Neural Networks with Attention for Word Sense Induction

Oleg Struyanskiy[1] and Nikolay Arefyev[1,2]

[1] Moscow State University, Moscow, Russia
[2] Samsung Moscow Research Center, Moscow, Russia
oleg.fox@gmail.com, ezhick179@gmail.com

Abstract
Attentional neural networks have achieved remarkable results for a number of tasks in the past few years. The fascinating success of neural networks with attention mechanism in natural language processing, especially in machine translation, suggests that these models can capture the meaning of ambiguous words considering their context. In this paper we introduce a new method for constructing vectors of ambiguous words occurrences for word sense induction based on the recently introduced model Transformer that achieved state of the art results for machine translation. Similar to the CBOW model for constructing word embeddings we train the Transformer to predict a word from it's context and use its trained parameters for word sense induction. On some datasets the proposed method outperforms the simple but hard-to-beat baseline, which was among the best three methods in the recent shared task on word sense induction for the Russian language *RUSSE-WSI* 2018. On one of the datasets our method beats the top result from the competition. Furthermore, we explore how different methods of weighing word embeddings affect the performance in word sense induction. Together with weighted sums of word2vec vectors, we explore the performance of vectors from Transformer's hidden layers and introduce a combined approach that improves previous results.

Keywords: word sense induction, attention, Transformer, neural network.

1 Introduction

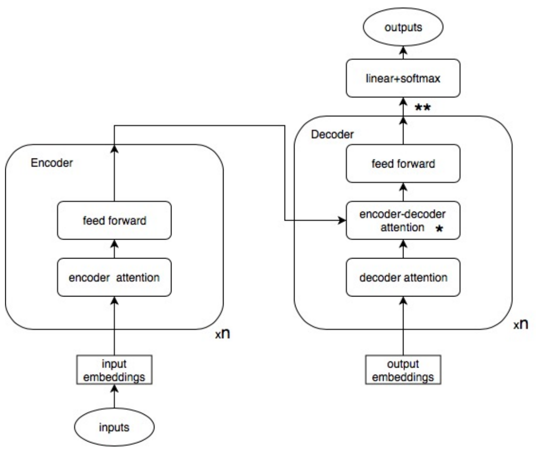
Word sense induction is a problem of clustering contexts, i.e. short texts containing a polysemous word into clusters depending on the sense of the word. The recent competition on word sense induction for the Russian language *RUSSE-WSI* [2] has shown that existing approaches work for homonyms but fail for complex polysemous words. We follow the unsupervised approach to word sense induction starting with building vector representations of contexts and then running a clustering algorithm to distinguish contexts that contain the ambiguous word in different senses. One of the ways to construct a vector representation of a context is to compute a weighted sum of word embeddings for the context. The main question in this approach is how to determine the weights. One simple yet effective method is to use weights based on word frequencies. For example the model proposed in [4] which we use as baseline employs tf-idf weights. But models based on word frequencies do not take into account complex linguistic relationships and can assign large weights to relatively unimportant words. In this paper we propose a more sophisticated approach for determining the weights using one of the newest neural network models.

2 Model

2.1 Transformer

Transformer [3] is a recently proposed sequence transduction model which shows state of the art results for several tasks including machine translation, text summarization etc. Similar to previous best method for machine translation (sequence to sequence models with attention) Transformer consists of encoder and decoder. The main novelty of Transformer is that it does not use recurrent neural networks, its encoder and decoder are based on a combination of feed forward networks and attention mechanism which is responsible for packing sequences of variable length into vectors of fixed size. Attention can be defined as a mechanism that computes weights of all the elements in the input sequence. It is assumed that elements are represented as real value vectors. The weights represent the importance of elements, and are typically used to build a weighted sum of the element vectors. There are different types of attention mechanism, depending on how the element weights are computed; in Transformer dot product attention is used [3]. The connection between the encoder and the decoder is also based on attention, therefore the model uses 3 different types of attention: self attention in encoder, masked self attention in decoder and the encoder-decoder attention. The model has *n* identical blocks where *n* is a hyperparameter (Fig 1). Every attention block in the model is multihead, which means that *h* independent attention layers operate in parallel and the results are combined using concatenation and a linear transformation. The number of heads is also a hyperparameter and is the same in all 3 attention blocks of the model.

Fig. 1. Transformer architecture



It is impossible to train a model to predict sense labels of polysemous words due to the lack of enough sense labeled texts. One way to solve this problem is to train a model to perform a side task and then use trained parameters for other purposes. This technic was used in the widely known word embedding tool word2vec. [5] The CBOW model is trained to predict words by their context and the weights of the model are then used as word representations. We propose a similar solution training the Transformer to predict words by their contexts with the aim to use attention weights for word sense induction afterwards. So, the input of the model is a text fragment with all occurrences of a specific word replaced by a special token *CENTERWORD*, and the desired output is the word which occurrences were replaced.

2.2 Context vectors and word sense induction

Weights from the encoder-decoder attention mechanism (* on figure 1), taken from the timestep when the model is generating the prediction of the center word, directly point out, how much each word in context contributes to the prediction. These weights were extracted during the processing of target datasets and used for the weighting of word embeddings when building context vectors. We hope that context words that the model has learned to attend to for the missing word prediction will also be useful for discriminating between that word's senses. We considered different hyperparameter values for Transformer, varying the number of model layers and the number of attention heads. From the simple variant of the model with just one layer and one head, attention weights were extracted without any aggregation, as there is only one vector of weights per input sequence. For more complex models with several layers and attention heads the weights from the first layer were extracted, and then for each word in the input sequence a maximum over weights from different attention heads were calculated. The idea behind such aggregation was that different heads would attend to different parts of the context, thus the maximum over all heads would determine how much the model attends to a particular word in general.

We consider two methods of determining word weights, one relying solely on attention weights, and another one using a combination of tf-idf and attention weights. The weights are raised to the specific power so a context vector is formed as follows:

$$v_{context} = \sum_{w \in context} {t_w}^{tf\_pow} \cdot {a_w}^{att\_pow} \cdot v_w$$

where t_w and a_w are tf-idf and attention weights of the word w respectively, tf_pow and att_pow are hyperparameters, v_w is the word2vec embedding for a context word w.

Also we explore the word sense induction performance of Transformer output embedding for an ambiguous word. Specifically, we take the vectors from the output of the decoder (** on fig 1), taken at the timestep when the model predicts the center word. We hypothesize that this vector is a good representation for the sense of the predicted word because it summarizes the whole context. Finally, our best performing method uses the concatenation of Transformer output embedding for an ambiguous word and the weighted sum for the words of its context.

The actual task of word sense induction is performed by clustering of the context vectors using agglomerative clustering algorithm [6]. The exact number of clusters was selected on the train sets for every dataset individually.

3 Experiments

For evaluation of our model we used the datasets and the evaluation scripts of the word sense induction for the Russian language shared task *RUSSE-WSI* [2]. The task provides three datasets (bts-rnc, active-dict and wiki-wiki) based on different corpora and sense inventories, each was split into train and test parts. We used the official evaluation script of the task, which calculates adjusted Rand index (ARI); it equals 0 for a random clustering and 1 for the gold standard clustering.

The train set for transformer was built from 25% of librusec [1] text collection. We extracted 12 M contexts (4.5 GB) containing any of 341 ambiguous words from the datasets of the shared task. All occurrences of the ambiguous words were replaced with a special token *CENTERWORD.* The average length of contexts in the train set is 20 words, the same as in *RUSSE-WSI* datasets. The preprocessing for all data included converting to lowercase and inserting separating spaces between words and punctuation signs**.** The size of the dataset was chosen to keep the training time reasonable. For this same reason we only considered the 341 polysemous words from *RUSSE-WSI* datasets as possible center words. To control the process of training, a development set with 10 000 examples was sampled from this train set.

The Transformer model was trained until the accuracy on the development set stopped increasing. We trained two models: 1 layer 1 head and 2 layers 4 heads.[1] The final list of hyperparameters includes powers of tf-idf and attention weights, the number of clusters and Transformer architecture. For each dataset we picked the optimal hyperparameters for our new method and the baseline on train set and evaluated them on test set.

We compared our models with the simple but hard to beat baseline that achieved second best results on bts-rnc and active-dict and third best on wiki-wiki on *RUSSE-WSI* task [4]. The method in question also relies on weighted sums of word embeddings for building context vectors and uses combinations of tf-idf and chi-squared weights.

4 Results and discussion

Table 1 lists the results achieved by our models in comparison with the baseline and *RUSSE-WSI* best result.

[1] All other hyperparameters for Transformer model taken from the *transformer_small* configuration. (https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/models/transformer.py)

Table 1. Model comparison

Model	Bts-rnc		Active-dict		Wiki-wiki	
	Train	Test	Train	Test	Train	Test
w2v,tf-idf	0.195	0.239	0.239	0.208	0.788	0.651
w2v, attention	0.196	0.178	0.140	0.067	0.828	0.651
output embeddings (2 layers 4 heads)	0.174	0.202	0.196	0.203	0.390	0.430
w2v,tf-idf,attention (2 layers 4 heads)	0.282	0.235	0.262	0.246	0.835	0.651
w2v, tf-idf, attention (1 layer 1 head)	0.220	0.187	0.224	0.195	**0.918**	0.651
w2v,tf-idf,attention + output embeddings (2 layers 4 heads)	**0.285**	0.316	**0.268**	**0.306**	0.664	0.651
Baseline, best results	0.283	0.281	0.253	0.236	0.814	0.964
Best results from *RUSSE-WSI*		**0.350**		0.264		**1.0**

The best results are highlighted in bold, our top score on active-dict beats the best result from *RUSSE-WSI* competition, and on bts-rnc our result is the second best[2]. Combination of tf-idf and attention weights works considerably better than each type of weight. Our method outperforms the baseline on two of the test sets

Transformer output embeddings did not show good results when used on their own, however combined with weighted averages of word2vec vectors they helped to improve results in a number of cases. Remarkably, most of our best results were obtained with the combined approach.

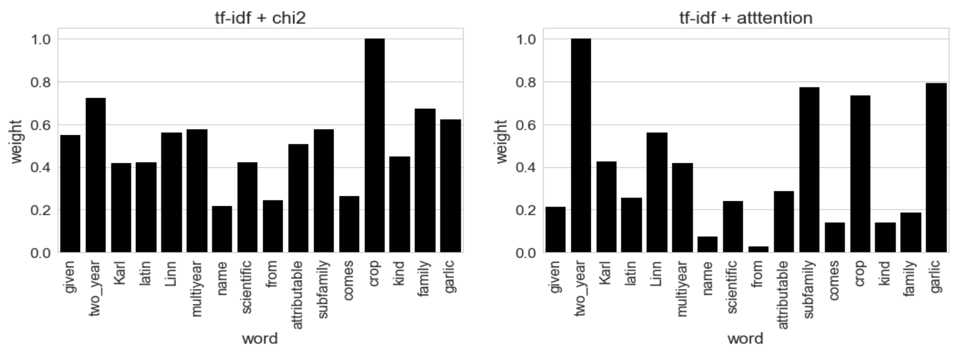
After evaluating all different hyperparameter values on the train sets from *RUSSE-WSI* we found that top 10 results on active-dict and bts-rnc datasets all used Transformer with 2 layers and 4 attention heads. This suggests that using several layers and attention heads can be crucial for achieving good results. The powers of the weights vary among the best results, which indicates that these hyperparameters need to be adjusted to a particular dataset. The hyperparameters used to achieve best results on test are as follows: tf_idf_pow = 1.5, attention_pow = 0.75, 2 clusters for bts-rnc; tf_idf_pow = 1.5, attention_pow = 0.25, 3 clusters for active-dict; tf_idf_pow = 0.5, attention_pow = 0.125, 2 clusters for wiki-wiki.

Considering the big difference in results of different models we explored the weight distributions to find, what plays a major role in the quality of word sense induction when using weighted sums of word embeddings. We observed many examples, all of which indicate that the combination of tf-idf weights with attention weights helps to reduce noise when compared to tf-idf and chi-squared weights. Figure 2 shows the weight distribution given by different models for the same context: “Onion (crop) - a species of two_year and multiyear crop, attributable to the subfami-

[2] Top scores on test datasets are submitted to *RUSSE-WSI* post-competition
active-dict - https://competitions.codalab.org/competitions/public_submissions/17806
bts-rnc - https://competitions.codalab.org/competitions/public_submissions/17809
wiki-wiki - https://competitions.codalab.org/competitions/public_submissions/17810

ly of onion family. Scientific latin name given by Karl Linn comes from latin name of garlic.”.

Fig. 2. Weights distribution



This example illustrates that the proposed model, comparing to the baseline, more vividly selects important words («two_year», «subfamily», «crop», «garlic») that indicate the sense of the word «onion».

References

1. Arefyev N., Panchenko A., Lukanin A., Lesota O., Romanov P. Evaluating three corpus-based semantic similarity systems for Russian. Proceedings of the International Conference on Computational Linguistics and Intelligent Technologies "Dialogue", June 2015 Moscow, Russia
2. Panchenko A., Lopukhina A., Ustalov D., Lopukhin K., Leontyev A., Arefyev N., Loukachevitch N.: (2018): RUSSE’2018: A Shared Task on Word Sense Induction and Disambiguation for the Russian Language. In Proceedings of the 24rd International Conference on Computational Linguistics and Intellectual Technologies (Dialogue’2018). May 30 – June 2, Moscow, Russia
3. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones J., Gomez A., Kaiser L., Polosukhin I. Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
4. Arefyev N., Ermolaev P., Panchenko A. HOW MUCH DOES A WORD WEIGHT: WEIGHTING WORD2VEC FOR WORD SENSE INDUCTION Proceedings of the International Conference on Computational Linguistics and Intelligent Technologies "Dialogue" 2018 Moscow Russia [in print]
5. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. Conference on Neural Information Processing Systems (NIPS 2013), Long Beach, CA, USA.
6. Joe H. WARD, Jr. Hierarchical Grouping to Optimize an Objective Function. Journal of the American Statistical Association. 1963.