTER
5	It’s light as aluminum:[OBJECT_A] , but harder than steel:[OBJECT_B] , he said.	NONE	OTHER	BETTER

The class distribution (figure 2.3.7) is similar to the prestudy. As expected, the majority of sentences is not comparative. The class BETTER is more than twice as big as the class WORSE, which will complicate the classification.

Figure 2.3.7: Class distribution for all



In the end, the crowdsourcing task was successful. For the majority of sentences

(71.08%) the annotators could agree on one class, while the amount of totally unclear sentences is small (2.67%).

3 Classification of Comparative Sentences

3.1 Experiments

The data collected from the crowdsourcing task was used as training data for classification problems. In the first problem, the machine learning algorithms were trained so that they were able to assign one of the three original classes (see section 2.3.1) to the data. The second problem is a simplification of the first one as it is designed as a binary classification problem. The classes `BETTER` and `WORSE` were merged into the class `ARG`.

The data was split into a training set (5937 examples; 4270 `NONE`, 1158 `BETTER` and 509 `WORSE`) and a held-out set. The held-out set stayed untouched until the final evaluation presented in section 3.6. During the development, the experiments were evaluated using stratified k-fold cross-validation where k equals five. The evaluation was done with the scikit-learn implementation. As the evaluation metric, the weighted mean¹ of the F1 scores of all classes is used.

3.2 Choice of Algorithms

To find the best performing classification algorithms, eleven (see table 3.2) were selected and compared. For all algorithms under test, the unigrams of the whole sentence were used as binary features (the implementation is described in section 3.3.1). Unigrams are a simple yet efficient feature for text classification ([Cavnar et al., 1994]) which makes them suitable as a baseline feature for comparisons. Stratified k-fold with k equals five was used to assess the quality of the algorithm. The unweighted average of the five F1 scores was used to make the different algorithms more comparable.

For all algorithms except XGBoost, the implementation available from scikit-learn ([Pedregosa et al., 2011]) was used. Tree-based methods and simple linear models work good. The *Support Vector Machine* without a linear kernel was not able to classify the examples at all as it assigns `NONE` to all training examples. This is true for all non-linear kernel functions available in scikit-learn (RBF, polynomial and Sigmoid).

Naive Bayes classifies examples correctly into all classes, yet the performance is still same as the baseline shown in section 3.4.1.

As XGBoost and Logistic Regression already work in a pleasing way, no further investigations on the performance of the *Support Vector Machine* and *Naive Bayes* were done. In

¹weighted by number of examples per class

Table 3.2.1: Classification Algorithms

Algorithm	F1 Score	Used in
XGBoost	0.76	-
Logistic Regression	0.75	[Dusmanu et al., 2017, ?, Lippi and Torroni, 2016]
AdaBoost	0.74	[Aker et al., 2017]
Linear SVC	0.73	[Aker et al., 2017]
Decision Tree	0.72	[Stab and Gurevych, 2014, Lippi and Torroni, 2016]
Stochastic Gradient Descent	0.72	-
Random Forest	0.72	[Dusmanu et al., 2017, Stab and Gurevych, 2014, Eckle-Kohler et al., 2015, Aker et al., 2017, Lippi and Torroni, 2016]
Extra Trees Classifier	0.72	-
K Neighbors	0.72	[Aker et al., 2017]
Support Vector Machine (non-linear kernel)	0.60	[Stab and Gurevych, 2014, Eckle-Kohler et al., 2015, Park and Blake, 2012b, Lippi and Torroni, 2016, Habernal and Gurevych, 2016]
Naive Bayes	0.60	[Stab and Gurevych, 2014, Eckle-Kohler et al., 2015, Aker et al., 2017, Park and Blake, 2012b, Lippi and Torroni, 2016]

the following sections, all experiments were done using XGBoost.

A set of hyper-parameters for XGBoost were tested using exhaustive grid search and randomized search. However, no significant increase in the F1 score could be achieved.

3.3 Features

The following section briefly describes each feature used for both classification tasks.

Two observations are true for all features. First, the F1 score increases by at least five points if only the part of the sentence between both objects of interest is used (from now on called “*middle part*”).

Second, the objects do not make any significant difference. All features were tested with four versions of the sentence (eventually with the middle part only).

1. the sentence stayed unchanged
2. the objects of interest removed completely from the sentence
3. both objects replaced with *OBJECT*
4. the first object replaced with *OBJECT_A*, the second object replaced with *OBJECT_B*

Most of the time, the F1 score for the different versions differ - if at all - in the third decimal place. In exceptional cases, using the fourth option increases the F1 score by X points. If not stated differently, all features used the unchanged middle part of the original sentence.

3.3.1 N-Gram Models

Uni-, bi- and trigram models were used as features. The respective n-grams were computed using *textacy*², a higher level library building upon *spaCy*³. The basis for the n-grams were all training instances of the given fold. Each n-gram was modelled as a binary feature.

The idea behind using trigram models despite the short text length was to capture trigrams like “*is better than*” or “*is worse because*” which would indicate a comparison. However, the performance of trigram models was not satisfying.

3.3.2 Part of Speech

Several boolean features were used to model the appearance of parts-of-speech in the sentence. The part-of-speech tagging was done with *spaCy*. The feature capturing the appearance of the part-of-speeches *comparative adverb (RBR)* and *comparative adjectives (JJR)*⁴ has an equal performance to the unigram model. Other part-of-speech tags did not get a result above the baseline (for instance *superlative adjective (RBS)*) However, using this feature in isolation, no training example is assigned to the class

3.3.3 Mean Word Embeddings

3.3.4 Sentence Embeddings

3.3.5 Dependency Features

3.3.6 Other Features

A couple of features did not contribute to the classification at all, be it alone or in combination with other features. This includes the length of the sentence or its parts, statistics over punctuation or named entities.

3.4 Classification with three classes

3.4.1 Baseline

As described in section 1.2.2, there is no task which is similar enough to this one which could be used as a baseline. Thus, two baselines using the obtained data were created.

²<https://github.com/chartbeat-labs/textacy>

³<https://github.com/explosion/spaCy>

⁴https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

The first baseline, shown in table 3.4.1, assigns all sentences to the class `NONE`.

Table 3.4.1: Majority Class Baseline (three classes)

	Worst			Average			Best		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BETTER	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WORSE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NONE	0.72	1.00	0.84	0.72	1.00	0.84	0.72	1.00	0.84
average	0.52	0.72	0.60	0.52	0.72	0.60	0.52	0.72	0.60

The second baseline is created by assigning classes to the data at random, respecting the distribution of classes in the original data. The results are shown in table 3.4.2.

Table 3.4.2: Random Baseline (stratified, three classes)

	Worst			Average			Best		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BETTER	0.17	0.17	0.17	0.21	0.23	0.22	0.18	0.18	0.18
WORSE	0.10	0.09	0.09	0.05	0.04	0.04	0.15	0.14	0.15
NONE	0.71	0.71	0.71	0.72	0.71	0.72	0.73	0.74	0.74
average	0.55	0.55	0.55	0.56	0.56	0.56	0.57	0.58	0.58

For both baselines, the scikit-learn's `DummyClassifier` was used.

3.4.2 Results

3.5 Binary classification

3.5.1 Baseline

Table 3.5.2; Table 3.5.1

Table 3.5.1: Random Baseline (stratified, binary)

	Worst			Average			Best		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
ARG	0.25	0.26	0.25	0.29	0.29	0.29	0.30	0.30	0.30
NONE	0.71	0.70	0.71	0.72	0.72	0.72	0.73	0.72	0.72
average	0.58	0.58	0.58	0.60	0.60	0.60	0.61	0.60	0.60

Table 3.5.2: Majority Class Baseline

	Worst			Average			Best		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
ARG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NONE	0.72	1.00	0.84	0.72	1.00	0.84	0.72	1.00	0.84
average	0.52	0.72	0.60	0.52	0.72	0.60	0.52	0.72	0.60

3.5.2 Results

3.6 Final results

3.7 Discussion

4 Conclusion

Bibliography

- [Aker et al., 2017] Aker, A., Sliwa, A., Ma, Y., Lui, R., Borad, N., Ziyaei, S., and Ghobadi, M. (2017). What works and what does not: Classifier and feature analysis for argument mining. In *Proceedings of the 4th Workshop on Argument Mining*, pages 91–96.
- [Cavnar et al., 1994] Cavnar, W. B., Trenkle, J. M., et al. (1994). N-gram-based text categorization. *Ann arbor mi*, 48113(2):161–175.
- [Daxenberger et al., 2017] Daxenberger, J., Eger, S., Habernal, I., Stab, C., and Gurevych, I. (2017). What is the essence of a claim? cross-domain claim identification. *CoRR*, abs/1704.07203.
- [Dusmanu et al., 2017] Dusmanu, M., Cabrio, E., and Villata, S. (2017). Argument mining on twitter: Arguments, facts and sources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2307–2312.
- [Eckle-Kohler et al., 2015] Eckle-Kohler, J., Kluge, R., and Gurevych, I. (2015). On the role of discourse markers for discriminating claims and premises in argumentative discourse. In *EMNLP*, pages 2236–2242.
- [Fiszman et al., 2007] Fiszman, M., Demner-Fushman, D., Lang, F. M., Goetz, P., and Rindflesch, T. C. (2007). Interpreting comparative constructions in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 137–144. Association for Computational Linguistics.
- [Gupta et al., 2017] Gupta, S., Mahmood, A. A., Ross, K., Wu, C., and Vijay-Shanker, K. (2017). Identifying comparative structures in biomedical text. *BioNLP 2017*, pages 206–215.
- [Habernal et al., 2014] Habernal, I., Eckle-Kohler, J., and Gurevych, I. (2014). Argumentation mining on the web from information seeking perspective. In *ArgNLP*.
- [Habernal and Gurevych, 2016] Habernal, I. and Gurevych, I. (2016). Argumentation mining in user-generated web discourse. *CoRR*.
- [Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.
- [Lippi and Torroni, 2016] Lippi, M. and Torroni, P. (2016). Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol.*, 16(2):10:1–10:25.

- [Panchenko et al., 2017] Panchenko, A., Ruppert, E., Faralli, S., Ponzetto, S. P., and Bie-
mann, C. (2017). Building a web-scale dependency-parsed corpus from commoncrawl.
- [Park and Blake, 2012a] Park, D. H. and Blake, C. (2012a). Identifying comparative claim
sentences in full-text scientific articles. In *Proceedings of the Workshop on Detecting Struc-
ture in Scholarly Discourse*, pages 1–9. Association for Computational Linguistics.
- [Park and Blake, 2012b] Park, D. H. and Blake, C. (2012b). Identifying comparative claim
sentences in full-text scientific articles. In *Proceedings of the Workshop on Detecting Struc-
ture in Scholarly Discourse, ACL '12*, pages 1–9, Stroudsburg, PA, USA. Association for
Computational Linguistics.
- [Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion,
B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Pas-
sos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn:
Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- [Šnajder, 2017] Šnajder, J. (2017). Social media argumentation mining: The quest for de-
liberateness in raucousness.
- [Stab and Gurevych, 2014] Stab, C. and Gurevych, I. (2014). Identifying argumentative
discourse structures in persuasive essays. In *EMNLP*, pages 46–56.
-

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den