A Joint Language Model With Fine-grain Syntactic Tags

Denis Filimonov[1]
[1]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park
den@cs.umd.edu

Mary Harper[1,2]
[2]Human Language Technology Center of Excellence
Johns Hopkins University
mharper@umiacs.umd.edu

Abstract

We present a scalable joint language model designed to utilize fine-grain syntactic tags. We discuss challenges such a design faces and describe our solutions that scale well to large tagsets and corpora. We advocate the use of relatively simple tags that do not require deep linguistic knowledge of the language but provide more structural information than POS tags and can be derived from automatically generated parse trees – a combination of properties that allows easy adoption of this model for new languages. We propose two fine-grain tagsets and evaluate our model using these tags, as well as POS tags and SuperARV tags in a speech recognition task and discuss future directions.

1 Introduction

In a number of language processing tasks, particularly automatic speech recognition (ASR) and machine translation (MT), there is the problem of selecting the best sequence of words from multiple hypotheses. This problem stems from the *noisy channel* approach to these applications. The noisy channel model states that the observed data, e.g., the acoustic signal, is the result of some input translated by some unknown stochastic process. Then the problem of finding the best sequence of words given the acoustic input, not approachable directly, is transformed into two separate models:

$$\underset{w_1^n}{argmax}\ p(w_1^n|A) = \underset{w_1^n}{argmax}\ p(A|w_1^n) \cdot p(w_1^n) \quad (1)$$

where A is the acoustic signal and w_1^n is a sequence of n words. $p(A|w_1^n)$ is called an acoustic model and $p(w_1^n)$ is the language model[1].

Typically, these applications use language models that compute the probability of a sequence in a generative way:

$$p(w_1^n) = \prod_{i=1}^{n} p(w_i|w_1^{i-1})$$

Approximation is required to keep the parameter space tractable. Most commonly the context is reduced to just a few immediately preceding words. This type of model is called an *ngram* model:

$$p(w_i|w_1^{i-1}) \approx p(w_i|w_{i-n+1}^{i-1})$$

Even with limited context, the parameter space can be quite sparse and requires sophisticated techniques for reliable probability estimation (Chen and Goodman, 1996). While the ngram models perform fairly well, they are only capable of capturing very shallow knowledge of the language.

There is extensive literature on a variety of methods that have been used to imbue models with syntactic and semantic information in different ways. These methods can be broadly categorized into two types:

- The first method uses surface words within its context, sometimes organizing them into deterministic classes. Models of this type include: (Brown et al., 1992; Zitouni, 2007), which use semantic word clustering, and (Bahl et al., 1990), which uses variable-length context.
- The other method adds stochastic variables to express the ambiguous nature of surface words[2]. To obtain the probability of the next

[1]Real applications use $argmax_{w_1^n}\ p(A|w_1^n) \cdot p(w_1^n)^\alpha \cdot n^\beta$ instead of Eq. 1, where α and β are set to optimize a heldout set.

[2]These variables have to be predicted by the model.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1114–1123,
Singapore, 6-7 August 2009.

word we need to sum over all assignments of the stochastic variables, as in Eq. 2.

$$p(w_i|w_1^{i-1}) = \sum_{t_1 \ldots t_i} p(w_i t_i|w_1^{i-1} t_1^{i-1}) \quad (2)$$
$$= \frac{\sum_{t_1 \ldots t_i} p(w_i t_i|w_1^{i-1} t_1^{i-1}) p(w_1^{i-1} t_1^{i-1})}{\sum_{t_1 \ldots t_{i-1}} p(w_1^{i-1} t_1^{i-1})}$$

Models of this type, which we call *joint* models since they essentially predict joint events of words and some random variable(s), include (Chelba and Jelinek, 2000) which used POS tags in combination with "parser instructions" for constructing a full parse tree in a left-to-right manner; (Wang et al., 2003) used SuperARVs (complex tuples of dependency information) without resolving the dependencies, thus called *almost parsing*; (Niesler and Woodland, 1996; Heeman, 1999) utilize part of speech (POS) tags. Note that some models reduce the context by making the following approximation:

$$p(w_i t_i|w_1^{i-1} t_1^{i-1}) \approx p(w_i|t_i) \cdot p(t_i|t_1^{i-1}) \quad (3)$$

thus, transforming the problem into a standard HMM application. However, these models perform poorly and have only been able to improve over the ngram model when interpolated with it (Niesler and Woodland, 1996).

Although joint models have the potential to better express variability in word usage through the introduction of additional latent variables, they do not necessarily perform better because the increased dimensionality of the context substantially increases the already complex problem of parameter estimation. The complexity of the space also makes computation of the probability a challenge because of space and time constraints. This makes the choice of the random variables a matter of utmost importance.

The model presented in this paper has some elements borrowed from prior work, notably (Heeman, 1999; Xu and Jelinek, 2004), while others are novel.

1.1 Paper Outline

The message we aim to deliver in this paper can be summarized in two theses:

- *Use fine-grain syntactic tags in a joint LM.* We propose a joint language model that can be used with a variety of tagsets. In Section 2, we describe those that we used in our experiments. Rather than tailoring our model to these tagsets, we aim for flexibility and propose an information theoretic framework for quick evaluation for tagsets, thus simplifying the creation of new tagsets. We show that our model with fine-grain tagsets outperform the coarser POS model, as well as the ngram baseline, in Section 5.

- *Address the challenges* that arise in a joint language model with fine-grain tags. While the idea of using joint language modeling is not novel (Chelba and Jelinek, 2000; Heeman, 1999), nor is the idea of using fine-grain tags (Bangalore, 1996; Wang et al., 2003), none of prior papers focus on the issues that arise from the combination of joint language modeling with fine-grain tags, both in terms of reliable parameter estimation and scalability in the face of the increased computational complexity. We dedicate Sections 3 and 4 to this problem.

In Section 6, we summarize conclusions and lay out directions for future work.

2 Structural Information

As we have mentioned, the selection of the random variable in Eq. 2 is extremely important for the performance of the model. On one hand, we would like for this variable to provide maximum information. On the other hand, as the number of parameters grow, we must address reliable parameter estimation in the face of sparsity, as well as increased computational complexity. In the following section we will compare the use of SuperARVs, POS tags, and other structural tags derived from parse trees.

2.1 POS Tags

Part-of-speech tags can be easily obtained for unannotated data using off-the-shelf POS taggers or PCFG parsers. However, the amount of information these tags typically provide is very limited,

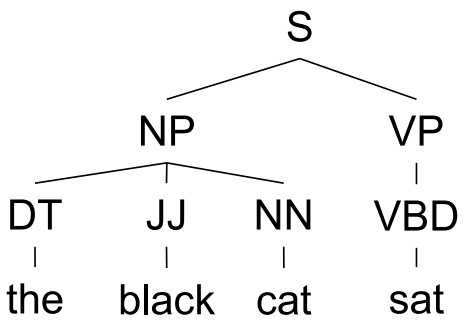

Figure 1: A parse tree example

e.g., while it is helpful to know whether *fly* is a verb or a noun, knowing that *you* is a personal pronoun does not carry the information whether it is a subject or an object (given the Penn Tree Bank tagset), which would certainly help to predict the following word.

2.2 SuperARV

The SuperARV essentially organizes information concerning one consistent set of dependency links for a word that can be directly derived from its syntactic parse. SuperARVs encode lexical information as well as syntactic and semantic constraints in a uniform representation that is much more fine-grained than POS. It is a four-tuple $(C; F; R+; D)$, where C is the lexical category of the word, F is a vector of lexical features for the word, $R+$ is a set of governor and need labels that indicate the function of the word in the sentence and the types of words it needs, and D represents the relative position of the word and its dependents. We refer the reader to the literature for further details on SuperARVs (Wang and Harper, 2002; Wang et al., 2003).

SuperARVs can be produced from parse trees by applying deterministic rules. In this work we use SuperARVs as individual tags and do not cluster them based of their structure. While SuperARVs are very attractive for language modeling, developing such a rich set of annotations for a new language would require a large amount of human effort.

We propose two other types of tags which have not been applied to this task, although similar information has been used in parsing.

2.3 Modifee Tag

This tag is a combination of the word's POS tag and the POS tag of its governor role. We designed it to resemble dependency parse structure. For example, the sentence in Figure 1 would be tagged: *the/DT-NN black/JJ-NN cat/NN-VBD sat/VBD-root*. Henceforth, we will refer to this kind of tag as *head*.

2.4 Parent Constituent

This tag is a combination of the word's POS tag with its immediate parent in the parse tree, along with the POS tag's relative position among its siblings. We refer to this type of tags as *parent*. The example in Figure 1 will be tagged: *the/DT-NP-start black/JJ-NP-mid cat/NN-NP-end sat/VB-VP-single*. This tagset is designed to represent constituency information.

Note that the *head* and *parent* tagsets are more language-independent (all they require is a treebank) than the SuperARVs which, not only utilized the treebank, but were explicitly designed by a linguist for English only.

2.5 Information Theoretic Comparison of Tags

As we have mentioned in Section 1, the choice of the tagset is very important to the performance of the model. There are two conflicting intuitions for tags: on one hand they should be specific enough to be helpful in the language model's task; on the other hand, they should be easy for the LM to predict.

Of course, in order to argue which tags are more suitable, we need some quantifiable metrics. We propose an information theoretic approach:

- To quantify how hard it is to predict a tag, we compute the conditional entropy:

$$\begin{aligned} H_p(t_i|w_i) &= H_p(t_i w_i) - H_p(w_i) \\ &= \sum_{w_i t_i} p(t_i w_i) \log p(t_i|w_i) \end{aligned}$$

- To measure how helpful a tagset is in the LM task, we compute the reduction of the conditional cross entropy:

$$\begin{aligned} & H_{\tilde{p},q}(w_i|w_{i-1}t_{i-1}) - H_{\tilde{p},q}(w_i|w_{i-1}) = \\ & \quad - \sum_{w_{i-1}^i t_{i-1}} \tilde{p}(w_{i-1}^i t_{i-1}) \log q(w_i|w_{i-1}t_{i-1}) \\ & \quad + \sum_{w_{i-1}^i} \tilde{p}(w_{i-1}^i) \log q(w_i|w_{i-1}) \\ = & \quad - \sum_{w_{i-1}^i t_{i-1}} \tilde{p}(w_{i-1}^i t_{i-1}) \log \frac{q(w_i|w_{i-1}t_{i-1})}{q(w_i|w_{i-1})} \end{aligned}$$

Note that in this case we use conditional *cross* entropy because conditional entropy has the tendency to overfit the data as we select more and more fine-grain tags. Indeed, $H_p(w_i|w_{i-1}t_{i-1})$ can be reduced to zero if the tags are specific enough, which would never happen in reality. This is not a problem for the former metric because the context there, w_i, is fixed. For this metric, we use a smoothed distribution $\tilde{p}$ computed on the training set[3] and the test distribution q.

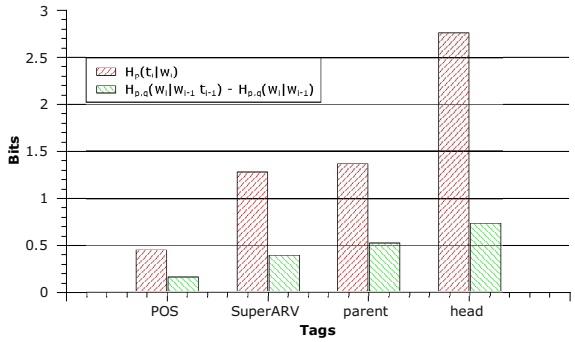

Figure 2: Changes in entropy for different tagsets

The results of these measurements are presented in Figure 2. POS tags, albeit easy to predict, provide very little additional information about the following word, and therefore we would not expect them to perform very well. The *parent* tagset seems to perform somewhat better than SuperARVs – it provides 0.13 bits more information while being only 0.09 bits harder to predict based on the word. The *head* tagset is interesting: it provides 0.2 bits more information about the following word (which would correspond to 15% perplexity reduction if we had perfect tags), but on the other hand the model is less likely to predict these tags accurately.

This approach is only a crude estimate (it uses only unigram and bigram context) but it is very useful for designing tagsets, e.g., for a new language, because it allows us to assess relative performance of tagsets without having to train a full model.

[3]We used *one-count* smoothing (Chen and Goodman, 1996).

3 Language Model Structure

The size and sparsity of the parameter space of the joint model necessitate the use of dimensionality reduction measures in order to make the model computationally tractable and to allow for accurate estimation of the model's parameters. We also want the model to be able to easily accommodate additional sources of information such as morphological features, prosody, etc. In the rest of this section, we discuss avenues we have taken to address these problems.

3.1 Decision Tree Clustering

Binary decision tree clustering has been shown to be effective for reducing the parameter space in language modeling (Bahl et al., 1990; Heeman, 1999) and other language processing applications, e.g., (Magerman, 1994). Like any clustering algorithm, it can be represented by a function H that maps the space of histories to a set of equivalence classes.

$$p(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \approx p(w_i t_i | H(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1})) \quad (4)$$

While the tree construction algorithm is fairly standard – to recursively select binary questions about the history optimizing some function – there are important decisions to make in terms of which questions to ask and which function to optimize. In the remainder of this section, we discuss the decisions we made regarding these issues.

3.2 Factors

The Factored Language Model (FLM) (Bilmes and Kirchhoff, 2003) offers a convenient view of the input data: it represents every word in a sentence as a tuple of factors. This allows us to extend the language model with additional parameters. In an FLM, however, all factors have to be deterministically computed in a joint model; whereas, we need to distinguish between the factors that are given or computed and the factors that the model must predict stochastically. We call these types of factors *overt* and *hidden*, respectively. Examples of overt factors include surface words, morphological features such as suffixes, case information when available, etc., and the hidden factors are POS, SuperARVs, or other tags.

Henceforth, we will use *word* to represent the set of overt factors and *tag* to represent the set of hidden factors.

3.3 Hidden Factors Tree

Similarly to (Heeman, 1999), we construct a binary tree where each tag is a leaf; we will refer to this tree as the *Hidden Factors Tree* (HFT). We use Minimum Discriminative Information (MDI) algorithm (Zitouni, 2007) to build the tree. The HFT represents a hierarchical clustering of the tag space. One of the reasons for doing this is to allow questions about subsets of tags rather than individual tags alone[4].

Unlike (Heeman, 1999), where the tree of tags was only used to create questions, this representation of the tag space is, in addition, a key feature of our decoding optimizations, which we discuss in Section 4.

3.4 Questions

The context space is partitioned by means of binary *questions*. We use different types of questions for hidden and overt factors.

- Questions about surface words are constructed using the Exchange algorithm (Martin et al., 1998). This algorithm takes the set of words that appear at a certain position in the training data associated with the current node in the history tree and divides the set into two complementary subsets greedily optimizing some target function (we use the average entropy of the marginalized word distribution, the same as for question selection). Note that since the algorithm only operates on the words that appear in the training data, we need to do something more to account for the unseen words. Thus, to represent this type of question, we create the history tree structure depicted in Fig. 4.

  For other overt factors with smaller vocabularies, such as suffixes, we use equality questions.

- As we mentioned in Section 3.3, we use the Hidden Factors Tree to create questions about hidden factors. Note that every node in a binary tree can be represented by a binary path from the root with all nodes under an inner node sharing the same prefix. Thus, a question about whether a tag belongs to a subset of tags dominated by a node can be expressed as whether the tag's path matches the binary prefix.

[4]Trying all possible subsets of tags is not feasible since there are $2^{|T|}$ of them. The tree allows us to reduce the number to $O(T)$ of the most meaningful (as per the clustering algorithm) subsets.

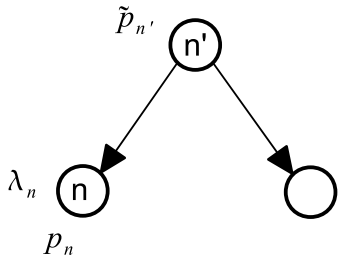

Figure 3: Recursive smoothing: $\tilde{p_n} = \lambda_n p_n + (1 - \lambda_n)\tilde{p_{n'}}$

3.5 Optimization Criterion and Stopping Rule

To select questions we use the average entropy of the marginalized word distribution. We found that this criterion significantly outperforms the entropy of the distribution of joint events. This is probably due to the increased sparsity of the joint distribution and the fact that our ultimate metrics, i.e., WER and word perplexity, involve only words.

3.6 Distribution Representation

In a cluster H_x, we factor the joint distribution as follows:

$$p(w_i t_i | H_x) = p(w_i | H_x) \cdot p(t_i | w_i, H_x)$$

where $p(t_i|w_i, H_x)$ is represented in the form of an HFT, in which each leaf has the probability of a tag and each internal node contains the sum of the probabilities of the tags it dominates. This representation is designed to assist the decoding process described in Section 4.

3.7 Smoothing

In order to estimate probability distributions at the leaves of the history tree, we use the following recursive formula:

$$\tilde{p}_n(w_i t_i) = \lambda_n p_n(w_i t_i) + (1 - \lambda_n)\tilde{p}_{n'}(w_i t_i) \quad (5)$$

where n' is the n-th node's parent, $p_n(w_i t_i)$ is the distribution at node n (see Figure 3). The

root of the tree is interpolated with the distribution $p_{unif}(w_i t_i) = \frac{1}{|V|} p_{ML}(t_i|w_i)$[5]. To estimate interpolation parameters λ_n, we use the EM algorithm described in (Magerman, 1994); however, rather than setting aside a separate development set of optimizing λ_n, we use 4-fold cross validation and take the geometric mean of the resulting coefficients[6]. We chose this approach because a small development set often does not overlap with the training set for low-count nodes, leading the EM algorithm to set $\lambda_n = 0$ for those nodes.

Let us consider one leaf of the history tree in isolation. Its context can be represented by the path to the root, i.e., the sequence of questions and answers $q_1, \ldots q_{(n')'} q_{n'}$ (with q_1 being the answer to the topmost question):

$$\tilde{p}_n(w_i t_i) = \tilde{p}(w_i t_i | q_1 \ldots q_{(n')'} q_{n'})$$

Represented this way, Eq. 5 is a variant of Jelinek-Mercer smoothing:

$$\begin{aligned}\tilde{p}(w_i t_i | q_1 \ldots q_{n'}) \;\; = \;\; & \lambda_n p(w_i t_i | q_1 \ldots q_{n'}) \; + \\ & (1 - \lambda_n) \tilde{p}(w_i t_i | q_1 \ldots q_{(n')'})\end{aligned}$$

For backoff nodes (see Fig. 4), we use a lower order model[7] interpolated with the distribution at the backoff node's grandparent (see node A in Fig. 4):

$$\begin{aligned}&\tilde{p}_B(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) = \\ &\quad \alpha_A \tilde{p}_{bo}(w_i t_i | w_{i-n+2}^{i-1} t_{i-n+2}^{i-1}) \; + \; (1 - \alpha_A) \tilde{p}_A(w_i t_i)\end{aligned}$$

How to compute α_A is an open question. For this study, we use a simple heuristic based on observation that the further node A is from the root the more reliable the distribution $\tilde{p}_A(w_i t_i)$ is, and hence α_A is lower. The formula we use is as follows:

$$\alpha_A = \frac{1}{\sqrt{1 + distanceToRoot(A)}}$$

[5]We use this distribution rather than uniform joint distribution $\frac{1}{|V||T|}$ because we do not want to allow word-tag pairs that have never been observed. The idea is similar to (Thede and Harper, 1999).

[6]To avoid a large number of zeros due to the product, we set a minimum for λ to be 10^{-7}.

[7]The lower order model is constructed by the same algorithm, although with smaller context. Note that the lower order model can back off on words or tags, or both. In this paper we backoff both on words and tags, i.e., $p(w_i t_i | w_{i-2}^{i-1} t_{i-2}^{i-1})$ backs off to $p(w_i t_i | w_{i-1} t_{i-1})$, which in turn backs off to the unigram $p(w_i t_i)$.

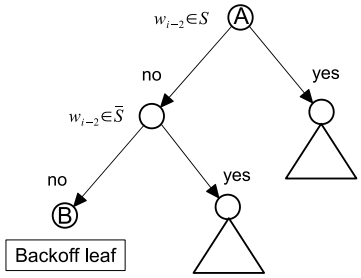

Figure 4: A fragment of the decision tree with a backoff node. $S \cup \bar{S}$ is the set of words observed in the training data at the node A. To account for unseen words, we add the backoff node B.

4 Decoding

As in HMM decoding, in order to compute probabilities for i-th step, we need to sum over $|T|^{n-1}$ possible combinations of tags in the history, where T is the set of tags and n is the order of the model. With $|T|$ predictions for the i-th step, we have $O(|T|^n)$ computational complexity per word. Straightforward computation of these probabilities is problematic even for a trigram model with POS tags, i.e., $n = 3, |T| \approx 40$. A standard approach to limit computational requirements is to use beam search where only $\mathcal{N}$ most likely paths are retained. However, with fine-grain tags where $|T| \approx 1,500$, a tractable beam size would only cover a small fraction of the whole space, leading to search errors such as pruning good paths.

Note that we have a history clustering function (Eq. 4) represented by the decision tree, and we should be able to exploit this clustering to eliminate unnecessary computations involving equivalent histories. Note that words in the history are known exactly, thus we can create a projection of the clustering function H in Eq. 4 to the plane $w_{i-n+1}^{i-1} = const$, i.e., where words in the context are fixed to be whatever is observed in the history:

$$H(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \;\Rightarrow\; \hat{H}_{w_{i-n+1}^{i-1} = const}(t_{i-n+1}^{i-1}) \tag{6}$$

The number of distinct clusters in the projection $\hat{H}$ depends on the decision tree configuration and can vary greatly for different words w_{i-n+1}^{i-1} in the history, but generally it is relatively small:

$$|\hat{H}_{w_{i-n+1}^{i-1} = const}(t_{i-n+1}^{i-1})| \;\ll\; |T^{n-1}| \tag{7}$$

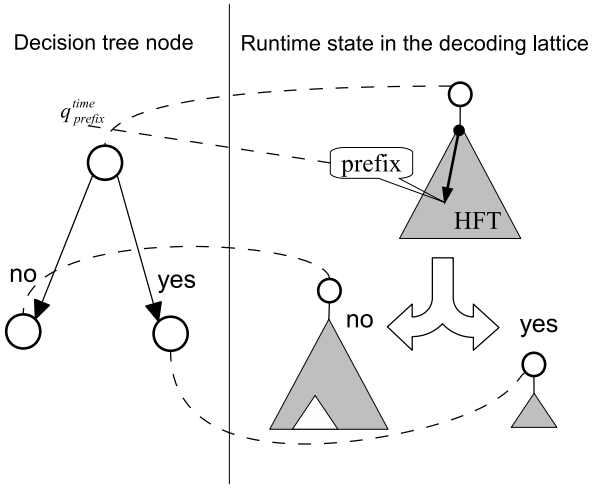

Figure 5: Questions about hidden factors split states (see Figure 6) in the decoding lattice represented by HFTs.

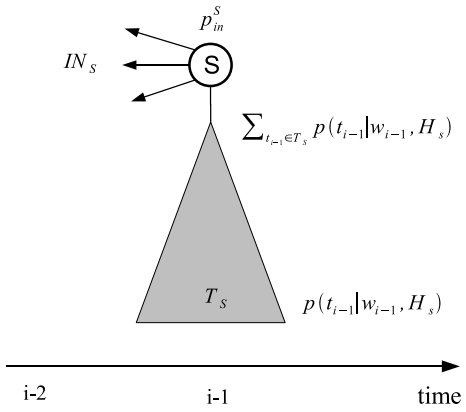

Figure 6: A state S in the decoding lattice. p_{in}^{S} is the probability of reaching the state S through the set of links IN_S. The probabilities of generating the tags $p(t_{i-1}|w_{i-1}, H_s), (t_{i-1} \in T_S)$ are represented in the form of the HFT.

thus, the number of probabilities that we need to compute is $|\hat{H}_{w_{i-n+1}^{i-1}=const}| \cdot |T|$.

Our decoding algorithm works similarly to HMM decoding with the exception that the set of hidden states is not predetermined. Let us illustrate how it works in the case of a bigram model. Recall that the set of tags T is represented as a binary tree (HFT) and the only type of questions about tags is about matching a binary prefix in the HFT. Such a question dissects the HFT into two parts as depicted in Figure 5. The cost of this operation is $O(\log |T|)$.

We represent states in the decoding lattice as shown in the Figure 6, where p_{in}^{S} is the probability of reaching the state S:

$$p_{in}^{S} = \sum_{S' \in IN_S} \left[p_{in}^{S'}\, p(w_{i-2}|H_{S'}) \sum_{t \in T_{S'}} p(t|w_{i-2}H_{S'}) \right]$$

where IN_S is the set of incoming links to the state S from the previous time index, and $T_{S'}$ is the set of tags generated from the state S' represented as a fragment of the HFT. Note, that since we maintain the property that the probability assigned to an inner node of the HFT is the sum of probabilities of the tags it dominates, the sum $\sum_{t \in T_{S'}} p(t|w_{i-2}H_{S'})$ is located at the root of $T_{S'}$, and therefore this is an $O(1)$ operation.

Now given the state S at time $i-1$, in order to generate tag predictions for i-th word, we apply questions from the history clustering tree, starting from the top. Questions about overt factors always follow either a *true* or *false* branch, implicitly computing the projection in Eq. 6. Questions about hidden factors, can split the state S into two states S_{true} and S_{false}, each retaining a part of T_S as shown in the Figure 5.

The process continues until each fragment of each state at the time $i-1$ reaches the bottom of the history tree, at which point new states for time i are generated from the clusters associated with leaves. The states at $i-1$ that generate the cluster $H_{\tilde{S}}$ become the incoming links to the state $\tilde{S}$.

Higher order models work similarly, except that at each time we consider a state S at time $i-1$ along with one of its incoming links (to some depth according to the size of the context).

5 Experimental Setup

To evaluate the impact of fine-grain tags on language modeling, we trained our model with five settings: In the first model, questions were restricted to be about overt factors only, thus making it a tree-based word model. In the second model, we used POS tags. To evaluate the effect of fine-grain tags, we train two models: *head* and *parent* described in Section 2.3 and Section 2.4 respectively. Since our joint model can be used with any kind of tags, we also trained it with SuperARV tags (Wang et al., 2003). The SuperARVs were created from the same parse trees that were used to produce POS and fine-grain tags. All our models, including SuperARV, use trigram context. We include standard trigram, four-gram, and five-

gram models for reference. The ngram models were trained using SRILM toolkit with interpolated modified Kneser-Ney smoothing.

We evaluate our model with an nbest rescoring task using 100-best lists from the DARPA WSJ'93 and WSJ'92 20k open vocabulary data sets. The details on the acoustic model used to produce the nbest lists can be found in (Wang and Harper, 2002). Since the data sets are small, we combined the 93et and 93dt sets for evaluation and used 92et for the optimization[8]. We transformed the nbest lists to match PTB tokenization, namely separating possessives from nouns, *n't* from auxiliary verbs in contractions, as well as contractions from personal pronouns.

All language models were trained on the NYT 1994-1995 section of the English Gigaword corpus (approximately 70M words). Since the New York Times covers a wider range of topics than the Wall Street Journal, we eliminated the most irrelevant stories based on their trigram coverage by sections 00-22 of WSJ. We also eliminated sentences over 120 words, because the parser's performance drops significantly on long sentences. After parsing the corpus, we deleted sentences that were assigned a very low probability by the parser. Overall we removed only a few percent of the data; however, we believe that such a rigorous approach to data cleaning is important for building discriminating models.

Parse trees were produced by an extended version of the Berkeley parser (Huang and Harper, 2009). We trained the parser on a combination of the BN and WSJ treebanks, preprocessed to make them more consistent with each other. We also modified the trees for the speech recognition task by replacing numbers and abbreviations with their verbalized forms. We pre-processed the NYT corpus in the same way, and parsed it. After that, we removed punctuation and downcased words. For the ngram model, we used text processed in the same way.

In *head* and *parent* models, tag vocabularies contain approximately 1,500 tags each, while the SuperARV model has approximately 1,400 distinct SuperARVs, most of which represent verbs (1,200).

In these experiments we did not use overt factors other than the surface word because they split <unk>, effectively changing the vocabulary thus making perplexity incomparable to models without these factors, without improving WER noticeably. However, we do plan to use more overt factors in Machine Translation experiments where a language model faces a wider range of OOV phenomena, such as abbreviations, foreign words, numbers, dates, time, etc.

Models	WER
trigram (baseline)	17.5
four-gram	17.7
five-gram	17.8
Word Tree	17.3
POS Tags	17.0
Head Tags	16.8
Parent Tags	**16.7**
SuperARV	16.9

Table 1: WER results, optimized on 92et set, evaluated on combined 93et and 93dt set. The Oracle WER is 9.5%.

Table 1 summarizes performance of the LMs on the rescoring task. The *parent* tags model outperforms the trigram baseline model by 0.8% WER. Note that four- and five-gram models fail to outperform the trigram baseline. We believe this is due to the sparsity as well as relatively short sentences in the test set (16 words on average).

Interestingly, whereas the improvement of the POS model over the baseline is not statistically significant ($p < 0.10$)[9], the fine-grain models outperform the baseline much more reliably: $p < 0.03$ (SuperARV) and $p < 0.007$ (*parent*).

We present perplexity evaluations in Table 2. The perplexity was computed on Section 23 of WSJ PTB, preprocessed as the rest of the data we used. The *head* model has the lowest perplexity outperforming the baseline by 9%. Note, it even outperforms the five-gram model, although by a small 2% margin.

Although the improvements by the fine-grain tagsets over POS are not significant (due to the small size of the test set), the reductions in perplexity suggest that the improvements are not random.

[8]We optimized the LM weight and computed WER with scripts in the SRILM and NIST SCTK toolkits.

[9]For statistical significance, we used SCTK implementation of the *mapsswe* test.

Models	PPL
trigram (baseline)	162
four-gram	152
five-gram	150
Word Tree	160
POS Tags	154
Head Tags	**147**
Parent Tags	150
SuperARV	150

Table 2: Perplexity results on Section 23 WSJ PTB

6 Conclusion and Future Work

In this paper, we presented a joint language modeling framework. Unlike any prior work known to us, it was not tailored for any specific tag set, rather it was designed to accommodate any set of tags, especially large sets ($\sim 1,000$), which present challenges one does not encounter with smaller tag sets, such at POS tags. We discussed these challenges and our solutions to them. Some of the solutions proposed are novel, particularly the decoding algorithm.

We also proposed two simple fine-grain tagsets, which, when applied in language modeling, perform comparably to highly sophisticated tag sets (SuperARV). We would like to stress that, while our fine-grain tags did not significantly outperform SuperARVs, the former use much less linguistic knowledge and can be automatically induced for any language with a treebank.

Because a joint language model inherently predicts hidden events (tags), it can also be used to generate the best sequence of those events, i.e., tagging. We evaluated our model in the POS tagging task and observed similar results: the fine-grain models outperform the POS model, while both outperform the state-of-the-art HMM POS taggers. We refer to (Filimonov and Harper, 2009) for details on these experiments.

We plan to investigate how parser accuracy and data selection strategies, e.g., based on parser confidence scores, impact the performance of our model. We also plan on evaluating the model's performance on other genres of speech, as well as in other tasks such as Machine Translation. We are also working on scaling our model further to accommodate amounts of data typical for modern large-scale ngram models. Finally, we plan to apply the technique to other languages with treebanks, such as Chinese and Arabic.

We intend to release the source code of our model within several months of this publication.

7 Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 and NSF IIS-0703859. Any opinions, findings and/or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the institutions where the work was completed.

References

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. 1990. A tree-based statistical language model for natural language speech recognition. *Readings in speech recognition*, pages 507–514.

Srinivas Bangalore. 1996. 'Almost parsing' technique for language modeling. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1173–1176.

Jeff A. Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of HLT/NACCL, 2003*, pages 4–6.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling for speech recognition. *CoRR*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, Morristown, NJ, USA. Association for Computational Linguistics.

Denis Filimonov and Mary Harper. 2009. Measuring tagging performance of a joint language model. In *Proceedings of the Interspeech 2009*.

Peter A. Heeman. 1999. POS tags and decision trees for language modeling. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 129–137.

Zhongqiang Huang and Mary Harper. 2009. Self-Training PCFG grammars with latent annotations across languages. In *Proceedings of the EMNLP 2009*.

David M. Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, Stanford, CA, USA.

Sven Martin, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. In *Speech Communication*, pages 1253–1256.

Thomas R. Niesler and Phil C. Woodland. 1996. A variable-length category-based n-gram language model. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:164–167 vol. 1, May.

Scott M. Thede and Mary P. Harper. 1999. A second-order hidden markov model for part-of-speech tagging. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 175–182.

Wen Wang and Mary P. Harper. 2002. The SuperARV language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 238–247, Morristown, NJ, USA. Association for Computational Linguistics.

Wen Wang, Mary P. Harper, and Andreas Stolcke. 2003. The robustness of an almost-parsing language model given errorful training data. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Peng Xu and Frederick Jelinek. 2004. Random forests in language modeling. In *in Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Imed Zitouni. 2007. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88–104.