Automatic Detection of Linguistic Quality Violations

Jonathan Oberländer

1st August 2014

1 Introduction

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*.

2 Related Work

3 Finding Ungrammaticality

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.

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 for sentence splitting.

Comment: besser?

4.1 Detecting Unknown Tokens

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 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

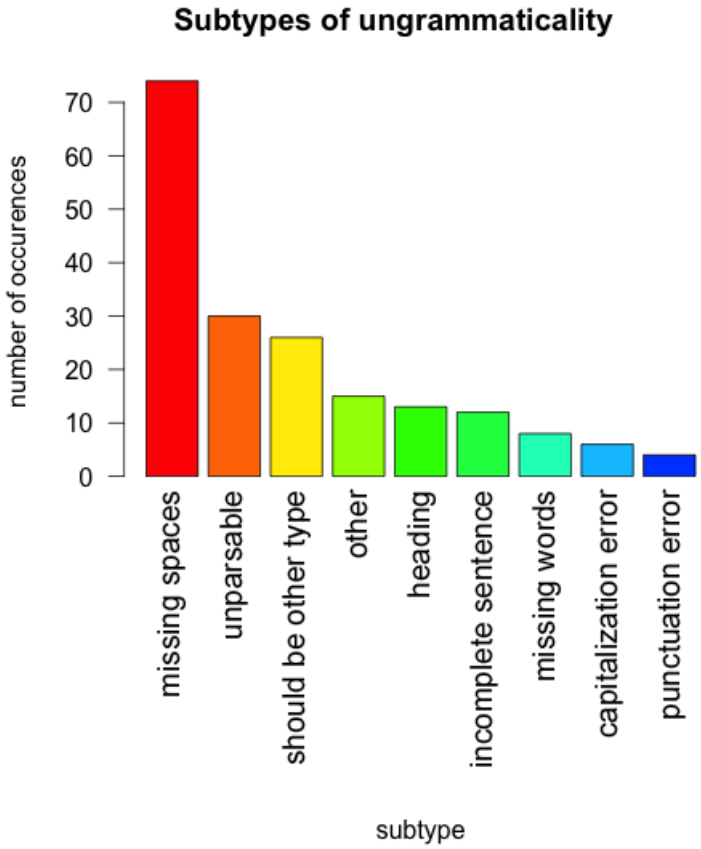

Figure 1: **Subtypes of ungrammaticality in *dev-2*.** Where the annotator didn't know what subtype to annotate the sentence with, they selected "other".

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 ($\textbf{UnknownTokens}_{\textbf{wn}}$); 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 $\textbf{UnknownTokens}_{\textbf{wn+heur+ner+wiki}}$ is what we call $\textbf{UnknownTokens}_{\textbf{general}}$.

4.1.2 Evaluation

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 **UnknownTokens$_{\text{general}}$** and **UnknownTokens$_{\text{source}}$** improve vastly over the baseline.

4.2 DecisionTree

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.

4.2.1 Method

The free WEKA software (Hall et al., 2009) is an allround machine learning toolkit.

We provide three features for WEKA:

Language Model is a numeric feature, the surprisal value for a sentence, returned by SRILM (Stolcke et al., 2002), a language modeling toolkit, trained on the same Gigaword corpus we used as a word source in **UnknownTokens$_{\text{general}}$** (section 4.1).

missing spaces is the classification of **UnknownTokens$_{\text{source}}$** for a given sentence, as described above.

Finally, **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.

4.2.2 Evaluation

We performed 10-fold cross-validation using the WEKA Scheme *RandomForest* , which lead to a weighted F-score of 86.3%. The summarized evaluation can be seen in Table 2.

Comment: cite & gehört zu method

4.3 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.

4.3.1 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

	Missing spaces			No missing spaces			Micro-average	Macro-average		
	P	**R**	**F**	**P**	**R**	**F**	**F**	**P**	**R**	**F**
Baseline	0.0	0.0	0.0	94.79	100	97.33	94.79	47.40	50.0	48.67
$\mathbf{UT_{wn}}$	14.96	98.65	25.98	99.89	69.14	81.72	70.68	57.43	83.90	68.19
$\mathbf{UT_{wn+heur}}$	30.51	97.30	46.45	99.83	87.81	93.43	88.30	65.17	92.56	76.49
$\mathbf{UT_{wn+heur+ner}}$	35.47	97.30	51.99	99.84	90.26	94.81	90.63	67.66	93.78	78.61
$\mathbf{UT_{wn+heur+ner+wiki}}$	70.30	95.95	81.15	99.77	97.77	98.76	97.67	85.04	96.86	90.57
$\mathbf{UT_{source}}$	95.89	94.59	95.24	99.70	99.70	99.70	99.50	97.80	97.15	97.47

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

Correctly Classified Instances	1215 (88.1073%)
Incorrectly Classified Instances	164 (11.8927%)
Kappa statistic	0.3351
Mean absolute error	0.1491
Root mean squared error	0.3094
Relative absolute error	65.2325%
Root relative squared error	91.6119%
Coverage of cases (0.95 level)	96.2292%
Mean rel. region size (0.95 level)	66.4612%
Total Number of Instances	1379

Table 2: Evaluation of the DecisionTree method, as reported by WEKA, using 10-fold cross-validation and the RandomForest scheme. TODO add to Table 1

```
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.

4.3.2 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.

References

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.

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.

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*.

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

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

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.