Integrating Semantic 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 low-resource language modeling in 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 scarcity 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, 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 use of factors or additional information has been limited in neural network models. Also, in those cases the extra 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 factors, 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. Using word classes and semantic features help both sparsely inflected languages(e.g. English, Chinese) [1] as well as low-resource languages. In the model we use LSTMs to take into account both independent side information and local context information for the model prediction.

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. One disadvantage of factored input is that additional factors must match the surface words in space. As a consequence, surface words that uses 1-of-n encoding cannot have factors with continuous space representations. However, this is often necessary to model more complex structures, e.g. topic distributions. In [6], a topic-conditioned RNNLM is proposed which takes a real-valued input vector as an additional input in association with each word. This vector is used to convey local context information based on previous sentences using latent dirichlet allocation. 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. However, the model cannot take side information associated with a sentence that contains the current word, which is what we studied in the second part.

3. Integration of Side Information

We studied two approaches to integrate encyclopediatic information into neural language models. First, we extracted for each word a corresponding label from Wikipedia. For this, we exploite the existing hierarchical page structure of Wikipedia and use the category name 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 encyclopedia for which we will present the results later on.

3.1. Word Level Information

A Wikipedia page has an title and 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 case of multiple categories we pick the first one, which is usually the most general one. Figure .. gives an example of this. In implementation, we utilized the downloadable Wikipedia dump to create a offline mapping of pages and their category. Note that we skip categories that only have one element to reduce the overall number of associated categories. In case no category is found at all, we resort to the part-of-speech tag, a word's default label. 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). For this reason, we can significantly minimize the number of category lookups by restricting those to the words of the most common POS groups without taking big losses in model quality. We will show results for looking up every word versus just nouns.

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 the most similar documents, which are the articles, for each query, which are the sentences, based on the underlying models.

3.2.1. TF-IDF

Tf-idf [7] reflects how important a word is to a document or a whole collection of documents, its 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:

$$tf(t, d) = \frac{count_d(t)}{|d|} \tag{1}$$

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:

$$idf(t, D) = \log_2 \frac{|D|}{|\{d \in D : t \in d\}|} \tag{2}$$

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{3}$$

Calculating tf-idf for each word in a vector gives an tf-idf vector. In gensim, this vector is normalized 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:

$$\cos\theta_i = \frac{q \cdot d_i}{|q| \cdot |d_i|} \tag{4}$$

The closer this value is to 1 the better the match. For a predefined number of top documents n we find the best n documents to a query and compute the average of their tfidf which we denote with top-doc-tfidf(q, D, n). This vector constitutes the sentence level semantic information which fed into a RNNLM together with the surface data stream. Shortcomings of this model are that it does not 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) [8]. LSI is a method for discovering hidden concepts in document data. Those hidden concepts are found using Single Value Decomposition(SVD) based on our documents. The derived LSI features are a linear combination of the original tf-idf features. To calculate the similarity between a query q to any document d using equation 4, we express both 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 $o_{i,j}$ is the tfidf-value of term i with respect to document j, that is $tfidf(t_i, d_j, D)$. SVD decomposing A into U, Σ, V so that $A = U\Sigma V^T$. U and V are orthonormal matrices and Σ is a diagonal matrix whose diagonal elements are called single values. LSI uses the decomposition to find a low-rank approximation, that is, a matrix A_k of a predefined lower rank k closes 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. However, crucially k is much smaller than the original space dimension, which is probably about the number of total documents. Previous papers show that for Wikipedia data containing between hundreds of thousands and a few million articles k should be chosen between 200 to 500 [9]. We chose 300 for k.

3.2.3. Latent Dirichlet Allocation

Finally, we used Latent Dirichlet Allocation (LDA) [10], 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. Having the number of topics to be learned set to K, following assumptions are made to generate a document **w** of length N in 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 $\beta_{ij} = p(w^j = 1|z^i = 1)$

The model is learning with Bayesian inference using collapsed Gibbs Sampling and expectation propagation. As for K, [11] 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. We chose 100 for K.

4. Experiments

4.1. System Description

4.2. English-Chinese

4.3. English-Romanian

5. Conclusion

6. Acknowledgements

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.