Using Suffix Arrays as Language Models: Scaling the n-gram

Herman Stehouwer Menno van Zaanen

TiCC, Tilburg University, Tilburg, The Netherlands
{J.H.Stehouwer, M.M.vanZaanen}@uvt.nl

Abstract

In this article, we propose the use of suffix arrays to implement n-gram language models with practically unlimited size n. These unbounded n-grams are called ∞-grams. This approach allows us to use large contexts efficiently to distinguish between different alternative sequences while applying synchronous back-off.

From a practical point of view, the approach has been applied within the context of spelling confusibles, verb and noun agreement and prenominal adjective ordering. These initial experiments show promising results and we relate the performance to the size of the n-grams used for disambiguation.

1 Introduction

When writing texts, people often use spelling checkers to reduce the number of mistakes in their texts. Many spelling checkers concentrate on non-word errors, which can be identified (relatively) easily in texts because they consist of character sequences that are not part of the language. For example, in English *woord* is not part of the language, hence a non-word error. A possible correction would be *word*. A simple example of a word error is for instance *the cat run*, a possible correction would be to modify the noun (*cat*) to a plural, or to modify the verb (*run*) to the singular form.

Even when a text does not contain any non-word errors, there is no guarantee that the text is error-free. There are several types of errors in which words are part of the language, but used incorrectly in context. We call these errors *contextual errors*. Note that these kinds of errors are much harder to recognize than non-word errors, as information from the context is required to recognize and correct them. In contrast, non-word errors can be recognized without context.

Here, we introduce an approach that can be used to make decisions about different types of contextual errors. This approach concentrates on the use of large n-grams and as such can be seen as an extension to more conventional n-gram models, such as a simple trigram model.

The approach generates all possible corrections for each potential contextual error and finds the most likely of these according to the language model. The language model encodes the context of the contextual error in the shape of large n-grams.

The underlying assumption of the language model is that if large n-grams can be found as correct examples of an alternative in the model, this is more indicative of the correct alternative than the situation where only smaller n-grams of the context are found. This thought follows from the idea that it is harmful to forget about parts of the data in language learning [6]. In short, we assume that using more (precise) information pertaining to the decision is better.

To be able to capture context of all sizes, we introduce ∞-grams, which are n-grams of unbounded size. We implement these using suffix arrays and will show in the rest of the article that this is a viable approach.

First, we will introduce the suffix array based n-gram disambiguation approach. We will discuss how we generate and use the language model, which includes an explanation of the use of ∞-grams and the synchronous back-off method. Section 3 describes the three tasks we have used to evaluate the method. Next, we discuss the results in Section 4 and finally we conclude.

$$
\begin{array}{ccc}
P(a\ b\ c) = 0? & \text{and} & P(a\ d\ c) = 0? \\
\Downarrow & & \Downarrow \\
P(a\ b) \times P(b\ c) = 0? & \text{and} & P(a\ d) \times P(d\ c) = 0? \\
\Downarrow & & \Downarrow \\
P(a) \times P(b) \times P(c) & & P(a) \times P(d) \times P(c)
\end{array}
$$

Figure 1: Computation of probabilities of sequences using synchronous back-off is done in parallel. If all probabilities with a particular n-gram size are zero, back off to a lower order n-gram until at least one of the probabilities is non-zero.

i	suffix	lcp	S[suffix]
0	2	0	aaacatat$
1	3	2	aacatat$
2	0	1	acaaacatat$
3	4	3	acatat$
4	6	1	atat$
5	8	2	at$
6	1	0	caaacatat$
7	5	2	catat$
8	7	0	tat$
9	9	1	t$
10	10	0	$

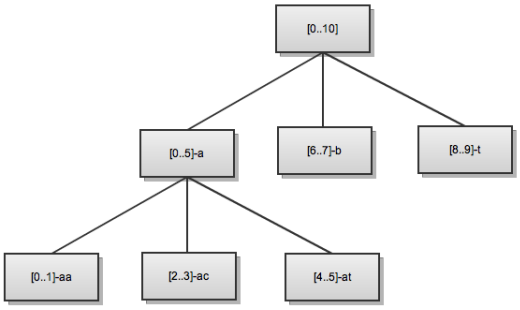

Figure 2: An enhanced suffix array on the string *S= acaaacatat* on the left, and its corresponding lcp-interval tree on the right. The *acaacatat* example is taken from [1].

2 Approach

To tackle the problem of contextual errors in text, we generate all alternative solutions at positions where an error can occur that we can correct. We then use a language model to select the most likely sequence. The sequence with the highest score is selected as the most probable correct form.

The language model we use here is based on ∞-grams, which are n-grams of arbitrary length. The score of a sequence is computed by multiplying the probabilities of the ∞-gram for each position in the sequence.

Obviously, when using n-grams with very large n, data sparseness is an issue. The training data will probably not contain any occurrence of the particular sequence of n symbols, even though the sequence is correct. The probability extracted from the training data will be zero, even though the correct probability should be non-zero (albeit small). To reduce the impact of this problem we can use techniques such as smoothing or back-off. Smoothing [4] redistributes probability mass to estimate the probability of previously unseen word sequences. In the case of back-off, probabilities of lower order n-grams are used to approximate the probability of the sequence.

In this article, we use the synchronous back-off method [17] to deal with data sparseness. This method, as illustrated in Figure 1, analyzes alternative n-grams of the same size in parallel. If all n-grams have zero probability, the method considers $n-1$-grams for all alternatives. Backing off continues until at least one alternative has a non-zero probability. This implements the idea that, assuming the training data is sufficient, if a probability is zero the sequence is not in the language.

The synchronous back-off model uses probabilities of the same model at the same position for all alternative sequences. For this the highest-order model is used that has at least one non-zero probability on one of the sequences at the position in question.

For instance when we look at trigrams at the focus position as shown in Figure 1. If option b has a non-zero probability and option d has a probability of zero the synchronous back-off method will halt and will assign the probability of the model to both b and d. This will result in option d being assigned a zero probability. When both have a probability of zero, a back-off to a lower-order model is performed for both alternatives. This is in line with the idea that if a probability is zero and the training data is sufficient, that the sequence is not in the language. In other words, it means that we have seen evidence that supports one of the alternative sequences. It is important to note that the output of the system is not a proper probability as, over a sequence, as it combines probabilities from different sized n-grams.

Task	# test-cases
Confusibles	221,301
Verb & noun agreement	5,632,210
Prenominal adjective ordering	412,974

Table 1: Number of test cases for the different tasks.

Keeping track of all n-grams of all sizes is implemented by storing the training data in a suffix tree. A suffixtree is a trie-based data structure (as described succinctly in [10, Chapter 6.3]) in which all suffixes occupy a single path from the root to a leaf. It stores all suffixes of a sequence in such a way that a suffix (and similarly an infix) can be found in linear time in the length of the suffix. Construction only needs to be performed once. The largest suffix and therefore the largest n-gram stored in both data-structures is the full training data.

An alternative data structure to store the data is using a suffix array, which is a flat data-structure containing a sorted list of all suffixes in the training sequence [14]. An enhanced suffix array contains an implicit suffixtree structure [1].

An enhanced suffix array extends a regular suffix array with a data-structure allowing for the implicit access of the longest-common-prefix (lcp) intervals [1]. An lcp interval represents a virtual node in the implicit suffixtree. A simple enhanced suffix array with its corresponding implicit suffixtree is shown in Figure 2 as an example.

There is a tradeoff between using a suffix array and a suffixtree. The suffix array occupies significantly less space to a suffixtree even with all the enhancements. Both of these relations are an order of magnitude in difference. As our main constraint is memory use we opt for the suffix array in this paper.

Due to the way suffix arrays are constructed, we can efficiently find the number of occurrences of subsequences (used as n-grams) of the training data. Starting from the entire suffix array we can quickly identify the interval that pertains to the particular n-gram query. The interval specifies exactly the number of occurrences of the subsequence in the training data. Effectively, this means that we can find the largest non-zero probability n-gram efficiently.

Since n-grams of all lengths are stored in the suffix array, we can use suffix arrays to efficiently implement language models of n-grams of any size without suffering storage difficulties. Effectively, this means that we can always find the largest non-zero probability n-gram efficiently.

3 Experimental settings

To evaluate the ∞-gram approach, three contextual error problems will be tackled, namely, the confusibles, verb and noun agreement, and prenominal adjective ordering tasks. We will first describe these three tasks briefly, followed by a description of the data and the experimental setup used. In Table 1 we show the number of different test-cases available for each task. In all experiments we assume our corpus to be correct.

3.1 Confusibles

The class of *confusibles* [9] consists of words that belong to the language, but are used incorrectly with respect to their context. For instance, *She owns to cars* contains the confusible *to*, which is valid and part of the language, but used incorrectly in this context. Alternatives *too*, *two*, and *to* are often confused and hence are placed in the same confusible set. We show an single example of the process for the confusible set {*then*,*than*} in Figure 3.

A typical approach to the problem of confusibles is to train machine learning classifiers for each confusible set. Most of the work in this area has concentrated on confusibles due to homophony or similar spelling [15, 20]. However, some research has also touched upon inflectional or derivational confusibles such as *I* versus *me* [7].

Most work on confusible disambiguation using machine learning concentrates on hand-selected sets of notorious confusibles. The confusible sets are typically very small (two or three elements) and the machine learner will only see training examples of the members of the confusible set. We approach the problem by finding all members of confusible sets and generating an alternative sequence for each member of the appropriate confusible set. We use the 21 confusible sets from [7].

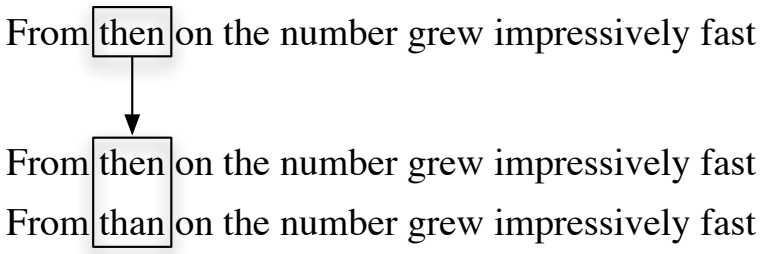

Figure 3: An example of a generated alternative set for the confusible set $\{then, than\}$. Example taken from the BNC, selected for the brevity of the sentence.

Our method considers the entire training data, so it sees examples of all members of all confusible sets. It also the information about the yet-to-be-defined confusible sets. Once trained on a corpus the suffix array can be used for all tasks, the training phase is not specialized. Therefore it is trivial to dynamically add or remove confusible sets while running without retraining.

3.2 Verb and noun agreement

The second problem deals with the detection and correction of (simple) agreement errors. Since many words can have agreement errors with respect to other words in the sentence, we concentrate on agreement between verbs and nouns only. For example, *The man speak* has an agreement error between the noun and the verb which could be solved in two ways: *the men speak* or *the man speaks*.

Over the years several approaches have been tried, such as using trigram probabilities [21], using multi-level features with support vector machines [**?**], reducing the sentence to its stems and rebuilding it [11], template matching on parse trees [12], pattern discovery with supervised learning [19], constraint grammar rules [3], co-occurrence statistics with the MI metric [5], maximum entropy models for article usage correction [8] to name a few.

We approach the problem by finding all verbs and nouns in the sequence and generating an alternative set for each. This alternative set contains a separate sequence for each inflection of the verb or noun. This list of inflections is generated using information from the English Celex-2 database [2].

3.3 Prenominal adjective ordering

When modifying nouns using multiple adjectives, one has to select the ordering of the adjectives. As a rule native speakers "know" the correct ordering. However, a formal description of the exact ordering and division of prenominal adjectives in semantic classes has generated many conflicting proposals [13, 16], although the correct ordering seems to be a fairly strict.

In this article we perform experiments similar to that of [13] and [16]. We try to find evidence in the available language model that indicates a single, correct ordering of adjectives out of all potential orderings. Note that if there are n consecutive prenominal adjectives this results in $n!$ potential orderings.

3.4 Experimental setup

To evaluate the effectiveness of the ∞-gram language model approach, we use a modular system with standardized data representations. This allows us to exchange modules for each of the tasks locally. The system consists of three phases. Firstly, test data for a specific task is transformed to sets of alternative sequences. Secondly, the language model is applied to these sets of alternatives and the most likely is selected based on the computed probability. Finally, the results are evaluated.

To generate test instances, we consider all alternative solutions for each instance extracted from a corpus. To identify test instances for confusibles, we use the 21 confusible sets from [7]. Alternative sets for verbs and nouns agreement contain all possible inflections, generated using the English Celex-2 database.

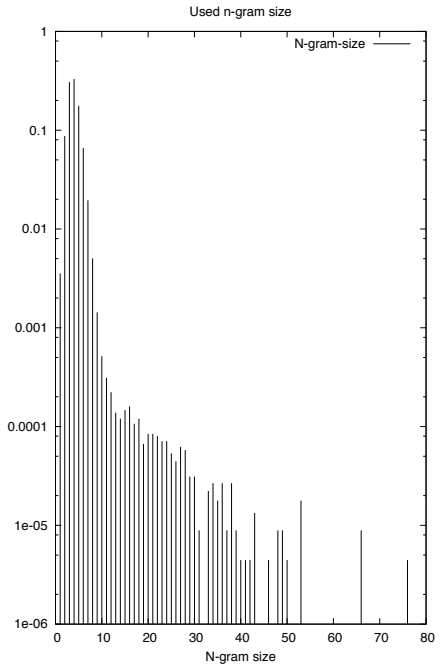

Figure 4: Distribution of the used n-gram sizes in the confusibles problem. The vertical-axis uses a log scale and denotes the percentage of n-grams of length n. Where n is shown on the horizontal-axis.

Prenominal adjective test instances are all potential orderings of sequences of two or more prenominal adjectives.

To compare the performance of the ∞-gram model, we run the experiments for each of the different tasks with different values for n, namely $\{1, 2, 3, 4, 5, 10, \infty\}$. The use of synchronous back-off means that these values for n are maximum values. Prenominal adjective problem experiments are not run with $n = 1$ as all alternatives would receive the same probability.

The training and testing data is taken from the British National Corpus (BNC). This is a corpus of approximately 100 million words of both spoken and written English. We use a consecutive chunk of 10% of the corpus as testing material and the rest as training material. Table 1 contains the number of found test-examples.

4 Results

Precision, recall and F-score results of the ∞-gram and restricted n-gram experiments are summed up in Table 2. This table shows the increase in performance on the tasks ends around $n = 3$ or 4, but also that using larger n-grams, such as ∞-grams does not decrease performance. This is due to the robustness of synchronous back-off [18].

One of the underlying assumptions is that larger n-grams provide more precise information. To test this, we look at the precision grouped by n-gram sizes used for classification. In Table 3, we show that larger n-grams perform better than smaller ones. However, larger matching n-grams are rare. In case of the largest n-grams, this often points at duplicate sentences in the corpus. n-grams in the $n = 6$–10 range perform better than smaller n-grams and are still fairly frequent, supporting our assumption that forgetting part of the data is harmful.

Figure 4 shows a histogram of the n-gram size used for classification of each set of sequences in the confusible problem. The average n-gram size used for the calculation of the probability is 3.9 for the confusible problem, 2.9 for verb and noun agreement and 2.8 for adjective ordering. This shows the balance between imprecise, frequent n-grams (with small n) and precise, infrequent n-grams (with large n).

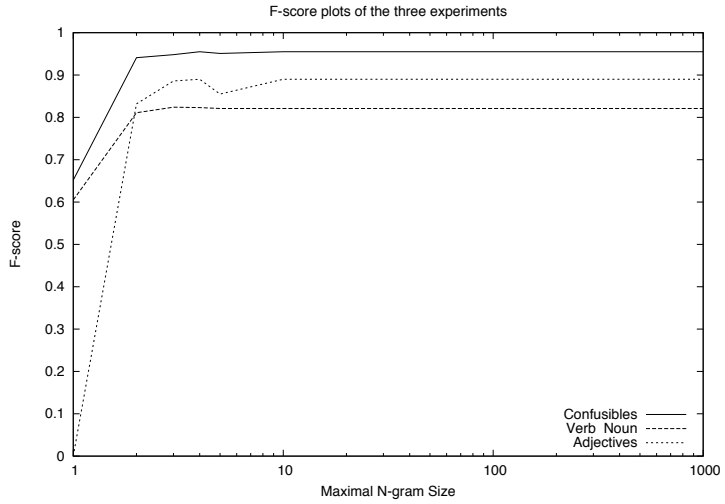

Figure 5: Average F-score for each task plotted against the maximum n-gram size. The 999 position shows results for the ∞-grams.

n-gram size		1	2	3	4	5	10	∞
Confusibles	P	0.753	0.942	0.949	0.955	0.953	0.955	0.955
	R	0.575	0.939	0.946	0.955	0.949	0.955	0.955
	F	0.652	0.941	0.948	0.955	0.951	0.955	0.955
Verb & noun	P	0.563	0.815	0.829	0.829	0.827	0.827	0.827
	R	0.655	0.807	0.818	0.817	0.815	0.815	0.815
	F	0.606	0.811	0.824	0.823	0.821	0.821	0.821
Adjectives	P	-	0.839	0.891	0.894	0.864	0.894	0.894
	R	-	0.826	0.882	0.886	0.845	0.885	0.885
	F	-	0.832	0.886	0.890	0.855	0.890	0.890

Table 2: Average precision (P), recall (R) and F-score (F) for each task with varying sizes of n. This is the maximum size of the n-gram, not the actual size of the n-gram that was used for classification.

n-gram size	Precision		
	Confusibles	Verb &Noun	Adjectives
1	0.585	0.694	0.505
2	0.865	0.747	0.714
3	0.938	0.786	0.836
4	0.957	0.806	0.885
5	0.962	0.820	0.909
6–10	0.968	0.846	0.922
11–20	0.986	0.975	0.966
20+	1	0.982	0.997

Table 3: Precision given the size of the n-gram used. The size is the size of the n-gram used for the actual classification, not the upper bound on the n-gram size.

5 Conclusion

In this article, we introduced the ∞-gram language model, which allows for the use of arbitrary length n-grams. It was evaluated on three different contextual error problems, leading to interesting results. Overall, ∞-grams perform well, which is shown by the precision of large n-grams. However, large n-grams are too infrequent to increase overall performance compared to limited size n-grams. When they do occur their results are much more accurate.

In the future to reduce impact of the data sparseness problem, we will look at incorporating other levels of information, such as part-of-speech, which allows for a more gradual back-off as occurrence of large n-grams of part-of-speech tags is more likely than occurrence of such n-grams of words.

References

[1] M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.

[2] R. H. Baayen, R. Piepenbrock, and H. van Rijn. *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA, 1993.

[3] E. Bick. A constraint grammar based spellchecker for danish with a special focus on dyslexics. In *A Man of Measure: Festschrift in Honour of Fred Karlsson on his 60th Birthday; a special supplement to SKY Journal of Linguistics*, pages 387–396, 2006.

[4] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modelling. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 310–318. ACL, June 1996.

[5] M. Chodorow and C. Leacock. An unsupervised method for detecting grammatical errors. In *In Proceedings of NAACL'00*, pages 140–147, 2000.

[6] W. Daelemans, A. Van den Bosch, and J. Zavrel. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41, 1999.

[7] A.R. Golding and D. Roth. A Winnow-Based Approach to Context-Sensitive Spelling Correction. *Machine Learning*, 34(1–3):107–130, 1999.

[8] N. Han, M. Chodorow, and C. Leacock. Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*, 12(02):115–129, 2006.

[9] J. H. Huang and D. W. Powers. Large scale experiments on correction of confused words. In *Australasian Computer Science Conference Proceedings*, pages 77–82, Queensland AU, 2001. Bond University.

[10] D. E. Knuth. *The art of computer programming*, volume 3: Sorting and searching (second edition). Addison-Wesley, Reading, MA, 1998.

[11] J. Lee and S. Seneff. Automatic Grammar Correction for Second-Language Learners. In *Ninth International Conference on Spoken Language Processing*. ISCA, 2006.

[12] J. Lee and S. Seneff. Correcting misuse of verb forms. In *Proceedings of ACL-08: HLT*, pages 174–182, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[13] R. Malouf. The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 85–92, New Brunswick, NJ, 2000. ACL.

[14] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

[15] D. Sandra, F. Daems, and S. Frisson. Zo helder en toch zoveel fouten! wat leren we uit psycholinguïstisch onderzoek naar werkwoordfouten bij ervaren spellers? *Tijdschrift van de Vereniging voor het Onderwijs in het Nederlands*, 30(3):3–20, 2001.

[16] J. Shaw and V. Hatzivassiloglou. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 135–143, College Park, Maryland, USA, June 1999. Association for Computational Linguistics.

[17] H. Stehouwer and A. Van den Bosch. Putting the t where it belongs: Solving a confusion problem in Dutch. In S. Verberne, H. van Halteren, and P.-A. Coppen, editors, *Computational Linguistics in the Netherlands 2007: Selected Papers from the 18th CLIN Meeting*, pages 21–36, Nijmegen, The Netherlands, 2009.

[18] H. Stehouwer and M. Van Zaanen. Language models for contextual error detection and correction. In *Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference*, pages 41–48, Athens, Greece, 2009.

[19] G. Sun, X. Liu, G. Cong, M. Zhou, Z. Xiong, J. Lee, and C. Lin. Detecting erroneous sentences using automatically mined sequential patterns. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 81–88, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[20] A. Van den Bosch, G.J. Busser, S. Canisius, and W. Daelemans. An efficient memory-based morpho-syntactic tagger and parser for Dutch. In P. Dirix, I. Schuurman, V. Vandeghinste, and F. Van Eynde, editors, *Computational Linguistics in the Netherlands: Selected Papers from the Seventeenth CLIN Meeting*, pages 99–114, Leuven, Belgium, 2007.

[21] L. A. Wilcox-O'Hearn, G. Hirst, and A. Budanitsky. Real-Word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. In A. Gelbukh, editor, *Proceedings of the Computational Linguistics and Intelligent Text Processing 9th International Conference, CICLing 2008*, volume LNCS 4919, pages 605–616, Berlin, Germany, 2008. Springer Verlag.