March 2015 Technical Report: KUL/ESAT/PSI/1501

Extending n-gram language models based on syntactic equivalences

Lyan Verwimp, Joris Pelemans,
Hugo Van hamme, Patrick
Wambacq

PSI

Kasteelpark Arenberg 10 – box 2441
B-3001 Heverlee, Belgium

KU LEUVEN

Telephone: +32-(0)16-32.17.13 Fax: +32-(0)16-32.17.23 http://www.esat.kuleuven.be/psi

Extending n-gram language models based on syntactic equivalences

1. Introduction

Language models are used in many areas of natural language processing. In speech recognition, *n-gram* language models are used to predict the probability that a certain word occurs given $n - 1$ previous words. However, n-gram models have some drawbacks: it is impossible to encode every possible word sequence that can occur in a language ("data sparsity"), and not every dependency can be captured within a window of five words (five is usually the maximum number of n that is used). In our work, we will try to deal with the former problem by adding n-grams to an existing language model in a syntactically motivated way.

In Dutch (and other languages), there exist combinations of syntactic relations that can occur in different word orders. For example, the order of subject + verb (SV) in a regular head clause is reversed in a question or if the first position of the sentence is already occupied by another element ("inversion", VS):

1. *Hij eet een boterham.* "He eats a sandwich."
2. *Eet hij een boterham?* "Does he eat a sandwich?"
3. *Daarom eet hij een boterham.* "That is why he eats a sandwich".

Another example is the combination of a verb and direct object: in a head clause the verb comes first (VO), whereas in a subordinate clause it comes last (OV).

4. *Hij schrijft een boek.* "He writes a book."
5. *Het klopt dat hij een boek schrijft.* "It is correct that he writes a book."

A last example are conjunctions: different parts of the conjunctions can be switched:

6. *De jongen en het meisje gingen wandelen.* "The boy and the girl went for a walk."
7. *Het meisje en de jongen gingen wandelen.* "The girl and the boy went for a walk."

In many other languages, there exist patterns in which the word order can be changed. For some of them, the word order is free such that it can always be changed, whereas for others, it depends on the (syntactic) context which word order occurs (see Dryer 2013a for an overview of the word order of SV, Dryer 2013b for OV and other chapters on word order on http://wals.info/chapter).

The purpose of this work is to automatically generate more n-grams based on the same amount of training text, by searching for such patterns that can be reversed, and adding them to the language model if they were not yet present. Since these patterns were not yet present, the probability assigned by the language model would otherwise be based on back-off probabilities. By adding the higher order n-gram to the language model, we have a more reliable estimate of its probability. For example, if there is some doubt about whether the word following *hij schrijft een* "he writes a" is *boek* "book" or *doek* "canvas", and if the difference between the unigram probabilities of *boek* and *doek* is very small, adding the n-gram *schrijft een boek* (the result of reversing the order of *een boek schrijft* "writes a book") would increase the probability of *boek* in this context. In this way, we are able to partially alleviate the data sparsity problem of n-gram language models. Moreover, since our language models are based upon written language but are

meant to be used for the recognition of spoken language, we would like to capture the differences in frequency of sentence types in both kinds of language. For example, it is natural that subordinate clauses occur more frequently in written language, and that questions occur more frequently in spoken language (where there is more interaction between people). By for instance reversing subject – verb patterns, we can generate more data that is suitable for (likely to occur in) spoken language recognition.

There exist some n-gram language models that use syntactic (parsing) information (Chelba 1997, Chelba et al. 1997, Gubbins and Vlachos 2013, Pauls and Klein 2012, Sidorov et al. 2014, Zhang and Wang 2014), but in these works the test set is also parsed. In our work, we only parse the training set and use the syntactic information we obtain to generate n-grams, so that our model is still an n-gram model, which can be evaluated quickly.

In what follows, we will first explain briefly the methods, data sets and tools we used (section 2), then present the results (section 3) and finally a conclusion (section 4).

2. Methodology

In this section, we will first mention which data sets and tools we used for our research (section 2.1). After that, we will explain how we created our adapted language models (section 2.2) and give some problems that this approach might encounter (section 2.3).

2.1 Data sets and tools

Unless we specify otherwise, the training and test sets we used will always be the same: as a training set, we used normalized extracts from *De Standaard* (a Flemish newspaper), which contain approximately 128M word tokens. This corpus was parsed with *Alpino*, a dependency parser for Dutch (Van Noord 2006). As test set for perplexity evaluations, we used extracts from another Flemish newspaper, *De Morgen* (containing ca. 135M word tokens). To create count files and language models and to evaluate the perplexity of the language models, we used the *SRILM* toolkit (Stolcke 2002).

For speech recognition experiments, we tested on part of component o (read speech) of the *Corpus of Spoken Dutch* (CGN) and made use of the *SPRAAK* recognizer (Demuynck et al. 2008). We always used a vocabulary containing the 100 000 most frequent words in *De Standaard*, and the default smoothing is modified Kneser-Ney (Chen and Goodman 1999).

2.2 Method

How did we search for patterns in which the word order can be reversed? First, we parsed our training set with *Alpino*. The output of the syntactic parser consists of an XML-file for each sentence, encoding a dependency parse. Each node contains a lot of information such as the POS-tag, the dependency relation, the lemma, the ID of the word, the beginning and end ID of the child nodes (if it has some), other specific information depending on the POS-tag (e.g. “case” if it is a substantive), etc. From these XML-files, we extracted only the words themselves, their POS-tags and the dependency relations with the aid of *lxml* (a Python library for parsing among others XML). We also extracted the dependency relations of the parent nodes because the relation of constituents consisting of more than one word is encoded in the parent node and not in the child nodes (e.g. the parent node of *een boek* has the relation “obj1” (direct object), but the nodes of *een* and *boek* do

not). We used the POS-tag and the dependency relation to search for the correct patterns: e.g. a word or constituent with the relation “subj” (subject) followed by a word with the relation “hd” (head) that has a verbal POS-tag. If such a pattern was found, the word order was reversed. These reversed n-grams can be added to a count file or to a language model.

2.3 Issues

As we already mentioned, there are some issues that might (and indeed did) arise when using this approach. The first (and the most important) issue is the frequency (if you add them to the count file) or probability (if you add them to the language model) that these new n-grams should get. If we add them to the count file, we can give them the same frequency as their original n-gram (so the total count increases), give them a fraction of the frequency of the original n-gram (the total count increases, but not so much as for the first approach) or we can distribute the count of the original n-gram over the original and the new one (the frequency of the original n-gram changes and the total count remains the same). Adding the reversed n-grams directly to the language model is more difficult, since we would have to normalize to ensure that the probabilities still sum up to one.

Although the dependency parsing nicely indicates the direct relations between constituents, it can also cause some problems. Firstly (and naturally), the parses are not always correct:

8. *Over de vluchtroute is voorlopig niets bekend.* “For now, nothing is known about the air route.”

In the above sentence, the parser incorrectly classifies *voorlopig niets* as the subject of the sentence. Moreover, since dependency parsing puts the head of a phrase in front and since the head of a sentence is the finite verb, *is* is moved to the front in the parse:

9. *is voorlopig niets over de vluchtroute bekend*

These word order changes occur quite frequently because in Dutch, the finite verb of a subordinate clause is always at the end of the clause. There are three possible approaches that one could try concerning the word order changes: you could either add all reversed n-grams, only add them if the corresponding original n-gram did not change in word order (due to the parsing) or even add all new n-grams that were found (so also the “original” n-grams in which the word order changed). We tried all three approaches, although the first and the third one are likely to introduce ungrammatical n-grams. Since the results of all three approaches were quite similar, we decided to include only the results for the second approach (only add reversed n-grams if the corresponding original one did not change in word order) in this work.

Another issue which occurs specifically for Dutch (but also for e.g. German and Afrikaans), are separable verbs. In a head clause, a separable verb is split: the finite verb occurs before the object and its particle after the object:

10. *Zij lossen dat op*. “They fix it.”

In a subordinate clause on the other hand, the verb occurs in a final position and is pronounced and written as one word:

11. *Het klopt dat zij dat oplossen*. “It is correct that they fix it.”

If one would reverse the word order of an OV pattern without splitting the verb, one would get an ungrammatical n-gram: **oplossen dat*. So ideally, the separable verbs are detected and either split (OV --> VO) or merged (VO --> OV). The *Alpino* parser however, only marks a verb as separable if it is already separated: the particle then gets the special tag "svp" (separable verb particle). The VO n-grams that are found can thus be reversed and corrected by merging the particle with the finite verb. For the OV patterns however, we need another strategy to detect whether the verb is separable or not.

Finally, the conjunctions that our script detected were not included in the language model. Many of the conjunctions were longer than 5 words (the maximum *n* we used): 50 789 out of a total of 199 411 n-grams were 5 words or shorter. Moreover, if we randomly select some of the n-grams, many errors occur:

12. *twee duizend en twee* "two thousand and two"
    -> ? *twee en twee duizend*
13. *voor eens en voor altijd* "for once and for all"
    -> ? *voor altijd en voor eens*
14. *het waren maar enkele jongeren* "it were only a couple of youngsters"
    -> **enkele jongeren maar het waren*

Numbers containing *en* are interpreted as conjunctions by the parser and reversing the parts leads to word sequences that are less acceptable and infrequent[1] (12). The same[2] applies to reversing the parts of fixed phrases (13), and the parser often wrongly interprets *maar* as a conjunction. That is why we decided not to include the conjunctions in our language model.

3. Results

3.1 Perplexity experiments

Most of our adapted language models were made by adding the new reversed n-grams to an existing count file and creating the language model based upon the new count file. The perplexity of this new language model on our test was then compared with a baseline language model (section 3.1.1). We also interpolated the baseline language model with either the adapted language model or a language model trained solely upon the new n-grams (section 3.1.2). Finally, we will discuss the possibility of adding the new n-grams directly to the language model (3.1.3).

3.1.1 Add n-grams to count file

As we already explained (see section 2.3), it is not certain which strategy is better: changing the count or probability of the original n-gram, or not changing it. We will first present the results of language models in which the original n-gram had the same frequency as before (section 3.1.1.1), and then for language models for which we did change the frequency of the original n-gram (section 3.1.1.2).

3.1.1.1 No change in the frequency of the original n-gram

1 We get only 2 hits at *Google* for the sequence *twee en tweeduizend* (*tweeduizend* is split by the normalization of the training set in order to generalize over numbers: otherwise every possible number would have to be included in the training set): https://www.google.be/?gws_rd=ssl#q=%22twee+en+tweeduizend%22 (hits found on 18/03/2015).

2 No hits were found at *Google* for *voor altijd en voor eens* (18/03/2015).

Since the reversed n-grams were not present in the training set, they will probably not be very frequent in other samples too. So they should either get the same frequency as their corresponding original n-gram, or a lower frequency. We will first present language models in which the new n-gram received the same frequency.

If we compare the baseline language model with one to which the new n-grams are added with the same frequency as their original n-grams, the baseline model performs slightly better:

Perplexity (same count, test set = *De Morgen*)		
	Baseline	Adapted (same count)
Order = 3	**215.803**	216.249
Order = 4	**195.528**	196.095
Order = 5	**192.427**	193.094

We get similar results if we test on a different test set, the Flemish magazine *Knack* (the default test set is *De Morgen*, as explained in section 2.1):

Perplexity (same count, test set = *Knack*)		
	Baseline	Adapted
Order = 3	**245.668**	246.364
Order = 4	**225.588**	226.479
Order = 5	**222.719**	223.739

Since our approach is meant to generate more information based upon the same amount of training text, we would expect it to function better if the training set was smaller. So we limited our training set to approximately 30 % of the complete corpus of *De Standaard*, and our test set to the first 1M lines of *De Morgen*:

Perplexity (same count, training set = ca. 30 % of *De Standaard*, test set = 1M lines of *De Morgen*)		
	Baseline	Adapted
Order = 3	**252.906**	253.101
Order = 4	**237.199**	237.919
Order = 5	**235.258**	236.124

As can be expected, the baseline model performs worse than the one trained on the complete corpus, but the adapted language model still does not improve it. If we reduce the training set until approximately 5 % of *De Standaard* and the test set to 300 000 lines of *De Morgen*, the difference becomes even larger: the baseline model has a perplexity of 745.758 whereas the adapted model has

a perplexity of 751.39.

Since our training corpus consists of newspaper articles, it is likely that questions will be less frequent than in spoken language. That is why we decided to test our adapted language model on a small corpus of questions with inversion: our adapted language model should perform better than the baseline language model because it contains more inversion patterns, created by reversing the word order of sentences with a canonical word order (SV --> VS). We created a small corpus of 247 questions with inversion with the aid of *GrETEL* (*Greedy Extraction of Trees for Empirical Linguistics)*, a tool with which you can create an XPath query based on a natural language example (Augustinus et al. 2012). With this query you can search in treebanks such as *LASSY* (*Large Scale Syntactic Annotation of written Dutch*, Van Noord et al. 2006, Van Noord 2009). We queried for the combination of a finite verb followed by a subject[3]. But also on this test set, our adapted language model performs slightly worse than the baseline:

Perplexity (same count, test set = corpus of inversion questions)		
	Baseline	Adapted
Order = 3	**241.879**	242.36
Order = 4	**230.27**	231.637
Order = 5	**227.623**	229.074

In addition to changing the training and test sets, we did some other experiments. The *SRILM* toolkit offers the possibility to print probability information about separate sentences and words when evaluating a language model. If *-debug 2* is used in combination with *-ppl* for testing the perplexity on a test set, the n-gram that is used to evaluate each word together with its probability is printed. Sometimes this information has to be adapted, because for n-grams with *n* higher than two only two words are given. We completed each n-gram with the aid of the full sentence, which is also given. We then checked which new n-grams were actually used in the evaluation on the test set of *De Morgen*. These n-grams were extracted and added to the baseline count file with the same frequency as their corresponding original n-gram, thus creating a language model to which only relevant n-grams are added. This new model was again tested on *De Morgen*, just to see how much improvement we could get if our approach only added relevant n-grams.

Perplexity (only add n-grams if used in evaluation)		
	Baseline	Adapted
Order = 3	215.803	**215.533**
Order = 4	195.528	**195.519**
Order = 5	192.427	**192.42**

The results of this new model are given in the table above: it shows that the addition of the reversed n-grams causes an extremely small improvement in perplexity. We can conclude that part of the new n-grams contribute something to the language model, but that this contribution is almost negligeable.

3 XPath query: //node[@cat="top" and node[@rel="--" and @cat="sv1" and node[@rel="hd" and @wvorm="pv" and @pos="verb" and number(@begin) < number(../node[@rel="su"]/@begin)] and node[@rel="su" and number(@begin) < ../number(../node[@rel="--" and @word="?" and @root="?" and @pos="punct"]/@begin)]] and node[@rel="--" and @word="?" and @root="?" and @pos="punct"]]

Another strategy we tried to increase the accuracy of our new n-grams, was separating the separable verbs. As we already explained in section 2.3, these verbs cannot be detected if they occur after their direct object. One possible way to detect separable verb is to use a morphological analyzer such as *Frog* (Van den Bosch et al. 2007). However, there also exist verbs in Dutch which consist of multiple morphemes but that are not separable (e.g. *betaal: be-taal* "pay"), so this would wrongly separate non-separable verbs. Another possibility was creating a list of separable verbs and their conjugations: from *Wiktionary* (Wiktionary 2015) we extracted a list of separable verbs in Dutch. Unfortunately, the *Wiktionary* entries of the verbs do not always provide their conjugations, or they sometimes provide conjugations that are obsolete and/or rarely used. That is why we used a list of lemmas and their inflected forms, based upon the *SoNaR* corpus (Oostdijk et al. 2008). If a separable verb in the *Wikipedia* list occurred in this *SoNaR* list, the lemma and all its conjugations together with their separated forms (which were created with the aid of a list of prefixes) were added to a file. We then checked whether the verb in each OV combination occurred in this list and separated the verb if it did. The new (3-gram) language model created with these n-grams did not improve over the baseline: its perplexity was 216.256 compared to 215.803 for the baseline.

Instead of giving the new n-grams the same frequency as their corresponding original n-gram, we now give them only a fraction of the original frequency. Since the original n-grams did occur in the training set and the new ones do not, it is perhaps more logical to give the new n-grams a lower weight.

Perplexity (new n-grams: fraction of frequency of original n-grams)					
Baseline	1/2	1/3	1/4	1/5	1/8
215.803	216.051	216.007	215.99	215.98	215.967

Only the results for a 3-gram model are given. The numbers above each column indicate the fraction of the original frequency that is given to the new n-gram. We have to note that these fractions have been rounded to make sure that the counts were integers (otherwise Witten-Bell smoothing had to used instead of modified Kneser-Ney, see section 3.1.1.2). Yet again, the adapted language models show no improvement with respect to the baseline.

If we add a new n-gram to the count file, he *SRILM* toolkit does not automatically add all possible corresponding lower order n-grams to the language model. Apparently, it only adds them if they are a prefix of an n-gram: for example, the trigram *valse profeten beginnen* occurs in the training text whereas its reversed counterpart – *beginnen valse profeten* – is not and is thus added to the count file. The corresponding bigram *beginnen valse* was also not yet present, and is added as well because it is a prefix for *profeten*. Another trigram that was added is *alle asbest verbiedt* (generated from *verbiedt alle asbest*), but here the bigram *asbest verbiedt* cannot be found in the adapted language model since it does not form a prefix. We adapted our script such that it makes sure that all bigrams are added for a trigram, and that the counts of all uni- and bigrams that were already present are adapted. The n-grams are added to the count file with the same frequency as their original n-grams (e.g. v*erbiedt alle asbest* has frequency 1 so *alle asbest verbiedt* and *asbest verbiedt* also get frequency 1). The perplexity of this language model on *De Morgen* is 216.325, which shows still no improvement.

3.1.1.2 Change in the frequency of the original n-gram

In the experiments that have been described above, the total frequency of all n-grams increased because we added n-grams with a certain count without reducing the count of other n-grams. The total frequency increases the most when the new n-grams received the same frequency as their corresponding original ones: with 2 578 198 or 0.72 % of the original frequency (360 562 749). We will now investigate whether maintaining the total frequency by subtracting counts of already existing n-grams gives better results.

First, we give a fraction of the frequency of the original n-gram to the new n-gram, while the original n-gram keeps the rest of the fraction (the smallest part is obviously always given to the new n-gram). For this approach, however, we have to use Witten-Bell smoothing (Bell, Cleary & Witten 1990, Witten & Bell 1991) rather than modified Kneser-Ney smoothing, because *SRILM* does not support the combination of Kneser-Ney with fractional counts[4]. Note that it would also be possible to round the frequencies as we did in the last experiment of the previous section, but then the total frequency would still change.

Perplexity (largest fraction of original frequency to original n-gram, smallest to new n-gram)								
	Baseline	Same count	2/3 -1/3	4/5 – 1/5	9/10 – 1/10	19/20 -1/20	29/30 -1/30	99/100 – 1/100
3	234.958	235.997	234.713	234.605	234.551	234.53	234.525	**234.522**
4	232.078		**227.623**					
5	233.058		**232.34**					

For the sake of completeness, we also added the perplexity of a language model in which the new n-grams received the same frequency as their corresponding original ones, but that has been smoothed with the Witten-Bell method (column “Same count”). This shows again a small deterioration compared to the baseline, while the fractional language models all show a small improvement. The more frequency is given to the original n-grams, the better the perplexity (but the differences are very small). We see a similar phenomenon for the separated separable verbs:

Perplexity (largest fraction of original frequency to original n-gram, smallest to new n-gram, separated separable verbs)				
Baseline	1/3 – 2/3	19/20 – 1/20	29/30-1/30	99/100-1/100
234.958	234.708	234.538	234.533	**234.53**

Just as for the language model to which only the relevant n-grams were added, these results seem to suggest that the contribution of the reversed n-grams to the language model is extremely small.

Another experiment also suggests this: in order to be able to use modified Kneser-Ney smoothing, we redistributed the total frequency in another way. We subtracted 1 from the frequency of the original n-gram and gave the new n-gram a frequency of 1. As for the previous experiment, the perplexity dropped a bit: 215.787 instead of 215.803 for the baseline. If we add corresponding lower order n-grams or adapt their frequency (see the last paragraph of the previous section), the results are similar: the perplexity is 215.864. One could argue whether it is useful to try other (integer) fractions, but since other approaches showed that giving a lower frequency to the new n-grams gives better results and we already gave the lowest frequency possible (1) to the new

4 Probably because modified Kneser-Ney uses counts of counts: fractional counts would lead to a possibly infinite amount of counts of counts.

n-grams, we did not further experiment with this.

3.1.2 Interpolation

We investigated two approaches for interpolation: either interpolating the baseline language model with a baseline model to which the new n-grams are added (section 3.1.2.1), or interpolating with a language model that is solely based on the new reversed n-grams (section 3.1.2.2).

3.1.2.1 Interpolation with an adapted language model

The results of the interpolation show again that a low weight for the new n-grams gives the best results. The optimal weight for the baseline language model is 0.9, and the perplexity is then 215.719 (baseline 215.803). If both language models receive an equal weight (0.5), the perplexity increases to 215.856.

3.1.2.2 Interpolation with a language model only based on new n-grams

There are three possibilities with regard to language models based solely on new n-grams: a model based on both SV and OV patterns, one based only on SV patterns and one based only on OV patterns.

The first approach performs the best, but even with a high weight for the baseline model (B), the perplexity increases:

Perplexity (interpolation baseline (B), SV language model and OV language model)		
Baseline	B = 0.8; SV = 0.1; OV = 0.1	B = 0.9; SV = 0.05; OV = 0.05
215.803	231.456	219.825

As can be expected, only using SV patterns is even worse (the numbers above each column indicate the weight for the baseline language model):

Perplexity (interpolation baseline and SV language model)			
Baseline	B = 0.1	B = 0.5	B= 0.9
215.803	1057.94	308.337	229.038

The same applies for a language model trained on OV patterns, but the perplexity is slightly better than for SV patterns if the baseline model receives a high weight.

Perplexity (interpolation baseline and OV language model)			
Baseline	B= 0.1	B= 0.5	B = 0.9
215.803	1067.26	305.871	226.241

We can conclude that the weight or frequency that is given to the new n-grams should be low enough. Since both the baseline model and the adapted model contain the original n-grams, an interpolation of these two models gives a much higher weight to the original n-grams (and consequently, a lower weight to the new n-grams) than an interpolation of the baseline with a model

only based on new n-grams.

3.1.3 Add n-grams to language model

Rather than computing the frequency of the new n-gram based on the frequency of its corresponding non-reversed n-gram, we could also compute it based on the back-off probability assigned by the baseline language model. Since the new n-gram is not yet present in the baseline model, the back-off probability would give us an idea how much probability the sequence would receive if it occurred in a test set. Naturally, we would like our new n-gram to have a probability that is higher than this back-off probability, because it is supposed to be an acceptable n-gram. So we could multiply the back-off probability with a certain factor (or add the logarithm of that factor for the log probabilities) and add the new n-gram directly to the language model with that probability. However, a big issue with this approach is then that the sum of the probabilities will not sum up to 1 anymore. As far as we now, the *SRILM* toolkit does not have an option to automatically re-normalize all probabilities (*-renorm* only recomputes the back-off weights). A possibility for future research would be to try to do this ourselves.

3.2 Speech recognition experiments

For two of the adapted language models that we described in the previous sections, we also did some speech recognition experiments. We have to note that the part of *CGN* we tested on seems to be particularly hard to recognize: the word error rates for the baseline language model are pretty high.

Let us first compare the baseline model with the one to which we added the new n-grams with the same frequency:

Word error rate (same count)		
	Baseline	Adapted (same count)
Order = 3	**38.4211**	38.598
Order = 4	**37.803**	37.8429
Order = 5	**37.9773**	38.2997

For this case, perplexity clearly is a good measure for the language model, since the same relation holds for the word error rates as for the perplexities: the adapted language model performs slightly worse.

This does not hold for the language model with fractional counts: the perplexity of the adapted model in which the original n-gram received 2/3 of the original frequency and the new n-gram 1/3, shows an improvement for each order of *n*. The word error rate however, is only better for the 3-gram model (note that for each case, we are talking about extremely small differences):

Word error rate (largest fraction of original frequency to original n-gram, smallest to new n-gram)		
	Baseline	Adapted (1/3 -2/3)

Order = 3	39.2995	**39.2712**
Order = 4	**38.6488**	38.7173
Order = 5	**39.1102**	39.2239

3.3 Other experiments

In this section, we explain some other experiments that were conducted in order to find out why our approach does not cause a (larger) improvement. First, we will compare the number of n-grams in our baseline language model with the number of n-grams in the adapted models (3.3.1). We also randomly checked some of the new n-grams that were created to see whether they are acceptable or not (3.3.2). Finally, we investigate whether our approach does what is is supposed to do: does the new language model actually use higher order n-grams than the baseline model uses and if it does, do these n-grams have a higher probability then the lower order n-grams that were originally used (3.3.3)?

3.3.1 Number of n-grams

Naturally, the adapted language model contains more n-grams than the baseline model:

Number of n-grams in baseline model versus adapted model (same count)		
	Baseline	Adapted
Number of 1-grams	100 000	100 000
Number of 2-grams	11 743 568	11 862 825 (4: 11 807 181; 5: 11 780 091)
Number of 3-grams	41 593 463	42 372 941 (4: 41 806 632; 5: 41 717 304)
Number of 4-grams	66 390 395	66 897 054 (5: 66 559 915)
Number of 5-grams	74 369 712	74 664 906

The adapted model is here the one to which the n-grams have been added with the same count as their original n-gram. If the numbers differ for language models of different orders, they are mentioned between brackets: e.g. the 3-gram adapted language model contains 42 372 941 3-grams, whereas the 4-gram adapted language model only contains 41 806 632 3-grams. As can be seen in the table, these numbers only differ for the adapted language model. This is caused by the fact that n-grams that are shorter than the maximum *n* used in the current language model and that do not have a corresponding higher order n-gram in which they appear, are not added. For example the trigram *zijn technologie ontwikkelde* “developed his technology” is a reversed n-gram and is thus added to the count files for 3-gram, 4-gram and 5-gram models. However, if we look at the 4-gram model (or the 5-gram model), this trigram is not added whereas it is present in the 3-gram model. A small toy example shows how this comes: if we also add a corresponding 4-gram like *zijn technologie ontwikkelde </s>* or *zijn technologie ontwikkelde hij* to the count file, the trigram does end up in the 4-gram model. That is why our adapted 4-gram model contains less trigrams than our adapted 3-gram model (the same reasoning applies to other n-grams with *n* lower than the

maximum *n* of the model). A solution for this problem is not straightforward: one could just add the surrounding context if you reverse the word order of SV or OV patterns, but then you risk creating ungrammatical n-grams:

15. *Daarom eet hij graag taart.* “That is why he likes to eat cake.”

In the above sentence, for example, switching the order of verb and subject and putting the reversed n-gram in the same context, would lead to ungrammatical sequences:

16. **Daarom hij eet graag taart*[5].

One could also consider adding beginning- and/or end-of-sentence symbols in front and/or at the end of the n-gram, but this is also not acceptable in all contexts:

17. Hij zet zijn computer aan. “He turns on his computer.”
18. *<s> zet hij </s>

In this sentence, a transitive verb is used, so not adding its direct object (*zijn computer*) is unacceptable (if the order or subject and verb is switched). A last possibility that we can think of to solve this problem, is to just create random higher order n-grams (with a frequency of 1) and increasing the minimum cut-off for this higher order of *n* (to 2). A drawback of this approach is that other n-grams of that order with frequency 1 will be discarded too. Since the n-grams that we add mostly have low frequencies/probabilities, we decided not to try this solution.

Let us now investigate the number of n-grams in the adapted language models that slightly outperform the baseline model. These were models in which we subtracted an amount – either a fraction of it or just 1 – from the frequency of the original n-grams and assigned that amount to the new n-grams, and the model to which we only added the n-grams that were used for evaluating the test set. Since the latter model was just made to check to which extent we could maximally improve the perplexity of our model on *De Morgen*, we will not discuss it here.

If we distribute the frequency of the original n-gram over the original and the new one by taking a fraction of that frequency, the adapted language model actually contains less n-grams than the baseline model. For example, if the original n-gram keeps 2/3 of its probability and the new one gets 1/3, the adapted 3-gram language model contains 540 279 less n-grams (53 437 031 n-grams in the baseline compared to 52 896 752 n-grams in the new model). The frequency of the infrequent n-grams is lowered and even drops below 1 if it was originally 1, so these infrequent n-grams are not added anymore. This causes a small improvement in perplexity, since it is in any case unlikely that these infrequent n-grams occur in the test set.

We see a similar phenomenon if we subtract 1 from the frequency of the original n-gram and give the new n-gram a frequency of 1. In this case, the count file already contains less n-grams since the new n-gram is only added if the original frequency is higher than 1 (otherwise the original n-gram would get a frequency of 0). This results in 267 607 new n-grams in the count file (compared to 1 451 206 for the count file where new n-grams got the same frequency as the original n-grams), and only 164 644 (0.31 %) new n-grams in the language model. The minor improvement for this language model is thus probably due to the fact that less new n-grams are added. Since they are only added if the frequency of their original n-gram is high enough, it is also more likely that the ones that do end up in the count file are more relevant.

5 An ungrammatical sentence is marked with an *.

3.3.2 Grammaticality of new n-grams

Not all of the new n-grams that were created by reversing the word order are grammatical. We randomly checked some of them and found four types of errors.

The first two types occur mainly for n-grams that are longer than five words, and that consequently are not added to the language model. Firstly, long sequences often lead to parsing errors:

19. *<s> ze pleitte schuldig op de aantijgingen dat hogere kaderleden en andere werknemers in de eerste helft van de jaren negentig tientallen miljoenen dollars registreerden als inkomsten en reserves van de bank terwijl het eigenlijk ging om ongeclaimde cheques en andere tegoeden van klanten </s>*

In the sentence above, *in de eerste helft van de jaren negentig* and *tientallen miljoenen dollars* are two separate phrases, but according to *Alpino*, *negentig* belongs to the same phrase as *tientallen miljoenen dollars*. A second problem with longer sequences is caused by the nature of dependency parsing: the parser always puts the head of a phrase in front. In a verbal phrase, the head is naturally the finite verb. However, in a Dutch subordinate clause, the finite verb occurs at the end of the verbal phrase:

20. *<s> er bestond een soort stilzwijgende overeenkomst dat wie banen schiep fiscaal een potje mocht breken </s>*

This finite verb is moved to the front of the verbal phrase by the parser:

21. ** ... overeenkomst dat wie banen schiep mocht fiscaal een potje breken*

The complete sentence would be ungrammatical in Dutch, although parts of it would be correct (e.g. *mocht fiscaal een potje breken* could be part of a head clause). Since these two errors – parsing errors and word order errors – occurred quite frequently in our random selection of long new n-grams, we decided not to extract lower order n-grams from the new n-grams that were longer than five words. Unfortunately, these errors also occur in n-grams that were added to the adapted language model (e.g. *negentig tientallen miljoenen dollars registreerden* is reversed to *registreerden negentig tientallen miljoenen dollars* and added to the language model).

Another error that sometimes creeps in the new n-grams, is due to an error in the normalization of the training corpus:

22. *<s> van onze redacteur brussel De euro moest gisteren nog voor de middag al bijna de helft van het terrein prijsgeven dat maandag op de dollar werd veroverd </s>*

The above sentence should be split before *De euro* (the first part does not belong to the sentence) and the first letter of *De* should be lower case. Our script finds *De euro moest* as an example of an SV pattern and generates the reversed n-gram *moest De euro*, which is not acceptable.

A fourth type of error was already mentioned in the previous sections: in the first language models that were created, the separable verbs were not separated when they should be. For example,

the OV pattern *een goede dynamiek meegaf* is detected and the ungrammatical n-gram *meegaf een goede dynamiek* is generated. We tried to correct the majority of these errors, but strangely enough the language model created with the separated separable verbs does not perform better.

3.3.3 Differences in evaluation

Finally, we investigated whether our adapted language model actually uses a higher order of n-gram for some word sequences. Again with the aid of the output of *-ppl -debug 2* in *SRILM*, which prints the n-grams and probabilities that are used to evaluate each word sequence, we found that for 131 080 word sequences the adapted 3-gram language model (in which the new n-grams have the same frequency as their original ones) indeed uses a higher order n-gram than the baseline (*De Morgen* contains ca. 135M word tokens). This is between 0.19 % (assuming that the n-gram that is used is each time a bigram) and 0.29 % (if it is a trigram that is used each time).

Do these new n-grams also have a higher probability than the lower order n-grams that were originally used? It appears that this is the case for the majority of them: 106 968 (81.6 %) indeed have higher probability. For the 24 112 other new n-grams, the difference between the probability of the new n-gram and the probability of the original one is small: for 19 674 n-grams the difference is smaller than 0.00001 and for 4106 n-grams it is smaller than 0.0001 (so 332 new n-grams have a probability which is more than 0.0001 lower than the probability of the lower order n-gram that was originally used).

Let us investigate the results for the corpus of inversion questions (see section 3.1.1.1) in a more detailed way. If we compare the result of *-ppl -debug 2* for the baseline model and the adapted model (same count), we see that the adapted language model uses another n-gram in six cases, among which five are inversion patterns such as (15) and one is an SV pattern (16).

15. *Zijn die theorieën nu uitgesloten?* “Are those theories now excluded?”
16. *Begrijpen de leerlingen wel goed wat de docent zegt?* “Do the students actually understand what the teacher says?”

For both sentences above, the adapted model uses a trigram instead of a bigram (this is the case for six out of seven changed n-grams). So our adapted language model performs as it should: it uses a higher order n-gram. In five out of six cases, the probability of this higher order n-gram is also higher than the one of the lower order n-gram (only the probability for *houdt de tegenslag (nooit op)* is lower than the probability for *de tegenslag*). There are however two drawbacks: first, the scope of our changes is not large enough. For example, the sentence below is the only one for which the adapted model uses a bigram rather than a unigram to evaluate the inversion pattern:

17. *Wordt Alan Haydock dan door louter onheil bezocht?* “Is it only misfortune that happens to Alan Haydock?”

But our corpus of inversion questions contains fourteen other sentences for which the baseline model uses a unigram to evaluate the inversion pattern, such as (18) and (19).

18. *Was Bono iets waard als redacteur?* “Was Bono of any use as editor?”
19. *Ontstond epilepsie op jonge leeftijd?* “Did epilepsy start at a young age?”

So in only one out of fifteen cases where improvement is possible, our adapted language model

actually performs better. A second drawback is that the probability of the sentences in which a higher order n-gram is used increases, but that the probabilities of all other sentences decrease. For 184 sentences, the probability assigned by the baseline model is higher, whereas the adapted model outperforms the baseline for only 64 sentences.

We can conclude that our approach does what we expected it to do: the new n-grams are indeed used for the evaluation on the test set, although in only a small percentage of the cases. Since the amount of n-grams that are added is also small compared to the total amount of n-grams in the baseline model, this is not unexpected. Moreover, the majority of the n-grams have a higher probability than their corresponding lower order n-grams, which is also desirable.

4. Conclusion

We can conclude that the amount of effort invested in this approach is not really worth the gains we get: only if a small amount of (probably) relevant new n-grams are added, we get a minor improvement. It is not very likely that continuing tweaking the parameters and trying out different frequencies or probabilities will cause a larger improvement. However, we still believe that using syntactic information might help language modeling. Possibly other kinds of models than n-gram models might be able to make better use of it, as Levy and Goldberg (2014) have already shown.

5. References

Alumäe, T. & M. Kurimo. 2010. “Efficient Estimation of Maximum Entropy Language Models with N-gram features: an SRILM extension”. *Proceedings Interspeech*, p. 1820-1823. Makuhari, Japan.

Augustinus, L., V. Vandeghinste & F. Van Eynde. 2012. “Example-Based Treebank Querying”. *Proceedings of the 8th International Conference on Language Resources and Evaluation* (LREC-2012), p. 3161-3167. Istanbul, Turkey. (GrETEL online: http://nederbooms.ccl.kuleuven.be/eng/gretel).

Bell, T.C., Cleary, J.G. & I.H. Witten. 1990. *Text Compression.* Prentice Hall, Englewood Cliffs, N.J.

Chelba, C., Engle, D., Jelinek, F., Jimenez, V., Khudanpur, S., Mangu, H., Printz, E., Ristad, S., Rosenfeld, R., Stolcke, A. & D. Wu. 1997. “Structure and Performance of Dependency Language Model”. In: *Proceedings of Eurospeech*, p. 2775-2778. Rhodes, Greece.

Chelba, C. 1997. “A Structured Language Model”. In: *Proceedings of ACL-EACL*, p. 498-500. Madrid, Spain.

Chen, S.F. & J. Goodman. 1999. “An empirical study of smoothing techniques for language modeling”. In: *Computer Speech and Language* 13, p. 359-394.

Corpus Gesproken Nederlands. (http://tst-centrale.org/producten/corpora/corpus-gesproken-nederlands/6-17)

Demuynck, K., Roelens, J., Van Compernolle, D. & P. Wambacq. 2008. “SPRAAK: An Open

Source SPeech Recognition and Automatic Annotation Kit". In: *Proceedings International Conference on Spoken Language Processing*, p. 495. Brisbane, Australia.

Demuynck, K., Puurula, A., Van Compernolle, D. & P. Wambacq. 2009. "The ESAT 2008 System for N-Best Speech Recognition Benchmark". In: *IEEE Workshop on Automatic Speech Recognition & Understanding*, p. 339-444.

Dryer, M.S. 2013a. "Order of Subject and Verb". In: Dryer, M. S. & Haspelmath, M. (eds.) *The World Atlas of Language Structures Online*. Leipzig: Max Planck Institute for Evolutionary Anthropology. (Available online at http://wals.info/chapter/82, accessed on 16/03/2015.)

Dryer, M.S. 2013b. "Order of Object and Verb". In: Dryer, M.S. & Haspelmath, M. (eds.) *The World Atlas of Language Structures Online*. Leipzig: Max Planck Institute for Evolutionary Anthropology. (Available online at http://wals.info/chapter/83, accessed on 16/03/2015.)

Gubbins, J. & A. Vlachos. 2013. "Dependency Language Models for Sentence Completion". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, p. 1405-1410. Seattle, Washington, USA.

Jijkoun, V., Mur. J. & M. De Rijke. 2004. "Information extraction for question answering: Improving recall through syntactic patterns". In: *COLING 2004*. Geneva, Switserland.

Levy, O. & Y. Goldberg. 2014. "Dependency-Based Word Embeddings". In: *ACL*, p. 302-308.

Oostdijk, N., Reynaert, M., Monachesi, P., Van Noord, G., Ordelman, R.J.F., Schuurman, I. & V. Vandeghinste. 2008. "From D-Coi to SoNaR: A reference corpus for Dutch". In: *Proceedings of the sixth international conference on language resources and evaluation (LREC)*, 28-30 May 2008, p. 1437-1444. Marrakech, Marocco.

Pauls, A. & D. Klein. 2012. "Large-scale syntactic language modeling with treelets". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* Vol 1, p. 959-968. Jeju Island, Korea.

Schukat-Talamazzini, E.G., Hendrych, R., Kompe, R. & H. Niemann. 1995. "Permugram language models". In: *Fourth European Conference on Speech Communication and Technology (EUROSPEECH)*, p. 1773-1776. Madrid, Spain.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A. & L. Chanona-Hernández. 2014. "Syntactic N-grams as machine learning features for natural language processing". In: *Expert Systems with Applications* 41, p. 852-860.

Stolcke, A. 2002. "SRILM – An Extensible Language Modeling Toolkit". In: *Proceedings International Conference Spoken Language Processing*, p. 901-904. Denver, Colorado.

Stolcke, A., Zheng, J., Wang, W. & V. Abrash. 2011. "SRILM at Sixteen: Update and Outlook". In: *Proceedings IEEE Automatic Speech Recognition and Understanding Workshop.* Waikoloa, Hawaii.

Van den Bosch, A., Busser, G.J., Daelemans, W. & S. Canisius. 2007. "An efficient memory-based morphosyntactic tagger and parser for Dutch". In: Van Eynde, F., Dirix, P., Schuurman, I. & V. Vandeghinste (eds.), S*elected Papers of the 17th Computational Linguistics in the Netherlands*

Meeting, p. 99-114. Leuven, Belgium. (*Frog* online: http://ilk.uvt.nl/frog/).

Van der Plas, L. & G. Bouma. 2005. “Syntactic contexts for finding semantically related words”. In: Van der Wouden, T., Poss, M., Reckman, H. & C. Cremer (eds.), *Computational Linguistics in the Netherlands 2004. Selected Papers from the fifteenth CLIN meeting*, p. 173-186. LOT, Netherlands Graduate School of Linguistics, Utrecht.

Van Eynde, F. 2004. *Part of Speech Tagging en Lemmatisering van het Corpus Gesproken Nederlands*. (Available online: http://tst-centrale.org/images/stories/producten/documentatie/cgn_website/doc_Dutch/topics/project/pos_tagging/index.htm).

Van Noord, G. 2006. "At Last Parsing Is Now Operational" In: *TALN 2006*, p. 20-42.

Van Noord, G., Schuurman, I. & G. Bouma. 2011. Lassy Syntactische Annotatie. (Available online: http://www.let.rug.nl/vannoord/Lassy/sa-man_lassy.pdf).

Van Noord. G. 2008. “Huge Parsed Corpora in LASSY”. In: F. van Eynde, A. Frank, K. de Smedt, G. van Noord (eds), *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7)*, p. 115-126. Groningen, The Netherlands. LOT Occasional Series.

Wiktionary, 2015. *Category: Dutch separable verbs*. (http://en.wiktionary.org/wiki/Category:Dutch_separable_verbs, accessed on 10/02/2015).

Witten, I.H. & T.C. Bell. 1991. “The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression”. In: *IEEE Transactions on Information Theory* 37(4), p. 1085-1094.

Zhang, L. & H. Wang, 2014. “Go Climb a Dependency Tree and Correct the Grammatical Errors”. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), p. 266-277. Doha, Qatar.