Towards the automatic detection of syntactic differences

Kroon, M.S.

CHAPTER 2

A filter for syntactically incomparable parallel sentences

A version of this chapter was published as:

Kroon, M., Barbiers, S., Odijk, J., & van der Pas, S. (2019). A filter for syntactically incomparable parallel sentences. *Linguistics in the Netherlands, 36*, 147-161. `https://doi.org/10.1075/avt.00029.kro`

Author contributions: MK, SB, JO and SvdP conceptualized the research; MK designed and wrote the tools; MK and SvdP labelled the data; MK analyzed the data and wrote the paper; SB, JO and SvdP supervised and critically reviewed the research.

Abstract

Massive automatic comparison of languages in parallel corpora will greatly speed up and enhance comparative syntactic research. Automatically extracting and mining syntactic differences from parallel corpora requires a pre-processing step that filters out sentence pairs that cannot be compared syntactically, for example because they involve "free" translations. In this paper we explore four possible filters: the Damerau-Levenshtein distance between POS-tags, the sentence-length ratio, the graph-edit distance between dependency parses, and a combination of the three in a logistic regression model. Results suggest that the dependency-parse filter is the most stable throughout language pairs, while the combination filter achieves the best results.

2.1 Introduction

An important goal of comparative syntactic research is to identify the syntactic differences between languages and the correlations between these differences. This should lead to an explanation of the locus and limits of syntactic variation (cf. Barbiers 2009). Massive automatic syntactic comparison of languages will greatly speed up and enhance this research. This is necessary given the enormous number of language varieties and syntactic variables involved. Parallel corpora such as Europarl (Koehn 2005), containing sentence aligned versions of the proceedings of the European Parliament in 21 languages, provide excellent data for automatic syntactic comparison.[1] The advantage of using a parallel corpus over a non-parallel corpus for this goal is that in a parallel corpus one can also identify in which contexts the differences occur, whereas in a non-parallel corpus one can only identify quantitative differences (cf. Wiersma, Nerbonne and Lauttamus 2011 for an example of the latter).

We are developing a pipeline to make automatic syntactic comparison of parallel sentences possible, as part of the *DeSDA* project.[2] The first step in this pipeline is to filter out syntactically incomparable parallel sentences. Steps two and three include the extraction of syntactic differences from the remaining sentences and the application of data mining techniques to discover possible correlations. This paper describes and evaluates the first step.

When extracting syntactic differences from parallel corpora, it is essential only to compare sentence pairs that are syntactically sufficiently similar. A method and measure is needed to filter out sentence pairs that are syntactically too different, such as “free” translations. Any extracted differences from too dissimilar sentence pairs will lead to noisy and uninterpretable results. In other works, researchers manually discard incorrect translations or those that are too free (among others, van der Klis, Le Bruyn and De Swart 2017; Abzianidze et al. 2017), whereas we aim for the automatization of the task, which, to the best of our knowledge, has not been attempted before.

In this paper we present four ways to automatically filter out parallel sentence pairs that are not sufficiently similar syntactically,[3] and report on experiments using these filters on manually labelled datasets of English, Dutch and German parallel sentences taken from the Europarl corpus – one filter based on the Levenshtein distance (Levenshtein 1966), one on the sentence-length ratio, one on the graph-edit distance between dependency parses and one that combines the other three in a logistic regression model.

All four filters use a threshold value beyond which a sentence pair is filtered out. In the case of the combination filter, this threshold is automatically derived from the logistic regression model, which is trained in a supervised manner, therefore requiring a gold standard dataset of labelled sentence pairs. The

[1] `http://opus.nlpl.eu/`

[2] `https://www.universiteitleiden.nl/en/humanities/centre-for-digital-humanities/projects/past-lucdh-research-projects#outline-of-the-desda-project`

[3] The code is made available on `https://github.com/mskroon/DeSDA`

other three filters can either take a manually set threshold, or find an optimal threshold value through supervised learning (see Section 2.4.5).

Syntactic comparability is hard to define and also depends on the research goals. We shall therefore first discuss this concept in more detail. We then describe the data and the filters, and how they were evaluated. Results are presented thereafter. Finally, we discuss and conclude.

2.2 Syntactic Comparability

It is difficult to define syntactic comparability. The sentence pair (1a,b), from the Europarl parallel corpus, involves "free" translation and is clearly not syntactically comparable.

(1) a. That is what will make us strong.

b. Dan zijn wij sterk.
then are we strong
'Then we are strong.' (Koehn 2005)

A similar problem arises with idiomatic expressions, as in (2). Such cases must be filtered out.

(2) a. ... I hope that this report will not be allowed **to bite the dust** on account of this...

b. ... hoffe ich, dass dieser Bericht nicht deswegen **zu Fall gebracht**
hope I that this report not therefore to fall brought

wird...
is

(Koehn 2005)

Switching a sentence's voice may also cause problems, as in the pair (3a,b). The English fragment shows an active construction, the Dutch fragment a passive one. This example should preferably be filtered out for syntactic research, as there is no reason other than (e.g.) a stylistic preference for changing the voice of the verb: (3b′) shows an equally natural yet active translation.

(3) a. This can double the available resources...

b. Hierdoor kunnen de beschikbare middelen worden verdubbeld...
by.this can the available resources be doubled
Through this, the available resources can be doubled. (Koehn 2005)

b′. Dit kan de beschikbare middelen verdubbelen...
this can the available resources double
This can double the available resources...

In this paper, we consider these kinds of examples to be syntactically incomparable. We realize, however, that other researchers may want to use different constraints and definitions in selecting comparable material. The filters can then still be used by manually or automatically setting a threshold that better suits their wishes.

2.3 Data

In order to evaluate the filters, we compiled a dataset of 400 randomly selected English-German-Dutch sentence triples from the Europarl corpus and labelled for each pair whether its sentences are syntactically comparable. This gave us three datasets of 400 sentence pairs, containing the same sentences so as to ensure that the results would be comparable between different language pairs.

These datasets were all labelled by several annotators. The English-Dutch dataset was annotated by three people, in which the inter-annotator agreement (Fleiss' κ (Fleiss and Cohen 1973) was 0.61). The judgement of the majority was taken as truth. The German datasets were annotated by two people each, with a Cohen's κ (Cohen 1960) of 0.55 and 0.26 for the German-Dutch and the German-English datasets, respectively. For the German sets, labelling as done by the first author was taken as truth.

Before annotation, annotators were given the following rule of thumb, which is in line with our definition of syntactic comparability but contains some language-specific examples:

> Two parallel sentences are considered syntactically comparable if: All content words in sentence A have an alignment with a word in sentence B and all content words in sentence B have an alignment with a word in sentence A, ignoring word order, and there is no voice shift, such as active to passive, an idiomatic construction in one language or a (pseudo-)cleft in one language.

The datasets were somewhat imbalanced. The exact distribution of labels ('Y' for syntactically comparable, 'N' for incomparable) can be found in Table 2.1.

2.4 Filters

In this section, we describe the filters in more detail. Each filter calculates a specific value for every sentence pair. The filters determine on the basis of this

	Y	N
German-English	131	269
German-Dutch	106	294
English-Dutch	173	227

Table 2.1: The distributions of the labels in the three datasets.

value whether to keep the sentence pair or to discard it. They are supervised learners, and learn a threshold value, above which sentence pairs are filtered out, from training data. They can also, however, use manually set thresholds or a pre-trained model.

Given that languages differ particularly in the domain of function words and the goal of comparative syntax is to identify syntactic variation, we give users of the filters the option to automatically ignore specific functional material, as based on the words' POS tags.

2.4.1 Levenshtein distance on POS-tags

Using the Levenshtein distance (Levenshtein 1966) on POS tags, which represent morphosyntactic properties of word tokens in context, is a simple approach to filtering for syntactic comparability. The Levenshtein distance is the minimum number of edit operations (in terms of insertion, deletion and substitution) needed to change one sequence into the other, e.g. DET NOUN to ADJ NOUN requires one substitution of DET to ADJ (hence the Levenshtein distance is 1). Intuitively, if the Levenshtein distance between two sentences is low, the sentences are probably syntactically comparable − if it is high, they probably are not.

Users can use the Damerau-Levenshtein distance (Bard 2007) as opposed to the classic Levenshtein distance, adding transpositions to the allowable operations. This will yield lower edit distances between a language where adjectives are prenominal and a language where they are postnominal, for example.

A weakness of the (Damerau-)Levenshtein distance is its sensitivity to whole constituents or phrases moving around. In a comparison of an SVO and an SOV language, the threshold will likely have to be very high to find syntactically comparable sentences, but at the same time having a high threshold will lead to many undesirable sentence pairs not being filtered out. For example, the sentence pair in Figure 2.1 yields a (Damerau-)Levenshtein distance of 4, while if it knew that the object phrase and the verbal cluster were transposed as a whole, the Damerau-Levenshtein distance would only be 2 (1 transposition of the phrases and 1 transposition between AUX VERB–VERB AUX).

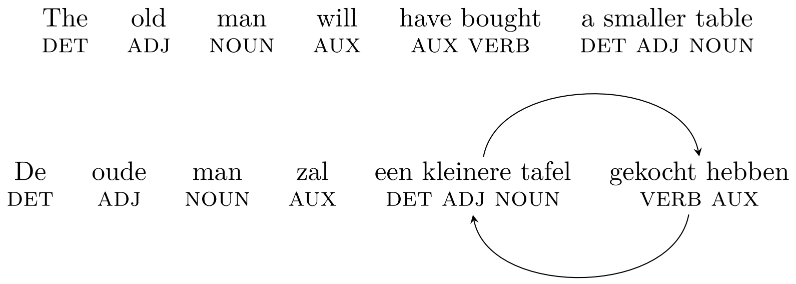

Figure 2.1: The Levenshtein distance is very sensitive to transposing phrases.

2.4.2 Sentence-length ratio

Filtering based on sentence-length ratio is another simple approach: if the sentence-length ratio of a sentence pair is very high or very low, they probably are not syntactically comparable. The sentence-length ratio is defined as the number of words in the source sentence divided by the number of words in the target sentence.

However, some languages use fewer words, for example because they are highly inflectional or do not have articles. Therefore, a language-pair specific threshold is defined in terms of percentiles, where the $n\%$ most extreme sentence-length ratios (relative to the median sentence-length ratio of a language pair) are considered syntactically incomparable – i.e. the left and right tails of the histogram in Figure 2.2 are filtered out. Note that the percentile-based cut-off allows for asymmetric decision rules, where e.g. a sentence is incomparable if it is twice as long in A as in B or three times as long in B as in A.

A sentence-length ratio-based filter is computationally cheap but it does not use any syntactic information, making it very coarse-grained: e.g., the English-Dutch pair *The next item is the vote.–Wij gaan over tot de stemming.* is syntactically incomparable, but the sentences have the same number of words and will not be filtered out, since syntactic information is not taken into account.

Very short sentences are another concern. The pair in (4) is syntactically comparable, but the Dutch and Italian sentences have a sentence-length ratio of 2. A ratio of 2 probably indicates syntactic incomparability when comparing two sentences with 12 and 6 words. This potential issue can be remedied by ignoring function words such as pronouns.

(4) a. ik eet
b. mangio
'I eat.'

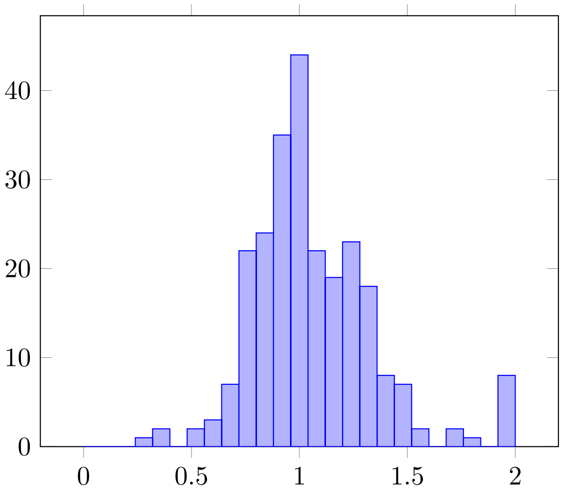

Figure 2.2: A histogram for sentence-length ratios for English and Dutch. The left and right tail are filtered out. If the threshold is, e.g., 10%, the cut-offs would be 0.74 and 1.38: sentence pairs where the English sentence is more than 1.38 times as long or less than 0.74 times as long as the Dutch counterpart will be discarded.

2.4.3 Graph edit distance on dependency trees

Whereas the Levenshtein distance calculates an edit distance between two linear sequences, a graph edit distance (GED) can be applied to hierarchically structured graphs. This has the benefit that it is insensitive to phrases or constituents transposing.

The filter applies Abu-Aisheh et al.'s (2015) exact GED algorithm on dependency parses, where the parses are represented as unordered directed trees (as implemented in `networkx`, Hagberg, Schult and Swart 2008) with labelled edges from heads to dependencies (cf. Figure 2.3). Importantly, both languages should use the same tag set. Nodes, i.e. words, are considered equal if they have the same POS tag; edges, i.e. syntactic relations, are considered equal if they have the same label. Node and edge insertion, deletion and substitution are all defined as 1.

The fact that graphs are unordered should make the algorithm more robust between different languages and language families, as it ignores the linear order between any two words, irrespective of whether these have a grammatical relation between them. The linear order between a word, phrase or constituent and its head is also not represented. Consequently, it is unimportant for the GED whether a direct object is on the left or right of its (head) verb, nor is it important what the linear order of the subject and the object is: transposition costs between sister nodes are therefore 0. This makes it easier to correctly clas-

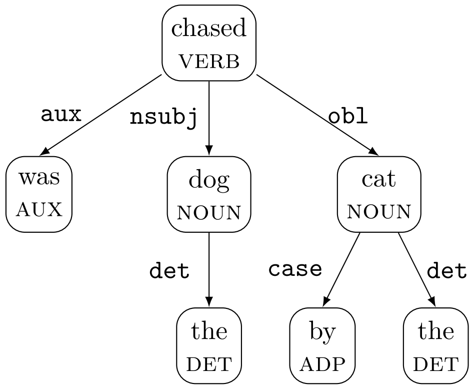

Figure 2.3: An example of a dependency parse (in Universal Dependencies) as an unordered directed tree. Every edge is labelled and directed and the surface order of the words is not represented in the graph.

sify sentence pairs between SOV and SVO (or other) languages as syntactically comparable.[4]

We provide users with the option to ignore function words, similar to the other filters. Importantly, to-be-ignored POS tags are removed from the graph. If a node has children nodes, the edge leading to it is contracted. Any edge leading from the removed POS tag now leads directly from its head to its children (cf. Figure 2.4). The root of the sentence is never removed, sentences always remain one connected component.

Although this approach uses syntactic structure to filter for syntactic comparability and is insensitive to phrase transposition, it is sensitive to parse accuracy. If a dependency parse is erroneous or even slightly off, it will influence the filter as it will yield noisy GED values. Apart from that, it requires the existence of a dependency parser for both languages, which must use the same annotation guidelines and tag set. In our experiments the filter tags and parses sentence pairs using UDPipe (Straka and Straková 2017).

2.4.4 Combination filter

Filtering is essentially a binary classification task. The final filter therefore combines the other filters by fitting a logistic regression model on a pre-labelled dataset of parallel sentence pairs. Each pair is binarily labelled as syntactically (in)comparable. The values calculated by the other filters are then the *features*. Given a pre-labelled dataset all sentence pairs are passed to the other filters,

[4] The linear order of nodes in the tree is only important when discovering syntactic differences, but since this filter is designed only to select sentence pairs from which to extract syntactic differences in a later stage, the linear order can (and should) be ignored by the filter.

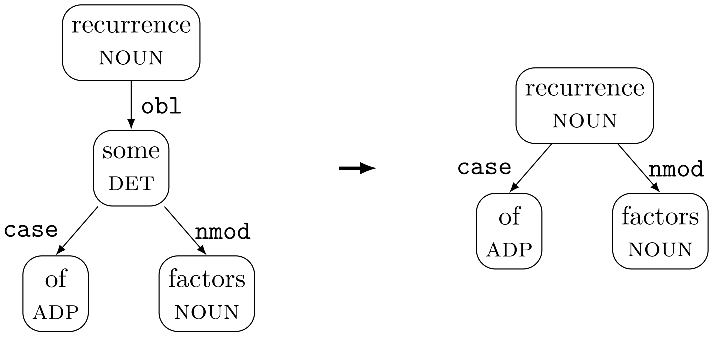

Figure 2.4: An example of contracting edges when DETs are removed. The graphs represent a fragment of *the recurrence of some of these factors*, as found in the Europarl corpus and parsed in UDPipe (Straka and Straková 2017).

which calculate a value. These values are then passed back to a logistic regressor, combined with the labels, to fit a model. This model can be used to calculate the probability of a sentence pair either being or not being syntactically comparable, and to predict a sentence pair's syntactic comparability – if the calculated probability that the sentence pair is syntactically comparable is too low, it will be filtered out. This filter has a clear drawback in that it must have a pre-labelled dataset, which is not always available.

The user can choose which other filters to use for the feature calculation. For every filter, users can select the same options described above. Importantly, ignored functional material need not be the same for each filter.

2.4.5 Automatically setting a threshold

Threshold values can be automatically set with a pre-labelled dataset of sentence pairs using a receiver operating characteristic curve (ROC curve), which plots the true positive rate against the false positive rate at various threshold values – true positive are sentence pairs correctly labelled as syntactically comparable; false positive are those incorrectly not filtered out. The threshold value is found where there is a compromise between the false positive rate and the true positive rate, which can be calculated with Youden's J statistic (Youden 1950).

2.5 Evaluation of the filters

We evaluate the filters on all three datasets, and use them as test sets in order to assess the filters' performance. The sentences used for evaluation were POS tagged and parsed in Universal Dependencies (UD)[5] – a programme that aims

[5] http://universaldependencies.org/

at cross-linguistically consistent tagging and annotation of dependency trees (Nivre et al. 2016) – so that all languages used the same tag set. Tagging and parsing was done automatically using UDPipe (Straka and Straková 2017).[6] Aiming for cross-linguistic consistency, UD defines a handful of coarse-grained POS tags that only capture a word's category; morphological information is not included in these tags.[7]

The filters were tested in all possible set-ups and compared to a baseline, which was a bare Levenshtein distance on POS tags: not ignoring any functional material, no transpositions allowed.

When ignoring functional material, we tested all combinations of closed set POS tags of the UD programme,[8] to see which subset of functional POS tags would render the best results when ignored, because the subset of POS tags that are to be ignored is dependent on the language pair in question.

The combination filter was tested by fitting models for all possible combinations of two or three filters in every set-up.

The filters were evaluated in terms of the area under the ROC curve (AUC). For the combination filters, a ROC curve was plotted with the calculated probabilities as threshold values in order to make the results comparable.

2.6 Results

The baselines performed with an AUC of 0.74 on the German-English and English-Dutch sets and with an AUC of 0.76 on the German-Dutch set. Its best thresholds (as found with Youden's J statistic) were 10, 7 and 5, respectively.

Runs with the non-bare Levenshtein distance filters don't clearly outperform the baseline, achieving only slightly higher AUCs. The best runs were also rather divergent in their parameter settings and thresholds.

In general it was observed that the sentence-length filter performed significantly worse, with AUCs of on average about 0.05 lower than the baseline. Ignored functional material differed greatly between the datasets.

The GED filter also did not clearly outperform the baseline in case of German-English and German-Dutch. However, on the English-Dutch dataset it performed somewhat better, with an AUC of 0.77.

More striking, however, is that it is more consistent in its parameter settings throughout the datasets. All best runs included all functional material. Also the threshold was consistent, discarding all sentence pairs with more than 4 edits.

The AUC and parameter settings are summarized in Table 2.3.

[6] `https://github.com/ufal/udpipe`

[7] Although UDPipe also does morphological tagging in the form of attributes, we only used the coarse-grained POS tags in our evaluation. The set of morphological features used by the three languages was too heterogeneous to achieve satisfying results.

[8] UD defines eight POS-tags as being in the closed set: ADP (adpositions), AUX (auxiliaries), CCONJ (coordinating conjunctions), DET (determiners), NUM (numerals), PART (particles), PRON (pronouns) and SCONJ (subordinating conjunctions).

	German-English			German-Dutch			English-Dutch		
	Lev.	**Sent. length**	**GED**	**Lev.**	**Sent. length**	**GED**	**Lev.**	**Sent. length**	**GED**
AUC	0.75	0.66	0.75	0.77	0.68	0.75	0.74	0.73	0.77
Threshold	9	24%	4	5	24.375%	4	7	20.625%	4
Ignored functional material	AUX, CCONJ, NUM	ADP, NUM	–	ADP, AUX, CCONJ, NUM, PART	SCONJ	–	AUX, CCONJ, NUM	AUX, NUM	–
Transpositions	No	–	–	No	–	–	Yes	–	–

Table 2.3: Overview of the results of the filters: AUC, and parameters per language pair.

The combination filter more clearly outperformed the baseline, with AUCs of on average about 0.06 higher than the baseline. It benefited from using all other filters, all best runs using all three single filters, though interestingly with different parameters.

In the best German-English run, the Levenshtein filter did not use transpositions, and ignored CCONJ, NUM, and PART instead of AUX. The sentence-length filter ignored ADP, but no NUM. The GED filter ignored NUM and SCONJ. In this setup, it achieved an AUC of 0.79.

As for the German-Dutch dataset, the best run, with an AUC of 0.80, was achieved by combining a Levenshtein filter that ignores ADP, CCONJ, NUM, PART and SCONJ and allows transpositions, a sentence-length filter that ignores all functional material but NUM, and a GED filter that ignores CCONJ.

Finally, the best run for the English-Dutch dataset achieved an AUC of 0.81, combining a transposing, CCONJ ignoring Levenshtein filter, a sentence-length filter that ignores nothing and a GED filter that ignores ADP, DET, NUM and SCONJ.

2.7 Discussion

The results suggest chiefly that filtering for syntactic comparability is a hard task, as corroborated by the annotations' relatively low κ values. Nevertheless, we believe that the presented filters are useful tools for automatizing the selection of syntactically comparable sentences from a parallel corpus, especially since it allows users to manually set thresholds and parameters and to work with other definitions of syntactic comparability.

The results further suggest that German, English and Dutch are rather similar syntactically. If not, we would have expected a larger performance gap between the GED and Levenshtein filters, due to the Levenshtein distance's sensitivity to constituents or phrases transposing. On the other hand, the difference in parameters between the Levenshtein runs does point towards syntactic dissimilarity of the languages, since, if the languages were more similar, the sets of ignored function words would have been smaller.

The filters' sensitivity weakly suggest that there is a syntactic difference to be found in the use of auxiliaries, adpositions and conjunctions. The fact that numerals are often ignored can be explained by the difference in how numerals are tagged by UDPipe: in English and German ordinal numerals are often tagged as adjectives (e.g. *second*) or adverbs (e.g. *thirdly*) – which are both rather frequent in the Europarl corpus – whereas in Dutch they are always tagged as numerals. This emphasizes the importance of uniform tagging conventions between compared languages. Although UD aims for consistent tagging, there are subtle differences from language to language. These differences, however subtle, lead to issues for our filters.

Overall, the best filter is the combination filter. It necessitates, though, the existence of a pre-labelled dataset – and if such a dataset is available, doing a grid search to find which parameters yield the best results is computationally expensive. Also, the risk of overfitting on the dataset is high.

If a pre-labelled dataset is not available, the other filters can still be used with reasonable results by setting thresholds manually. While the baseline is not clearly outperformed by the other filters, the GED filter's robustness in its parameters, thresholds and performance throughout the different language pairs suggests that it is most stable in all aspects. Its parameter robustness even suggests that the settings found could be used as a default for other language pairs. We also expect it to outperform other approaches more clearly when supplied with more accurate parses.

The Levenshtein filter performs similarly, but has the advantage of not requiring a parser model. If such a model is not available, the Levenshtein filter could still be used, but it is likely to perform well on closely related languages only and requires more parameter fine-tuning.

The sentence-length filter did not give satisfying results, as expected since it does not use any syntactic information. Interestingly, the combination filter did use sentence-length ratio. This makes sense, as using the sentence-length ratio to filter out the most extreme sentence pairs allows for the model to more finely tune the weights for the Levenshtein distance and GED, yielding more informed decisions.

A point of improvement could be the edge contraction conventions when nodes are removed in the GED; in our current design, the `obl` relation in Figure 2.4 is lost entirely, which may be undesirable.

It will be most useful to improve the filter such that it also selects sentence *fragments* that are syntactically comparable. Now an entire sentence can be filtered out despite being almost completely syntactically comparable. A possible design of such a filter is top-down: if a sentence-pair exceeds the edit threshold, the algorithm can search for two pairs of maximally large subtrees that do not exceed the threshold.

The way the combination filter operates could perhaps be improved upon, too. Now the other filters are combined in parallel. A sequential fashion may yield better results, discarding sentence pairs that exceed some sentence-length ratio before optimizing a threshold for the GED filter, for example.

2.8 Conclusion

Automatic extraction of cross-linguistic syntactic differences from parallel corpora will greatly speed up comparative syntactic research. Automatic extraction requires a pre-processing step to filter out syntactically incomparable sentence pairs, e.g., because they involve “free” translations. In this paper we evaluated four possible filters. The best results were achieved with a filter that combines the other three in a regression model, but it has the downside of requiring a pre-labelled training set, more so than the others which allow for manual tuning. Alternatively our filter based on the Levenshtein distance or our GED filter can be used to achieve reasonable results, but both have their own weaknesses. Our last filter, based on sentence length, did not achieve satisfying results in itself, as expected.