# Predictive Incremental Parsing Helps Language Modeling

**Arne Köhn** and **Timo Baumann**
Department of Informatics
Universität Hamburg
{koehn,baumann}@informatik.uni-hamburg.de

## Abstract

Predictive incremental parsing produces syntactic representations of sentences as they are produced, e. g. by typing or speaking. In order to generate connected parses for such unfinished sentences, upcoming word types can be hypothesized and structurally integrated with already realized words. For example, the presence of a determiner as the last word of a sentence prefix may indicate that a noun will appear somewhere in the completion of that sentence, and the determiner can be attached to the predicted noun. We combine the forward-looking parser predictions with backward-looking N-gram histories and analyze in a set of experiments the impact on language models, i. e. stronger discriminative power but also higher data sparsity. Conditioning N-gram models, MaxEnt models or RNN-LMs on parser predictions yields perplexity reductions of about 6 %. Our method (a) retains online decoding capabilities and (b) incurs relatively little computational overhead which sets it apart from previous approaches that use syntax for language modeling. Our method is particularly attractive for modular systems that make use of a syntax parser anyway, e. g. as part of an understanding pipeline where predictive parsing improves language modeling at no additional cost.

## 1 Introduction

N-gram models are the best performing language models capable of online decoding. Interestingly, they perform well despite discarding all information contained in a sentence except the last N-1 words to predict the next one. At least intuitively, it seems implausible that N-grams are a natural model for sentences. In particular, humans seem to use strong predictive skills that rely on non-local aspects of pragmatics (i.e., the conversational context), semantics (i.e., meaningful continuations), or syntax (i.e., syntactically correct continuations). Sturt and Lombardo (2005) show that humans have strong expectations about upcoming words based on the predicted syntactic structure of the sentence. This includes effects spanning long distances in the sentence. For example, reading a word that does not match expectations slows down the reading process.

Syntactic dependency trees provide long-spanning context information not contained in N-grams. They have, however, not widely been used for language modeling. Previous work (see Sec. 2) has integrated parsing into the language modeling process either by post-hoc (re-)scoring of full sentences or by parsing the partial sentence including the word to be predicted. The first approach is not suitable for online decoding at all and the second is computationally expensive as outlined below.

We propose to use a predictive dependency parser that produces syntactic structure for yet-unfinished sentences. The parser is able to include upcoming ('virtual') nodes in the predicted structure (see Sec. 3). We use this forward-looking information to condition our models. In a range of experiments, we confirm the value of parser predictions using a very simple setup in Sec. 4. In Sec. 5, we integrate the predictions with several types of language modeling approaches, and obtain perplexity reductions in all of them.

## 2 Related Work

Existing approaches that have exploited syntax for language modeling fall into 3 clusters: post-hoc reranking, the application of tree probabilities, and the use of predictions (as in our work).

Recent research focused on the reranking of n-best hypotheses or lattices either based on the probability of the syntax tree for the full sentence (Filimonov and Harper, 2009; Charniak, 2001; Tan et al., 2012) or by training a discriminative model (Collins et al., 2005). A special case of reranking is the task of sentence completion, where one word of a sentence is missing and the correct word out of five possibilities needs to be selected to fill the gap (Zweig and Burges, 2011). For evaluation purposes, the possible words are selected specifically to have a similar probability under a plain N-gram model. Zhang et al. (2016) propose a model based on LSTMs that predicts dependency trees instead of word sequences. They obtain state of the art results for the sentence completion challenge by computing the probabilities of the alternatives and choosing the best one. However, these models do not cope with sentence prefixes, and hence they can only be applied non-incrementally in a rescoring fashion, e.g. to improve ASR results after a first pass over the input data; thus, they are not applicable to interactive use-cases which require incremental processing.

Parsers that assign probabilities to the derived parse tree can be used for language modeling by comparing parse tree probabilities for all possible continuations of the sentence prefix, e. g using a limited-domain hand-written PCFG with learned rule-weights (Jurafsky et al., 1995), or by learning structure using a hierarchical HMM (Schwartz et al., 2011). As an alternative, Roark (2001) estimates the probability of a sentence prefix by measuring the probability of all parse trees (within a beam) that are derivable for the prefix ending in the queried N-gram divided by the probability of all parse trees for the prefix ending at the N-1'th word. For these approaches, sentence prefixes including the N'th word must be computed, which is computationally expensive. When obtaining a probability distribution for the next word, the cost grows with the vocabulary size. In contrast, our approach parses only once, up to the N-1'th word (i. e. the context portion of the prefix when determining the N'th word).

Chelba and Jelinek (1998) use a tree adjoining grammar to parse the sentence prefix up to (but not including) the word being queried, and the parser state (the top two elements of the stack) is used to condition the language model upon. The probability is summed over all possible trees for the prefix. Thus, again many parses are required for a single query, resulting in a high complexity. Nevertheless, this approach shows many similarities to ours as parsing is done incrementally and the parser state implicitly encodes information about upcoming structure. In our model, we make the predictions explicit which may also be useful to other modules in a system.

The main contributions of our work are as follows: first, our method only needs raw text (not tree-banks) for estimating syntactic LMs (as our parser is trained independently) while retaining high run-time performance on single machines. Tan et al. (2012) use a similar amount of data but need 400 servers to apply their admittedly more sophisticated models which also include semantics and topic for full-document modelling. Second, our approach is applicable to online decoding for incremental systems such as highly interactive spoken dialogue systems, whereas other methods are often limited to post-hoc lattice rescoring.

## 3 Predictive Parsing

Dependency parsers generate the syntax tree for a sentence by attaching every token (the dependent of the relation) to another token (the head) or a root node. For sentence prefixes, the head of a dependent may not yet be available resulting in an incomplete tree (as in Figure 1 (a)).

Trees can be completed by using what Beuck et al. (2013) call virtual nodes which serve as stand-ins for upcoming words. Words without a matching head in the prefix can be attached to a virtual node (see Figure 1 (b)) which is fully integrated into the syntactic structure. Such nodes can also be introduced when no word needs to be attached to them, e.g. because otherwise a valency can not be satisfied (see Figure 1 (c)).

Each virtual node has a type corresponding to the PoS of the words it can stand for. We use the prediction about the upcoming word types for language modeling, following the intuition that e. g. a verb might be more likely when the parser predicts that a verb is missing to complete the sentence.

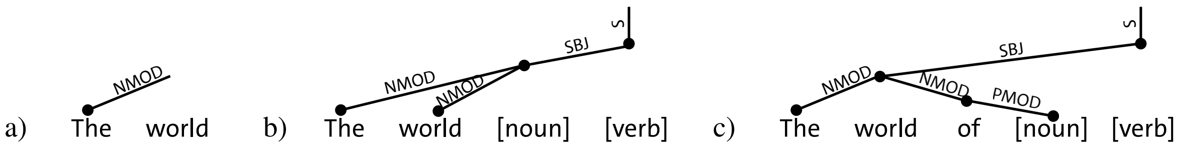


Figure 1: Dependency trees for prefixes of the sentence *The world of politics has no rules.* showing: (a) an incomplete tree (no head exists for *world*), (b) a predicted noun and verb head (lexicalization leads the parser to assume that *world* is part of a more complex noun phrase), and (c) an additional predicted noun dependent when *of* is added. (b) and (c) are actual outputs of the predictive parser.

We use a parser based on TurboParser (Martins et al., 2013) which was extended by Köhn and Menzel (2014) to produce predictive analyses of sentence prefixes. The parser predicts up to one upcoming verb and one upcoming noun, which results in a good coverage/precision trade-off for English. The parser is trained on the Penn-Treebank (Marcus et al., 1994) and not specifically tuned for language modeling nor for the corpus used for estimating the LM. Predictions of $vn$ nodes reach an F-score of .68 (precision: .79; recall: .60). The parser output for full sentences is as accurate as the original TurboParser.

Throughout this paper, we abbreviate the prediction of the virtual verb and noun as $vn$ where $v$ stands for the verb and $n$ for the noun. For specific predictions, $\overset{+}{v}$ and $\overset{+}{n}$ denotes that the corresponding virtual node was predicted, $\bar{v}$ and $\bar{n}$ that it was not. The distribution of predictions for sentence prefixes is shown in Table 1; as can be seen, both $v$ and $n$ split the prefixes roughly equally into four parts, yielding an information of 1.90 bit. Other forward-looking syntactic features (e.g. dependency direction of an expected virtual node or the dependency type) would likely be less informative, hence we focus on only these two features in our work. We experiment with models that use both or just one of the predictions.

| sentence prefixes | | verb predicted | | |
|---|---|---|---|---|
| | | $\bar{v}$ | $\overset{+}{v}$ | Sum |
| noun predicted | $\bar{n}$ | 41.6 | 22.7 | **64.3** |
| | $\overset{+}{n}$ | 18.7 | 17.0 | **35.7** |
| | Sum | **60.3** | **39.7** | |

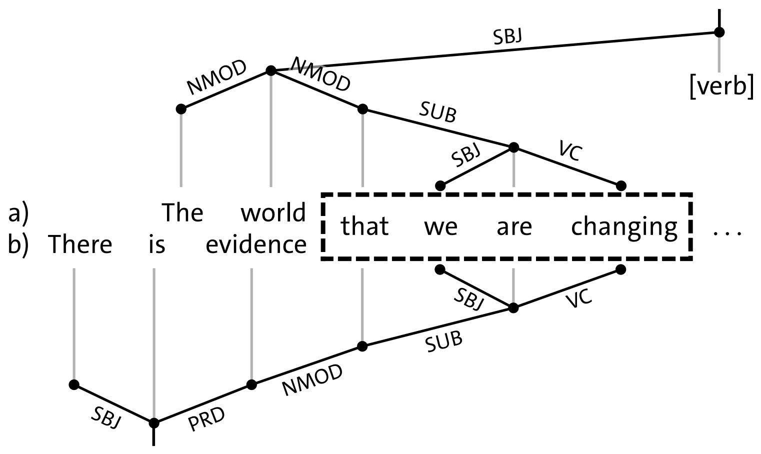
Table 1: Distribution (in percent) of virtual node predictions for sentence prefixes in the Billion Word Corpus.

## 4 Basic Idea and Proof-of-Concept

Our experiments are based on the assumption that probabilities for a word (such as "likes") depend on whether a verb (or noun) is structurally required to complete the sentence. In addition, we believe that this *forward-looking* structural information is not well-captured by the *backward-looking* local context of an N-gram. Notice that we do not make any assumption on the kinds of probability shifts but only the assumption that *some* shift occurs, i.e. "likes" might as well be preferred when a (plural) noun is predicted.

A language model computes the probability of a sentence: $P(w_1 \ldots w_n)$. Applications such as speech recognition perform online decoding and use the decomposition according to the chain rule $P(w_1 \ldots w_n) = \prod_{i=1}^{n} P(w_i|w_1 \ldots w_{i-1})$, as well as the (N-gram) approximation based on the Markov assumption ($P(w_i|w_1 \ldots w_{i-1}) \approx P(w_i|w_{i-N} \ldots w_{i-1})$). This approximation limits language models to local contexts but makes the probability estimation problem tractable by reducing data sparsity. We enhance the model by including parser predictions $p_{i-1} \in \{\overset{+}{v}\overset{+}{n}, \overset{+}{v}\bar{n}, \bar{v}\overset{+}{n}, \bar{v}\bar{n}\}$ extracted from the predictive parse for the *full* history of the sentence $w_1 \ldots w_{i-1}$. Thus, the parser predictions encode additional information about the continuation that enhance the local history in a very dense form – using only two bits.

The non-locality of parsing predictions is exemplified in Figure 2 (the example is restricted to just the verb prediction $v$). In the two sentences given in the figure, the local N-gram context is identical for either sentence prefix. In fact, the cause for different parser predictions can be arbitrarily far into the past (for example by exchanging *we* for a complex noun phrase in the example). For the given N-gram context, we queried language models that make use of the (non-)prediction of verbs ($\overset{+}{v}$ vs. $\bar{v}$, as described below) and differences in $\log_{10}$ probabilities for some continuations of either sentence are shown in tabular form. As expected, we find that plausible continuations of Sentence 2 (a) are much more probable in the $\overset{+}{v}$ case, and plausible continuations of 2 (b) are more probable in the $\bar{v}$ case.



| | continuation | $\log_{10}$prob difference between $\overset{+}{v}/\bar{v}$ model |
|---|---|---|
| $\overset{+}{v}$ | has/hasn | $+1.18$ / $+1.12$ |
| | will | $+1.10$ |
| | is/isn | $+1.08$ / $+0.97$ |
| | and | $+0.05$ |
| | in | $-0.06$ |
| $\bar{v}$ | worlds | $-0.34$ |
| | it | $-1.28$ |
| | a/an | $-1.41$ / $-1.35$ |
| | our | $-1.69$ |
| | lives | $-2.52$ |

Figure 2: Predictive parses for two sentence beginnings (left). While the parser predicts a verb to be neccessary to complete (a), it does not predict a verb for (b). The differences in the sentences (that cause the different parses) lie outside of the local N-gram context (dashed box). Possible continuations and the relative differences in assigned probabilities depending on $\overset{+}{v}/\bar{v}$ context are shown in tabular form (right).

To make use of the predictions in an N-gram model, we split the N-gram counts by the parser predictions of the corresponding prefix histories: an N-gram $(w_{i-N} \dots w_{i-1} w_i)$ is sorted according to $p_{i-1}$.[1] With the prefixes split into four bins ($\overset{+}{v}\overset{+}{n}$, $\overset{+}{v}\bar{n}$, $\bar{v}\overset{+}{n}$, and $\bar{v}\bar{n}$), we train four separate N-gram models. During decoding, we switch between these four models: given a sentence prefix $(w_1 \dots w_{i-1})$, we extract $p_{i-1}$ and query the corresponding model for the probability of $w_i$. Note that the output of the parser does not depend on $w_i$ and we therefore only need to parse once for every sentence prefix even if we query the full probability distribution (as in speech recognition). We also perform experiments with only two bins by splitting either by $v$ or by $n$, obtaining two N-gram models.

All experiments in this paper are carried out on the Billion Word Corpus (BWC; Chelba et al. (2013)), for which we parsed every sentence prefix.[2] N-gram models are generated with SRILM (Stolcke et al., 2011) using standard settings (modified Kneser-Ney smoothing (Chen and Goodman, 1996) and interpolation, a limited vocabulary of 100,000 words, and dropping singleton N-grams for N>2).[3]

### 4.1 Exemplary Effects of Parsing Splits on Trigrams

We first illustrate the effects of parser predictions on probability distributions for prefixes ending in *the world*. The probabilities of the most frequent continuations for *the world* (i.e., standard N-gram probabilities) as well as probabilities under the four different split models are shown in Table 2 (middle part). We discuss the highlighted numbers: *the world 's* has a high probability when a noun is predicted (particularly $\bar{v}\overset{+}{n}$). This subsumes cases where a verb has a valency to which *world* is a bad fit.[4] Hence, *the world* is regarded as a possessive adjective to the predicted argument and *'s* makes the connection between the two. The continuation "." has a high probability when neither a verb nor a noun is predicted to continue the sentence. A comma is more likely under the $\overset{+}{v}\overset{+}{n}$ model, presumably because this constellation dominates at the end of introductory prepositional phrases. Lastly, *and* is most likely under $\bar{v}\bar{n}$, similarly to the full stop, as it connects material to an already complete structure.

While these observations may or may not be interesting linguistically, our main point is that probabilities shift (sometimes radically, as highlighted in the right part of the table) when dissecting the N-gram history by our longer-range $vn$ information. This increased discriminatory power is what our method is about.

[1]Simply including parser prediction in an N-gram model like a context word (using the N-grams $(w_{i-N} \dots w_{i-1} p_{i-1} w_i)$) would break the count-of-count assumptions for Kneser-Ney smoothing.

[2]This takes 30 ms per prefix, totalling about one CPU year; parsing results as well as code to replicate our experiments are available at `https://nats-www.informatik.uni-hamburg.de/PIPLM` to encourage further research.

[3]SRILM allows to manipulate N-gram counts and is still able to compute correct backoff values unlike e.g. KenLM which, however, would allow to include singleton N-grams. Data sparsity (see below) would be less grave if singletons were included.

[4]Note that since our parser is lexicalized, it automatically learns valency relations.

| | probabilities | | | | | relative change | | | |
|---|---|---|---|---|---|---|---|---|---|
| tri-gram | avg | $\overset{+}{v}\overset{+}{n}$ | $\overset{+}{v}\overset{-}{n}$ | $\overset{-}{v}\overset{-}{n}$ | $\overset{-}{v}\overset{+}{n}$ | $\overset{+}{v}\overset{+}{n}$ | $\overset{+}{v}\overset{-}{n}$ | $\overset{-}{v}\overset{-}{n}$ | $\overset{-}{v}\overset{+}{n}$ |
| the world 's | .33 | .41 | .37 | .28 | **.81** | ×1.2 | ×1.1 | ÷1.2 | ×2.5 |
| the world . | .14 | .02 | .03 | **.19** | .01 | **÷7.0** | **÷4.9** | ×1.4 | **÷13** |
| the world , | .10 | **.17** | .07 | .11 | .02 | ×1.6 | ÷1.5 | ×1.1 | **÷4.9** |
| the world and | .02 | .01 | .01 | **.03** | .004 | ÷2.2 | ÷2.2 | ×1.3 | **÷5.3** |
| the world *everything else* | .41 | .39 | .52 | .39 | .16 | ÷1.1 | ×1.3 | ÷1.1 | ÷2.6 |

Table 2: Some tri-grams and their probabilities in the full corpus (avg) vs. split by parser prediction.

## 4.2 The Detrimental Effect of Data Sparsity

The splitting method outlined above increases data sparsity in the sub-models for each of the four $vn$ bins. We here differentiate between gains from parsing prediction-induced discriminatory power and losses from splitting-induced data sparsity. For this purpose, we devise a baseline model which replaces the parser predictions with random bins, using the same distribution as in Table 1. If the parser's predictions were to provide no additional information, both models should obtain the same perplexity, as they are comparably impacted by data sparsity.

Table 3 shows the cross-entropies obtained by our syntax splitting model, the baseline model, and standard N-gram models trained on the full material.[5] We find that data sparsity inhibits the results for N>2-grams. While our method achieves a gain from syntax predictions (e. g. ~.3 bit for 4/5-grams), additional data sparsity deteriorates results slightly more (~.4 bit for 4/5-grams). Nevertheless, we find that syntax gain even increases with N-gram order, that is higher-order N-grams do not implicitly encode longer-range information in the same way that syntax predictions do. We also find that both $v$ and $n$ help similarly and that their gains roughly add up (i. e. they are orthogonal). In total, each of $v$/$n$ in isolation shows slightly better performance than combined $vn$ splitting given their lower data sparsity.

## 4.3 Exemplary Comparison of Split and Standard Model

To further analyze the strengths and weaknesses of the split model, we look word-by-word at the probability assignments of standard 5-gram models and $vn$ state-specific 5-gram models. We restrict this analysis to 500 test sentences with 6-14 tokens and focus attention on sentences with large differences.

In the qualitative analysis, we find some patterns that are illustrated by the sentence plots in Figure 3. Each plot shows for every word the probability assigned by either model, the parser's prediction for the prefix leading up to this word, and the backoff order necessary if the queried 5-gram is not contained in the model. The backoff order is a strong indicator for data sparsity pressure.

[5]Entropies reported in Table 3 are higher than for the final N-gram results in Section 5.1, as the preliminary models employed here are not order-interpolated and sentence-ends are excluded from perplexity computation.

| | | syntax pred. A: | random splits B: | standard C: | syntax gain B-A: | splitting loss B-C: |
|---|---|---|---|---|---|---|
| 2-grams | $vn$ | 7.819 | 7.968 | 7.881 | 0.149 | 0.087 |
| | $v$ | 7.837 | 7.918 | | 0.081 | 0.037 |
| | $n$ | 7.861 | 7.917 | | 0.057 | 0.036 |
| 3-grams | $vn$ | 6.988 | 7.209 | 6.938 | 0.221 | 0.271 |
| | $v$ | 6.969 | 7.071 | | 0.102 | 0.133 |
| | $n$ | 6.959 | 7.067 | | 0.108 | 0.129 |
| 4-grams | $vn$ | 6.735 | 7.025 | 6.633 | 0.290 | 0.392 |
| | $v$ | 6.703 | 6.839 | | 0.136 | 0.206 |
| | $n$ | 6.671 | 6.830 | | 0.159 | 0.197 |
| 5-grams | $vn$ | 6.685 | 6.994 | 6.566 | 0.309 | 0.428 |

Table 3: Cross-entropies (in bit; lower is better) obtained by splitting the training data according to the parser's prediction (verb, noun, or both), vs. random splitting and comparison to standard models.

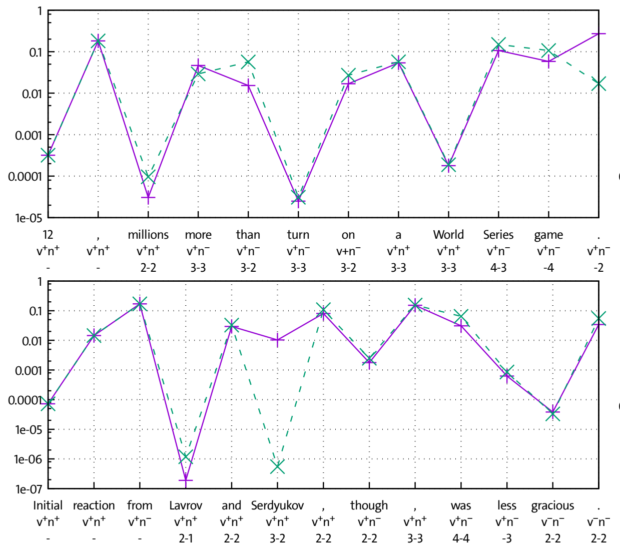


Figure 3: Sentences with their per-word probabilities in standard and split models. For each word, we show syntax prediction and the N-gram order backed off by the standard and split model respectively, if backoff is needed (e.g., “Lavrov $\overset{+}{v}\overset{+}{n}$ 2-1” indicates the parser predicting both $\overset{+}{v}$ and $\overset{+}{n}$ for the prefix up to “from” and both models backing off: the standard model to bigrams, the split model even to unigrams).

Probability estimates are **identical for the N-1 sentence-initial words** as syntax predictions are implicitly contained in the N-gram history and both models provide the same information (lower N if a model is forced into backoff). Models perform similarly under no or little data sparsity, i. e. no backoff (or just to 4-grams). **Moderate improvements** often occur **under data sparsity** when both models back off to the same order. The first sentence shows some improvements for the syntax predictions even when using lower N-gram orders (syntax being more useful than one more word of history) but fails at predicting the sentence final full stop (the parser still expects a verb and this could be classified as a parsing error), in this case due to the incompleteness of the sentence (it is both authors’ feeling that this sentence *should* go on).

Inherently, the standard model is never impacted more strongly by data sparsity than the split model. Thus, **critical cases occur when the split model needs to back off more** than the standard model. While positive effects are also noticeable to some extent for the second and third sentence, these sentences are hurt badly by not ‘knowing’ infrequent co-occurrences, i. e. N-grams in which the last word has a high conditional probability given the full history of the N-gram: “North American stores” (where “stores” is a likely continuation for “North American” but not for “American” *per se*), and “Lavrov and Serdyukov” (where “Lavrov” must be in the history to assign a relevant probability, yet the split model backs off to “and Serdyukov”). Further analysis of this last example shows that there are two occurrences of the trigram in the training data, which fall into two different prediction states ($\overset{+}{v}\overset{+}{n}$ and $\overset{-}{v}\overset{+}{n}$). As singleton N-grams are ignored, both occurrences of the trigram are lost and not used in the split model.

A histogram of $\log_{10}$prob differences is shown in Figure 4 (left side; notice the logarithmic y-axis). The majority of the differences are positive (55 %), indicating that our method is helpful more often than not. However, the very long tail of rare gross mistakes results in a overall slightly negative $\log_{10}$prob loss of -0.025, in line with our analysis of the examples in Figure 3 and the overall loss in Table 3.

In summary, we find that most often, probability estimates are similar or slightly improve with syntax predictions as used by the split model due to the better discrimination from non-local context. On the other hand, splitting increases the random variation of N-gram counts in particular for rarely seen N-grams; more often than not, the positive effect appears to outweigh the negative effect. Unfortunately, few but large errors are introduced systematically by the fact that each of the split model’s sub-models is trained on less data. We deal with data sparsity in the following section, where we build prediction-enhanced models that outperform their baselines.

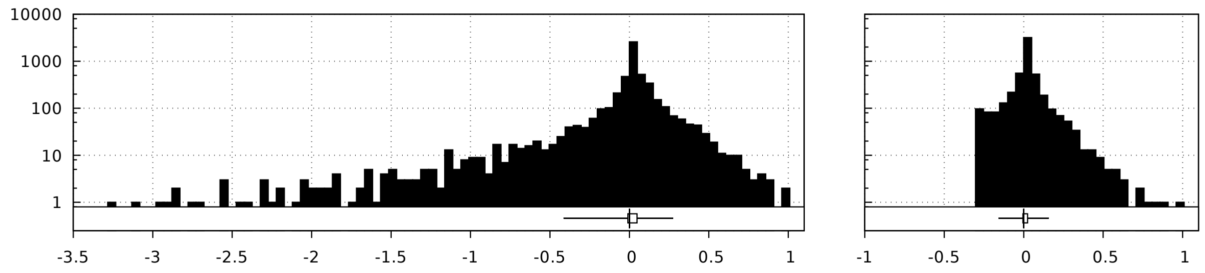


Figure 4: Left: Histogram over $\log_{10}$prob differences between the split and the standard model. Right: Differences between the interpolated and standard model (see Section 5.1). Notice the logarithmic y-axis. The boxplots below show 25/75% (box) and 5/95% (whiskers) intervals and median.

## 5 Experiments

To assess the merit of predictive parsing, we perform a wide range of experiments. We use SRILM (Stolcke et al., 2011) for N-grams on the BWC, for which Chelba et al. (2013) quote a cross-entropy of 6.08 bit for unpruned interpolated Kneser-Ney smoothed 5-grams. We build MaxEnt N-gram models (Rosenfeld, 1994) as well as RNN-LMs (Mikolov et al., 2011) using faster-rnnlm.[6] For the BWC, faster-rnnlm in its best setting (after extensive hyper-parameter search) results in a cross-entropy of 6.8 bit and combined with its MaxEnt model of 6.4 bit (the MaxEnt model alone has not previously been benchmarked).[6]

The numbers reported below are not necessarily directly comparable to reference numbers for several reasons: we use limited-vocabulary N-grams and separate OOV handling as we plan to apply our models to speech recognition. In contrast, faster-rnnlm by default discards all sentences containing OOV tokens which results in lower perplexity. We use SRILM defaults (pruned N-grams) to create models that easily fit in memory, and do not search for optimal RNN hyper-parameters. Nonetheless, given the breadth of our experiments, the general findings of our work are likely independent of these particularities.

### 5.1 Interpolated N-gram models

We first present interpolated N-gram models that reduce the impact of additional data sparsity in the plain syntax splitting models presented in Section 4.2. Interpolation exploits the averaging effect when combining models with different characteristics. As noted above, the syntax-enhanced model is slightly better *often* but much worse sometimes (see Figure 4, left side). Using interpolation, the outliers can be greatly reduced in magnitude by averaging the models; in particular, a probability from the interpolated model is at worst half as probable as the probability from the better model. Therefore, no $\log_{10}$prob difference is smaller than $log_{10}(0.5) \approx -0.3$ (cmp. Figure 4, right side). Although the magnitude of improvements is also reduced, that effect is much smaller and the overall average advantage over the standard model turns from negative to positive. In effect, interpolation allows to optimize the tradeoff between discrimination and accumulation of evidence under data sparsity, and we try several strategies:

**1.** The split model uses two binary predictions, thus each of the sub-models is trained on (roughly) one quarter of the training data, only. One possible route to improvements is to build separate sub-models for the $v$ and the $n$ variable (**1a**), each of which can be trained with roughly half of the training data, i.e. with less data sparsity. We can then select the two models that match the $vn$ state and interpolate between them. An alternative approach for reducing data sparsity (**1b**) is to join the N-gram counts of the corresponding $v$ and $n$ state and to build pre-aggregated models. (E. g. the model for $\bar{v}\overset{+}{n}$ would accumulate counts from the $\bar{v}$ and the $\overset{+}{n}$ bins.) Either way, we account for the fact that an N-gram may be seen during testing in a different $vn$ state than in training (that is nevertheless related by one of the variables).

**2.** We interpolate between the standard model which is trained on all data and the split models which are more discriminative (**2a**). We also combine this method with **1b** as **2b**.

The cross-entropies of the plain syntax-splitting method (**0**) and the standard 5-gram model as well as the experimental conditions described above are given in Table 4. We find that interpolating the $vn$

[6]github.com/yandex/faster-rnnlm

| model type | setting | standard | syntax | gain |
|---|---|---|---|---|
| 5-grams | **0.** plain splitting | 6.253 | 6.374 | −0.12 |
| | **1a.** interp. from $v \times n$ | | 6.254 | −0.001 |
| | **1b.** joint counts | | 6.187 | 0.07 |
| | **2a.** $vn$ interp. w/ std | | 6.234 | 0.02 |
| | **2b.** joint counts interp. w/ std | | 6.170 | **0.08** |
| MaxEnt | 2-grams | 8.41 | 8.31 | 0.10 |
| | 3-grams | 7.86 | 7.78 | 0.08 |
| | 4-grams | 7.88 | 7.87 | 0.01 |
| | 4-grams (large GPGPU) | 7.55 | 7.49 | **0.06** |
| RNN-LM | syntax in input | 7.94 | 7.90 | 0.04 |
| | syntax in NCE | | 7.89 | **0.05** |

Table 4: Cross-entropies (in bit; lower is better) of the experiments presented in Section 5, for 5-grams, MaxEnt models, and RNN-LMs, as discussed in the respective sub-sections, with the standard model, syntax-enhanced model, and gain (or loss) between the two. As can be seen, the baseline standard models are outperformed for all model types.

models from $v$ and $n$ leads to a large improvement, in particular when using more training data for each individual model (**1b**) rather than interpolating from $v$ and $n$ models (**1a**). Interpolating between the standard and plain syntax-splitting models (**2a**) leads to good results (which peaked at an interpolation weight of .367 for the splitting model). Finally, interpolation of **1b** with the standard model (weight .675) outperforms the standard model by 0.083 bit (a perplexity improvement of 6 % from 76.3 down to 72.0).

## 5.2 Maximum Entropy Models

We extend the faster-rnnlm MaxEnt model with parser predictions. In the model, the score for a word $w_i$ is computed by summing over all N-gram scores: $s(w_i) = \sum_{j=0}^{n} f(w_i \dots w_{i-j})$. During decoding, the softmax over all possible words $w_i$ yields a probability distribution. $f$ hashes the N-gram (as well as lower order N-grams) and then looks up their scores in a fixed-size table. We extend $s$ by adding $vn$-annotated (N…1)-grams to the summation: $s(w_i) = \sum_{j=0}^{n} f(w_i \dots w_{i-j}) + f(p_{i-1} w_i \dots w_{i-j})$ We use a hash table with 400M elements, the maximum for our GPGPU card.

Results are presented in Table 4, showing averages of several runs (to account for random initialization). We find that the $vn$-enhanced models perform significantly better than the standard ones.[7] Performance did not improve with N>3 with our setup, likely due to an increase in hash collisions, as our method puts almost five times as many items into the table. We tested 4-grams on a larger GPGPU card allowing a hash table with 1,600M elements and see that both overall performance and syntax gain improve as expected.

## 5.3 RNN-based Models

We explore how to integrate parser predictions into a RNN-LM in two different ways as depicted in Figure 5. Either, we add $vn$ as two additional dimensions to the input layer, with positive weight if the corresponding virtual node was predicted and with negative weight otherwise. For example, $\bar{v}\overset{+}{n}$ translates to $(-0.5, 0.5)$. Alternatively, we add $vn$ to the output layer, which is used for noise contrastive estimation (NCE) because intuitively, parser predictions do not need to be fed forward to the next prediction.

For all experiments, we use faster-rnnlm with noise contrastive estimation and a sigmoid layer of size 100 (we ignore MaxEnt here and test RNN in isolation). To account for random initialization (and to allow the estimation of significance), each experiment condition is run 6 times. For both versions, we compared results with a corresponding standard RNN model as reported in Table 4. Both methods significantly[7] outperform the baseline model, and the addition of $vn$ to the NCE layer is slightly better. We conclude that explicitly encoded syntactic long-range information even helps for RNN-LMs which in principle are able to capture longer-range dependencies, at least implicitly.

[7]Wilcoxon rank sum test based on the multiple runs of each experiment condition, $p < .01$.

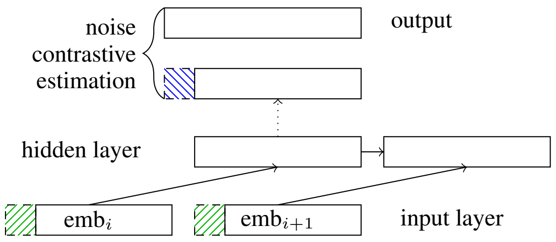


Figure 5: Choice for RNN-LM enhancement, hatched. Either as input (bottom, ▨) or in NCE (top, ▨).

## 6 Summary and Discussion

We have explored the integration of syntactic structure for online decodable language models, which we base on predictive parsing. This approach incurs a far lower computational cost than previous work on integrating parsing into language models as we parse just once for the history and use the prediction to generate the full probability distribution over possible next words. Our model is built with loose coupling in mind: the parser model is tuned for high syntax parsing accuracy and in no way for language modeling. It can hence be used for other purposes (such as incremental understanding) in an integrated system as well. If a predictive parser is already used in such a system, our proposed approach yields a perplexity improvement for free by feeding back parser predictions to the language model.

Predictions encode information about the next words that are not present in N-grams, i. e. they are able to transport information over long distances in a dense form. Explicit syntax predictions also help RNN-LMs which at least implicitly already model longer-range dependencies. We find that both verb and noun predictions are valuable for language modeling. Combined, they provide a gain of ~0.3 bit for 4- and 5-grams over the split baseline (cmp. Section 4.2). Despite the additional data sparsity introduced by the $vn$ predictions, using count-merging and interpolation yields a perplexity improvement of 6 % over standard N-grams. We also integrated predictions into MaxEnt and RNN-LMs and find consistent and modest improvements for both.

We have used a limited vocabulary and discarded singleton N-grams for computational efficiency. We do not believe that our results will be fundamentally different at larger vocabularies or with unpruned models, but we plan to improve our setup to assess the effect nonetheless in the future. We also plan to integrate our models with online speech recognition to asses whether the observed perplexity reduction carries over to WER. Additional information extracted from the dependency tree may yield further improvements, especially with RNN models, which are robust against data sparsity. Finally, improvements to the parser that result in better predictions should transfer to language modeling as well.

We plan to extend our work beyond English data, as predictive parsing is language independent. However, Beuck and Menzel (2013) find a larger set of virtual nodes to be optimal for German for which we want to assess the merit to our method.

**Acknowledgements** We would like to thank Wolfgang Menzel for helpful discussion and comments, and the anonymous reviewers for valuable suggestions and remarks. This work is supported by a Daimler and Benz Foundation PostDoc Grant to the second author.

## References

Niels Beuck and Wolfgang Menzel. 2013. Structural prediction in incremental dependency parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 7816 of *Lecture Notes in Computer Science*, pages 245–257. Springer, Berlin.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2013. Predictive incremental parsing and its evaluation. In Kim Gerdes, Eva Hajičová, and Leo Wanner, editors, *Computational Dependency Theory*, volume 258 of *Frontiers in Artificial Intelligence and Applications*, pages 186 – 206. IOS press.

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

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th Int. Conf. on Computational Linguistics*, volume 1, pages 225–231, Montréal, Canada, August. ACL.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, USA, June. ACL.

Michael Collins, Brian Roark, and Murat Saraclar. 2005. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 507–514, Ann Arbor, USA, June. ACL.

Denis Filimonov and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09): Volume 3*, pages 1114–1123, Singapore, August. ACL.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchaman, and Nelson Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Acoustics, Speech, and Signal Processing. ICASSP-95., International Conference on*, volume 1, pages 189–192, Detroit, USA, May. IEEE.

Arne Köhn and Wolfgang Menzel. 2014. Incremental predictive parsing with TurboParser. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2: Short Papers, pages 803–808, Baltimore, USA, June. ACL.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Plainsboro, USA, March. ACL.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August. ACL.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201, Waikoloa, USA, December. IEEE.

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

Ronald Rosenfeld. 1994. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, USA.

Lane Schwartz, Chris Callison-Burch, William Schuler, and Stephen Wu. 2011. Incremental syntactic language models for phrase-based translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 620–631, Portland, USA, June. ACL.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, Waikoloa, USA, December. IEEE.

Patrick Sturt and Vincenzo Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science*, 29(2):291–305.

Ming Tan, Wenli Zhou, Lei Zheng, and Shaojun Wang. 2012. A scalable distributed syntactic, semantic, and lexical language model. *Computational Linguistics*, 38(3):631–671.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 310–320, San Diego, USA, June. ACL.

Geoffrey Zweig and Christopher J.C. Burges. 2011. The Microsoft Research sentence completion challenge. Technical Report MSR-TR-2011-129, Microsoft Research, Redmond, USA, December.