Character-Level Neural Language Models with Part of Speech Labeling

Jonathan Gill
Department of Electrical Engineering (ECE 523, Spring 2017)
University of Arizona
Tucson, AZ 85721
jtgill@email.arizona.edu

Abstract

Neural network language models (NNLM's) have experienced a major increase in popularity due to their flexibility, convenience, and accuracy. State of the art prediction results can be achieved by simply training these models on sequences of words. This work attempts to augment the predictive power of the NNLM by training it on sequences of words *and* parts of speech. We introduce a method of estimating a joint probability distribution over word and part of speech, and compare the results of our model to a word-only LM.

1 Theory

The goal of this project is to include part-of-speech tagging to enhance the predictive power of a LSTM language model. What sets this model apart from other word-level language models is the network's ability to allow POS tags to inform the prediction of the next word, and vice versa. Let $\vec{w}_t$ represent a one-hot encoding of a particular word at a location t in a sentence, and $\vec{p}_t$ that word's part of speech. It is desired for the network to produce the predictions over the values $\vec{w}_{t+1}$ and $\vec{p}_{t+1}$ given that all word and POS pairs from times 1 to t have been presented to the network. In a traditional RNNLM a softmax is simply taken over the logits of $\boldsymbol{w}_{t+1}$, and the cross-entropy between the model's prediction and the target probability is minimized.

$$P(\boldsymbol{w}_{t+1}|\boldsymbol{w}_1, \dots, \boldsymbol{w}_t) = P(\boldsymbol{w}_{t+1}|W_t),$$
$$H(P,Q) = \frac{1}{N}\sum_{t=1}^{N} P(\boldsymbol{w}_{t+1}|W_t)\log[Q(\boldsymbol{w}_{t+1}|W_t)], \qquad \text{Eq 1}$$

where Q represents the model's predictions, P is the target probability, and we have let W_t represent the conjunction of all words at the previous timesteps, $W_t = \bigcap_{i=1}^{t} \boldsymbol{w}_i$.

In our model, we must minimize the cross-entropy between the joint conditional probability of $\boldsymbol{w}_{t+1}$ and $\boldsymbol{p}_{t+1}$. Unfortunately, our model only allows us to recover the marginal conditional probabilities for $\boldsymbol{w}_{t+1}$ and $\boldsymbol{p}_{t+1}$, so we make a small simplifying assumption. We now let X_t represent the conjunction of all words *and* POS tags at the previous timesteps, $X_t = \bigcap_{i=1}^{t}(\boldsymbol{w}_i, \boldsymbol{p}_i)$, and we seek to minimize the cross-entropy of the joint probability given by

$$Q(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}|X_t) = Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1}, X_t)Q(\boldsymbol{w}_{t+1}|\, X_t). \qquad \text{Eq 2}$$

Qualitatively, we want to know the probability of seeing a particular word $\boldsymbol{w}_{t+1}$ with part of speech $\boldsymbol{p}_{t+1}$, given that we've seen t previous words and POS's. Here we assume that the probability of seeing a particular part of speech for a given word is independent of the history of previous words and POS's, which allows us to write

$$Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1}, X_t) \approx Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1}). \qquad \text{Eq 3}$$

In most cases this is a safe assumption to make; there are many words whose part of speech remains constant regardless of the context (conjunctions, articles, etc.). A particularly alluring facet of this estimation is that it can be empirically computed using frequency analysis on a large, annotated corpus (discussed in XXX). We now have an estimation of the full joint probability:

$$Q(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}|X_t) \approx Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1})Q(\boldsymbol{w}_{t+1}|\, X_t). \qquad \text{Eq 4}$$

To make use of this estimation, we modify the cross-entropy loss function to accommodate the joint probability,

$$H(P, Q) = \frac{1}{N}\sum_{t=1}^{N} P(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}|X_t) \log[Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1})Q(\boldsymbol{w}_{t+1}|\, X_t)]. \qquad \text{Eq 5}$$

This estimation has one major drawback. The intent of our model is to produce a prediction over a part of speech as well as a word (figure XXX). In order to do this, we take a softmax over the portion of the output layer set aside for POS tags. By inspection of (Eq 5), it’s easy to notice that the estimation doesn’t take the model’s prediction over POS into account. To ameliorate this, we make use of the definition of conditional probability and make another simplifying assumption.

We start by rewriting (Eq 2) in an equivalent form

$$Q(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}|X_t) = Q(\boldsymbol{w}_{t+1}|\boldsymbol{p}_{t+1}, X_t)Q(\boldsymbol{p}_{t+1}|\, X_t), \qquad \text{Eq 6}$$

which explicitly takes the prediction over POS into account. Taking note of the fact that the expressions in the right-hand-sides of (Eq 2) and (Eq 6) are exactly equivalent, we can write (Eq 2) as

$$\begin{aligned} Q(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}|X_t) = \frac{1}{2}[&Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1}, X_t)Q(\boldsymbol{w}_{t+1}|\, X_t) \\ &+ Q(\boldsymbol{w}_{t+1}|\boldsymbol{p}_{t+1}, X_t)Q(\boldsymbol{p}_{t+1}|\, X_t)]. \end{aligned} \qquad \text{Eq 7}$$

We can insert the approximation from (Eq 4) into (Eq 7), but this leaves us with yet another untenable mathematical expression. The troublesome term is the prediction over the current word given the current POS and the history of words and POS’s,

$$Q(\boldsymbol{w}_{t+1}|\boldsymbol{p}_{t+1}, X_t). \qquad \text{Eq 8}$$

This term could be approximated similarly to (Eq 3), however, this approximation does not enjoy the same intuitive interpretability.

$$Q(\boldsymbol{w}_{t+1}|\boldsymbol{p}_{t+1}, X_t) \approx Q(\boldsymbol{w}_{t+1}|\boldsymbol{p}_{t+1}). \qquad \text{Eq 9}$$

Our argument for the rejection of this estimation is that sampling from the resulting distribution would be akin to playing a game of MadLibs: given a part of speech, select the most likely word to fit that part of speech. Instead, we assume that the predicted word is independent of the conditioning on the predicted part of speech.

$$Q(\boldsymbol{w}_{t+1}|\boldsymbol{p}_{t+1}, X_t) \approx Q(\boldsymbol{w}_{t+1}|\, X_t). \qquad \text{Eq 10}$$

This convenient assumption allows us to estimate (Eq 7) using quantities readily available to us:

$$Q(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}|X_t) \approx \frac{1}{\alpha} Q(\boldsymbol{w}_{t+1}|X_t)[Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1}) + Q(\boldsymbol{p}_{t+1}|\, X_t)]. \qquad \text{Eq 11}$$

The factor $1/\alpha$ is inserted to normalize the distribution, and can be computed as such:

$$\alpha = \sum_{\substack{\boldsymbol{w}_{t+1} \in W, \\ \boldsymbol{p}_{t+1} \in P}} Q(\boldsymbol{w}_{t+1}|X_t)[Q(\boldsymbol{p}_{t+1}|\boldsymbol{w}_{t+1}) + Q(\boldsymbol{p}_{t+1}|\, X_t)]. \qquad \text{Eq 12}$$

The value of the normalizing factor could change at every position in the training sequence, so it is computed before (Eq 11) is inserted into the cross-entropy loss.

2 Procedure

Our model was trained, validated, and tested on separate portions of the Brown corpus. The model was implemented using TensorFlow, and POS tags were generated using the Natural Language Toolkit. The vocabulary for the word-level language model (LM) was pulled from the most frequently used 9,999 words in the PTB corpus, with one vocabulary word representing "unknown" words not present in the vocabulary. Words in the corpus were not lemmatized, nor were words converted to lowercase before being added to the vocabulary. Therefore, the words "It", and "it" can appear as separate entries in the training vocabulary.

Two LM architectures were created for comparison: one incorporating information from both POS and word, and another using solely words. Both architectures used two stacked LSTM layers with 256 nodes, as well as dimensionality reducing embeddings for the sparsely populated and high-dimensional input vectors. POS embeddings were generated using an auto-encoder, reducing the number of dimensions by a factor of ten. Word embeddings were created with an auto-encoder as well. The embeddings are only weight matrices; they include no bias terms. Once inserted into the full models, these embeddings were not modified.

To obtain an empirical estimation for the quantities in (Eq 3), the NLTK library was used to tag every word in the Brown corpus with a POS from the PTB tag set. All word and POS pairs were counted and normalized to give an estimation of the joint probability $Q(\boldsymbol{p},\boldsymbol{w})$. This allowed the conditional probabilities to be retrieved by simply employing the definition of conditional probability. To this end, the probability of each word appearing in the corpus, $Q(\boldsymbol{w})$, was calculated by simply normalizing the number of occurrences of each word. The top twenty-five most probable word-POS pairs in the Brown corpus account for roughly forty percent of the joint probability mass. Unsurprisingly, "the" with its part of speech "DT" (determiner) is at the top of the list with a probability of roughly 0.05.

At training time, 32 sequences of 50 word, POS pairs were generated from the Brown training data. The starting points of the sequences were sampled with replacement from the training corpus. Batches of word sequences and batches of POS sequences were presented to the network as separate rank 3 tensors. Gradients were generated that minimized the joint probability cross-entropy loss function (Eq 5). Updates to the model parameters were carried out using RMSProp (XXX). The learning rate halving scheme employed by Mikolov (XXX) was also used here; pseudocode for the halving operation is shown in (Figure 2). The only differences for the word-level LM were: (1)

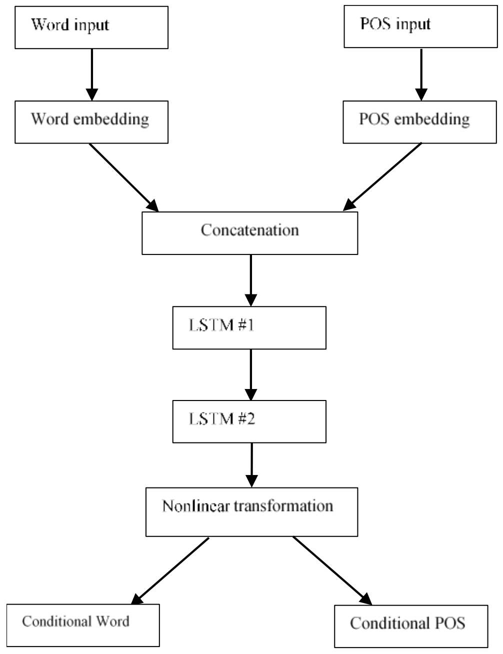

Figure 1: Overview of the word+POS LM architecture. One-hot encodings for word and POS are received on the word and POS inputs, respectively. The result of the non-linear transformation on the output of the second LSTM is split into two separate softmax operations: one over words, the other over POS tags.

exclusion of the POS batch from training, and (2) gradients were calculated through minimization of the cross-entropy function shown in Eq 1.

Each LM's perplexity on the validation set was saved after a fixed number of training rounds. Perplexity was calculated by averaging the per-word perplexity of each sequence in a training batch.

$$PPL = \frac{1}{B}\sum_{i=1}^{B} p_i$$

$$p_i = \exp\left\{-\frac{1}{N}\sum_{j=1}^{N} \ln\left[P\left(\boldsymbol{w}_j \middle| X_{j-1}\right)\right]\right\}$$

where B is the number of sequences in a batch, N is the number of words in a sequence, $\boldsymbol{w}_j$ is the chosen word at position j in the sequence, and X_{j-1} is the conjunction of all previously selected POS's and words. Note that the definition of perplexity is the exponentiation of batch-averaged cross-entropy loss with one-hot encodings for the target probabilities.

Once the maximum number of learning rate halvings had been achieved, each trained network was tested against the hold-out section of the PTB corpus. The perplexity of each model was saved and compared. In addition to performing learning based on rate halving, training was also carried out using a small, static learning rate over a fixed number of epochs.

3 Practical Issues

There were three major difficulties encountered with this project: (1) finding an annotated corpus with a limited POS tag set, (2) integrating the approximation of the joint probability in (Eq 4) into the cross-entropy minimization, and (3) implementing the word+POS approximation without encountering memory constraints. Another impediment was making a choice on the trade-off between using a small corpus with exactly correct annotations, and using a large, un-annotated corpus with NLTK's built-in tagging methods.

Penn Treebank's tag set was chosen because it contained a very limited number of POS tags, and because it didn't implement compound POS tagging, such as the Brown tag set. Another advantage is that the default auto-annotation system in the NLTK library uses the PTB tagset. This meant that it would have been possible to use other corpora for training or testing.

The joint probability estimation in (Eq 4) is the most critical piece of this model, and ensuring the estimation's veracity was crucial to the completion of this project. The problem with calculating the relevant quantities was the fact that the result was heavily dependent on the size of the training corpus. The automatic tagging feature of NLTK was used to create a version of the Brown corpus with PTB annotations (as opposed to the verbose Brown annotations). This introduced its own potential for error; automatic annotation systems are prone to misclassification of homonyms (e.g. "refuse" could be tagged as a verb or a noun depending on its pronunciation). Despite this, the Brown corpus with PTB annotations was used to estimate (Eq 3).

```
Input a tensor of one-hot character encodings
Output a PMF over the possible words in the vocabulary
build TensorFlow computation graph
learning_rate ← 0.01, num_halvings ← 0, i ← 0
while num_halvings < 15 and i < 1e4
    $error_i$ ← network.train(corpus.batch(32, 50), learning_rate)
    if $(error_{i-1} - error_i)/error_{i-1} < \varepsilon$
        learning_rate ← learning_rate/2
    end
    i ← i + 1
end
```

Figure 2: Pseudocode for the rate halving scheme employed by the training portion of the LM.

	Word LM	Word+POS LM
Input	10,000	10,049
Embedding layer	100	104
Recurrent layers	LSTM, 2×256	LSTM, 2×256
Output layer	10,000	10,049
Optimization	Cross-entropy of word conditional: $Q(\boldsymbol{w}_{t+1}\|W_t)$	Cross-entropy of joint approximation: $Q(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}\|X_t)$

Table 1 – Network configurations for the word and word+POS LM's. The embedding layer entry indicates the severity of the dimensionality reduction. Both networks used two layers of LSTM's with 256 nodes in each LSTM cell.

Initially, the PTB corpus was used for training, testing, and validation. Naturally, the word-level vocabulary that was chosen reflected the highest frequency words within PTB. However, the top N words in the PTB corpus did not wholly intersect with the top N words from the Brown corpus. This caused the marginal probabilities of some words in the training vocabulary to drop to zero. Specifically, words that were used in PTB and *not* used Brown had zero probability of even occurring within the corpus. This would have resulted propagating zeros through the network whenever any of these words were encountered.

Programmatically implementing the approximation from (Eq 4) was also challenging, given that model was implemented inside of the TensorFlow framework. The estimated conditional probability from (Eq 3) took the form of a matrix whose rows indices corresponded to POS tags, and whose columns indices corresponded to words from the vocabulary. The row and column indices of the conditional probability matrix had to correspond exactly to the word and POS tags used for training, and everything had to be represented in terms of an immutable TensorFlow matrix. In short, this part was just difficult because it required coordination between vocabularies and indices.

Using minibatch SGD with the word+POS LM required the addition of another axis to the training batches because cross-entropy of the joint probability was used to perform back-propagation. This increased the volume of the training tensor by a factor of 49 (given that there were 49 POS's used in training), which yielded training matrices requiring 2.9GB of RAM. Reducing the sequence length and number of batches by a factor of one half drove the required memory for a training batch down to 0.73GB. Despite this reduction, the intermediate matrices in the neural network pushed the memory requirements past the bounds of the local computing resources.

Using the same minibatch reduction, an AWS EC2 GPU unit (P2.xlarge) was used to carry out all POS+word LM training procedures. The lowest-tier GPU available was an NVidia Tesla K80 with 12GB of VRAM, which was more than sufficient for the POS+word LM training. Unfortunately, this instance is not available as part of the free tier, and costs of $0.90/hour rendered its prolonged use prohibitive. The combined time required to test the code, run it, and correct errors prevented the addition of useful features that could have aided in further analysis of the results.

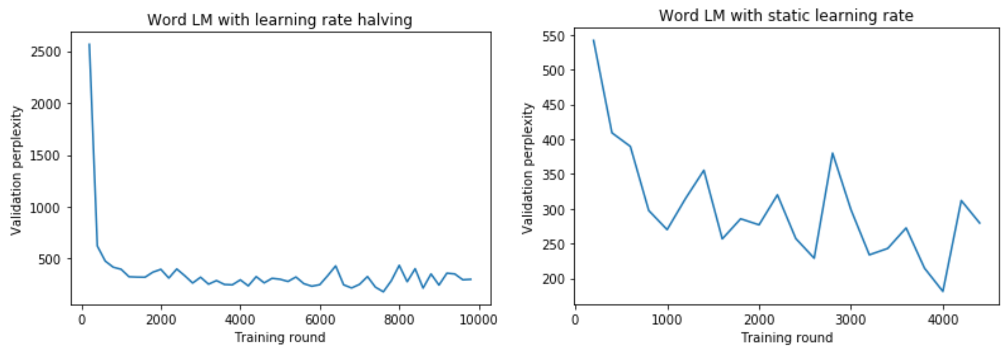

Figure 3 – Perplexity of the conditional probability of the chosen word for the word LM, obtained from the validation set. This model does not take POS tags into consideration.

4 Results

The following experiments were carried out: (1) word LM with constant learning rate, (2) word LM with rate halving, (3) word+POS LM with constant learning rate, (4) word+POS LM with rate halving. The same rate halving scheme was used for both LM's, and adheres to the pseudocode shown in Figure 2. The constant learning rate training runs used a learning rate of 0.001.

Figure 4 shows the word+POS LM's perplexity on the joint probabilities, $Q(\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}|X_t)$. In both cases, the LM's settled to a perplexity of $\approx 2{,}100$, though the implementation that made use of learning rate halving converged more quickly than the model that used a static learning rate. A common interpretation of perplexity is the number of "likely alternatives" to the target word. Though this suggests a possible $2{,}000$ likely alternatives, this is within a context of $490{,}000$ potential combinations of $\boldsymbol{w}_{t+1}, \boldsymbol{p}_{t+1}$. However, we do not seek a joint probability over next word and next POS; our desired output is a marginalized distribution over the possible next words.

Figure 5 shows the perplexity resulting from marginalized distributions of the next word given a history of previous words and parts of speech. Consistent with the previous set of results, the model that made use of rate halving converged to a perplexity value more quickly than the model that used a constant learning rate. The values reported by these models, however, are distressing. The halved scheme achieved a perplexity of ≈ 1.0, which implies that each word predicted by the model was nearly perfectly accurate. This also suggests that the model has completely memorized the sequences that it's been presented, but this explanation seems to lack plausibility when we compare to the results of the pure word LM.

Comparatively, the pure word-level LM produced minimal perplexities of ≈ 200 even with the aid of learning rate halving (Figure 3). These results are far from state-of-the-art, so we take them as an indication that the model is not overfitting the data. It's worth noting that the networks used to create these models have extremely similar parameters (Table 1), the chief differences being the addition of extra nodes to the input and output layer, and the presence of a POS embedding in the WPOSLM.

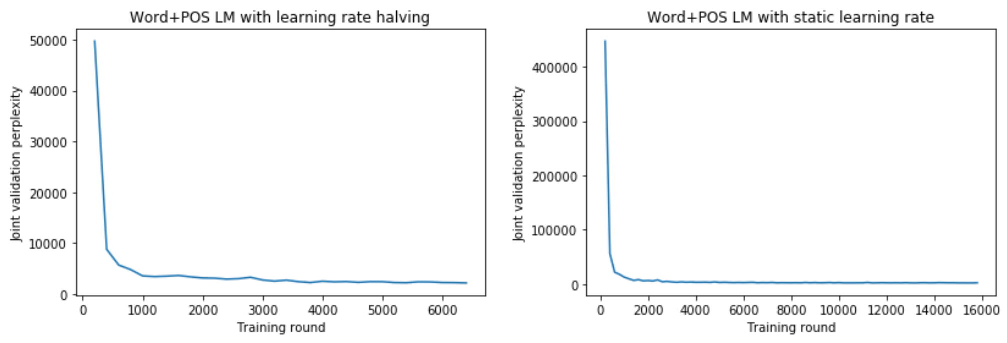

Figure 4: Perplexity of the joint probability of the selected word and POS tag (conditioned on the recent history of POS's and words) for the word+POS LM, obtained from the validation set. The minimum perplexity achieved was 2,100 by the model that used learning rate halving.

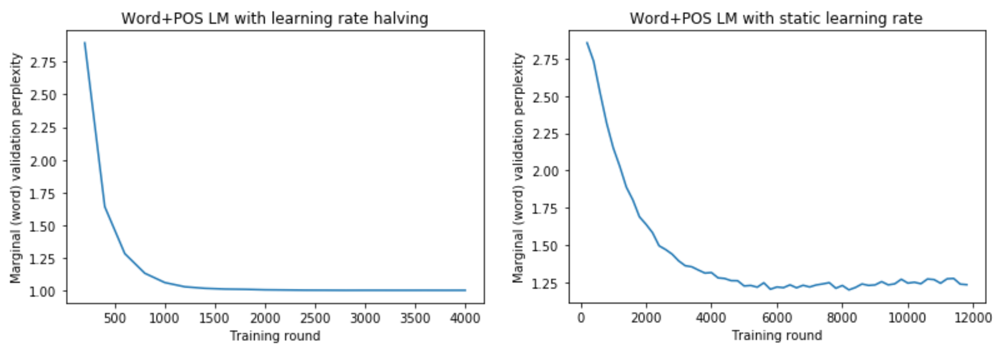

Figure 5: Perplexity of the chosen word (conditioned on the recent history of POS tags and words) for the word+POS LM. The perplexity in this case is lower-bounded by 1.0, a rather outrageous value.

Intuitively we'd expect that increasing the dimensionality of our data by a *factor* of 49 would make the WPOSLM almost immune to overfitting.

5 Conclusion

We expected the WPOSLM to perform slightly better than the traditional word LM; it was assumed that the addition of a POS tag would aid in the model in predicting the next word in a sequence of tagged words. Taken naively, our results indicate that our model was successful, but the magnitude of our model's outperformance warrants a degree of skepticism. We will present some possible explanations for our model's performance and discuss methods for further evaluation.

One use of traditional neural LM's is using a sequence of words to predict the following word, but this requires a drastic simplification of language in general. Namely, that the probability of a word occurring in a corpus is wholly dependent on the N words that were uttered previously. The cornerstone of this approximation is the idea that small values of N will result in a high-variance prediction, meaning that immediate neighbors are poor predictors. We state without proof that parts of speech are much better suited to making small N predictions due to their relatively low dimensionality and inherently sequential nature. We therefore propose that our model was able to make accurate predictions due to the synthesis of short-term information provided by the POS tags, and long-term information provided by words.

We have two proposals for testing our model's veracity: Firstly, validate the model on ungrammatical sentences. This would consist of pulling a random bag of tagged words from the training vocabulary and presenting it to the network. In this case we would expect the model to return large perplexities, as ungrammatical sentences wouldn't conform to any sentence pattern in its training corpus.

Secondly, the core of our intuition could be tested by modifying the network architecture shown in Figure 1. This would involve removing the prediction over part of speech and asking the network to only make predictions over words given a history of words and POS's. The appeal of this idea is two-fold: (1) it doesn't require the substantial number of manipulations and approximations shown in the Theory section and (2) would not require massive GPU power to train and run the model. From an ease-of-implementation standpoint, this method should be attempted first. There are a preponderance of highly accurate POS taggers available for various programming languages (NLTK has several varieties available); this would allow for the exploitation of other, larger corpora in the training process.

Additionally, we propose that an attention model is well-suited to this problem. Rather than having the network separately make predictions over POS's and words, the network's POS prediction could be used to directly inform the word prediction. The core of this idea is to keep the softmax POS prediction, and use the resulting probability as "blurry" selection for the subset of words belonging to that particular POS.

6 References

[1] J. F. A. Peter A. Heeman, "Incorporating POS Tagging Into Language Modeling," in *Proceedings of the 5th European Conference on Speech Communication and Technology*, 1997.

[2] O. V. M. S. N. S. Y. W. Rafal Jozefowicz, "Exploring the Limits of Language Modeling," 11 February 2016. [Online]. Available: arXiv:1602.02410v2.

[3] T. Mikolov, "Statistical Language Models Based on Neural Networks," BRNO, 2012.

[4] O. V. Q. V. L. Ilya Sutskever, "Sequence to Sequence Learning with Neural Networks," in *NIPS '14 Proceedings of the 27th International Conference on Neural Information Processing Systems*, Montreal, 2014.