Joint Language and Translation Modeling with Recurrent Neural Networks

Michael Auli, Michel Galley, Chris Quirk, Geoffrey Zweig
Microsoft Research
Redmond, WA, USA
{michael.auli,mgalley,chrisq,gzweig}@microsoft.com

Abstract

We present a joint language and translation model based on a recurrent neural network which predicts target words based on an unbounded history of both source and target words. The weaker independence assumptions of this model result in a vastly larger search space compared to related feed-forward-based language or translation models. We tackle this issue with a new lattice rescoring algorithm and demonstrate its effectiveness empirically. Our joint model builds on a well known recurrent neural network language model (Mikolov, 2012) augmented by a layer of additional inputs from the source language. We show competitive accuracy compared to the traditional channel model features. Our best results improve the output of a system trained on WMT 2012 French-English data by up to 1.5 BLEU, and by 1.1 BLEU on average across several test sets.

1 Introduction

Recently, several feed-forward neural network-based language and translation models have achieved impressive accuracy improvements on statistical machine translation tasks (Allauzen et al., 2011; Le et al., 2012b; Schwenk et al., 2012). In this paper we focus on recurrent neural network architectures, which have recently advanced the state of the art in language modeling (Mikolov et al., 2010; Mikolov et al., 2011a; Mikolov, 2012), outperforming multi-layer feed-forward based networks in both perplexity and word error rate in speech recognition (Arisoy et al., 2012; Sundermeyer et al., 2013). The major attraction of recurrent architectures is their potential to capture long-span dependencies since predictions are based on an *unbounded history* of previous words. This is in contrast to feed-forward networks as well as conventional n-gram models, both of which are limited to fixed-length contexts. Building on the success of recurrent architectures, we base our joint language and translation model on an extension of the recurrent neural network language model (Mikolov and Zweig, 2012) that introduces a layer of additional inputs (§2).

Most previous work on neural networks for speech recognition or machine translation used a rescoring setup based on n-best lists (Arisoy et al., 2012; Mikolov, 2012) for evaluation, thereby side stepping the algorithmic and engineering challenges of direct decoder-integration.[1] Instead, we exploit *lattices*, which offer a much richer representation of the decoder output, since they compactly encode an exponential number of translation hypotheses in polynomial space. In contrast, n-best lists are typically very redundant, representing only a few combinations of top scoring arcs in the lattice. A major challenge in lattice rescoring with a recurrent neural network model is the effect of the unbounded history on search since the usual dynamic programming assumptions which are exploited for efficiency do not hold up anymore. We apply a novel algorithm to the task of rescoring with an unbounded language model and empirically demonstrate its effectiveness (§3).

The algorithm proves robust, leading to significant improvements with the recurrent neural network language model over a competitive n-gram baseline across several language pairs. We even observe consistent gains when pairing the model with a large n-gram model trained on up to 575 times more

[1] One notable exception is Le et al. (2012a) who rescore reordering lattices with a feed-forward network-based model.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1044–1054,
Seattle, Washington, USA, 18-21 October 2013.

data, demonstrating that the model provides complementary information (§4).

Our joint modeling approach is based on adding a *continuous space representation* of the foreign sentence as an additional input to the recurrent neural network language model. With this extension, the language model can measure the consistency between the source and target words in a context-sensitive way. The model effectively combines the functionality of both the traditional channel and language model features. We test the power of this new model by using it as the only source of traditional channel information. Overall, we find that the model achieves accuracy competitive with the older channel model features and that it can improve over the gains observed with the recurrent neural network language model (§5).

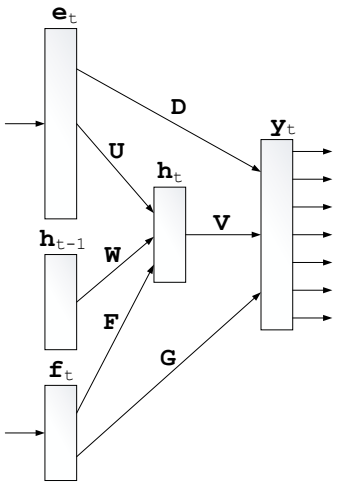

Figure 1: Structure of the recurrent neural network model, including the auxiliary input layer $\mathbf{f}_t$.

2 Model Structure

We base our model on the recurrent neural network language model of Mikolov et al. (2010) which is factored into an input layer, a hidden layer with recurrent connections, and an output layer (Figure 1). The input layer encodes the target language word at time t as a 1-of-N vector $\mathbf{e}_t$, where $|V|$ is the size of the vocabulary, and the output layer $\mathbf{y}_t$ represents a probability distribution over target words; both of size $|V|$. The hidden layer state $\mathbf{h}_t$ encodes the history of all words observed in the sequence up to time step t. This model is extended by an *auxiliary input layer* $\mathbf{f}_t$ which provides complementary information to the input layer (Mikolov and Zweig, 2012). While the auxiliary input layer can be used to feed in arbitrary additional information, we focus on encodings of the foreign sentence (§5).

The state of the hidden layer is determined by the input layer, the auxiliary input layer and the hidden layer configuration of the previous time step $\mathbf{h}_{t-1}$. The weights of the connections between the layers are summarized in a number of matrices: $\mathbf{U}$, $\mathbf{F}$ and $\mathbf{W}$, represent weights from the input layer to the hidden layer, from the auxiliary input layer to the hidden layer, and from the previous hidden layer to the current hidden layer, respectively. Matrix $\mathbf{V}$ represents connections between the current hidden layer and the output layer; $\mathbf{G}$ represents direct weights between the auxiliary input and output layers.

The hidden and output layers are computed via a series of matrix-vector products and non-linearities:

$$\mathbf{h}_t = s(\mathbf{U}\mathbf{e}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{F}\mathbf{f}_t)$$
$$\mathbf{y}_t = g(\mathbf{V}\mathbf{h}_t + \mathbf{G}\mathbf{f}_t)$$

where

$$s(z) = \frac{1}{1+\exp\{-z\}}, \quad g(z_m) = \frac{\exp\{z_m\}}{\sum_k \exp\{z_k\}}$$

are sigmoid and softmax functions, respectively. Additionally, the network is interpolated with a maximum entropy model of sparse n-gram features over input words (Mikolov et al., 2011a).[2] The maximum entropy weights are added to the output activations before computing the softmax.

The model is optimized via a maximum likelihood objective function using stochastic gradient descent. Training is based on the back propagation through time algorithm, which unrolls the network and then computes error gradients over multiple time steps (Rumelhart et al., 1986). After training, the output layer represents posteriors $p(e_{t+1}|e_{t-n+1}^t, \mathbf{h}_t, \mathbf{f}_t)$; the probabilities of words in the output vocabulary given the n previous input words e_{t-n+1}^t, the hidden layer configuration $\mathbf{h}_t$ as well as the auxiliary input layer configuration $\mathbf{f}_t$.

[2]While these features depend on multiple input words, we depicted them for simplicity as a connection between the current input word vector $\mathbf{e}_t$ and the output layer ($\mathbf{D}$).

Naïve computation of the probability distribution over the next word is very expensive for large vocabularies. A well established efficiency trick uses word-classing to create a more efficient two-step process (Goodman, 2001; Emami and Jelinek, 2005; Mikolov et al., 2011b) where each word is assigned a unique class. To compute the probability of a word, we first compute the probability of its class, and then multiply it by the probability of the word conditioned on the class:

$$p(e_{t+1}|e_{t-n+1}^{t}, \mathbf{h}_t, \mathbf{f}_t) = \\ p(c_i|e_{t-n+1}^{t}, \mathbf{h}_t, \mathbf{f}_t) \times p(e_{t+1}|c_i, e_{t-n+1}^{t}, \mathbf{h}_t, \mathbf{f}_t)$$

This factorization reduces the complexity of computing the output probabilities from $\mathcal{O}(|V|)$ to $\mathcal{O}(|C| + \max_i |c_i|)$ where $|C|$ is the number of classes and $|c_i|$ is the number of words in class c_i. The best case complexity $\mathcal{O}(\sqrt{|V|})$ requires the number of classes and words to be evenly balanced, i.e., each class contains exactly as many words as there are classes.

3 Lattice Rescoring with an Unbounded Language Model

We evaluate our joint language and translation model in a lattice rescoring setup, allowing us to search over a much larger space of translations than would be possible with n-best lists. While very space efficient, lattices also impose restrictions on the context available to features, a particularly challenging setting for our model which depends on the entire prefix of a translation. In the ensuing description we introduce a new algorithm to efficiently tackle this issue.

Phrase-based decoders operate by maintaining a set of *states* representing competing translations, either partial or complete. Each state is scored by a number of features including the n-gram language model. The independence assumptions of the features determine the amount of *context* each state needs to maintain in order for it to be possible to assign a score to it. For example, a trigram language model is indifferent to any context other than the two immediately preceding words. Assuming the trigram model dominates the Markov assumptions of all other features, which is typically the case, then we have to maintain at least two words at each state, also known as the *n-gram context*.

```
1: function RescoreLattice(k, V, E, s, T)
2:   Q ← topologically-sort(V)
3:   for all v in V do                    ▷ Heaps of split-states
4:     H_v ← minHeap()
5:   end for
6:   h_0 ← 0⃗                              ▷ Initialize start-state
7:   H_s.add(h_0)
8:   for all v in Q do                    ▷ Examine outgoing arcs
9:     for ⟨v, x⟩ in E do
10:      for h in H_v do                  ▷ Extend LM states
11:        h' ← scoreRNN(h, phrase(h))
12:        parent(h') ← h                 ▷ Backpointers
13:        if H_x.size() ≥ k∧             ▷ Beam width
14:          H_x.min()<score(h') then
15:            H_x.removeMin()
16:        if H_x.size()<k then
17:            H_x.add(h')
18:      end for
19:    end for
20:  end for
21:  I = maxHeap()
22:  for all t in T do                    ▷ Find best final split-state
23:    I.merge(H_t)
24:  end for
25:  return I.max()
26: end function
```

Figure 2: Push-forward rescoring with a recurrent neural network language model given a beam-width for language model split-states k, decoder states V, edges E, a start state s and final states T.

However, a recurrent neural network language model makes much weaker independence assumptions. In fact, the predictions of such a model depend on *all* previous words in the sentence, which would imply a potentially very large context. But storing all words is an inefficient solution from a dynamic programming point of view. Fortunately, we do not need to maintain entire translations as context in the states: the recurrent model compactly encodes the entire history of previous words in the hidden layer configuration $\mathbf{h}_i$. It is therefore sufficient to add $\mathbf{h}_i$ as context, instead of the entire translation. The language model can then simply score any new words

based on $\mathbf{h}_i$ from the previous state when a new state is created.

A much larger problem is that items, that were previously equivalent from a dynamic programming perspective, may now be different. Standard phrase-based decoders (Koehn et al., 2007) *recombine* decoder states with the same context into a single state because they are equivalent to the model features; usually recombination retains only the highest scoring candidate.[3] However, if the context is large, then the amount of recombination will decrease significantly, leading to less variety in the decoder beam. This was confirmed in preliminary experiments where we simulated context sizes of up to 100 words but found that accuracy dropped by between 0.5-1.0 BLEU.

Integrating a long-span language model naïvely requires to keep context equivalent to the entire left prefix of the translation, a setting which would permit very little recombination. Instead of using inefficient long-span contexts, we propose to maintain the usual n-gram context and to keep a fixed number of hidden layer configurations k at each decoder state. This leads to a new split-state dynamic program which splits each decoder state into at most k new items, each with a separate hidden layer configuration representing an unbounded history (Figure 2). This maintains diversity in the explored translation hypothesis space and preserves high-scoring hidden layer configurations.

What is the effect of this strategy? To answer this question we measured translation accuracy for various settings of k on our lattice rescoring setup (see §4 for details). In the same experiment, we compare lattices to n-best lists in terms of accuracy, model score and wall time impact.[4] The results (Table 1 and Figure 3) show that reranking accuracy on lattices is not significantly better, however, rescoring lattices with $k = 1$ is much faster than n-best lists. Similar observations have been made in previous work on minimum error-rate training (Macherey et al., 2008). The recurrent language model adds an overhead of about 54% at $k = 1$ on top of the time to produce the baseline 1-best output, a considerable but not necessarily prohibitive overhead. Larger values of k return higher probability solutions, but there is little impact on accuracy: the BLEU score is nearly identical when retaining up to 100 histories compared to keeping only the highest scoring.

While surprising at first, we believe that this effect is due to the high similarity of the translations represented by the histories in the beam. Each history represents a different translation but all translation hypothesis share the same n-gram context, and, more importantly, they are translations of the *same foreign words*, since they have exactly the same coverage vector. These commonalities are likely to result in similar recurrent histories, which in turn reduces the effect of aggressive pruning.

	BLEU	oracle	sec/sent
Baseline	28.25	-	0.173
100-best	28.90	37.22	0.470
1000-best	28.99	40.06	3.920
lattice ($k = 1$)	29.00	43.50	0.093
lattice ($k = 10$)	29.04	43.50	0.599
lattice ($k = 100$)	29.03	43.50	4.531

Table 1: Rescoring n-best lists and lattices with various language model beam widths k. Accuracy is based on the news2011 French-English task. Timing results are in *addition* to the baseline.

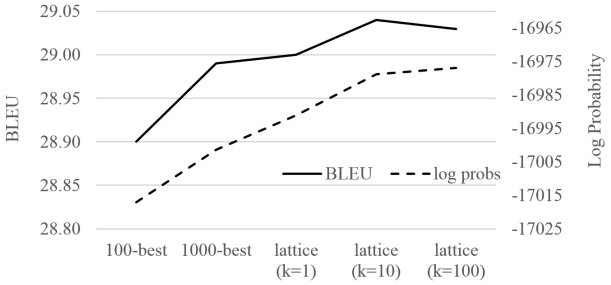

Figure 3: BLEU vs. log probabilities of 1-best translations when rescoring n-best lists and lattices (cf. Table 1).

4 Language Model Experiments

Recurrent neural network language models have previously only been used in n-best rescoring

[3] Assuming a max-translation decision rule. In a minimum-risk setting, we may assign the sum of the scores of all candidates to the retained item.

[4] We measured running times on an HP z800 workstation equipped with 24 GB main memory and two Xeon E5640 CPUs with four cores each, clocked at 2.66 GHz. All experiments were run single-threaded.

settings and on small-scale tasks with baseline language models trained on only 17.5m words (Mikolov, 2012). We extend this work by experimenting on lattices using strong baselines with n-gram models trained on over one billion words and by evaluating on a number of language pairs.

4.1 Experimental Setup

Baseline. We experiment with an in-house phrase-based system similar to Moses (Koehn et al., 2003), scoring translations by a set of common features including maximum likelihood estimates of source given target mappings $p_{MLE}(e|f)$ and vice versa $p_{MLE}(f|e)$, as well as lexical weighting estimates $p_{LW}(e|f)$ and $p_{LW}(f|e)$, word and phrase-penalties, a linear distortion feature and a lexicalized reordering feature. Log-linear weights are estimated with minimum error rate training (Och, 2003).

Evaluation. We use training and test data from the WMT 2012 campaign and report results on French-English, German-English and English-German. Translation models are estimated on 102m words of parallel data for French-English, 91m words for German-English and English-German; between 3.5-5m words are newswire, depending on the language pair, and the remainder are parliamentary proceedings. The baseline systems use two 5-gram modified Kneser-Ney language models; the first is estimated on the target-side of the parallel data, while the second is based on a large newswire corpus released as part of the WMT campaign. For French-English and German-English we use a language model based on 1.15bn words, and for English-German we train a model on 327m words. We evaluate on the newswire test sets from 2010-2011 containing between 2034-3003 sentences. Log-linear weights are estimated on the 2009 data set comprising 2525 sentences. We rescore the lattices produced by the baseline systems with an aggressive but effective context beam of $k = 1$ that did not harm accuracy in preliminary experiments (§3).

Neural Network Language Model. The vocabularies of the language models are comprised of the words in the training set after removing singletons. We obtain word-classes using a version of Brown-Clustering with an additional regularization term to optimize the runtime of the language model (Brown et al., 1992; Zweig and Makarychev, 2013). Direct connections use maximum entropy features over unigrams, bigrams and trigrams (Mikolov et al., 2011a). We use the standard settings for the model with the default learning rate $\alpha = 0.1$ that decays exponentially if the validation set entropy does not increase after each epoch. Back propagation through time computes error gradients over the past twenty time steps. Training is stopped after 20 epochs or when the validation entropy does not decrease over two epochs. We experiment with varying training data sizes and randomly draw the data from the same corpora used for the baseline systems. Throughout, we use a hidden layer size of 100 which provided a good trade-off between time and accuracy in initial experiments.

4.2 Results

Training times for neural networks can be a major bottleneck. Recurrent architectures are particularly hard to parallelize due to their inherent dependence on the previous hidden layer configuration. One straightforward way to influence training time is to change the size of the training corpus.

Our results (Table 2, Table 3 and Table 4) show that even small models trained on only two million words significantly improve over the 1-best decoder output (Baseline); this represents only 0.6 percent of the data available to the n-gram model used by the baseline. Models of this size can be trained in only about 3.5 hours. A model trained on 50m words took 63 hours to train. When paired with an n-gram model trained on 25 times more data, accuracy improved by up to 0.7 BLEU on French-English.

5 Joint Model Experiments

In the next set of experiments, we turn to the joint language and translation model, an extension of the recurrent neural network language model with additional inputs for the foreign sentence. We first introduce two continuous space representations of the foreign sentence (§5.1). Using these representations we evaluate the accuracy of the joint model in the lattice rescoring setup and compare against the traditional translation channel model features (§5.2). Next, we establish an upper bound on accuracy for the joint model via an oracle experiment (§5.3). Inspired by the results of the oracle experiment we

	dev	news2010	news2011	newssyscomb2011	Avg(test)
Baseline	26.6	27.6	28.3	27.5	27.8
+RNNLM (2m)	27.5	28.1	28.6	28.1	28.3
+RNNLM (50m)	27.7	28.2	29.0	28.1	28.5

Table 2: French-English results when rescoring with the recurrent neural network language model; the baseline relies on an n-gram model trained on 1.15bn words.

	dev	news2010	news2011	newssyscomb2011	Avg(test)
Baseline	21.2	20.7	19.2	20.6	20.0
+RNNLM (2m)	21.8	20.9	19.4	20.9	20.3
+RNNLM (50m)	22.1	21.1	19.7	21.0	20.5

Table 3: German-English results when rescoring with the recurrent neural network language model.

	dev	news2010	news2011	newssyscomb2011	Avg(test)
Baseline	15.2	15.6	14.3	15.7	15.1
+RNNLM (2m)	15.7	15.9	14.6	16.0	15.4
+RNNLM (50m)	15.8	15.9	14.7	16.1	15.5

Table 4: English-German results when rescoring with the recurrent neural network language model; the baseline relies on an n-gram model trained on 327m words.

train a transform between the source words and the reference representations. This leads to the best results improving 1.5 BLEU over the 1-best decoder output and adding 0.2 BLEU on average to the gains achieved by the recurrent language model (§5.4).

Setup. Conventional language models can be trained on monolingual or bilingual data; however, the joint model can only be trained on the latter. In order to control for data size effects, we restrict training of all models, including the baseline n-gram model, to the target side of the parallel corpus, about 102m words for French-English. Furthermore we train recurrent models only on the newswire portion (about 3.5m words for training and 250k words for validation) since initial experiments showed comparable results to using the full parallel corpus, available to the baseline. This is reasonable since the test data is newswire. Also, it allows for more rapid experimentation.

5.1 Foreign Sentence Representations

We represent foreign sentences either by latent semantic analysis (LSA; Deerwester et al. 1990) or by word encodings produced as a by-product of training the recurrent neural network language model on the source words.

LSA is widely used for representing words and documents in low-dimensional vector space. The method applies reduced singular value decomposition (SVD) to a matrix M of word counts; in our setting, rows represent sentences and columns represent foreign words. SVD reduces the number of columns while preserving similarity among the rows, effectively mapping from a high-dimensional representation of a sentence, as a set of words, to a low-dimensional set of *concepts*. The output of SVD is an approximation of M by three matrices: T contains single word representations, R represents full sentences, and S is a diagonal scaling matrix:

$$M \approx TSR^T$$

Given vocabulary V and n sentences, we construct M as a matrix of size $|V \times n|$. The ij-th entry is the number of times word i occurs in sentence j, also known as the term frequency value; the entry is also weighted by the inverse document frequency, the relative importance of word i among all sentences, expressed as the negative logarithm of the fraction of sentences in which word i occurs.

As a second representation we use single *word*

embeddings implicitly learned by the input layer weights $\mathbf{U}$ of the recurrent neural network language model (§2), denoted as **RNN**. Each word is represented by a vector of size $|\mathbf{h}_i|$, the number of neurons in the hidden layer; in our experiments, we consider concatenations of individual word vectors to represent foreign word contexts. These encodings have previously been found to capture syntactic and semantic regularities (Mikolov et al., 2013) and are readily available in our experimental framework via training a recurrent neural network language model on the source-side of the parallel corpus.

5.2 Results

We first experiment with the two previously introduced representations of the source-side sentence. Table 5 shows the results compared to the 1-best decoder output and an RNN language model (target-only). We first try LSA encodings of the entire foreign sentence as 80 or 240 dimensional vectors (sent-lsa-dim80, sent-lsa-dim240). Next, we experiment with single-word RNN representations of sliding word-windows in the hope of representing relevant context more precisely. Word-windows are constructed relative to the source words aligned to the current target word, and individual word vectors are concatenated into a single vector. We first try contexts which do not include the aligned source words, in the hope of capturing information not already modeled by the channel models, starting with the next five words (ww-rnn-dim50.n5), the five previous and the next five words (ww-rnn-dim50.p5n5) as well as the previous three words (ww-rnn-dim50.p3). Next, we experiment with word-windows of up to five aligned source words (ww-rnn-dim50.c5). Finally, we try contexts based on LSA word vectors (ww-lsa-dim50.n5, ww-lsa-dim50.p3).[5]

While all models improve over the baseline, none significantly outperforms the recurrent neural network language model in terms of BLEU. However, the perplexity results suggest that the models utilize the foreign representations since all joint models improve vastly over the target-only language model. The lowest perplexity is achieved by the context covering the aligned source words (ww-rnn-dim50.c5) since the source words are a better predictor of the target words than outside context.

[5]We ignore the coverage vector when determining word-windows which risks including already translated words. Building word-windows based on the coverage vector requires additional state in a rescoring setting meant to be light-weight.

	$-p(e\|f)$	$-p(f\|e)$ $-p(e\|f)$
Baseline without CM	24.0	22.5
+ target-only	24.5	22.6
+ sent-lsa-dim240	24.9	23.3
+ ww-rnn-dim50.n5	24.9	24.0
+ ww-rnn-dim50.p5n5	24.6	23.7
+ ww-rnn-dim50.p3	24.6	22.3
+ ww-rnn-dim50.c5	24.9	24.0
+ ww-lsa-dim50.n5	24.8	23.9
+ ww-lsa-dim50.p3	23.8	23.2

Table 6: Comparison of the joint model and the channel model features (CM) by removing channel features corresponding to $-p(e|f)$ from the lattices, or both directions $-p(e|f), -p(f|e)$ and replacing them by various joint models. We re-tuned the log-linear weights for different feature-sets. Accuracy is based on the average BLEU over news2010, newssyscomb2010, news2011.

The experiments so far measured if the joint model can improve *in addition* to the four channel model features used by the baseline, that is, the maximum likelihood and lexical translation features in both translation directions. The joint model clearly overlaps with these features, but how well does the recurrent model perform compared *against* the channel model features? To answer this question, we removed channel model features corresponding to the same translation direction as the joint model, specifically $p_{MLE}(e|f)$ and $p_{LW}(e|f)$, from the lattices and measured the effect of adding the joint models.

The results (Table 6, column $-p(e|f)$) clearly show that our joint models are competitive with the channel model features by outperforming the original baseline with all channel model features (24.7 BLEU) by 0.2 BLEU (ww-rnn-dim50.n5, ww-rnn-dim50.c5). As a second experiment, we removed all channel model features (column $-p(e|f), p(f|e)$), diminishing baseline accuracy to 22.5 BLEU. In this setting, the best joint model is able to make up 1.5 of the 2.2 BLEU lost due to removal of the channel

	dev	news2010	news2011	newssyscomb2010	Avg(test)	PPL
Baseline	24.3	24.4	25.1	24.3	24.7	341
target-only	25.1	25.1	26.4	25.0	25.6	218
sent-lsa-dim80	25.2	25.2	26.3	25.1	25.6	147
sent-lsa-dim240	25.1	25.0	26.2	24.9	25.4	126
ww-rnn-dim50.n5	24.9	25.0	26.3	24.8	25.4	61
ww-rnn-dim50.p5n5	25.0	24.8	26.2	24.7	25.3	59
ww-rnn-dim50.p3	25.1	25.1	26.5	24.9	25.6	143
ww-rnn-dim50.c5	24.8	24.9	26.0	24.8	25.3	16
ww-lsa-dim50.n5	25.0	25.0	26.2	24.8	25.4	76
ww-lsa-dim50.p3	25.1	25.1	26.5	24.9	25.6	151

Table 5: Translation accuracy of the joint model with various encodings of the foreign sentence measured on the French-English task. Perplexity (PPL) is based on news2011.

model features, while modeling only a single translation direction. This setup also shows the negligible effect of the target-only language model in the absence of translation scores, whereas the joint models are much more effective since they do model translation. Overall, the best joint models prove very competitive to the traditional channel features.

5.3 Oracle Experiment

The previous section examined the effect of a set of basic foreign sentence representations. Although we find some benefit from these representations, the differences are not large. One might naturally ask whether there is greater potential upside from this channel model. Therefore we turn to measuring the upper bound on accuracy for the joint approach as a whole.

Specifically, we would like to find a bound on accuracy given an *ideal representation* of the source sentence. To answer this question, we conducted an experiment where the joint model has access to an LSA representation of the reference translation.

Table 7 shows that the joint approach has an oracle accuracy of up to 4.3 BLEU above the baseline. This clearly confirms that the joint approach can exploit the additional information to improve BLEU, given a good enough representation of the foreign sentence. In terms of perplexity, we see an improvement of up to 65% over the target-only model. It should be noted that since LSA representations are computed on reference words, perplexity no longer has its standard meaning.

	BLEU	PPL
Baseline	25.2	341
target-only	26.4	218
oracle (sent-lsa-dim40)	27.7	124
oracle (sent-lsa-dim80)	28.5	103
oracle (sent-lsa-dim160)	29.0	86
oracle (sent-lsa-dim240)	29.5	76

Table 7: Oracle accuracy of the joint model when using an LSA encoding of the references, measured on the news2011 French-English task.

5.4 Target Language Projections

Our experiments so far showed that joint models based on direct representations of the source words are very competitive to the traditional channel models (§5.2). However, these experiments have not shown any improvements over the normal recurrent neural network language model. The previous section demonstrated that good representations can lead to substantial gains (§5.3). In order to bridge the gap, we propose to learn a separate transform from the foreign words to an encoding of the reference target words, thus making the source-side representations look more like the target-side encodings used in the oracle experiment.

Specifically, we learn a linear transform $d_\theta : \mathbf{x} \rightarrow \mathbf{r}$ mapping directly from a vector encoding of the foreign sentence $\mathbf{x}$ to an l-dimensional LSA representation $\mathbf{r}$ of the reference sentence. At test and training time we apply d_θ to the foreign words and use the transformation instead of a direct

	dev	news2010	news2011	newssyscomb2010	Avg(test)	PPL
Baseline	24.3	24.4	25.1	24.3	24.7	341
target-only	25.1	25.1	26.4	25.0	25.6	218
proj-lsa-dim40	25.1	25.3	26.5	25.2	25.8	145
proj-lsa-dim80	25.1	25.3	26.6	25.2	25.8	134

Table 8: Translation accuracy of the joint model with a source-target transform, measured on the French-English task. Perplexity (PPL) is based on news2011; differences to target-only are significant at the $p < 0.001$ level.

source-side representation.

The transform models all foreign words in the parallel corpus except singletons, which are collapsed into a unique class, similar to the recurrent neural network language model. We train the transform to minimize the *squared error* with respect to the reference LSA vector using an SGD online learner:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{n} \Big(\mathbf{r}_i - d_\theta(\mathbf{x}_i) \Big)^2 \qquad (1)$$

We found a simple constant learning rate, tuned on the validation data, to be as effective as schedules based on constant decay, or reducing the learning rate when the validation error increased. Our feature-set includes unigram and bigram word features. The value of unigram features is simply the unigram count in that sentence; bigram features receive a weight of the bigram count divided by two to help prevent overfitting. Then the vector for each sentence was divided by its L2 norm. Both weighting and normalization led to substantial improvements in test set error. More complex features such as skip-bigrams, trigrams and character n-grams did not yield any significant improvements. Even this representation of sentences is composed of a large number of instances, and so we resorted to feature hashing by computing feature ids as the least significant 20 bits of each feature name. Our best transform achieved a cosine similarity of 0.816 on the training data, 0.757 on the validation data, and 0.749 on news2011.

The results (Table 8) show that the transform improves over the recurrent neural network language model on all test sets and by 0.2 BLEU on average. We verified significance over the target-only model using paired bootstrap resampling (Koehn, 2004) over all test sets (7526 sentences) at the $p < 0.001$ level. Overall, we improve accuracy by up to 1.5 BLEU and by 1.1 BLEU on average across all test sets over the decoder 1-best with our joint language and translation model.

6 Related Work

Our approach of combining language and translation modeling is very much in line with recent work on n-gram-based translation models (Crego and Yvon, 2010), and more recently continuous space-based translation models (Le et al., 2012a; Gao et al., 2013). The joint model presented in this paper differs in a number of key aspects: we use a recurrent architecture representing an unbounded history of both source and target words, rather than a feed-forward style network. Feed-forward networks and n-gram models have a finite history which makes predictions independent of anything but a small history of words. Furthermore, we only model the target-side which is different to previous work modeling both sides.

We introduced a new algorithm to tackle lattice rescoring with an unbounded model. The automatic speech recognition community has previously addressed this issue by either approximating long-span language models via simpler but more tractable models (Deoras et al., 2011b), or by identifying confusable subsets of the lattice from which n-best lists are constructed and rescored (Deoras et al., 2011a). We extend their work by directly mapping a recurrent neural network model onto the structure of the lattice, rescoring all states instead of focusing only on subsets.

7 Conclusion and Future Work

Joint language and translation modeling with recurrent neural networks leads to substantial gains over the 1-best decoder output, raising accuracy by up to 1.5 BLEU and by 1.1 BLEU on average across

several test sets. The joint approach also improves over the gains of the recurrent neural network language model, adding 0.2 BLEU on average across several test sets. Our models are competitive to the traditional channel models, outperforming them in a head-to-head comparison.

Furthermore, we tackled the issue of lattice rescoring with an unbounded recurrent model by means of a novel algorithm that keeps a beam of recurrent histories. Finally, we have shown that the recurrent neural network language model can significantly improve over n-gram baselines across a range of language-pairs, even when the baselines were trained on 575 times more data.

In future work we plan to directly learn representations of the source-side during training of the joint model. Thus, the model itself can decide which encoding is best for the task. We also plan to change the cross entropy objective to a BLEU-inspired objective in a discriminative training regime, which we hope to be more effective. We would also like to apply recent advances in tackling the vanishing gradient problem (Pascanu et al., 2013) using a regularization term to maintain the magnitude of the gradients during back propagation through time. Finally, we would like to integrate the recurrent model directly into first-pass decoding, a straightforward extension of lattice rescoring using the algorithm we developed.

Acknowledgments

We would like to thank Anthony Aue, Hany Hassan Awadalla, Jon Clark, Li Deng, Sauleh Eetemadi, Jianfeng Gao, Qin Gao, Xiaodong He, Will Lewis, Arul Menezes, and Kristina Toutanova for helpful discussions related to this work as well as for comments on previous drafts. We would also like to thank the anonymous reviewers for their comments.

References

Alexandre Allauzen, Hélène Bonneau-Maynard, Hai-Son Le, Aurélien Max, Guillaume Wisniewski, François Yvon, Gilles Adda, Josep Maria Crego, Adrien Lardilleux, Thomas Lavergne, and Artem Sokolov. 2011. LIMSI @ WMT11. In *Proc. of WMT*, pages 309–315, Edinburgh, Scotland, July. Association for Computational Linguistics.

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep Neural Network Language Models. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 20–28, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Josep Crego and Franois Yvon. 2010. Factored bilingual n-gram language models for statistical machine translation. *Machine Translation*, 24(2):159–175.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Anoop Deoras, Tomáš Mikolov, and Kenneth Church. 2011a. A Fast Re-scoring Strategy to Capture Long-Distance Dependencies. In *Proc. of EMNLP*, pages 1116–1127, Stroudsburg, PA, USA, July. Association for Computational Linguistics.

Anoop Deoras, Tomáš Mikolov, Stefan Kombrink, M. Karafiat, and Sanjeev Khudanpur. 2011b. Variational Approximation of Long-Span Language Models for LVCSR. In *Proc. of ICASSP*, pages 5532–5535.

Ahmad Emami and Frederick Jelinek. 2005. A Neural Syntactic Language Model. *Machine Learning*, 60(1-3):195–227, September.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2013. Learning Semantic Representations for the Phrase Translation Model. Technical Report MSR-TR-2013-88, Microsoft Research, September.

Joshua Goodman. 2001. Classes for Fast Maximum Entropy Training. In *Proc. of ICASSP*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of HLT-NAACL*, pages 127–133, Edmonton, Canada, May.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, Jun.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of EMNLP*, pages 388–395, Barcelona, Spain, Jul.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012a. Continuous Space Translation Models with

Neural Networks. In *Proc. of HLT-NAACL*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

Hai-Son Le, Thomas Lavergne, Alexandre Allauzen, Marianna Apidianaki, Li Gong, Aurélien Max, Artem Sokolov, Guillaume Wisniewski, and François Yvon. 2012b. LIMSI @ WMT12. In *Proc. of WMT*, pages 330–337, Montréal, Canada, June. Association for Computational Linguistics.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based Minimum Error Rate Training for Statistical Machine Translation. In *Proc. of EMNLP*, pages 725–734, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomáš Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *Proc. of Spoken Language Technologies (SLT)*, pages 234–239, Dec.

Tomáš Mikolov, Karafiát Martin, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proc. of INTERSPEECH*, pages 1045–1048.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký. 2011a. Strategies for Training Large Scale Neural Network Language Models. In *Proc. of ASRU*, pages 196–201.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. In *Proc. of ICASSP*, pages 5528–5531.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space-Word Representations. In *Proc. of NAACL*, pages 746–751, Stroudsburg, PA, USA, June. Association for Computational Linguistics.

Tomáš Mikolov. 2012. *Statistical Language Models based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL*, pages 160–167, Sapporo, Japan, July.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training Recurrent Neural Networks. *Proc. of ICML*, abs/1211.5063.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning Internal Representations by Error Propagation. In *Symposium on Parallel and Distributed Processing*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics.

Martin Sundermeyer, Ilya Oparin, Jean-Luc Gauvain, Ben Freiberg, Ralf Schlüter, and Hermann Ney. 2013. Comparison of Feedforward and Recurrent Neural Network Language Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8430–8434, Vancouver, Canada, May.

Geoff Zweig and Konstantin Makarychev. 2013. Speed Regularization and Optimality in Word Classing. In *Proc. of ICASSP*.