Dependency language models for sentence completion

Joseph Gubbins
Computer Laboratory
University of Cambridge
jsg52@cam.ac.uk

Andreas Vlachos
Computer Laboratory
University of Cambridge
av308@cam.ac.uk

Abstract

Sentence completion is a challenging semantic modeling task in which models must choose the most appropriate word from a given set to complete a sentence. Although a variety of language models have been applied to this task in previous work, none of the existing approaches incorporate syntactic information. In this paper we propose to tackle this task using a pair of simple language models in which the probability of a sentence is estimated as the probability of the lexicalisation of a given syntactic dependency tree. We apply our approach to the Microsoft Research Sentence Completion Challenge and show that it improves on n-gram language models by 8.7 percentage points, achieving the highest accuracy reported to date apart from neural language models that are more complex and expensive to train.

1 Introduction

The verbal reasoning sections of standardised tests such as the Scholastic Aptitude Test (SAT) feature problems where a partially complete sentence is given and the candidate must choose the word or phrase from a list of options which completes the sentence in a logically consistent way. Sentence completion is a challenging semantic modelling problem. Systematic approaches for solving such problems require models that can judge the global coherence of sentences. Such measures of global coherence may prove to be useful in various applications, including machine translation and natural language generation (Zweig and Burges, 2012).

Most approaches to sentence completion employ language models which use a window of immediate context around the missing word and choose the word that results in the completed sentence with the highest probability (Zweig and Burges, 2012; Mnih and Teh, 2012). However, such language models may fail to identify sentences that are locally coherent but are improbable due to long-range syntactic/semantic dependencies. Consider, for example, completing the sentence

I saw a tiger which was really very ...

with either *fierce* or *talkative*. A language model relying on up to five words of immediate context would ignore the crucial dependency between the missing word and the noun *tiger*.

In this paper we tackle sentence completion using language models based on dependency grammar. These models are similar to standard n-gram language models, but instead of using the linear ordering of the words in the sentence, they generate words along paths in the dependency tree of the sentence. Unlike other approaches incorporating syntax into language models (e.g., Chelba et al., 1997), our models are relatively easy to train and estimate, and can exploit standard smoothing methods. We apply them to the Microsoft Research Sentence Completion Challenge (Zweig and Burges, 2012) and show an improvement of 8.7 points in accuracy over n-gram models, giving the best results to date for any method apart from the more computationally demanding neural language models.

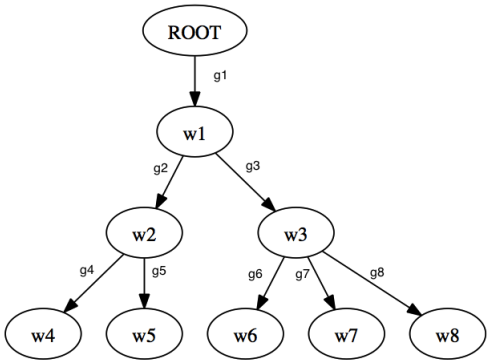


Figure 1: Dependency tree example

2 Unlabelled Dependency Language Models

In dependency grammar, each word in a sentence is associated with a node in a dependency tree (Figure 1). We define a dependency tree as a rooted, connected, acyclic directed graph together with a mapping from the nodes of the tree to a set of grammatical relation labels $\mathcal{R}$. We define a lexicalised dependency tree as a dependency tree along with a mapping from the vertices of the tree to a vocabulary $\mathcal{V}$.

We seek to model the probability distribution of the lexicalisation of a given dependency tree. We will use this as a language model; we neglect the fact that a given lexicalised dependency tree can correspond to more than one sentence due to variations in word order. Let S^T be a lexicalised dependency tree, where T is the unlexicalised tree and let $w_1 w_2 \dots w_m$ be an ordering of the words corresponding to a breadth-first enumeration of the tree. In order for this representation to be unique, when we parse a sentence, we will use the unique breadth-first ordering where the children of any node appear in the same order as they did in the sentence. We define w_0 to be a special symbol denoting the root of the tree. We denote the grammatical relation between w_k and its parent by $g_k \in \mathcal{R}$.
We apply the chain rule to the words in the tree in the order of this breadth-first enumeration:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|(w_k)_{k=0}^{i-1}, T] \tag{1}$$

Given a word w_i, we define the ancestor sequence $A(w)$ to be the subsequence of $(w_k)_{k=0}^{i-1}$ describing the path from the root node to the parent of w, where each element of the sequence is the parent of the next element. For example in Figure 1, $A(w_8) = (w_0, w_1, w_3)$. We make the following two assumptions:

- that each word w_i is conditionally independent of the words outside of its ancestor sequence $(w_k)_{k=0}^{i-1} \cap A(w_i)^c$, given the ancestor sequence $A(w_i)$;
- that the words are independent of the labels $(g_k)_{k=1}^{m}$.

Using these assumptions, we can write the probability as:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|A(w_i)] \tag{2}$$

Given a training data corpus consisting of sentences parsed into dependency trees, the maximum likelihood estimator for the probability $\mathbb{P}[w_i|A(w_i)]$ is given by the proportion of cases where the ancestor sequence $A(w_i)$ was followed by w_i. Let $C(\cdot)$ be the count of the number of observations of a pattern in the corpus. We have

$$\hat{\mathbb{P}}[w_i|A(w_i)] = \frac{C((A(w_i), w_i))}{\sum_{w \in \mathcal{V}} C((A(w_i), w))} \tag{3}$$

As is the case for n-gram language models, we can't hope to observe all possible sequences of words no matter how big the corpus. To deal with this data sparsity issue, we take inspiration from n-gram models and assume a Markov property of order $(N-1)$:

$$\mathbb{P}[w|A(w)] = \mathbb{P}[w|A^{(N-1)}(w)] \tag{4}$$

where $A^{(N-1)}(w)$ denotes the sequence of up to $(N-1)$ closest ancestors of w.

The maximum likelihood estimator for this probability is:

$$\hat{\mathbb{P}}[w_i|A^{(N-1)}(w_i)] = \frac{C((A^{(N-1)}(w_i), w_i))}{\sum_{w \in \mathcal{V}} C((A^{(N-1)}(w_i), w))}$$

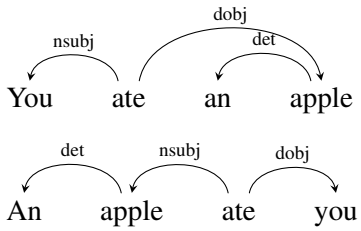
We have arrived at a model which is quite similar to n-gram language models. The main difference

is that each word in the tree can have several children, while in the n-gram models it can only be followed by one word. Thus the sum in the denominator above does not simplify to the count of the ancestor sequence in the way that it does for n-gram language models. However, we can calculate and store the denominators easily during training, so that we do not need to sum over the vocabulary each time we evaluate the estimator. We refer to this model as the **order N unlabelled dependency language model**.

As is the case for n-gram language models, even for low values of N, we will often encounter sequences $(A^{(N-1)}(w), w)$ which were not observed in training. In order to avoid assigning zero probability to the entire sentence, we need to use a smoothing method. We can use any of the smoothing methods used for n-gram language models. For simplicity, we use stupid backoff smoothing (Brants et al., 2007).

3 Labelled Dependency Language Models

We assumed above that the words are generated independently from the grammatical relations. However, we are likely to ignore valuable information in doing so. To illustrate this point, consider the following pair of sentences:



The dependency trees of the two sentences are very similar, with only the grammatical relations between *ate* and its arguments differing. The unlabelled dependency language model will assign the same probability to both of the sentences as it ignores the labels of grammatical relations. In order to be able to distinguish between them, the nature of the grammatical relations between the words in the dependency tree needs to be incorporated in the language model. We relax the assumption that the words are independent of the labels of the parse tree, assuming instead the each word is conditionally independent of the words and labels outside its ancestor path given the words and labels in its ancestor path. We define $G(w_i)$ to be the sequence of grammatical relations between the successive elements of $(A(w_i), w_i)$. $G(w_i)$ is the sequence of grammatical relations found on the path from the root node to w_i. For example, in Figure 1, $G(w_8) = (g_1, g_3, g_8)$. With our modified assumption we have:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|A(w_i), G(w_i)] \qquad (5)$$

Once again we apply a Markov assumption. Let $G^{(N-1)}(w)$ be the sequence of grammatical relations between successive elements of $(A^{(N-1)}(w), w)$. With an $(N-1)^{th}$ order Markov assumption, we have:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|A^{(N-1)}(w_i), G^{(N-1)}(w_i)]$$

The maximum likelihood estimator for the probability is once again given by the ratio of the counts of labelled paths. We refer to this model as the **order N labelled dependency language model**.

4 Dataset and Implementation Details

We carried out experiments using the Microsoft Research Sentence (MSR) Completion Challenge (Zweig and Burges, 2012). This consists of a set of 1,040 sentence completion problems taken from five of the Sherlock Holmes novels by Arthur Conan Doyle. Each problem consists of a sentence in which one word has been removed and replaced with a blank and a set of 5 candidate words to complete the sentence. The task is to choose the candidate word which, when inserted into the blank, gives the most probable complete sentence. The set of candidates consists of the original word and 4 imposter words with similar distributional statistics. Human judges were tasked with choosing imposter words which would lead to grammatically correct sentences and such that, with some thought, the correct answer should be unambiguous. The training data set consists of 522 19th century novels from Project Gutenberg. We parsed the training data using the Nivre arc-eager deterministic dependency parsing algorithm (Nivre and Scholz, 2004) as implemented in MaltParser (Nivre et al., 2006). We trained order N labelled and unabelled dependency

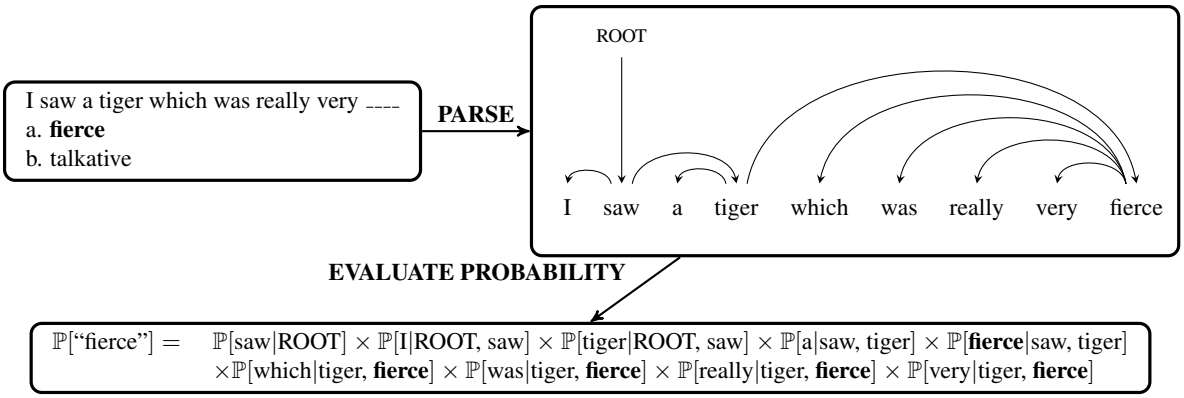


Figure 2: Procedure for evaluating sentence completion problems

N	Unlab-SB	Lab-SB	Ngm-SB	Ngm-KN
2	43.2%	43.0%	28.1%	27.8%
3	48.3%	49.8%	38.5%	38.4%
4	48.3%	**50.0%**	40.8%	41.1%
5	47.4%	49.9%	41.3%	40.8%

Table 1: Summary of results for Sentence Completion

Method	Accuracy
n-grams (Various)	39% - 41%
Skip-grams (Mikolov)	48%
Unlabelled Dependency Model	**48.3%**
Average LSA (Zweig)	49%
Labelled Dependency Model	**50.0%**
Log-bilinear Neural LM (Mnih)	54.7%
Recurrent Neural LM (Mikolov)	55.4%

Table 2: Comparison against previous results

language models for $2 \leq N \leq 5$. Words which occured fewer than 5 times were excluded from the vocabulary. In order to have a baseline to compare against, we also trained n-gram language models with Kneser-Ney smoothing and stupid backoff using the Berkeley Language Modeling Toolkit (Pauls and Klein, 2011).

To test a given language model, we calculated the scores it assigned to each candidate sentence and chose the completion with the highest score. For the dependency language models we parsed the sentence with each of the 5 possible completions and calculated the probability in each case. Figure 2 illustrates an example of this process for the order 3 unlabelled model.

5 Results

Table 1 summarises the results. **Unlab-SB** is the order N unlabelled dependency language model with Stupid Backoff, **Lab-SB** is the order N labelled dependency language model with Stupid Backoff, **Ngm-SB** is the n-gram language model with Stupid Backoff and **Ngm-KN** is the interpolated Kneser-Ney smoothed n-gram language model.

Both of the dependency language models outperfomed the n-gram language models by a substantial margin for all orders considered. The best result was achieved by the order 4 labelled dependency model which is 8.7 points in accuracy better than the best n-gram model. Furthermore, the labelled dependency models outperformed their unlabelled counterparts for every order except 2.

Comparing against previous work (Table 2), the performance of our n-gram baseline is slightly better than the accuracy reported by other authors (Mnih and Teh, 2012; Zweig et al., 2012) for models of this type. The performance of the labelled dependency language model is superior to the results reported for any single model method, apart from those relying on neural language models (Mnih and Teh, 2012; Mikolov et al., 2013) . However the superior performance of neural networks comes at the cost of long training times. The best result achieved in Zweig et al. (2012) using a single method was 49% accuracy with a method based on LSA. Mikolov et al. (2013) also reported accuracy of 48% for a method called skip-grams, which uses a log-linear classifier to predict which words will appear close to each other in sentences.

6 Related Work and Discussion

The best-known language model based on dependency parsing is that of Chelba et al. (1997). This model writes the probability in the familiar left-to-right chain rule decomposition in the linear order of the sentence, conditioning the probability of the next word on the linear trigram context, as well as some part of the dependency graph information relating to the words on its left. The language models we propose are far simpler to train and compute. A somewhat similar model to our unlabelled dependency language model was proposed in Graham and van Genabith (2010). However they seem to have used different probability estimators which ignore the fact that each node in the dependency tree can have multiple children. Other research on syntactic language modelling has focused on using phrase structure grammars (Pauls and Klein, 2012; Charniak, 2001; Roark, 2001; Hall and Johnson, 2003). The linear complexity of deterministic dependency parsing makes dependency language models such as ours more scalable than these approaches.

The most similar task to sentence completion is lexical substitution (McCarthy and Navigli, 2007). The main difference between them is that in the latter the word to be substituted provides a very important clue in choosing the right candidate, while in sentence completion this is not available. Another related task is selectional preference modeling (Séaghdha, 2010; Ritter et al., 2010), where the aim is to assess the plausibility of possible syntactic arguments for a given word.

The dependency language models described in this paper assign probabilities to full sentences. Language models which require full sentences can be used in automatic speech recognition (ASR) and machine translation (MT). The approach is to use a conventional ASR or MT decoder to produce an N-best list of the most likely candidate sentences and then re-score these with the language model. This was done by Chelba et al. (1997) for ASR using a dependency language model and by Pauls and Klein (2011) for MT using a PSG-based syntactic language model.

7 Conclusion

We have proposed a pair of language models which are probabilistic models for the lexicalisation of a given dependency tree. These models are simple to train and evaluate and are scalable to large data sets. We applied them to the Microsoft Research Sentence Completion Challenge. They performed substantially better than n-gram language models, achieving the best result reported for any single method except for the more expensive and complex to train neural language models.

Acknowledgments

Andreas Vlachos is funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 270019 (SPACEBOOK project www.spacebook-project.eu). The authors would like to thank Dr. Stephen Clark for his helpful comments.

References

Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858–867. Association for Computational Linguistics.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131. Association for Computational Linguistics.

Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, et al. 1997. Structure and performance of a dependency language model. In *Proceedings of Eurospeech*, volume 5, pages 2775–2778.

Yvette Graham and Josef van Genabith. 2010. Deep syntax language models and statistical machine translation. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 118–126. Coling 2010 Organizing Committee, August.

Keith Hall and Mark Johnson. 2003. Language modeling using efficient best-first bottom-up parsing. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 507–512. IEEE.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Andriy Mnih and Yee W Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 64. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Adam Pauls and Dan Klein. 2011. Faster and Smaller N-Gram Language Models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 258–267. Association for Computational Linguistics.

Adam Pauls and Dan Klein. 2012. Large-scale syntactic language modeling with treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 959–968. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, et al. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Diarmuid O Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics.

Geoffrey Zweig and Chris JC Burges. 2012. A challenge set for advancing language modeling. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 29–36. Association for Computational Linguistics.

Geoffrey Zweig, John C Platt, Christopher Meek, Christopher JC Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 601–610. Association for Computational Linguistics.