Improving Continuous Space Language Models using Auxiliary Features

Walid Aransa, Holger Schwenk, Loïc Barrault

LIUM, University of Le Mans, France

firstname.lastname@lium.univ-lemans.fr

Abstract

In this paper we introduce a novel method to improve the continuous space language models using auxiliary features. The suggested auxiliary features include text genre, line length, various types of context vector representations. We report perplexity improvements of around 7.5% of the English Penn Treebank data set. We also report an improvement on a translation task up to 1.1 BLEU point on test by re-scoring the n-best list generated by our phrase-based statistical machine translation system.

1. Introduction

The neural network LM (also known as continuous space LM or CSLM) tries to overcome the disadvantages of back-off n-gram LMs. One of these disadvantages is that the probabilities are estimated in a discrete space which does not allow directly the estimation of non-observed n-gram in the training data. In a neural network LM, the words are projected into a continuous space during the training. [1] proposes a multi-layer neural network model that jointly learns the word projection and the probability estimation. The basic architecture of this neural network is shown in Figure 1.

A CSLM has many advantages, it can be used to estimate the probability of long n-gram (also short n-gram) which can not be directly estimated using n-gram back-off LMs. Also, it can be trained using longer context with just small increase in the complexity which is not possible for n-gram back-off LMs.

The CSLM was successfully applied to large vocabulary speech recognition. It is usually used to rescore lattices and improvement of the word error rate by about one point were obtained for many languages and domains, for instance [3, 4, 5, 6]. More recently, the CSLM was also successfully applied to statistical machine translation [7, 8, 9, 10].

In this paper, we present improvements of the CSLM. The idea is to provide additional information at the input of the neural network in a similar for recurrent NN LM by [18]. We call these additional inputs "auxiliary features". We use different types of auxiliary features including line length, text genre, line context vector representation,... etc. By these means, better domain and context specific LM estimations can be obtained.

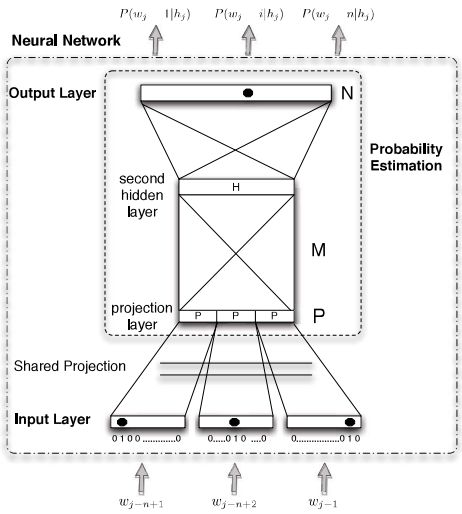

Figure 1: The neural network language model architecture. P, N and H are the size of one projection, one hidden layer and the output layer respectively. h_j denotes the context $w_{j-(n-1)}^{j-1}$.

We report the results using perplexity as well as when these improved CSLMs are integrated into an SMT system. This is performed by re-scoring the n-best list and adding an additional feature function.

2. Modified architecture

The basic architecture of a CSLM with auxiliary data is shown in Figure 2. The example in the figure shows only one additional auxiliary feature vector. This architecture would allow different auxiliary information for each n-gram, but since our goal is to model the topic or long-term context, we made the choice to keep the auxiliary data constant for all n-grams of one sentence. Therefore, the auxiliary data is loaded once for each sentence. If more than one auxiliary feature is desired, the dimension of the auxiliary feature vector will be equal to the sum of the individual feature dimensions. In this case the auxiliary feature vector will be the concatenation of two or more feature vectors. This architecture also allows us to use sentence-level features as well as document (or corpus) level features by using the same auxil-

iary vector for all lines in the document (or corpus).
The functionality of auxiliary features has been integrated in the open-source CSLM toolkit [1] [9].

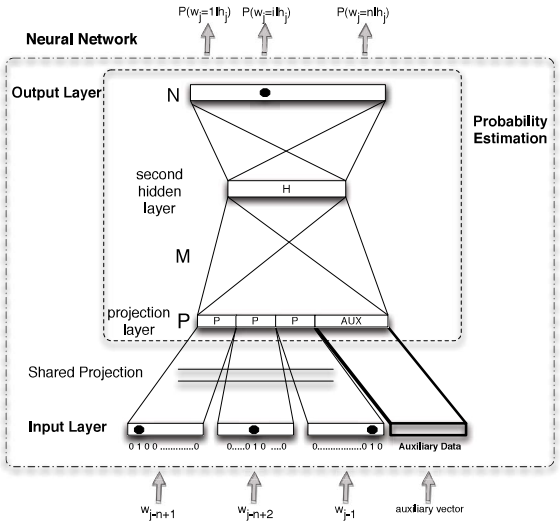

Figure 2: CSLM modified architecture with additional new auxiliary data input to the neural network

3. Related work

Although, we focus on improving CSLM in this work, some related research focus on improving the standard n-gram language models by integrating more context or semantic knowledge. Kuhn and De Mori [12] proposed to calculate the probabilities which correspond to the relative proportion of the last N words. They present a combined LM that interpolates a general trigram LM and another LM called a cache-based LM which is trained on the last N words. The relative interpolation weights assigned to each component are based on the POS of each word. The cache component assigns higher probability to recently encountered words. In our work, the context is represented as a continuous space vector. It can be one line or the whole history back to the beginning of the document. In the latter case more weight is given to recent lines.

Bellegarda [13] proposed a method to use more global constraints to improve LM since local constraints are already captured by the n-gram model. They use latent semantic analysis (LSA) which automatically discovers the semantic relationships between words and documents in a given corpus. In their approach, words and documents are mapped into a continuous semantic vector space, in which clustering techniques are used. This allows the characterization of parallel layers of semantic knowledge in the space, with variable granularity. The resulting LMs complement the conventional n-gram LMs. They suggested to use hybrid n-gram+LSA models to benefit from the advantages of several smoothing techniques.

In a similar work, Coccaro and Jurafsky [14] integrated semantic knowledge into an n-gram LM using LSA and a word similarity algorithm. Since LSA is a bad predictor of frequent words, they used a geometric instead of a linear combination based on a per-word confidence metric. In our work, instead of using LSA , we use the line context vector representations which is calculated using the embeddings of the words in this line. The word embeddings are the projections learned during CSLM training. We were motivated by what was reported recently by Baroni et al. [15] that the context predictive models (i.e. word embedding) outperform classic count-vector-based distributional semantic approaches.

Other works, like the work of Iyer and Ostendorf [16] focused on developing a sentence-level mixture language model that takes advantage of the topic constraints in a sentence or article. They proposed topic-dependent dynamic cache adaptation techniques in the framework of mixture models. An automatic clustering algorithm was used to classify text with two levels of mixture models for smoothing. In our work a predefined genre is assigned to different corpora, which is used as additional input to the neural network. However, it is also possible to use topics instead of genres and to assign the topic dynamically by using similar automatic clustering algorithm like the one used by [16].

Khudanpur and Wu [17] proposed an LM that combines collocational dependencies with the syntactic structure and the topic of the sentence. They integrate these dependencies using a maximum entropy technique. They report a substantial improvement in perplexity and in the accuracy of a speech recognition task. In our work, instead of using topic, we used the genre of the sentence. Since we are using auxiliary features on the sentence level, it could be envisioned to extend our work to use syntactic features.

Mikolov and Zweig [18] focused on improving the performance of recurrent neural network language models (RNNLMs) by using a topic-conditioned RNNLM. They used a contextual real-value input vector in association with each input word. This vector is used to convey contextual information about the sentence being modeled. They use Latent Dirichlet Allocation (LDA) to get a compact vector-space representation of a long span context which they conventionally interpreted as a topic representation. They argue that their approach has the key advantage of avoiding the data fragmentation associated with building multiple topic models on different data subsets. The main differences with our work are, that we used a feed-forward neural network and context vector representation instead of LDA. Also, we evaluated the impact of using various types of auxiliary feature as explained in Section 4.

[1] Available for download from https://github.com/hschwenk/cslm-toolkit

4. Auxiliary features

In this work, we experimented with two types of auxiliary features: the **first** one provides a feature of the current line itself (e.g. the number of words or genre) which allows us to train feature-conditioned continuous space language models. Some of these features are motivated by research in the machine translation quality estimation literature. The **second** type of auxiliary feature aims at providing a larger context. Table 1 summarizes the auxiliary features of this type that we have experimented with. Each auxiliary feature has a reference name that we are using in this paper.

One of the basic auxiliary feature we used is **LineLen** or the line length, expressed in number of words. We used an 1-of-n encoding to generate this feature vector. The ith value in the vector is set to 1 if the line length is equal to i, and zeros otherwise. We considered a maximum line length of $n = 200$, so if the line length exceeded 200 words, we use $n = 200$. In our experiments this 1-of-n encoding is projected into a continuous space like for the words.

Aux feature	Embeddings
CurrLine	words in the **current** line
PrecLine	words in the **preceding** line
PrecHCurrLines	**current** line and **h preceding** lines
AllPrecCurrWords	words in the **current** and **all preceding lines**
AllPrecWords	words in **all preceding** lines
AllPrecLines	**all preceding** lines

Table 1: *Auxiliary features using normalized weighted sum of different embeddings*

The **Genre** consists of a binary vector with dimension equal to the number of genres we have. As for LineLen, we used a 1-of-n encoding. In our training data, we have 5 genres as shown in Table 3.

For the context vector representation auxiliary features, We used various ways to compose them. One of the composition is **CurrLine** $\hat{\alpha}_l$ of a line l. This will be the normalized sum of the word embeddings e_w of all tokens $w \in l$ computed as follows:

$$\hat{\alpha}_l = \frac{\sum_{w \in l} e_w}{||\sum_{w \in l} e_w||} \quad (1)$$

Similarly, **PrecLine** auxiliary feature $\hat{\beta}_l$ is calculated as follows:

$$\hat{\beta}_l = \frac{\sum_{w \in l-1} e_w}{||\sum_{w \in l-1} e_w||} \quad (2)$$

For **PrecHCurrLines**, we calculate the weighted sum of the context vector representation of the current line $\hat{\alpha}_l$ and the preceding H lines. The farther the line is in the past, the lower the weight is. The vector of a line l is calculated as follows:

$$\hat{\eta}_{l,H} = \frac{\sum_{i=l-H}^{l} \hat{\alpha}_i \lambda^{l-i}}{||\sum_{i=l-H}^{l} \hat{\alpha}_i \lambda^{l-i}||} \quad (3)$$

In our experiments we used different values of H=10, 30, 50 and λ=0.95.

The differences between **AllPrecLines** and **PrecHCurrLines** is that the first one does not include the current line context vector representation in the calculation of its vector and that it uses all preceding lines not just the H preceding lines. The equation used to calculate the feature vector of AllPrecLines of a line l is as follows:

$$\hat{\omega}_l = \frac{\sum_{i=1}^{l-1} \hat{\alpha}_i \lambda^{l-i}}{||\sum_{i=1}^{l-1} \hat{\alpha}_i \lambda^{l-i}||} \quad (4)$$

For the first line, we used the context vector representation of itself (i.e. $\hat{\omega}_1 = \hat{\alpha}_1$). In our experiments, we used several weights: $\lambda = 0.85, 0.95, 0.98$.

For **AllPrecCurrWords**, the line context vector representation $\hat{\sigma}_l$ is calculated using all preceding words with a weight λ that gives more weight to the near history **words** and lower weight to the far history **words**. The equation used to calculate the feature vector of AllPrecCurrWords of a line l is the following:

$$\hat{\sigma}_l = \frac{\sum_{i=1}^{W'-1} e_{w_i} \lambda^{W'-i}}{||\sum_{i=1}^{W'-1} e_{w_i} \lambda^{W'-i}||} \quad (5)$$

where W' is the number of words in the current and all preceding lines. In our work we experiment with the following weights: $\lambda = 0.75, 0.85, 0.95$.

AllPrecWords is calculated in a similar way as **AllPrecCurrWords**, but excluding the words of the current line.

5. Evaluation on Penn Treebank

We first evaluated our work on the English Penn Treebank (PTB) corpus [19]. This is a very small corpus ($<$ 1 million words training data), but it has the advantage that many comparable results are published. We limited our evaluation on PTB to use only the preceding line auxiliary feature (i.e. PrecLine). The features **LineLen** and **CurrLine** can not be used when using perplexity to evaluate an LM since they provide information on the future. However, it is valid and useful to apply them in an n-best list re-scoring framework, as discussed later in this paper.

The perplexity values on PTB for several configurations are shown in Table 2. We experiment with different learning rate

scales for the first layer of the neural network as shown in the third column in Table 2. This means that the first layer learning rate is scaled by this value which means that the network learns the weights faster than other layers weights and possibly learns better projection weights. **Copy** means that no weights are learned and the auxiliary feature vector is copied to the next layer directly.
In CSLM1, using auxiliary features and unified learning rate scale decreased the perplexity slightly. The same happen when we replaced **Copy** by a **sequence of double hyperbolic tangent** in CSLM3, and when we increased the learning rate scale to 2 in CSLM4, comparing to Baseline2. Changing the learning rate scale to 3 in CSLM5, again, decreased the perplexity by 7.5 on dev and 7.2 on test vs. Baseline2. So the perplexity of CSLM5 compared to Baseline1 decreased by 7.6% on dev and 7.5% on test.

System	Aux layer	lrs	DevSet PPL	TestSet PPL
Baseline1 (No Aux)	-	1	133.19	127.66
Baseline2 (No Aux)	-	2	130.48	125.28
CSLM1	Copy	1	128.26	123.45
CSLM2	Copy	2	124.80	120.32
CSLM3	Seq. of two tanh	1	127.15	121.93
CSLM4	Seq. of two tanh	2	124.22	118.57
CSLM5	Seq. of two tanh	3	**122.98**	**118.08**

Table 2: *Perplexity on Penn Treebank using the PrecLine auxiliary feature with different auxiliary layer topology and learning rate scale (lrs) for the first layer.*

To understand these results, we compared systems with the same setup except for one variable. Comparing Baseline1 and Baseline2 shows the impact of increasing the learning rate scale from unified to 2. Also comparing CSLM1 and CSLM2 gives us the impact related to the increase of learning rate scale for word embeddings only since the **Copy** layer used for auxiliary feature does not have any weights. Also comparing CSLM1 and CSLM3, gives us the impact of using **sequence of double hyperbolic tangent** layer for auxiliary data instead of **Copy**. We observed that this allows the network to deeply learn from the auxiliary data. These three comparisons accumulated a perplexity decrease of 7.28 on dev and 7.03 on test. We concluded that using auxiliary feature decreases the perplexity with different meta-configuration and topology by around 7.5% on dev and test.

6. SMT experimental results

We evaluate the performance of our improved CSLMs which use auxiliary features in the context of SMT. This is done by using them to re-score the n-best list provided by an SMT system. A new CSLM score is added to the n-best list for each hypothesis and the coefficients of all feature functions are optimized. In the following subsections, we describe our baseline system and the rescoring results with some discussions.

6.1. Baseline system of SMS/Chat

The language pair of the baseline system is Arabic Egyptian dialect into the English. The translation task is SMS/Chat translation in the context of DARPA BOLT project. The system is a standard phrase-based system trained using Moses toolkit [21], SRILM [22], KenLM [23], and GIZA++ [20]. Log linear weights are optimized using MERT [20]. We evaluated the translation quality using BLEU [24].
We used the following technique to build our baseline SMT system:

- **Data selection**: We selected the most relevant sentences to the task from the bilingual corpora based on the work of [25] using XenC [26] open source toolkit . The selected sentences are used to train our phrase-based system. Since our SMT system is for SMS/Chat genre, the training data size using data selection is 4.7m words only as shown in Table 3 compared to the full available bilingual corpora size of 191.26m. Another advantage of using data selection is to have smaller translation model. Dev and test sets are shown in Table 4. Dev set is used for tuning the weights of the feature functions.

corpus	corpus genre	selected size Ar/En tokens
smschat	SMS/CHAT (Egyptian)	648k/845k
gale	Modern Standard Arabic (MSA)	128k/158k
e103		44k/46k
fix		73k/84k
ummah		36k/37k
isi		354k/348k
bolt	FORUM (Egyptian)	136k/165k
bbnturk		167k/177k
bbnlev	FORUM (Levantine)	111k/124k
un	FORMAL MSA (UN)	1.34m/1.27m
cts	CALLS (Egyptian)	1.24m/1.45m
Total	-	4.28m/4.7m

Table 3: The size of the selected data from bilingual corpora for SMS/CHAT SMT baseline system

type	# Arabic tokens	# English tokens	genre
dev	19.7k	25.6k	SMS/CHAT
test	19.4k	24.6k	SMS/CHAT

Table 4: *Development and test sets of SMS/Chat SMT system*

- **Data weighting**: This method is used to weight the bilingual sub-corpora models according to their importance to the translation task. We used a method based on the work of [27] using perplexity minimization given the development set. if $\overline{s}$ and $\overline{t}$ denote the source and target phrase respectively, we are instantly optimizing the weight of the four features: $p(\overline{s}|\overline{t})$, $lex(\overline{s}|\overline{t})$, $p(\overline{t}|\overline{s})$ and $lex(\overline{t}|\overline{s})$ in the Moses translation model.

- **Language modeling**: We used data selection method based on [28] to select the relevant monolingual data for our 4-gram back-off language model. The back-off LM was used in SMT decoding for generating the 1000-best translation output. We used this back-off LM also in CSLM re-scoring to calculate the probability of words not in the CSLM shortlist.

type	data set	# English tokens	genre
train	gale	5.01	MSA
	bolt	2.05m	FORUM (Egyptian)
	smschat	845k	SMS/CHAT
	Total	7.9m	-
dev	smschat dev	25.6k	SMS/CHAT

Table 5: *Training corpora and dev set used to train and tune the CSLM models*

6.2. Result and analysis of re-scoring the n-best list

CSLM models with various auxiliary features were trained using CSLM toolkit on three English corpora (total of 7.91m words) which are the target side of the bilingual corpora shown in Table 5.
The results obtained by re-scoring the n-best list created by the baseline system are summarized in Table 6. The table contains the best result for each auxiliary feature. Detailed results can be found in Tables 7 and 8. Since the test set BLEU scores of both **SMT Baseline** and **CSLM Baseline** without auxiliary data are the same, we decided to use SMT Baseline as the Baseline for the result analysis.

The CSLMs English training corpora used in these experiments is about 7.9m tokens (see Table 5). These results were obtained with the best meta-parameters (i.e. H and λ). In Table 6, we described the CSLM model, auxiliary feature dimension, auxiliary feature projection dimension along with the BLEU scores on dev and test. We used **projection** layer for **LineLen** auxiliary feature, **Copy** layer for **Genre** auxiliary feature, **sequence of double hyperbolic tangent** layer for the rest of auxiliary features. All experiments are trained with 24-gram context size.

System	Aux dim/proj.	Dev	Test
SMT Baseline	-	27.35	25.72
CSLM Baseline (No AuxData)	-	28.04	25.67
LineLen	1/200	28.65	26.14
Genre	5/-	**28.90**	**26.32**
CurrLine	320/-	28.29	26.09
PrecLine	320/-	28.67	**26.33**
PrecHCurrLines λ=0.95, h=50	320/-	**28.92**	26.26
AllPrecCurrWords λ=0.75	320/-	28.52	25.86
AllPrecWords λ=0.95	320/-	**28.77**	**26.82**
AllPrecLines λ=0.98	320/-	28.63	**26.52**

Table 6: BLEU *scores obtained when re-scoring the n-best list using different auxiliary data.*

Looking at Table 6, we observed a good improvement using *LineLen* auxiliary feature, but *Genre* has relatively better gain on both dev and test. This means that *Genre* is better discriminative auxiliary feature.

We observed that *PrecLine* provides better performance due to better context information compared to *CurrLine*. We also observed that CSLMs with auxiliary features which contain the current line (i.e. *AllPrecCurrWords*, *PrecHCurrLines*) generally have lower BLEU scores than CSLMs with auxiliary features which do not contain the current line. We concluded that using current line is not so useful for re-scoring n-best list because instead of predicting the next word, the CSLM would rather learn to find the next word from the input auxiliary feature making undesirable cycle in the model.

PrecLine has +0.6 BLEU gain on test. If one preceding line is useful, two or more preceding lines would be more useful (possibly weighted). We can verify this assumption by looking at *AllPrecLines* result, which uses auxiliary feature that does not contain the current line (i.e. both *AllPrecCurrWords*, *PrecHCurrLines* contain the current line). The results of *AllPrecLines* is 26.52 on test which is the second best BLEU score in Table 5, which confirms that our assumption is correct.

Looking at the additional results of *AllPrecLines* with different λ(s) in Table 7, we observed that larger λ weight improved the BLEU score on both dev and test sets. The best BLEU scores are obtained using *AllPrecWords* CSLM. The only difference between *AllPrecLines* and *AllPrecWords* is that the second one is weighted sum of words' embeddings, while the first one is the weighted sum of lines' embeddings.

System	λ	Dev	Test
SMT baseline	-	27.35	25.72
CSLM Baseline	-	28.04	25.67
CurrLine	-	28.29	26.09
PrecLine	-	28.67	26.33
AllPrecLines	0.85	28.06	25.52
AllPrecLines	0.95	28.59	26.42
AllPrecLines	0.98	**28.63**	**26.52**
AllPrecWords	0.75	28.37	26.36
AllPrecWords	0.85	28.74	26.49
AllPrecWords	0.95	**28.77**	**26.82**
AllPrecCurrWords	0.75	**28.52**	**25.86**
AllPrecCurrWords	0.85	28.23	25.59
AllPrecCurrWords	0.95	28.21	25.64

Table 7: BLEU *scores of re-scoring n-best list using AllPrecLines, AllPrecWords and AllPrecCurrWords auxiliary features with various weights. Auxiliary layer is a sequence of two tanh 320x320.*

It means that *AllPrecWords* auxiliary feature includes better and consistent context information. One possible reason for this is that for *AllPrecLines* auxiliary feature vector, each line has a different length, and hence the *weight* on each line controls the contribution of a variable number of words. This clearly is less stable than using the weighted sum of individual words embeddings and hence the auxiliary feature vector will be independent of individual lines lengths. In Table 7, we noticed the same relation between λ and the BLEU scores as we discussed for *AllPrecWords* auxiliary feature.
Looking at the results of *AllPrecCurrWords* auxiliary feature in Table 7, we observed that the results also are inconsistent on test, λ=0.75 gives better scores than λ=0.85, but also, λ=0.95 gives better scores than λ=0.85. We concluded that including word embeddings of both current line and preceding lines in the same auxiliary feature gives inconsistent results. For the results of *PrecHCurrLines* in Table 8, generally, we observed that including more preceding lines does not give better scores on test (we used maximum 50 preceding lines in these experiments), even with H=50, the scores are not better than just one preceding line **PrecLine**. We concluded that the reason is that this auxiliary feature includes the current line embeddings which cause inconsistent results on dev and almost no improvement on test.

System	H	Dev	Test
SMT baseline	-	27.35	25.72
CSLM Baseline	-	28.04	25.67
CurrLine	-	28.29	26.09
PrecLine	-	28.67	**26.33**
PrecHCurrLines	10	28.70	26.21
PrecHCurrLines	30	28.28	**26.26**
PrecHCurrLines	50	**28.92**	**26.26**

Table 8: BLEU *scores using PrecHCurrLines auxiliary feature with number of preceding lines H and* $\lambda = 0.95$. *Auxiliary layer is a sequence of two tanh 320x320.*

7. Conclusions

In this paper we introduced a novel method to improve the continuous space language model using auxiliary features. We used different features which some of them are motivated by the important features in machine translation quality estimation literature. The suggested auxiliary features include text genre, line length and various types of context vector representations.

We reported perplexity improvement around 7.5% on dev and test using the English Penn Treebank dataset. We also reported an improvement on a translation task up to 1.42 BLEU on dev and 1.1 on test by re-scoring n-best list of a strong baseline phrase-based SMT system. Also, the results show that the weighted sum of the word embeddings is more stable and outperforms the line level weighted sum of embeddings. These results need to be validated on other tasks with different language pairs, genres and data sets.

In future work, we would like to try using combined features and explore syntactic features. Also we would like to experiment with additional features like source language features and study their impact on the CSLM performance.

8. Acknowledgements

This research was partially financed by DARPA under the BOLT contract.
We would like to thank the reviewers of this paper for their helpful comments.

9. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944966

[2] H. Schwenk, "Efficient training of large neural networks for language modeling," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 4. IEEE, 2004, pp. 3059–3064.

[3] H. Schwenk, H. Schwenk, and J.-l. Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," *IN INTERNATIONAL CON-*

FERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, pp. 765–768, 2002.

[4] H. Schwenk, "Continuous Space Language Models," vol. 21, no. 3, pp. 492–518, 2007.

[5] J. Park, X. Liu, M. J. Gales, and P. C. Woodland, "Improved neural network based language modelling and adaptation." in *INTERSPEECH*, 2010, pp. 1041–1044.

[6] L. Lamel, J.-L. Gauvain, V. B. Le, I. Oparin, and S. Meng, "Improved models for mandarin speech-to-text transcription," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4660–4663.

[7] H. Schwenk, D. Déchelotte, and J.-L. Gauvain, "Continuous space language models for statistical machine translation," in *Proceedings of the COLING/ACL Conference*. Morristown, NJ, USA: Association for Computational Linguistics, 2006, pp. 723–730.

[8] H. Schwenk, "Investigations on large- scale lightly-supervised training for statistical machine translation," in *IWSLT*, 2008, pp. 182–189.

[9] ——, "Continuous space language models for statistical machine translation," in *The Prague Bulletin of Mathematical Linguistics, (93):137–146.*, 2010.

[10] H. S. Le, I. Oparin, A. Messaoudi, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Large vocabulary soul neural network language models." in *INTERSPEECH*, 2011, pp. 1469–1472.

[11] H. Schwenk, "Continuous space translation models for phrase-based statistical machine translation," in *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, M. Kay and C. Boitet, Eds. Indian Institute of Technology Bombay, 2012, pp. 1071–1080. [Online]. Available: http://aclweb.org/anthology/C/C12/C12-2104.pdf

[12] R. Kuhn and R. De Mori, "A cache-based natural language model for speech recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 6, pp. 570–583, 1990.

[13] J. R. Bellegarda, "Exploiting latent semantic information in statistical language modeling," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1279–1296, 2000.

[14] N. Coccaro and D. Jurafsky, "Towards better integration of semantic predictors in statistical language modeling." in *ICSLP*. Citeseer, 1998.

[15] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2014, pp. 238–247.

[16] R. M. Iyer and M. Ostendorf, "Modeling long distance dependence in language: Topic mixtures versus dynamic cache models," *Speech and Audio Processing, IEEE Transactions on*, vol. 7, no. 1, pp. 30–39, 1999.

[17] S. Khudanpur and J. Wu, "Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling," *Computer Speech & Language*, vol. 14, no. 4, pp. 355–372, 2000.

[18] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*. IEEE, 2012, pp. 234–239. [Online]. Available: http://dx.doi.org/10.1109/SLT.2012.6424228

[19] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[20] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Comput. Linguist.*, vol. 29, pp. 19–51, March 2003.

[21] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Meeting of the Association for Computational Linguistics*, 2007, pp. 177–180.

[22] A. Stolcke, "Srilm - an extensible language modeling toolkit," *In Proceesings of the 7th International Conference on Spoken Language Processing (ICSLP 2002*, pp. 901–904, 2002.

[23] K. Heafield, "KenLM: faster and smaller language model queries," in *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland, United Kingdom, July 2011, pp. 187–197. [Online]. Available: http://kheafield.com/professional/avenue/kenlm.pdf

[24] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 311–318.

[25] A. Axelrod, X. He, and J. Gao, "Domain adaptation via pseudo in-domain data selection," in *Proceedings*

of the 2011 Conference on Empirical Methods in Natural Language Processing. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 355–362.

[26] A. Rousseau, "Xenc: An open-source tool for data selection in natural language processing," *The Prague Bulletin of Mathematical Linguistics*, vol. 100, pp. 73–82, 2013.

[27] R. Sennrich, "Perplexity minimization for translation model domain adaptation in statistical machine translation," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon, France: Association for Computational Linguistics, April 2012, pp. 539–549. [Online]. Available: http://www.aclweb.org/anthology/E12-1055

[28] R. C. Moore and W. Lewis, "Intelligent selection of language model training data," in *Proceedings of the ACL 2010 Conference Short Papers*, ser. ACLShort '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 220–224. [Online]. Available: http://dl.acm.org/citation.cfm?id=1858842.1858883