Saarland University

Department of Computational Linguistics

Bachelor thesis

Automatic Detection of Linguistic Quality Violations

Author:
Jonathan Oberländer
Matriculation: 2539632

Supervisors:
Prof. Manfred Pinkal
Annemarie Friedrich
Alexis Palmer

Date of submission

Abstract ...Abstract here...

Contents

1 Introduction

With an ever-growing amount of information, it is becoming increasingly important to reduce that information to a processible amount, by summarizing it. Most of the time, it is infeasible to do this by hand, and there is thus a need for automating the process.

Automatic summarization systems take one or more documents and produce a summary that contains the most important information from these documents. Such systems can be divided in *extractive* and *abstractive*, and in *single-document* and *multi-document* summarization systems.

A *single-document* system takes only one document as an input and produces a summary of that document, while a *multi-document* system summarizes multiple documents on the same topic.

An *abstractive* system is one which infers knowledge from its input into an internal semantic representation and generates a new text based on that knowledge. The task of constructing an abstractive summarization system is very complex, which is why most systems used today are *extractive*, which means they work by taking phrases from their input and glue them together to produce an output.

We are only concerned with extractive, multi-document summarization.

Evaluation of automatic summarization systems has largely been done on content coverage, which is why generated summaries often lack linguistic quality (Nenkova et al., 2011), meaning that they contain many errors of various kinds. A new focus on linguistic quality (LQ) evaluation would presumably help decrease these issues and produce overall better summarization systems.

As a first step in this direction, Friedrich et al. (2014) created the *LQVCorpus*, which consists of multi-document summaries from the TAC 2011 shared task on Guided Summarization (Owczarzak and Dang, 2011), annotated manually with LQ violations on entity, clause and discourse level.

Manual annotation is an expensive and time-consuming process, for which an automatic system could be an alternative. Our work deals with creating such a system that manages at least part of the annotation automatically. We develop and evaluate our system on top of the *LQVCorpus*. We also limit ourselves to the clause-level violations listed in Friedrich et al. (2014), except for `no_semantic_relation`.

The rest of this paper is organized as follows:

The next section discusses related work. In Section 3, we take a look at the *LQVCorpus* and the challenges it poses for our experiments and methods, which are discussed in Section 4, where we also evaluate our experiments. We discuss our results in Section 5. In Section 6, we discuss future work and conclude.

2 Related Work

Most work in the field of automatic grammar checking seems to have been done in the area of second language learning. Bender et al. (2004) use the LKB parser and generator Copestake (2002) with the English Resource Grammar (Flickinger, 2000) to build *Arboretum*, a system that, when given an erroneous input sentence, detects that error and outputs a corrected version of the sentence. Their method uses manually selected mal-rules (Schneider and McCoy, 1998) to detect error types specific to second language learners of English.

While this seems to work for their specific domain, it does not transform well into ours: The mistakes language learners make, and the kind of ungrammaticality summarization systems produce, are very

different. Source texts that are being summarized are usually grammatical, and get their ungrammaticality during summarization from being compressed from two formerly grammatical sentences into one ungrammatical one, for example. Mistakes typical for learners of English as a second language either usually do not occur in automatic summarizations (such as misspelt words), or do happen, but due to a different cause (for example agreement).

Chodorow and Leacock (2000) build *ALEK*, a system to grade TOEFL essays. It does so by viewing the task as an extension of Word Sense Disambiguation, and collecting n-grams of POS-tags and function words, and then classifying using primitive statistical measures like *Mutual Information* and the χ^2 test. However, only the whole essay is classified into one of six possible scores, and this decision is only made by looking at the test subject's understanding of the three words *Concentrate*, *Interest* and *Knowledge* in context.

Sun et al. (2007), while also citing second language learning as their primary application, use a different approach: Instead of manually writing rules, they employ a supervised statistical learning method. They extract what they call *labeled sequential patterns* (ngrams associated with a class) from both erroneous and correct sentences in their training corpus, enrich them with other automatically computed features, and use them as input for classification models; an SVM (SVMlight, Joachims, 2002) and a Naïve Bayes classification model. The system achieves F-scores of around 80%.

There also has been work in the area of assessing grammaticality of automatically generated summaries, namely Vadlapudi and Katragadda (2010). They used n-grams on POS-tag level and evaluated their system by looking at how well their classification correlates to ROGUE scores (Lin, 2004). Vadlapudi and Katragadda achieve a Spearman's ρ correlation of up to 77% to ROGUE scores manually assigned by annotators.

Wagner et al. (2007) compare various methods for detecting ungrammaticality: The "flat" method of using n-grams, a parser-based method using the XLE LFG parser (Maxwell and Kaplan, 1996), and using decision trees trained on the output of each of these methods. Their best method achieves an F-score of about 67%.

The BLEU score (Papineni et al., 2002) is a method for evaluating the effectiveness of a machine translation system, which uses a number of (human) reference translations to score an automatic translation. They showed that their score correlates highly with human evaluation, which has lead to the BLEU score having become the standard evaluation metric for machine translation.

3 Corpus Study

As a first step, we inspect the corpus. We find that the type system of the LQVCorpus (Valeeva, 2013) is not detailed enough for our purposes and has a few shortcomings: Especially under the label of `other_ungrammatical_form` many different types of errors are combined. For this reason, we define a number of subtypes for this violation:

missing spaces

This is the most common type of error: The sentence contains a word that does not exist. In almost all cases, this happens when whitespace between two words is missing.

Example: *A strong earthquake measuring 7.8 magnitude struck* ***Wenchuancounty*** *of Sichuan Province on Monday, leaving at least* ***12,000people*** *died and thousands more injured.*

missing words

One or multiple words are clearly missing. These seem to most often be function words such as articles or pronouns, rather than content words. In the example, an underscore marks the position where a word, probably "was" is missing.

Example: *An Israeli woman _ killed and 11 others were wounded in the suicide bombing at a shopping mall in southern Israeli town of Dimona.*

punctuation error

Most of the time, this means: Punctuation is missing, but it can also mean there is something else wrong with punctuation in that sentence which makes it ungrammatical, as can be seen in the example: Unbalanced parantheses.

Example: *China has allocated 200 million yuan (million dollars for disaster relief work after an earthquake rocked the country's killing at least seven people, state reported on Tuesday.*

capitalization error

A word that should be capitalized is not or one that should not be capitalized is.

Example: *earlier on **m**onday **g**erman chancellor **a**ngela **m**erkel and foreign minister **f**rank **w**alter **s**teinmeier offered their condolences to **c**hina over the heavy loss of life in the powerful earthquake that hit **c**hina's southwestern province of **s**ichuan.*

unparsable

This subtype means that the (human) annotator was unable to make sense of the sentence and considers it not correctly parsable. Most instances of this subtype seem to happen when sentence fusion occurs; the summarization system combined two or more sentences in an ungrammatical or meaningless way.

Example: *All of those provinces and Chongqing, a special municipality _ deepest condolences to those who lost their loved ones in the devastating natural disaster.*

heading

The sentence contain (usually at the beginning) a sequence of capitalized or otherwise heading-like words.

Example: *THE CURRENT FIX: Internet applications such as firewalls and spam filters attempt to control security threats.*

incomplete sentence

The type system of the LQVCorpus (Valeeva, 2013) defines an `incomplete_sentence` violation as words being cut off at the end of a sentence. A couple of times though, this also occurs at the beginning of a sentence. We restructure the type system by treating all types of incomplete sentences as this subtype of `other_ungrammatical_form`.

Example: *A Palestinian suicide bomber detonated an explosive belt at a commercial center in Dimona on.*

should be other type

Sometimes sentences that are tagged as `other_ungrammaticality` should (also) be tagged as another type, such as sentences containing datelines. This doesn't necessarily mean that the sentence contains no other ungrammaticalities. This subtype be caused by inconsistencies in the corpus, or from the process of mapping the annotated violations to sentences. While the following example is an `incomplete sentence`, it also contains a dateline.

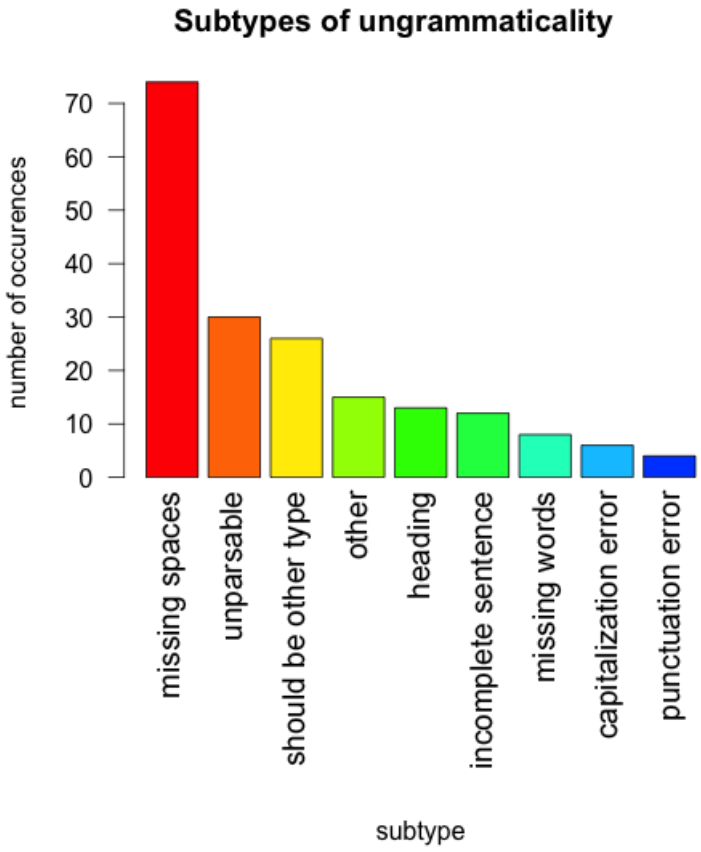

Figure 1. Subtypes of ungrammaticality in *dev-2*. Where the annotator didn't know what subtype to annotate the sentence with, they selected "other".

> Example: *its deepest sympathy to the Israeli people,**00 a.m. local time (0800 GMT)** as a suicide bomber detonated an explosive belt in the commercial center, according to local media reports.*

The corpus was split into five parts (*dev-1* through *dev-2*), each containing 8 topics, so that e.g. *dev-1* contains topics D1101 through D1108.

After manually counting and investigating these subtypes on *dev-1* we then mark these types in *dev-2* for every sentence that is tagged as `other_ungrammatical_form` in the corpus. We allow sentences to have multiple subtypes and also annotate `incomplete_sentence` violations with additional (sub)types, if applicable. The frequency distribution of the subtypes in *dev-2* can be seen in Figure 1.

4 Experiments

We conduct a number of experiments to find reliable ways to detect different kinds of ungrammaticality. The annotation, our experiments and the evaluation are all done on sentence basis. We use Stanford CoreNLP (Manning et al., 2014) for sentence splitting.

4.1 Detecting Ungrammaticality

Perhaps the most noticeable result is the large portion of *missing spaces* violations, meaning clauses that include tokens which aren't correct words. Almost all of these cases come from a missing space between two words forming tokens such as `reportsreaching` or `Wenchuancounty`. As there is such a large amount of this type, finding a reliable detection method for this subtype would significantly boost detection of ungrammaticality in general.

4.1.1 The UnknownTokens method

We call our approach to identifying missing spaces **UnknownTokens**: Tagging a sentence as containing a *missing spaces* ungrammaticality if and only if there is a token in the sentence that isn't a "known" token, i.e. one that doesn't exist in a list of known words. In our setting, we can easily deduce such a list from the source documents of the automatic summarization process. If the source document is available and used as a base for our vocabulary, we call that approach $\textbf{UnknownTokens}_{\textbf{source}}$.

In case the original documents aren't available or this method should be used in a different setting, we also investigate how to adapt our approach to that. This approach has a few steps, which we will all evaluate individually: First, an initial list of words is created from a large text corpus ($\textbf{UnknownTokens}_{\textbf{gw}}$); we use the first 20% of the AFE part of the gigaword corpus (Graff et al., 2003). If the corpus is large and general enough, this will probably cover a large vocabulary.

Because this will not include words for named entities yet, we only consider a token to be an unknown token, if it is not tagged as a Named Entity by a Named Entity Recognizer ($\textbf{UnknownTokens}_{\textbf{ner}}$). We employ the Stanford Named Entity Recognizer (Finkel et al., 2005), which is a widely used, state-of-the-art NER that comes with a model for the English language.

In order to improve upon the NER and further increase recall, we use two simple heuristics ($\textbf{UnknownTokens}_{\textbf{heur}}$): Firstly, we assume that words that start with a capital letter and are only followed by lower case letters are named entities and therefore no unknown tokens. Finally, we automatically check whether an unknown token has an entry on the online encyclopedia Wikipedia ($\textbf{UnknownTokens}_{\textbf{wiki}}$).

We automatically label any tokens that are neither on our list of known tokens from GigaWord, nor tagged as a named entity as "unknown" and tag the sentence as containing unknown tokens. The combined method $\textbf{UnknownTokens}_{\textbf{gw+heur+ner+wiki}}$ is what we call $\textbf{UnknownTokens}_{\textbf{general}}$.

4.1.2 Evaluation of UnknownTokens

We evaluate how effective **UnknownTokens** is in detecting `missing spaces` violations, both with ($\textbf{UnknownTokens}_{\textbf{source}}$) and without ($\textbf{UnknownTokens}_{\textbf{general}}$) access to the source documents, and compare it to a **baseline**, which always assumes the most common case (no `missing spaces` violation). The standard metrics precision, recall and F-score are used for evaluation.

Table 1 shows the results of the evaluation. Because the amount of sentences not containing a missing spaces violation is much larger than the ones that do, several weighted metrics are provided, too:

The micro-average method consists of summing up the individual values from the confusion matrix (True Positive, False Positive, True Negative and False Negative) and calculating new metrics based on the results. We only display the F-score here, since F-score, recall and precision are identical for a 2-class scenario: Since the true positives of the missing spaces class are identical with the true negatives of

the no missing spaces class (and vice versa), the new number of true positives and true negatives (for the micro average) is $tp + tn$ of any of the two classes. The same goes for false negatives and false positives. That means that when only dealing with two classes, in the micro-average method, the terms *positive* and *negative* can be used interchangeably. As precision is defined as $\frac{tp}{tp+fp}$ and recall as $\frac{tp}{tp+fn}$; as the equations only differ in whether they use fp or fn (which are exactly the same here) in the divisor, precision and recall (and therefore F-score, too, since the F-score is simply the harmonic mean of precision and recall) are identical.

Macro-averages are just arithmetic means of precision and recall. For both micro- and macro-average, the new F-score is calculated from the new scores.

Both $\textbf{UnknownTokens}_{\textbf{general}}$ and $\textbf{UnknownTokens}_{\textbf{source}}$ improve vastly over the baseline, but it is clear, that if the source text is available, it should be used to infer known words.

4.1.3 The RandomForest$_{\text{UT}}$ method

To detect more general forms of ungrammaticality, several other methods were investigated. Following Wagner et al. (2007), we learn decision trees from outputs of several systems that on their own have not proven sufficient for this task. For this experiment, the development set *dev-1* was used.

We train a language model on parts (the same parts we used as a word source in $\textbf{UnknownTokens}_{\textbf{general}}$ (section 4.1)) of the Gigaword corpus(Graff et al., 2003), extract perplexity scores for the sentences in our test set, and annotate them automatically. For the language model generation and the extraction of perplexity scores, we employ SRILM (Stolcke et al., 2002), a language modeling toolkit.

We then use a RandomForest (Breiman, 2001) to learn decision trees. For this task, we use the machine learning toolkit WEKA (Hall et al., 2009). We use three features for our decision tree:

- **LM_PP** is the perplexity calculated for a sentence by the language model, following Sun et al. (2007). This is a numeric feature. It proved to not be sufficient for detecting ungrammaticality on its own.
- **missing spaces** is the classification of $\textbf{UnknownTokens}_{\textbf{source}}$ for a given sentence, as described in section 4.1. This is a binary feature.
- **numberofwords** is the number of words in the sentence. We follow Wagner et al. (2007) here, the intuition is that ungrammatical sentences will often be extraordinarily long, or very short:

  Former US President George W Bush, Who Will attend a memorial service on Tuesday, Virginia, and the university of at least 33 people Were killed, in the Worst shooting rampage in modern history, the White House said.

  The report of a

4.1.4 Evaluation of RandomForest$_{\text{UT}}$

We performed 10-fold cross-validation, which lead to a weighted F-score of 86.3%. A total of 1215 (88.1%) instances was classified correctly, 164 (11.9%) incorrectly.

	Missing spaces			No missing spaces			Macro-average	Accuracy
	P	**R**	**F**	**P**	**R**	**F**	**F**	
Baseline	0.0	0.0	0.0	94.8	**100**	97.3	48.7	94.8
$\mathbf{UT_{gw}}$	15.0	**98.7**	26.0	**99.9**	69.1	81.7	68.2	70.7
$\mathbf{UT_{gw+heur}}$	30.5	97.3	46.5	99.8	87.8	93.4	76.5	88.3
$\mathbf{UT_{gw+heur+ner}}$	35.5	97.3	52.0	99.8	90.3	94.8	78.6	90.6
$\mathbf{UT_{gw+heur+ner+wiki}}$	70.3	96.0	81.2	99.8	97.8	98.8	90.6	97.7
$\mathbf{UT_{source}}$	**95.9**	94.6	**95.2**	99.7	99.7	**99.7**	**97.5**	**99.5**

Table 1. Evaluation of **UnknownTokens** on *dev-2*, in percent. $\mathbf{UT_{wordnet+heur+ner+wiki}}$ is equivalent to $\mathbf{UnknownTokens_{general}}$. $_{\mathbf{wn}}$ = Wordnet-based dictionary, $_{\mathbf{heur}}$ = title case heuristics, $_{\mathbf{ner}}$ = Stanford NER, $_{\mathbf{wiki}}$ = Wikipedia entry check

4.1.5 The RandomForest$_{\text{parser}}$ method

We also investigated whether parser output would improve the results. For this task, we utilized the Answer Constraint Engine (Packart, 2011) with the English Resource Grammar (Copestake, 2002) to parse the entire corpus and annotate it with output statistics of the parser: Following Wagner et al. (2007) loosely, we extracted the number of readings/parses, the amount of memory the parser used for the parse and the success status, which could be either "ok" or "ignored", the latter appearing when the parsing failed entirely for some reason.

We then added these three features to **RandomForest$_{\mathbf{UT}}$**.

4.1.6 Evaluation of RandomForest$_{\text{parser}}$

When evaluating the emerging decision tree on general ungrammaticality using the same method as in Section 4.1.4, it yielded worse results than **RandomForest$_{\mathbf{UT}}$** alone. We therefore inspected whether it could instead be used to find instances of the subtype *unparsable*. This lead to a precision of 86.6%, a recall of 87.9%, and an F-Score of 86.3%. A total of 87.9% of the instances was classified correctly.

4.2 Datelines

Source texts from news articles, like the ones in the TAC dataset, often contain datelines, which sometimes make it into the generated summary. While in the context of the article datelines made sense (often displayed smaller or otherwise visually distinctive), they are not needed, distracting and confusing in the summaries:

BLACKSBURG, Virginia 2007-04-16 18:34: 44 UTC *A gunman opened fire in a dorm and classroom at Virginia Tech on Monday, killing at least 30 people in the deadliest shooting rampage in U.S. history.*

4.2.1 Method

The emerging patterns are so regular, that we use regular expressions to detect them. A few iterations of the regex were done, but so far the highest results were achieved using the regular expression

```
UTC|^\d{4}-\d{2}-\d{2}|^[A-Z]{3,}|^(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct
    |Nov|Dec)
```

The expression has four parts, of which only one has to match to make the system classify a sentence as containing datelines: The first part matches the string "UTC" (Coordinated Universal Time, a common fragment in datelines), the second one numeric dates like `2014-04-13`, the third one capitalized words that are longer than 4 letters and are therefore expected not to be abbreviations (an example would be `LONDON, Feb. 4 (Xinhua)`), and the final part matches abbreviated names of months. Only "UTC" can occur anywhere, everything else needs to occur at the beginning of a sentence to be counted.

4.2.2 Evaluation

This very simple approach was evaluated on *dev-2* and leads to good results: The F-score of the class of dateline-containing sentences is 90.7% (precision: 90.3%, recall: 91.1%). When looking into the inaccurately detected instances, the inconsistencies of the LQVCorpus make up most of the misses:

IVORY TUSKS

is classified as containing a dateline by the regular expression, but is not annotated as such in the corpus. An example of a false negative would be:

00 a.m. local time (0800 GMT)

which our system doesn't match.

4.3 Redundancy

Automatic summaries constructed from multiple source texts on the same topic often suffer from redundancy, as sentences that contain the same or largely the same information are occasionally included in the summary. As a consequence, reading fluency is impaired.

While — in contrast to many other linguistic quality violations — this also has an impact on traditional evaluation scores for automatic summarization systems (because the information density is lower than it could be), it is also considered to be a problem of our domain.

In the LQVCorpus, different degrees of redundant sentence pairs occur. Some sentence pairs annotated as redundant contain similar information, but using completely different phrasing and focus:

According to a survey by the State Food and Drug Administration, 65 percent of the respondents worried about the food safety situation in China.

Food and drug safety has become a major concern of Chinese people.

Other pairs of sentences are apparently at least partly composed of the same source sentence, but are still different. One of the sentences might contain additional information and the two sentences might still be of very different length:

Cyclone Sidr, described as the worst storm in years to hit low-lying and disaster-prone Bangladesh, crashed into the southwestern coast Thursday night before sweeping north over the capital Dhaka.

The cyclone hit the southwestern coast of Bangladesh on Thursday before sweeping north to the capital Dhaka.

Finally, sometimes subsequent sentences have marginal differences: *An unknown number of cats and dogs suffered kidney failure and about 10 died after eating the affected pet food, Menu Foods said in announcing the North American recall.*

An unknown number of cats and dogs suffered kidney failure, and about 10 died after eating the affected pet food, Menu Foods said in announcing the North American recall.

We are seeking to find all of these sentences, but concentrate on the latter two cases where the redundancy is obvious for humans.

4.3.1 Method

Our method (**Unigrams**) consists of a simple unigram overlap score, which we found to outperform more sophisticated approaches:

First, we remove all non-alphanumeric characters from both sentences and split them into words. We then treat the words as a set and divide the cardinality of the intersection of each sentence's set by the number of words in the smaller sentence. A given threshold is then used to classify instances.

A number of other methods are also implemented: **Bigrams** is a variation of the **Unigrams** method which uses bigrams instead of unigrams. **Combined** is using a score that the mean of the **Unigrams** and **Bigrams** scores.

Levenshtein consists of the standard edit distance(Levenshtein, 1966) on word level. A threshold of 2 yielded best results on our development set, meaning that sentences are classified as containing redundancies if their word-level Levenshtein distance is 1 or 0. Prior to computation of the score, non-alphanumeric characters were again pruned from the sentences.

Finally, as a further **Baseline** we classify all sentence pairs as containing redundancies.

We briefly considered using BLEU scores (Papineni et al., 2002) for redundancy detection (with the rationale that if the second sentence is a "good translation" of the first one, it might be redundant), but BLEU scores get their performance from having multiple reference translations available, which is not the case here.

4.3.2 Evaluation

In the corpus, redundancy isn't annotated on sentence base, but rather on clause level, where "clause" is an arbitrary substring of the summary text. That means that redundancy annotations can span over many sentences, can skip sentences inbetween and can be contained in a single sentence.

For our purposes, we only consider redundancies that occur in two subsequent sentences and our evaluation metrics are based on those.

To obtain a good threshold for **Unigrams**, **Bigrams**, and **Combined**, we compare their performance on the development sets *dev-1* and *dev-2*. The results can be seen in Figure 2.

For recall, **Unigrams** outperforms **Bigrams**, while for precision **Bigrams** is predominant.

As is to be expected, higher thresholds correlate with higher precision and lower recall. And while the F-scores are given in Figure 2 for informational purposes, they are not what we want to optimize; while recall is desirable, we feel that precision is much more important for a system that annotates violations.

We therefore choose a threshold of 0.4 for the **Bigrams** and **Combined**, and one of 0.5 for **Unigrams**, because there appear to be local maxima at that point with still some amount of recall.

The results can be seen in Table 2. Evidently, the **Unigrams** method outperforms **Levenshtein** and the **Baseline** as well as the other n-gram methods.

5 Discussion

Detecting linguistic quality violations automatically is not a trivial task. It is not always easy for humans to produce consistent annotations and annotations aren't necessarily highly consistent across humans, as can be seen in the inter-annotator agreement in Friedrich et al. (2014).

Sometimes, however, the easier methods outperformed the more sophisticated approaches. This can be the case because...

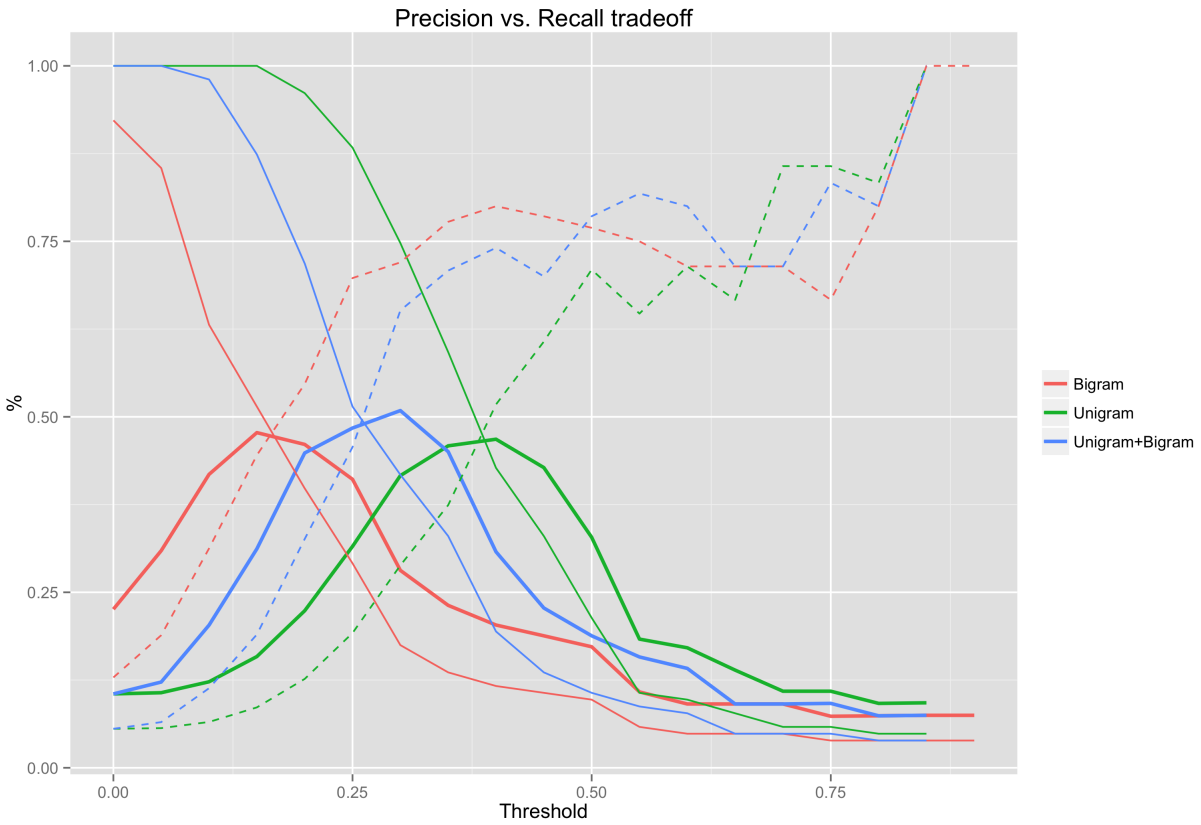

Figure 2. Performance of various thresholds for **Unigrams**, **Bigrams**, and **Combined** in *dev-1* and *dev-2*. Continuous lines are recall, dashed lines are precision and bold lines are F-score.

	Precision	Recall	F-Score
Baseline	4.5%	**100%**	8.7%
Levenshtein	15.8%	17.3%	3.1%
Unigrams	**58.0%**	28.2%	**37.0%**
Bigrams	55.6%	14.5%	22.9%
Combined	56.8%	24.3%	34.0%

Table 2. Evaluation of **Unigrams**, **Bigrams**, and **Combined** on everything but *dev-1* and *dev-2*

The corpus we used was not an optimal choice either.

The LQVCorpus that we used for all of our experiments is, unfortunately, not large enough to have a wide coverage of all violations. This can be seen most prominently when trying to detect redundancy (Section 4.3).

Apart from its size, the corpus is also often annotated inconsistently or at least inappropriate for our purposes. In general, annotations are done on the clause level, where clauses are arbitrary substrings of the summary. As we classify and evaluate on sentences produced by Stanford CoreNLP (Manning et al., 2014), this introduces noise.

Annotations aren't done consistently throughout the corpus, and often very different violations are annotated with the same label, the most obvious example of this being shown in Section 3. This is not automatically a weakness of the corpus itself, but rather an unfitness for our purposes. However, there's little to no alternatives available.

A legitimate criticism of our work is that our methods, experiments, and results are tailored to the LQVCorpus or to the summarization systems that participated in the TAC 2011 Guided Summarization task (Owczarzak and Dang, 2011).

6 Conclusion

In this paper, we have presented a solid method to detect the kind of ungrammaticality that occurs in automatically generated extractive multi-document summarizations. While the results are good, they do rely on a certain type of ungrammaticality occuring often (missing spaces). Future work could concentrate on improving other grammaticality violations, so that not only the accuracy rises, but also the generality. Such a system could potentially detect ungrammaticality in a much broader domain.

We have also shown a simple but high-performing method of detecting datelines. We believe our results are close to what a human could achieve and don't see much room for improvement here.

Finally we developed a basic method of detecting redundancy. With the lowest performance of all methods in this paper, redundancy detection is requiring much more work. While we based our approach only on the two subsequent sentences on which redundancy was to be tested, future work could include information available in the context of these sentences or even, as we did in $\mathbf{UnknownTokens_{source}}$, information from the source documents.

We only looked at a part of the violation types in Friedrich et al. (2014). Methods for detecting the last clause-level violation (no semantic relation), as well as the entity- and discourse-level violations have yet to be developed.

For the entity-level violations, one could use the output of a coreference resolution tool such as BART (Versley et al., 2008) or the Stanford coreference resolution system (Lee et al., 2011) to infer coreferences between entities in the summaries as well as the source documents and compare the originating coreference chains.

Along with this paper, the authors will provide a toolchain that can be used to annotate given documents with the methods described in here.

Acknowledgments

?

References

Bender, E. M., Flickinger, D., Oepen, S., Walsh, A., and Baldwin, T. (2004). Arboretum: Using a precision grammar for grammar checking in call. In *InSTIL/ICALL Symposium 2004*.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Chodorow, M. and Leacock, C. (2000). An unsupervised method for detecting grammatical errors. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 140–147. Association for Computational Linguistics.

Copestake, A. (2002). *Implementing typed feature structure grammars*, volume 110. CSLI publications Stanford.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Flickinger, D. (2000). On building a more effcient grammar by exploiting types. *Natural Language Engineering*, 6(01):15–28.

Friedrich, A., Valeeva, M., and Palmer, A. (2014). Lqvsumm: A corpus of linguistic quality violations in multi-document summarization.

Graff, D., Kong, J., Chen, K., and Maeda, K. (2003). English gigaword. *Linguistic Data Consortium, Philadelphia*.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M., and Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34. Association for Computational Linguistics.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

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

Maxwell, J. and Kaplan, R. (1996). An efficient parser for lfg. In *Proceedings of LFG*, volume 96, page 131.

Nenkova, A., McKeown, K., et al. (2011). Foundations and trends® in information retrieval. *Foundations and Trends® in Information Retrieval*, 5(2-3):103–233.

Owczarzak, K. and Dang, H. T. (2011). Overview of the tac 2011 summarization track: Guided task and aesop task. In *Proceedings of the Text Analysis Conference (TAC 2011), Gaithersburg, Maryland, USA, November*.

Packart, W. (2011). ACE: the Answer Constraint Engine. http://sweaglesw.org/linguistics/ace/. Accessed: 2014-08-20.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Schneider, D. and McCoy, K. F. (1998). Recognizing syntactic errors in the writing of second language learners. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1198–1204. Association for Computational Linguistics.

Stolcke, A. et al. (2002). Srilm-an extensible language modeling toolkit. In *INTERSPEECH*.

Sun, G., Liu, X., Cong, G., Zhou, M., Xiong, Z., Lee, J., and Lin, C.-Y. (2007). Detecting erroneous sentences using automatically mined sequential patterns. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 81. Citeseer.

Vadlapudi, R. and Katragadda, R. (2010). On automated evaluation of readability of summaries: capturing grammaticality, focus, structure and coherence. In *Proceedings of the NAACL HLT 2010 Student Research Workshop*, pages 7–12. Association for Computational Linguistics.

Valeeva, M. (2013). Annotation of factors of linguistic quality for multi-document summarization.

Versley, Y., Ponzetto, S. P., Poesio, M., Eidelman, V., Jern, A., Smith, J., Yang, X., and Moschitti, A. (2008). Bart: A modular toolkit for coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session*, pages 9–12. Association for Computational Linguistics.

Wagner, J., Foster, J., and van Genabith, J. (2007). A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. Association for Computational Linguistics.