Integrating Encyclopedic information into Neural Network Language Models

Firstname Lastname

Insitute for Anthropomatics
Karlsruhe Institute of Technology, Germany
firstname.lastname@iwslt.org

Abstract

Neural models have recently shown big improvements in the performance of phrase-based machine translation. Recurrent language models with different word factors, in particular, were a great success due to their ability to incorporate additional knowledge into the model. In this work, we want to integrate global semantic information extracted from large independent knowledge bases into neural network language models. We propose two approaches for doing this: word class extraction from Wikipedia and sentence level topic modeling. The new resulting models exhibit great potential in counteracting data sparsity problems with additional independent knowledge. This approach of integrating global context information is not restricted to language modeling but can also be easily applied to any model that profits from context or further data resources, e.g. neural machine translation. Using this model has improved rescoring quality of a state-of-the-art phrase-based translation system by ... BLEU points. We performed experiments on two language pairs.

1. Introduction

Recurrent neural network language models have recently shown great improvement in statistical machine translation, both during decoding and rescoring. The use of continuous word representations has achieved better generalizations of the data which effectively lowered data sparseness problems. Furthermore, the recurrent connections are able to model long range dependencies. Yet, most of these models strictly depend on monolingual and parallel data, which is sometimes not available in huge amounts or only for different domains, especially for low-resource languages. This has motivated neural network language models that take multiple parallel streams of data as input instead of just the single form of surface words. These so called factors can be used to add additional information, e.g. POS or automatic word clusters, which helps mainly with morphologically rich languages (e.g. Romanian, German). However, so far the additional feature only pertained to syntactic or local context knowledge around the current word. Especially for languages with low resources, it is essential to also facilitate the use of other knowledge sources, e.g. encyclopedia knowledge. It is a useful source especially for learning general concepts, even more after the emergence of the Internet has led to an explosion of textual data. These data sources give insights into a variety of human endeavors waiting to be computationally analyzed. In this paper, we study the integration of large independent knowledge bases in the form of encyclopedia, e.g. Wikipedia, into RNN-based language models and propose two solutions. First, we use the factored model to integrate extracted Wikipedia categories as one of the factors. In order to understand large unstructured datasets great achievements have been attained in latent concept learning in the area of text mining. Techniques include categorization of documents using latent semantic analysis and probabilistic topic modeling. In our case, we used techniques like tfidf, LSA and LDA to compute a real-valued topic vector for each sentence that is fed into the network as additional input. General word class and global topic features are used in combination with local contexts provided by recurrent neural models to improve lexical selection.

2. Related Work

Language models are a critical component of many application systems, e.g. ASR, MT and OCR. However, language models have always faced the problem of data sparseness. Factored Language Models [1] introduced the use of a bundle of factors associated with a word which outperformed previous n-gram models without expanding the training data. For factors morphs, stems, POS and word class obtained using the SRILM's n-gram-class tool were used. [2] replaced the single feature stream of surface words with multiple factors and integrated it into phrase-based statistical machine translation systems by breaking down the translation model into several steps that pertain to the translation of single factors which are all taken into account when the target word is generated. After recurrent neural network models became a success in language modeling [3], a factored input layer was employed in a model by [4] which uses a structured output layer based on word classes that was able to handle vocabulary of arbitrary size. Motivated by multi-task learning in NLP, [5] proposed a multi-factor recurrent neural network language model which jointly predicts different output factors by mapping the output of the LSTM-layer to as many softmax layers as there are output factors, thus creating multiple distributions at the output layer. In the rescoring of an n-best list, this model can be included as either one additional feature or several features depending on whether the output is treated as a joint probability or individual probabilities. However, a dedicated continuous space vector offers more flexibility and possibility to store additional side information, especially for complex higher-level concepts, e.g. topic distributions. In [6] a first approach to use an additional vector for higher-level concepts was proposed. In their work, a vector instead of a single factor is associated with each word. However, this vector depended only on the earlier local context of the word, neglecting the influence of the future context on a word's meaning. In fact, often the meaning of a word cannot be just derived from its preceding words but by content words in the entire sentence or surrounding sentences. Topic models play a great role in text mining because they summarize concepts in large amounts of documents into fewer topics by capturing word co-occurence information. Essentially, topic models can be divided into vector space models, e.g. LSA [7], and probability models, e.g. LDA [8]. The successful usage of Wikipedia to devise methods for computing semantic relatedness of documents was reported in [9] and [10]. Graphic models were employed in [11] to link named entities in text documents by using information extracted from Wikipedia. In [12],[13] vector space models are employed to resolve word disambiguations based on entities derived from Wikipedia. Similar approaches were applied on other knowledge sources, such as WordNet [14].

3. Integration of Side Information

We studied two approaches to integrate encyclopedic information into neural language models. First, we extracted for each word a corresponding label from Wikipedia. For this, we exploited the existing hierarchical page structure of Wikipedia and use the category label of a page as a factor input into the previously described factored neural language model. Second, for each sentence a ranking of similar encyclopedia documents is calculated using vector space models, such as tfidf or LSA, and topic models, such as LDA. Based on the same underlying models a feature vector of the most similar documents to the current sentence is computed which is fed into a neural language model as additional input. The second approach is not limited to Wikipedia, but can be applied to any encyclopedia. To show the efficiency of this method independent from the underlying knowledge source we have crawled a Chinese lexicon from zdic.net [1] for which we will present the results later on.

Neither was possible to determine only from their respective phrases in isolation

3.1. Word Level Information

The motivation behind using general word categories from Wikipedia to associate with words is to strengthen the relatedness of words in sentences that are correct translations but have a low score due to the translation model.

$$\text{“A journalist writes articles called columns”} \qquad (1)$$

Both of the following hypotheses, as shown in Example 1, are possible translation candidates in decoding:

Example 1: Two hypothese tagged with Wikipedia categories

1. $\underbrace{\text{一}}_{\text{CD}}\ \underbrace{\text{位}}_{\text{M}}\ \underbrace{\text{记者}}_{\text{新闻 (=news)}}\ \underbrace{\text{写的}}_{\text{VV}}\underbrace{\text{文章}}_{\text{作品}}\ \underbrace{\text{被}}_{\text{SB}}\ \underbrace{\text{称为}}_{\text{VV}}\ \underbrace{\text{柱子}}_{\text{新闻(=news)}}$
2. $\underbrace{\text{一}}_{\text{CD}}\ \underbrace{\text{位}}_{\text{M}}\ \underbrace{\text{记者}}_{\text{新闻 (=news)}}\ \underbrace{\text{写的}}_{\text{VV}}\underbrace{\text{文章}}_{\text{作品}}\ \underbrace{\text{被}}_{\text{SB}}\ \underbrace{\text{称为}}_{\text{VV}}\ \underbrace{\text{专栏}}_{\text{建筑(=archit.)}}$

Even though the second translation is obviously the correct one, because column in the context of journalist and article refers to a newspaper area, the first translation is more likely according to the translation model since generally a column describes more often a pillar than an article. Since the main contextual reference of column, which is article, is several words apart, translation tasks where the data does not originate from news or similar domains, or in case the data is so constrained in size that it has never seen the words "column" and "journalist" in the same context, the language model will fail to choose the correct translation. Out approach solves this problem by tagging the nouns in the sentence with according Wikipedia categories. By doing so, the bond between column and journalist is reinforced by the their common factor which contributes to a higher score for the correct translation.

3.1.1. Input representation

A Wikipedia page, labeled with a title, is either an article or a category page which encompasses one or multiple pages. We define the search space as the set of all page titles. Given a word, we search for the page with the same word as title and retrieve its category which is found at the bottom of the page. In implementation, we downloaded the freely available Wikipedia dump to create a mapping of pages and their category offline. Using a threshold for the minimum number of elements in a category, we pick recursively the parent of a category in case it does not meet the requirement, thus reducing the total number of associated articles which performed better in practice. In case no category is found at all, we resort to the default part-of-speech tag of a word. One typical characteristic of Wikipedia is that most of its articles only refer to a small group lexical categories. For example, there are lots more articles about objects (nouns) than about activities (verbs). Fortunately, topic words are usually nouns, so it suffices to tag nouns for an improvement in model quality as suggested in Example 1.

[1] http://www.zdic.net/

3.1.2. *Factored language model*

For training we used the factored language model proposed in [5], which takes in one or multiple factors at the input layer and offers the option to factorize output layer. After concatenating the embeddings of the input factors into a single word embedding, this vector is send through one or multiple LSTM-based layers before projected onto the factored output probabilities. The instance of the model we used for our purposes, takes two factors, the surface form and the Wikipedia word category. These are mapped to an embedding vector of size 100, which is the same size as the first LSTM-layer. For the second LSTM-layer we used 200 nodes. We only used one factor for the output, the surface form of the next word.

3.2. Sentence Level Information

To create a ranking of similar documents and their representation to a given sentence we employ vector space models, such as the tf-idf and latent semantic indexing model, as well as topic models, such as the latent dirichlet allocation. After representing the sentences and cleaned encyclopedia articles as bag-of-words with the underlying vocabulary from the training, we can find for each query, which are the sentences, the most similar documents from D, which are the articles, based on the underlying models.

3.2.1. *TF-IDF*

Tf-idf [15] reflects how important a word is to a document or a whole collection of documents, it measures the co-occurrence. The advantage of tf-idf is that it grades down terms that appear in multiple documents which makes them less relevant. The tf-idf is computed by multiplying a local component (term frequency or tf) with a global component (inverse document frequency or idf). Term frequency $tf(t, d)$ is defined as the number of times that term t appears in document d:

$$tf(t, d) = \frac{count_d(t)}{|d|} \tag{2}$$

Inverse document frequency $idf(t, D)$ measures how much information a term t provides regarding documents D, that is, whether it is common or rare in D:

$$idf(t, D) = \log_2 \frac{|D|}{|\{d \in D : t \in d\}|} \tag{3}$$

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{4}$$

Calculating tf-idf for each word in a vector gives an tf-idf vector. We used the python library gensim, which normalizes the vector at the end.

To compute a ranking of similar documents for a given sentence query, we compute the query's tf-idf vector q and compare it to each of the documents' tf-idf vectors $d_1, d_2, \ldots, d_{|D|}$ using the cosine to calculate the angle θ between vector pairs:

$$\cos\theta_i = \frac{q \cdot d_i}{|q| \cdot |d_i|} \tag{5}$$

The closer this value is to 1 the better the match. For a predefined number of top documents n we determine the averaged tf-idf value of the best n documents to a query, which we denote with top-doc-tfidf(q, D, n). This vector constitutes the sentence level semantic information which is fed into a RNNLM together with the surface data stream. Shortcomings of this model include the inability to reduce the description length of the document, since words are only replaced with values. Also, it does not tell much about the statistical structure within and between documents.

3.2.2. *Latent Semantic Indexing*

Another model we used is latent semantic analysis (LSA) or latent semantic indexing (LSI) [7]. LSI is a method for discovering hidden concepts in document data by using single value decomposition (SVD) on the set of documents D. To calculate the similarity between a query q and any document d according equation 5, we must first express q and d in a unified way corresponding to these concepts, so that each of their vector elements gives the degree of participation of query or document in the corresponding concept.

LSI uses a term-document matrix or occurrence matrix A, whose rows correspond to terms and columns correspond to documents. The entry $A_{i,j}$ equates to $tfidf(t_i, d_j, D)$. SVD decomposes A into U, Σ and V so that $A = U\Sigma V^T$ with U and V being orthonormal matrices and Σ a diagonal matrix with the single values on the diagonal. LSI uses the decomposition to find a low-rank approximation, that is, a matrix A_k of a predefined lower rank k closest in similarity to the original matrix A. This is done by deleting all but the k biggest single values and the according rows and columns in V and S. The degree of similarity is determined by the Frobenius Distance $\|A - A_k\|_F$, which is minimized by LSI. In our case, k is the number of hidden concepts we want to learn. The number of dimensions k is an empirical question. Essentially, k is much smaller than the original space dimension, which is usually the number of total documents. Previous papers show that for Wikipedia dumps containing between hundreds of thousands and a few million articles k should be chosen between 200 to 500 [16]. For our use we chose k as 300.

3.2.3. *Latent Dirichlet Allocation*

Finally, we used Latent Dirichlet Allocation (LDA) [8], a generative probabilistic model to automatically discover topics from a data collection. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. The model is a three-level hierarchical Bayesian model with the first level being the corpus-level, the second being the document-level and the third-level being the word-level. Setting the number of topics to be learned to K, the following assumptions are made to generate a document **w** of length N from a corpus D:

1. Choose the document length $N \sim Poisson(\xi)$
2. Choose document's distribution of topics $\theta \sim Dir(\alpha)$ with K dimensions
3. For each of the document word w_n:
   (a) Choose topic $z_n \sim Multinomial(\theta)$
   (b) Choose w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n. β is a $K \times V$ matrix with $\beta_{ij} = p(w^j = 1|z^i = 1)$

Learning is performed with Bayesian inference, e.g. by using collapsed Gibbs Sampling and expectation propagation. As for K, [17] discusses how to choose the number of topics. Given the computed model, we can predict the topic of a word or query using Bayes Theorem.

3.2.4. Extended language model

The simple recurrent neural network language model consists of an input layer, a hidden layer with recurrent connections that maintains a representations of the sentence history, and an output layer which produces the probability distribution over words. We propose a variation of the conventional recurrent language model by extending the basic model with an additional sentence based feature layer that is connected to the output layer. Since this real-valued feature vector stays the same for all words in the current sentence, it is replicated for each word before connected with the output layer. This way, the feature information is retained in the model while the same sentence is processed. For the m hidden layers we use LSTM-based layers. Given a sentence $\mathbf{w} = \{w_1, w_2, \ldots, w_n\}$, we denote the input representation of a word w_i with x_i, which is encoded using 1-of-N coding. The side information **f** computed for the sentence **w** with the above mentioned models is duplicated n times, yielding the features vectors $f_1, f_2, \ldots, f_n$. We denote the j-th hidden layer associated with the x_i with s_i^j and the output layer with y_i. Then for the i-th word w_i the hidden and output layers are computed as follows:

$$\begin{aligned} s_i^1 &= f_1(U_1 x_i + W_1 s_{i-1}^1) \\ s_i^j &= f_j(U_j s_i^{j-1} + W_j s_{i-1}^j), j \neq 1 \\ y_i &= g(V s_i^m + F f_i) \end{aligned} \tag{6}$$

where f_i represents activation functions, and g the softmax function. For training of the network, i.e. finding the weight matrices U, V, W, F, we use stochastic gradient descent according to the negative log-likelihood loss function. We also added an additional link from feature input to first hidden layer, which is shown in Figure 1.

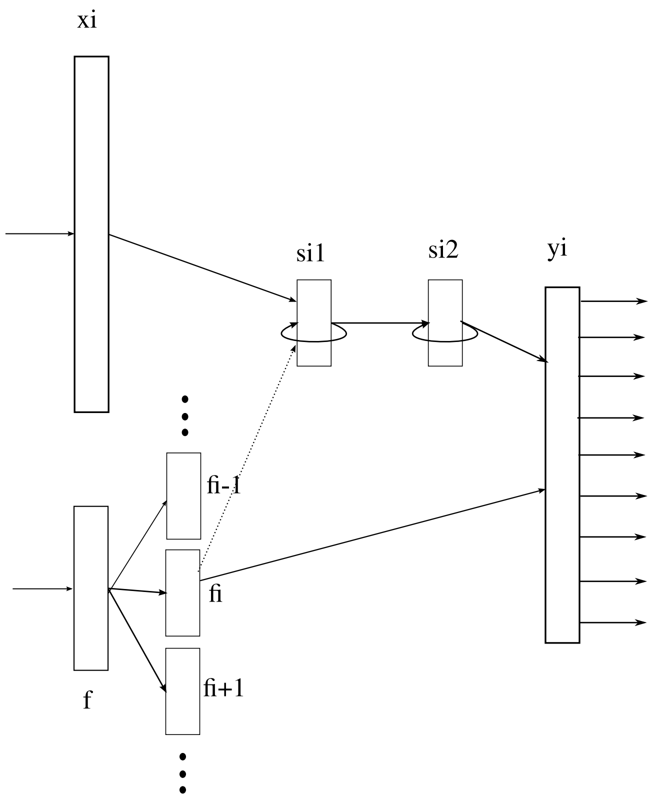

Figure 1: Extended recurrent language model with additional feature input **f** and two LSTM-based hidden layers. The dashed line from the feature input to the first hidden layer represents a later added connection

4. Experiments

We evaluated both the factored language model and the extended language model on English-Chinese and English-Romanian language pairs. For each language pair we created the n-best list using our phrase-based MT system and used the models as additional feature in rescoring.

4.1. System Description

The baseline system is an in-house implementation of the phrase-based approach and is used to generate n-best lists on all the available training data. The English-Chinese system is trained on the TED and UN corpus, optimized and rescored on the TED dev2010 and tested on the test2010 corpora. The English-Romanian system was trained on the corpora of WMT 2015 Shared Translation Task, optimized on the first half of news-dev 2016 and tested on the second half of news-dev 2016. In addition, for rescoring a subset of 2000 sentences of the SETimes corpus was used for further optimization. The English-Chinese baseline system uses two word-based language models, and 12 features in total to create an n-best list of size 3000. The English-Romanian baseline system uses two word-based language models, 2 cluster-based models using 50, 1000 or 1000 clusters, and a POS-based language model. In total 22-23 features were used to generate an n-best list of size 300. A full system description can be found in [5]. For decoding, both language pairs were optimized using minimum error rate training. The same is used for rescoring of English-Chinese, whereas English-Romanian used ListNet to rerank the n-best list. All RNN-based language models for both language pairs were trained on the target side of the parallel training data. The English-Chinese system uses a vocabulary of 10K, while the English-Romanian system uses a vocabulary of 5K words. For word level side information we used Wikipedia to extract the word categories as described in 3.1 as input into the factored language model. For sentence level side information, as described in 3.2, we additionally used a web-crawled Chinese lexicon from zdict for the English-Chinese system to show the model's performance independently from a specific encyclopedia.

4.2. English-Chinese

In the first experiment of English-Chinese we used the scores of the factored language model in addition to the baseline features, which is shown in Table 1. Since Chinese is a sparsely inflected language, it was sufficient to only tag nouns and names in this experiment. BLEU score improvements over the basic recurrent language model are displayed in brackets. When using both words and Wikipedia categories the model performs 0.66 BLEU score points better in testing than the basic recurrent model using only words as input. Combining this model with the scores of a factored model with words and POS as factors produces a further increase of 0.14 points, resulting in an overall improvement of 0.84 BLEU points over the basic recurrent model.

Table 1: English-Chinese Factored Language Models

Model	devdata	testdata
Baseline	14.69	17.05
+RNNLM	14.7	17.02
+Factored LM POS	14.77	16.97
+Factored LM Cat	14.89 (+0.12)	17.63 (+0.66)
+Factored LM Cat + POS	14.75 (-0.02)	17.81 (+0.84)

Table 2: Extended Language Models: Overview of different feature vectors

Model	devdata	testdata
Baseline+RNNLM	14.70	17.02
Baseline+TFIDF.tfidf	14.78 (+0.08)	17.68 (+0.63)
Baseline+LSA.tfidf	14.78 (+0.08)	17.31 (+0.29)
Baseline+LSA.lsa	14.83 (+0.13)	17.80 (+0.78)
Baseline+LDA.tfidf	14.79 (+0.09)	17.41 (+0.39)
Baseline+LDA.lda	14.79 (+0.09)	17.27 (+0.25)

In a second experiment, extended language models using different methods to compute the feature input vector are studied. Tfidf, LSA and LDA are used for similar document ranking as well as vector representation. It is worth mentioning that the method for ranking can be paired with a different choice for representation. The performance results of various combinations of pairings are presented in Table 2. Although all combinations result in better rescoring performance compared to the basic recurrent model, in most cases choosing the same method for both tasks achieve a noticeable higher boost. For example, tfidf creates an increase of 0.63 points on the basic recurrent model, LSA an increase of 0.78. The only exception to this rule is LDA. One reason for this could be suboptimal parameter picks for this generally more complex generative model. Despite the slightly better performance of LSA, tfidf is employed in the following experiments due to its simplicity and, thus, faster training and evaluation.

In order to test the efficiency of this model independently from the characteristics of the underlying encyclopedia source, a Chinese lexicon is crawled from zdict. It contrasts Wikipedia in the variety and length of definitions. In fact, zdict provides more but shorter compact explanations for all type of words, particularly verbs. Despite the difference in content, the use of zdict documents shows an increase of 0.56 BLEU points, similar to that of Wikipedia as shown in Table 3.

An overview of all the models and their perplexities is given in Table 4. All models exhibit an evident reduction in perplexity compared to the basic recurrent model, which goes along with the previous rescoring results.

As indicated by Figure 1, an additional connection was established between the feature layer and the first hidden layer in another experiment. Using tfidf the model gained a small increase of +0.16 BLEU on the model with single con-

Table 3: Extended Language Models: Comparison between different encyclopedia sources

Model	devdata	testdata
Baseline	14.69	17.05
+RNNLM	14.7	17.02
+Wiki	14.78 (+0.08)	17.68 (+0.66)
+Dict	14.91 (+0.21)	17.58 (+0.56)

Table 4: Model Perplexities

Model	PPL
RNNLM	128.17
Factored LM POS	110.86
Factored LM Cat	109.73
Factored LM Cat+POS	110.38
WIKI	120.60
ZDICT	118.85

nection and a total of +0.79 BLEU on the simple recurrent model, as shown in Table 5.

4.3. English-Romanian

Unlike Chinese Romanian is a morphologically rich language which is why investigated tagging only nouns and all word types in parallel for the first task. In previous work [5], the factored language model for English-Romanian integrated four factors: the word's surface form, POS, and word clusters with 100 and 1000 class respectively. Our experiment builds on top of [5], which used for all systems a vocabulary of 5K, and the same denotations are used to illustrate the following results. In the first experiment, only the factored language models are evaluated without the baseline system. The system from the previous work with all four factors as input and prediction reached a BLEU score of 28.54, as shown in Table 6. This model will serve as reference for our models. After adding our extracted Wikipedia word categories to the factors, we achieved improvement of 0.17 and 0.30 BLEU points respectively.

Table 7 shows the results of model features used in addition to the baseline system in three configurations, which are evaluated using just the joint probability. Adding word tags for all word types achieved an improvement of about 0.1 BLEU points in two configurations over the reference model.

Table 5: Extended Language Models: Connecting feature input with two layers

Model	devdata	testdata
Baseline+RNNLM	14.70	17.02
Baseline+TFIDF	14.78 (+0.08)	17.68 (+0.63)
Baseline+2Con TFIDF	14.74 (+0.04)	17.81 (+0.79)

Table 6: Factored Language Models: Single Scores

Input	Prediction	Single
Word	Word	27.88
All factors	All factors	28.54
+Cat (Nouns)	+Cat (Nouns)	28.71 (+0.17)
+Cat (All Words)	+Cat (All)	28.84 (+0.30)

Table 7: Factored Language Models: End Scores

Model	Conf1	Conf2	Conf3
Baseline	29.86	30.00	29.75
+All factors	29.94	30.01	30.01
+All factors + Nouns	29.94 (+0.00)	30.31 (+0.30)	29.99 (-0.02)
+All factors + All Words	29.95 (+0.01)	30.13 (+0.12)	30.14 (+0.13)

5. Conclusion

In this work, we proposed two approaches to integrate higher level information from encyclopediatic sources into neural network language models. The first takes advantage of Wikipedia, which is one of the fastest growing encyclopedias. The second approach takes a continuous space vectors as additional input which enables incorporation of more complex side information on the sentence level. In addition, we also introduce three different methods how to prepare the additional data and represent it in the correct form. By using of more complex information from large independent sources, we have improved the translation system on two different language pairs. This work plays a key role for low-ressource languages, that often lack sufficient training data.

6. Acknowledgements

This work was funded by the CLICS Exchange Program.

7. References

[1] J. A. Bilmes and K. Kirchhoff, "Factored language models and generalized parallel backoff," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*. Association for Computational Linguistics, 2003, pp. 4–6.

[2] P. Koehn and H. Hoang, "Factored translation models." in *EMNLP-CoNLL*, 2007, pp. 868–876.

[3] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[4] Y. Wu, X. Lu, H. Yamamoto, S. Matsuda, C. Hori, and

H. Kashioka, “Factored language model based on recurrent neural network,” 2012.

[5] J. Niehues, T.-L. Ha, E. Cho, and A. Waibel, “Using factored word representation in neural network language models.”

[6] T. Mikolov and G. Zweig, “Context dependent recurrent neural network language model.” in *SLT*, 2012, pp. 234–239.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[9] E. Gabrilovich and S. Markovitch, “Computing semantic relatedness using wikipedia-based explicit semantic analysis.” in *IJcAI*, vol. 7, 2007, pp. 1606–1611.

[10] M. Strube and S. P. Ponzetto, “Wikirelate! computing semantic relatedness using wikipedia,” in *AAAI*, vol. 6, 2006, pp. 1419–1424.

[11] X. Han and L. Sun, “An entity-topic model for entity linking,” in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 105–115.

[12] S. Cucerzan, “Large-scale named entity disambiguation based on wikipedia data.” in *EMNLP-CoNLL*, vol. 7, 2007, pp. 708–716.

[13] R. C. Bunescu and M. Pasca, “Using encyclopedic knowledge for named entity disambiguation.” in *EACL*, vol. 6, 2006, pp. 9–16.

[14] M. A. Hearst, “Automatic acquisition of hyponyms from large text corpora,” in *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1992, pp. 539–545.

[15] G. Salton and M. J. McGill, “Introduction to modern information retrieval,” 1986.

[16] R. B. Bradford, “An empirical study of required dimensionality for large-scale latent semantic indexing applications,” in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 153–162.

[17] M. Hoffman, F. R. Bach, and D. M. Blei, “Online learning for latent dirichlet allocation,” in *advances in neural information processing systems*, 2010, pp. 856–864.