Exploring the Limits of Language Modeling

Rafal Jozefowicz RAFALJ@GOOGLE.COM
Oriol Vinyals VINYALS@GOOGLE.COM
Mike Schuster SCHUSTER@GOOGLE.COM
Noam Shazeer NOAM@GOOGLE.COM
Yonghui Wu YONGHUI@GOOGLE.COM

Google Brain

Abstract

In this work we explore recent advances in Recurrent Neural Networks for large scale Language Modeling, a task central to language understanding. We extend current models to deal with two key challenges present in this task: corpora and vocabulary sizes, and complex, long term structure of language. We perform an exhaustive study on techniques such as character Convolutional Neural Networks or Long-Short Term Memory, on the One Billion Word Benchmark. Our best single model significantly improves state-of-the-art perplexity from 51.3 down to 30.0 (whilst reducing the number of parameters by a factor of 20), while an ensemble of models sets a new record by improving perplexity from 41.0 down to 23.7. We also release these models for the NLP and ML community to study and improve upon.

1. Introduction

Language Modeling (LM) is a task central to Natural Language Processing (NLP) and Language Understanding. Models which can accurately place distributions over sentences not only encode complexities of language such as grammatical structure, but also distill a fair amount of information about the knowledge that a corpora may contain. Indeed, models that are able to assign a low probability to sentences that are grammatically correct but unlikely may help other tasks in fundamental language understanding like question answering, machine translation, or text summarization.

LMs have played a key role in traditional NLP tasks such as speech recognition (Mikolov et al., 2010; Arisoy et al., 2012), machine translation (Schwenk et al., 2012; Vaswani et al.), or text summarization (Rush et al., 2015; Filippova et al., 2015). Often (although not always), training better language models improves the underlying metrics of the downstream task (such as word error rate for speech recognition, or BLEU score for translation), which makes the task of training better LMs valuable by itself.

Further, when trained on vast amounts of data, language models compactly extract knowledge encoded in the training data. For example, when trained on movie subtitles (Serban et al., 2015; Vinyals & Le, 2015), these language models are able to generate basic answers to questions about object colors, facts about people, etc. Lastly, recently proposed sequence-to-sequence models employ conditional language models (Mikolov & Zweig, 2012) as their key component to solve diverse tasks like machine translation (Sutskever et al., 2014; Cho et al., 2014; Kalchbrenner et al., 2014) or video generation (Srivastava et al., 2015a).

Deep Learning and Recurrent Neural Networks (RNNs) have fueled language modeling research in the past years as it allowed researchers to explore many tasks for which the strong conditional independence assumptions are unrealistic. Despite the fact that simpler models, such as N-grams, only use a short history of previous words to predict the next word, they are still a key component to high quality, low perplexity LMs. Indeed, most recent work on large scale LM has shown that RNNs are great in combination with N-grams, as they may have different strengths that complement N-gram models, but worse when considered in isolation (Mikolov et al., 2011; Mikolov, 2012; Chelba et al., 2013; Williams et al., 2015; Ji et al., 2015a; Shazeer et al., 2015).

We believe that, despite much work being devoted to small data sets like the Penn Tree Bank (PTB) (Marcus et al., 1993), research on larger tasks is very relevant as overfitting is not the main limitation in current language modeling, but is the main characteristic of the PTB task. Results on larger corpora usually show better what matters as many ideas work well on small data sets but fail to improve on

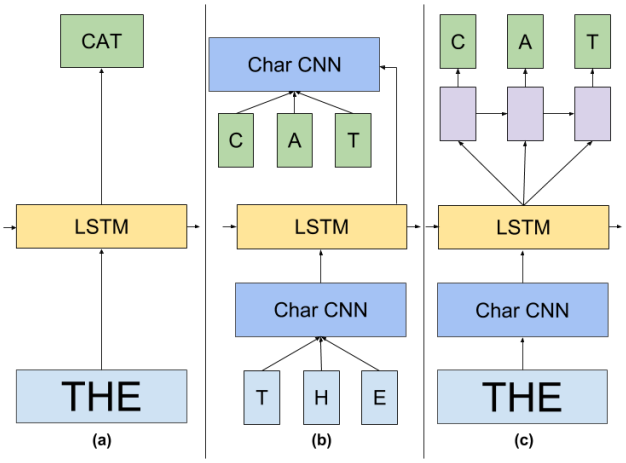

Figure 1. A high-level diagram of the models presented in this paper. (a) is a standard LSTM LM. (b) represents an LM where both input and Softmax embeddings have been replaced by a character CNN. In (c) we replace the Softmax by a next character prediction LSTM network.

larger data sets. Further, given current hardware trends and vast amounts of text available on the Web, it is much more straightforward to tackle large scale modeling than it used to be. Thus, we hope that our work will help and motivate researchers to work on traditional LM beyond PTB – for this purpose, we will open-source our models and training recipes.

We focused on a well known, large scale LM benchmark: the One Billion Word Benchmark data set (Chelba et al., 2013). This data set is much larger than PTB (one thousand fold, 800k word vocabulary and 1B words training data) and far more challenging. Similar to Imagenet (Deng et al., 2009), which helped advance computer vision, we believe that releasing and working on large data sets and models with clear benchmarks will help advance Language Modeling.

The contributions of our work are as follows:

- We explored, extended and tried to unify some of the current research on large scale LM.
- Specifically, we designed a Softmax loss which is based on character level CNNs, is efficient to train, and is as precise as a full Softmax which has orders of magnitude more parameters.
- Our study yielded significant improvements to the state-of-the-art on a well known, large scale LM task: from 51.3 down to 30.0 perplexity for single models whilst reducing the number of parameters by a factor of 20.
- We show that an ensemble of a number of different models can bring down perplexity on this task to 23.7, a large improvement compared to current state-of-art.
- We share the model and recipes in order to help and motivate further research in this area.

In Section 2 we review important concepts and previous work on language modeling. Section 3 presents our contributions to the field of neural language modeling, emphasizing large scale recurrent neural network training. Sections 4 and 5 aim at exhaustively describing our experience and understanding throughout the project, as well as emplacing our work relative to other known approaches.

2. Related Work

In this section we describe previous work relevant to the approaches discussed in this paper. A more detailed discussion on language modeling research is provided in (Mikolov, 2012).

2.1. Language Models

Language Modeling (LM) has been a central task in NLP. The goal of LM is to learn a probability distribution over sequences of symbols pertaining to a language. Much work has been done on both parametric (e.g., log-linear models) and non-parametric approaches (e.g., count-based LMs). Count-based approaches (based on statistics of N-grams) typically add smoothing which account for unseen (yet possible) sequences, and have been quite successful. To this extent, Kneser-Ney smoothed 5-gram models (Kneser & Ney, 1995) are a fairly strong baseline which, for large amounts of training data, have challenged other parametric approaches based on Neural Networks (Bengio et al., 2006).

Most of our work is based on Recurrent Neural Networks (RNN) models which retain long term dependencies. To this extent, we used the Long-Short Term Memory model (Hochreiter & Schmidhuber, 1997) which uses a gating mechanism (Gers et al., 2000) to ensure proper propagation of information through many time steps. Much work has been done on small and large scale RNN-based LMs (Mikolov et al., 2010; Mikolov, 2012; Chelba et al., 2013; Zaremba et al., 2014; Williams et al., 2015; Ji et al., 2015a; Wang & Cho, 2015; Ji et al., 2015b). The architectures that we considered in this paper are represented in Figure 1.

In our work, we train models on the popular One Billion Word Benchmark, which can be considered to be a medium-sized data set for count-based LMs but a very large data set for NN-based LMs. This regime is most interesting to us as we believe learning a very good model of human language is a complex task which will require large models,

and thus large amounts of data. Further advances in data availability and computational resources helped our study. We argue this leap in scale enabled tremendous advances in deep learning. A clear example found in computer vision is Imagenet (Deng et al., 2009), which enabled learning complex vision models from large amounts of data (Krizhevsky et al., 2012).

A crucial aspect which we discuss in detail in later sections is the size of our models. Despite the large number of parameters, we try to minimize computation as much as possible by adopting a strategy proposed in (Sak et al., 2014) of projecting a relatively big recurrent state space down so that the matrices involved remain relatively small, yet the model has large memory capacity.

2.2. Convolutional Embedding Models

There is an increased interest in incorporating character-level inputs to build word embeddings for various NLP problems, including part-of-speech tagging, parsing and language modeling (Ling et al., 2015; Kim et al., 2015; Ballesteros et al., 2015). The additional character information has been shown useful on relatively small benchmark data sets.

The approach proposed in (Ling et al., 2015) builds word embeddings using bidirectional LSTMs (Schuster & Paliwal, 1997; Graves & Schmidhuber, 2005) over the characters. The recurrent networks process sequences of characters from both sides and their final state vectors are concatenated. The resulting representation is then fed to a Neural Network. This model achieved very good results on a part-of-speech tagging task.

In (Kim et al., 2015), the words characters are processed by a 1-d CNN (Le Cun et al., 1990) with max-pooling across the sequence for each convolutional feature. The resulting features are fed to a 2-layer highway network (Srivastava et al., 2015b), which allows the embedding to learn semantic representations. The model was evaluated on small-scale language modeling experiments for various languages and matched the best results on the PTB data set despite having 60% fewer parameters.

2.3. Softmax Over Large Vocabularies

Assigning probability distributions over large vocabularies is computationally challenging. For modeling language, maximizing log-likelihood of a given word sequence leads to optimizing cross-entropy between the target probability distribution (e.g., the target word we should be predicting), and our model predictions p. Generally, predictions come from a linear layer followed by a Softmax non-linearity: $p(w) = \frac{\exp(z_w)}{\sum_{w' \in V} \exp(z_{w'})}$ where z_w is the logit corresponding to a word w. The logit is generally computed as an inner product $z_w = h^T e_w$ where h is a context vector and e_w is a "word embedding" for w.

The main challenge when $|V|$ is very large (in the order of one million in this paper) is the fact that computing all inner products between h and all embeddings becomes prohibitively slow during training (even when exploiting matrix-matrix multiplications and modern GPUs). Several approaches have been proposed to cope with the scaling issue: importance sampling (Bengio et al., 2003; Bengio & Senécal, 2008), Noise Contrastive Estimation (NCE) (Gutmann & Hyvärinen, 2010; Mnih & Kavukcuoglu, 2013), self normalizing partition functions (Vincent et al., 2015) or Hierarchical Softmax (Morin & Bengio, 2005; Mnih & Hinton, 2009) – they all offer good solutions to this problem. We found importance sampling to be quite effective on this task, and explain the connection between it and NCE in the following section, as they are closely related.

3. Language Modeling Improvements

Recurrent Neural Networks based LMs employ the chain rule to model joint probabilities over word sequences:

$$p(w_1, \ldots, w_N) = \prod_{i=1}^{N} p(w_i | w_1, \ldots, w_{i-1})$$

where the context of all previous words is encoded with an LSTM, and the probability over words uses a Softmax (see Figure 1(a)).

3.1. Relationship between Noise Contrastive Estimation and Importance Sampling

As discussed in Section 2.3, a large scale Softmax is necessary for training good LMs because of the vocabulary size. A Hierarchical Softmax (Mnih & Hinton, 2009) employs a tree in which the probability distribution over words is decomposed into a product of two probabilities for each word, greatly reducing training and inference time as only the path specified by the hierarchy needs to be computed and updated. Choosing a good hierarchy is important for obtaining good results and we did not explore this approach further for this paper as sampling methods worked well for our setup.

Sampling approaches are only useful during training, as they propose an approximation to the loss which is cheap to compute (also in a distributed setting) – however, at inference time one still has to compute the normalization term over all words. Noise Contrastive Estimation (NCE) proposes to consider a surrogate binary classification task in which a classifier is trained to discriminate between true data, or samples coming from some arbitrary distribution. If both the noise and data distributions were known, the

optimal classifier would be:

$$p(Y = true|w) = \frac{p_d(w)}{p_d(w) + kp_n(w)}$$

where Y is the binary random variable indicating whether w comes from the true data distribution, k is the number of negative samples per positive word, and p_d and p_n are the data and noise distribution respectively (we dropped any dependency on previous words for notational simplicity).

It is easy to show that if we train a logistic classifier $p_\theta(Y = true|w) = \sigma(s_\theta(w, h) - \log kp_n(w))$ where σ is the logistic function, then, $p'(w) = softmax(s_\theta(w, h))$ is a good approximation of $p_d(w)$ (s_θ is a logit which e.g. an LSTM LM computes).

The other technique, which is based on importance sampling (IS), proposes to directly approximate the partition function (which comprises a sum over all words) with an estimate of it through importance sampling. Though the methods look superficially similar, we will derive a similar surrogate classification task akin to NCE which arrives at IS, showing a strong connection between the two.

Suppose that, instead of having a binary task to decide if a word comes from the data or from the noise distribution, we want to identify the words coming from the true data distribution in a set $W = \{w_1, \ldots, w_{k+1}\}$, comprised of k noise samples and one data distribution sample. Thus, we can train a multiclass loss over a multinomial random variable Y which maximizes $\log p(Y = 1|W)$, assuming w.l.o.g. that $w_1 \in W$ is always the word coming from true data. By Bayes rule, and ignoring terms that are constant with respect to Y, we can write:

$$p(Y = k|W) \propto_Y \frac{p_d(w_k)}{p_n(w_k)}$$

and, following a similar argument than for NCE, if we define $p(Y = k|W) = softmax(s_\theta(w_k) - \log p_n(w_k))$ then $p'(w) = softmax(s_\theta(w, h))$ is a good approximation of $p_d(word)$. Note that the only difference between NCE and IS is that, in NCE, we define a binary classification task between true or noise words with a logistic loss, whereas in IS we define a multiclass classification problem with a Softmax and cross entropy loss. We hope that our derivation helps clarify the similarities and differences between the two. In particular, we observe that IS, as it optimizes a multiclass classification task (in contrast to solving a binary task), may be a better choice. Indeed, the updates to the logits with IS are tied whereas in NCE they are independent.

3.2. CNN Softmax

The character-level features allow for a smoother and compact parametrization of the word embeddings. Recent efforts on small scale language modeling have used CNN character embeddings for the input embeddings (Kim et al., 2015). Although not as straightforward, we propose an extension to this idea to also reduce the number of parameters of the Softmax layer. Recall from Section 2.3 that the Softmax computes a logit as $z_w = h^T e_w$ where h is a context vector and e_w the word embedding. Instead of building a matrix of $|V| \times |h|$ (whose rows correspond to e_w), we produce e_w with a CNN over the characters of w as $e_w = CNN(chars_w)$ – we call this a CNN Softmax. We used the same network architecture to dynamically generate the Softmax word embeddings without sharing the parameters with the input word-embedding sub-network. For inference, the vectors e_w can be precomputed, so there is no computational complexity increase w.r.t. the regular Softmax.

We note that, when using an importance sampling loss such as the one described in Section 3.1, only a few logits have non-zero gradient (those corresponding to the true and sampled words). With a Softmax where e_w are independently learned word embeddings, this is not a problem. But we observed that, when using a CNN, all the logits become tied as the function mapping from w to e_w is quite smooth. As a result, a much smaller learning rate had to be used. Even with this, the model lacks capacity to differentiate between words that have very different meanings but that are spelled similarly. Thus, a reasonable compromise was to add a small correction factor which is learned per word, such that:

$$z_w = h^T CNN(chars_w) + h^T M corr_w$$

where M is a matrix projecting a low-dimensional embedding vector $corr_w$ back up to the dimensionality of the projected LSTM hidden state of h. This amounts to adding a bottleneck linear layer, and brings the CNN Softmax much closer to our best result, as can be seen in Table 1, where adding a 128-dim correction halves the gap between regular and the CNN Softmax.

Aside from a big reduction in the number of parameters and incorporating morphological knowledge from words, the other benefit of this approach is that out-of-vocabulary (OOV) words can easily be scored. This may be useful for other problems such as Machine Translation where handling out-of-vocabulary words is very important (Luong et al., 2014). This approach also allows parallel training over various data sets since the model is no longer explicitly parametrized by the vocabulary size – or the language. This has shown to help when using byte-level input embeddings for named entity recognition (Gillick et al., 2015),

and we hope it will enable similar gains when used to map onto words.

3.3. Char LSTM Predictions

The CNN Softmax layer can handle arbitrary words and is much more efficient in terms of number of parameters than the full Softmax matrix. It is, though, still considerably slow, as to evaluate perplexities we need to compute the partition function. A class of models that solve this problem more efficiently are character-level LSTMs (Sutskever et al., 2011; Graves, 2013). They make predictions one character at a time, thus allowing to compute probabilities over a much smaller vocabulary. On the other hand, these models are more difficult to train and seem to perform worse even in small tasks like PTB (Graves, 2013). Most likely this is due to the sequences becoming much longer on average as the LSTM reads the input character by character instead of word by word.

Thus, we combine the word and character-level models by feeding a word-level LSTM hidden state h into a small LSTM that predicts the target word one character at a time (see Figure 1(c)). In order to make the whole process reasonably efficient, we train the standard LSTM model until convergence, freeze its weights, and replace the standard word-level Softmax layer with the aforementioned character-level LSTM.

The resulting model scales independently of vocabulary size – both for training and inference. However, it does seem to be worse than regular and CNN Softmax – we are hopeful that further research will enable these models to replace fixed vocabulary models whilst being computationally attractive.

4. Experiments

All experiments were run using the TensorFlow system (Abadi et al., 2015), with the exception of some older models which were used in the ensemble.

4.1. Data Set

The experiments are performed on the 1B Word Benchmark data set introduced by (Chelba et al., 2013), which is a publicly available benchmark for measuring progress of statistical language modeling. The data set contains about 0.8B words with a vocabulary of 793471 words, including sentence boundary markers. All the sentences are shuffled and the duplicates are removed. The words that are out of vocabulary (OOV) are marked with a special UNK token (there are approximately 0.3% such words).

4.2. Model Setup

The typical measure used for reporting progress in language modeling is perplexity, which is the average per-word log-probability on the holdout data set: $e^{-\frac{1}{N}\sum_i \ln p_{w_i}}$. We follow the standard procedure and sum over all the words (including the end of sentence symbol).

We used the 1B Word Benchmark data set without any preprocessing. Given the shuffled sentences, they are input to the network as a batch of independent streams of words. Whenever a sentence ends, a new one starts without any padding (thus maximizing the occupancy per batch).

For the models that consume characters as inputs or as targets, each word is fed to the model as a sequence of character IDs of preespecified length (see Figure 1(b)). The words were processed to include special begin and end of word tokens and were padded to reach the expected length. I.e. if the maximum word length was 10, the word "`cat`" would be transformed to "`$cat^ `" due to the CNN model.

In our experiments we found that limiting the maximum word length in training to 50 was sufficient to reach very good results while 32 was clearly insufficient. We used 256 characters in our vocabulary and the non-ascii symbols were represented as a sequence of bytes.

4.3. Model Architecture

We evaluated many variations of RNN LM architectures. These include the dimensionalities of the embedding layers, the state, projection sizes, and number of LSTM layers to use. Exhaustively trying all combinations would be extremely time consuming for such a large data set, but our findings suggest that LSTMs with a projection layer (i.e., a bottleneck between hidden states as in (Sak et al., 2014)) trained with truncated BPTT (Williams & Peng, 1990) for 20 steps performed well.

Following (Zaremba et al., 2014) we use dropout (Srivastava, 2013) before and after every LSTM layer. The biases of LSTM forget gate were initialized to 1.0 (Jozefowicz et al., 2015). The size of the models will be described in more detail in the following sections, and the choices of hyper-parameters will be released as open source upon publication.

For any model using character embedding CNNs, we closely follow the architecture from (Kim et al., 2015). The only important difference is that we use a larger number of convolutional features of 4096 to give enough capacity to the model. The resulting embedding is then linearly transformed to match the LSTM projection sizes. This allows it to match the performance of regular word embeddings but only uses a small fraction of parameters.

Table 1. Best results of single models on the 1B word benchmark. Our results are shown below previous work.

MODEL	TEST PERPLEXITY	NUMBER OF PARAMS [BILLIONS]
SIGMOID-RNN-2048 (JI ET AL., 2015A)	68.3	4.1
INTERPOLATED KN 5-GRAM, 1.1B N-GRAMS (CHELBA ET AL., 2013)	67.6	1.76
SPARSE NON-NEGATIVE MATRIX LM (SHAZEER ET AL., 2015)	52.9	33
RNN-1024 + MAXENT 9-GRAM FEATURES (CHELBA ET AL., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (NO DROPOUT)	37.9	3.3
LSTM-8192-2048 (50% DROPOUT)	32.2	3.3
2-LAYER LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN INPUTS	**30.0**	**1.04**
BIG LSTM+CNN INPUTS + CNN SOFTMAX	39.8	**0.29**
BIG LSTM+CNN INPUTS + CNN SOFTMAX + 128-DIM CORRECTION	35.8	**0.39**
BIG LSTM+CNN INPUTS + CHAR LSTM PREDICTIONS	47.9	**0.23**

Table 2. Best results of ensembles on the 1B Word Benchmark.

MODEL	TEST PERPLEXITY
LARGE ENSEMBLE (CHELBA ET AL., 2013)	43.8
RNN+KN-5 (WILLIAMS ET AL., 2015)	42.4
RNN+KN-5 (JI ET AL., 2015A)	42.0
RNN+SNM10-SKIP (SHAZEER ET AL., 2015)	41.3
LARGE ENSEMBLE (SHAZEER ET AL., 2015)	41.0
OUR 10 BEST LSTM MODELS (EQUAL WEIGHTS)	26.3
OUR 10 BEST LSTM MODELS (OPTIMAL WEIGHTS)	26.1
10 LSTMS + KN-5 (EQUAL WEIGHTS)	25.3
10 LSTMS + KN-5 (OPTIMAL WEIGHTS)	25.1
10 LSTMS + SNM10-SKIP (SHAZEER ET AL., 2015)	**23.7**

4.4. Training Procedure

The models were trained until convergence with an AdaGrad optimizer using a learning rate of 0.2. In all the experiments the RNNs were unrolled for 20 steps without ever resetting the LSTM states. We used a batch size of 128. We clip the gradients of the LSTM weights such that their norm is bounded by 1.0 (Pascanu et al., 2012).

Using these hyper-parameters we found large LSTMs to be relatively easy to train. The same learning rate was used in almost all of the experiments. In a few cases we had to reduce it by an order of magnitude. Unless otherwise stated, the experiments were performed with 32 GPU workers and asynchronous gradient updates. Further details will be fully specified with the code upon publication.

Training a model for such large target vocabulary (793471 words) required to be careful with some details about the approximation to full Softmax using importance sampling. We used a large number of negative (or noise) samples: 8192 such samples were drawn per step, but were shared across all the target words in the batch (2560 total, i.e. 128 times 20 unrolled steps). This results in multiplying (2560 x 1024) times (1024 x (8192+1)) (instead of (2560 x 1024) times (1024 x 793471)), i.e. about 100-fold less computation.

5. Results and Analysis

In this section we summarize the results of our experiments and do an in-depth analysis. Table 1 contains all results for our models compared to previously published work. Table 2 shows previous and our own work on ensembles of models. We hope that our encouraging results, which improved the best perplexity of a single model from 51.3 to 30.0 (whilst reducing the model size considerably), and set a new record with ensembles at 23.7, will enable rapid research and progress to advance Language Modeling. For

this purpose, we will release the model weights and recipes upon publication.

5.1. Size Matters

Unsurprisingly, size matters: when training on a very large and complex data set, fitting the training data with an LSTM is fairly challenging. Thus, the size of the LSTM layer is a very important factor that influences the results, as seen in Table 1. The best models are the largest we were able to fit into a GPU memory. Our largest model was a 2-layer LSTM with 8192+1024 dimensional recurrent state in each of the layers. Increasing the embedding and projection size also helps but causes a large increase in the number of parameters, which is less desirable. Lastly, training an RNN instead of an LSTM yields poorer results (about 5 perplexity worse) for a comparable model size.

5.2. Regularization Importance

As shown in Table 1, using dropout improves the results. To our surprise, even relatively small models (e.g., single layer LSTM with 2048 units projected to 512 dimensional outputs) can over-fit the training set if trained long enough, eventually yielding holdout set degradation.

Using dropout on non-recurrent connections largely mitigates these issues. While over-fitting still occurs, there is no more need for early stopping. For models that had 4096 or less units in the LSTM layer, we used 10% dropout probability. For larger models, 25% was significantly better. Even with such regularization, perplexities on the training set can be as much as 6 points below test.

In one experiment we tried to use a smaller vocabulary comprising of the 100,000 most frequent words and found the difference between train and test to be smaller – which suggests that too much capacity is given to rare words. This is less of an issue with character CNN embedding models as the embeddings are shared across all words.

5.3. Importance Sampling is Data Efficient

Table 3 shows the test perplexities of NCE vs IS loss after a few epochs of 2048 unit LSTM with 512 projection. The IS objective significantly improves the speed and the overall performance of the model when compared to NCE.

5.4. Word Embeddings vs Character CNN

Replacing the embedding layer with a parametrized neural network that process characters of a given word allows the model to consume arbitrary words and is not restricted to a fixed vocabulary. This property is useful for data sets with conversational or informal text as well as for morphologically rich languages. Our experiments show that using character-level embeddings is feasible and does not degrade performance – in fact, our best single model uses a Character CNN embedding.

Table 3. The test perplexities of an LSTM-2048-512 trained with different losses versus number of epochs. The model needs about 40 minutes per epoch. First epoch is a bit slower because we slowly increase the number of workers.

EPOCHS	NCE	IS	TRAINING TIME [HOURS]
1	97	60	1
5	58	47.5	4
10	53	45	8
20	49	44	14
50	46.1	43.7	34

Table 4. Nearest neighbors in the character CNN embedding space of a few out-of-vocabulary words. Even for words that the model has never seen, the model usually still finds reasonable neighbors.

WORD	TOP-1	TOP-2	TOP-3
INCERDIBLE	INCREDIBLE	NONEDIBLE	EXTENDIBLE
WWW.A.COM	WWW.AA.COM	WWW.AAA.COM	WWW.CA.COM
7546	7646	7534	8566
TOWNHAL1	TOWNHALL	DJC2	MOODSWING360
KOMARSKI	KOHARSKI	KONARSKI	KOMANSKI

An additional advantage is that the number of parameters of the input layer is reduced by a factor of 11 (though training speed is slightly worse). For inference, the embeddings can be precomputed so there is no speed penalty. Overall, the embedding of the best model is parametrized by 72M weights (down from 820M weights).

Table 4 shows a few examples of nearest neighbor embeddings for some out-of-vocabulary words when character CNNs are used.

5.5. Smaller Models with CNN Softmax

Even with character-level embeddings, the model is still fairly large (though much smaller than the best competing models from previous work). Most of the parameters are in the linear layer before the Softmax: 820M versus a total of 1.04B parameters.

In one of the experiments we froze the word-LSTM after convergence and replaced the Softmax layer with the CNN Softmax sub-network. Without any fine-tuning that model was able to reach 39.8 perplexity with only 293M weights (as seen in Table 1).

As described in Section 3.2, adding a "correction" word embedding term alleviates the gap between regular and

CNN Softmax. Indeed, we can trade-off model size versus perplexity. For instance, by adding 100M weights (through a 128 dimensional bottleneck embedding) we achieve 35.8 perplexity (see Table 1).

To contrast with the CNN Softmax, we also evaluated a model that replaces the Softmax layer with a smaller LSTM that predicts one character at a time (see Section 3.3). Such a model does not have to learn long dependencies because the base LSTM still operates at the word-level (see Figure 1(c)). With a single-layer LSTM of 1024 units we reached 49.0 test perplexity, far below the best model. In order to make the comparisons more fair, we performed a very expensive marginalization over the words in the vocabulary (to rule out words not in the dictionary which the character LSTM would assign some probability). When doing this marginalization, the perplexity improved a bit down to 47.9.

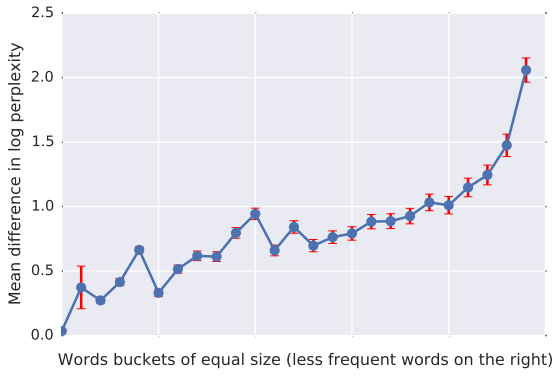

Figure 2. The difference in log probabilities between the best LSTM and KN-5 (higher is better). The words from the hold-out set are grouped into 25 buckets of equal size based on their frequencies.

5.6. Training Speed

We used 32 Tesla K40 GPUs to train our models. The smaller version of the LSTM model with 2048 units and 512 projections needs less than 10 hours to reach below 45 perplexity and after only **2 hours** of training the model beats previous state-of-the art on this data set. The best model needs about 5 days to get to 35 perplexity and 10 days to 32.5. The best results were achieved after 3 weeks of training. See Table 3 for more details.

5.7. Ensembles

We averaged several of our best models and we were able to reach 23.7 test perplexity (more details and results can be seen in Table 2), which is more than 40% improvement over previous work. Interestingly, including the best N-gram model reduces the perplexity by 1.2 point even though the model is rather weak on its own (67.6 perplexity). Most previous work had to either ensemble with the best N-gram model (as their RNN only used a limited output vocabulary of a few thousand words), or use N-gram features as additional input to the RNN. Our results, on the contrary, suggest that N-grams are of limited benefit, and suggest that a carefully trained LSTM LM is the most competitive model.

5.8. LSTMs are best on the tail words

Figure 2 shows the difference in log probabilities between our best model (at 30.0 perplexity) and the KN-5. As can be seen from the plot, the LSTM is better across all the buckets and significantly outperforms KN-5 on the rare words. This is encouraging as it seems to suggest that LSTM LMs may fare even better for languages or data sets where the number of rare words is larger than traditional N-gram models.

5.9. Samples from the model

To qualitatively evaluate the model, we sampled many sentences. We discarded short and politically incorrect ones, but the sample shown below is otherwise "raw" (i.e., not hand picked). The samples are of high quality – which is not a surprise, given the perplexities attained – but there are still some occasional mistakes.

Sentences generated by the ensemble (about 26 perplexity):

> $< S >$ With even more new technologies coming onto the market quickly during the past three years , an increasing number of companies now must tackle the ever-changing and ever-changing environmental challenges online . $< S >$ Check back for updates on this breaking news story . $< S >$ About 800 people gathered at Hever Castle on Long Beach from noon to 2pm , three to four times that of the funeral cortège . $< S >$ We are aware of written instructions from the copyright holder not to , in any way , mention Rosenberg 's negative comments if they are relevant as indicated in the documents , " eBay said in a statement . $< S >$ It is now known that coffee and cacao products can do no harm on the body . $< S >$ Yuri Zhirkov was in attendance at the Stamford Bridge at the start of the second half but neither Drogba nor Malouda was able to push on through the Barcelona defence .

6. Discussion and Conclusions

In this paper we have shown that RNN LMs can be trained on large amounts of data, and outperform competing models including carefully tuned N-grams. The reduction in perplexity from 51.3 to 30.0 is due to several key components which we studied in this paper. Thus, a large, regularized LSTM LM, with projection layers and trained with an approximation to the true Softmax with importance sampling performs much better than N-grams. Unlike previous work, we do not require to interpolate both the RNN LM and the N-gram, and the gains of doing so are rather marginal.

By exploring recent advances in model architectures (e.g. LSTMs), exploiting small character CNNs, and by sharing our findings in this paper and accompanying code and models (to be released upon publication), we hope to inspire research on large scale Language Modeling, a problem we consider crucial towards language understanding. We hope for future research to focus on reasonably sized datasets taking inspiration from recent advances seen in the computer vision community thanks to efforts such as Imagenet (Deng et al., 2009).

Acknowledgements

We thank Ciprian Chelba, Ilya Sutskever, and the Google Brain Team for their help and discussions. We also thank Koray Kavukcuoglu for his help with the manuscript.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Arisoy, Ebru, Sainath, Tara N, Kingsbury, Brian, and Ramabhadran, Bhuvana. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pp. 20–28. Association for Computational Linguistics, 2012.

Ballesteros, Miguel, Dyer, Chris, and Smith, Noah A. Improved transition-based parsing by modeling characters instead of words with lstms. *arXiv preprint arXiv:1508.00657*, 2015.

Bengio, Yoshua and Senécal, Jean-Sébastien. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722, 2008.

Bengio, Yoshua, Senécal, Jean-Sébastien, et al. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, 2003.

Bengio, Yoshua, Schwenk, Holger, Senécal, Jean-Sébastien, Morin, Fréderic, and Gauvain, Jean-Luc. Neural probabilistic language models. In *Innovations in Machine Learning*, pp. 137–186. Springer, 2006.

Chelba, Ciprian, Mikolov, Tomas, Schuster, Mike, Ge, Qi, Brants, Thorsten, Koehn, Phillipp, and Robinson, Tony. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

Filippova, Katja, Alfonseca, Enrique, Colmenares, Carlos A, Kaiser, Lukasz, and Vinyals, Oriol. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 360–368, 2015.

Gers, Felix A, Schmidhuber, Jürgen, and Cummins, Fred. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Gillick, Dan, Brunk, Cliff, Vinyals, Oriol, and Subramanya, Amarnag. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*, 2015.

Graves, Alex. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Graves, Alex and Schmidhuber, Jürgen. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5): 602–610, 2005.

Gutmann, Michael and Hyvärinen, Aapo. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pp. 297–304, 2010.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Ji, Shihao, Vishwanathan, S. V. N., Satish, Nadathur, Anderson, Michael J., and Dubey, Pradeep. Blackout: Speeding up recurrent neural network language models

with very large vocabularies. *CoRR*, abs/1511.06909, 2015a. URL http://arxiv.org/abs/1511.06909.

Ji, Yangfeng, Cohn, Trevor, Kong, Lingpeng, Dyer, Chris, and Eisenstein, Jacob. Document context language models. *arXiv preprint arXiv:1511.03962*, 2015b.

Jozefowicz, Rafal, Zaremba, Wojciech, and Sutskever, Ilya. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2342–2350, 2015.

Kalchbrenner, Nal, Grefenstette, Edward, and Blunsom, Phil. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Kim, Yoon, Jernite, Yacine, Sontag, David, and Rush, Alexander M. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*, 2015.

Kneser, Reinhard and Ney, Hermann. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pp. 181–184. IEEE, 1995.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Le Cun, B Boser, Denker, John S, Henderson, D, Howard, Richard E, Hubbard, W, and Jackel, Lawrence D. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.

Ling, Wang, Luís, Tiago, Marujo, Luís, Astudillo, Ramón Fernandez, Amir, Silvio, Dyer, Chris, Black, Alan W, and Trancoso, Isabel. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

Luong, Minh-Thang, Sutskever, Ilya, Le, Quoc V, Vinyals, Oriol, and Zaremba, Wojciech. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Mikolov, Tomáš. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 2012.

Mikolov, Tomas and Zweig, Geoffrey. Context dependent recurrent neural network language model. In *SLT*, pp. 234–239, 2012.

Mikolov, Tomas, Karafiát, Martin, Burget, Lukas, Cernockỳ, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, pp. 3, 2010.

Mikolov, Tomas, Deoras, Anoop, Kombrink, Stefan, Burget, Lukas, and Cernockỳ, Jan. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, number s 1, pp. 605–608, 2011.

Mnih, Andriy and Hinton, Geoffrey E. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pp. 1081–1088, 2009.

Mnih, Andriy and Kavukcuoglu, Koray. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pp. 2265–2273, 2013.

Morin, Frederic and Bengio, Yoshua. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pp. 246–252. Citeseer, 2005.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

Rush, Alexander M, Chopra, Sumit, and Weston, Jason. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Sak, Hasim, Senior, Andrew W, and Beaufays, Françoise. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pp. 338–342, 2014.

Schuster, Mike and Paliwal, Kuldip K. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

Schwenk, Holger, Rousseau, Anthony, and Attik, Mohammed. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pp. 11–19. Association for Computational Linguistics, 2012.

Serban, Iulian Vlad, Sordoni, Alessandro, Bengio, Yoshua, Courville, Aaron C., and Pineau, Joelle. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808, 2015. URL http://arxiv.org/abs/1507.04808.

Shazeer, Noam, Pelemans, Joris, and Chelba, Ciprian. Sparse non-negative matrix language modeling for skip-grams. *Proceedings of Interspeech*, pp. 1428–1432, 2015.

Srivastava, Nitish. *Improving neural networks with dropout*. PhD thesis, University of Toronto, 2013.

Srivastava, Nitish, Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015a.

Srivastava, Rupesh K, Greff, Klaus, and Schmidhuber, Jürgen. Training very deep networks. In *Advances in Neural Information Processing Systems*, pp. 2368–2376, 2015b.

Sutskever, Ilya, Martens, James, and Hinton, Geoffrey E. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Vaswani, Ashish, Zhao, Yinggong, Fossum, Victoria, and Chiang, David. Decoding with large-scale neural language models improves translation. Citeseer.

Vincent, Pascal, de Brébisson, Alexandre, and Bouthillier, Xavier. Efficient exact gradient update for training deep networks with very large sparse targets. In *Advances in Neural Information Processing Systems*, pp. 1108–1116, 2015.

Vinyals, Oriol and Le, Quoc. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

Wang, Tian and Cho, Kyunghyun. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*, 2015.

Williams, Ronald J and Peng, Jing. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

Williams, Will, Prasad, Niranjani, Mrva, David, Ash, Tom, and Robinson, Tony. Scaling recurrent neural network language models. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 5391–5395. IEEE, 2015.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.