A Review of Word Representations in Natural Language Processing

Anindya Sundar Das

This article is a brief review of two remarkable research papers (Mikolov et al., 2013) and (Peters et al., 2018) on word representations that gained a lot of attention due to the ingenious approaches and achieving the state-of-the-art performance on different NLP tasks.

The first paper (Mikolov et al., 2013) is renowned word2vec paper in which the authors introduced two novel yet simple model architectures for computing very efficient high dimensional word vector representations which capture the both semantic and syntactic characterises of words. **The major contribution of this paper is to present Continuous Bag-of-Words model (CBOW) and Continuous Skip-gram model. These models are used to obtain highly accurate word vectors from very large dataset with billions of words and a vocabulary with million words, while preserving the linear regularities (semantics and syntactic) among words.**

The paper first explores different popular approaches for word representations (Neural Net Language Model (NNLM) and Recurrent Neural Net Language Model (RNNLM)), discusses their computational complexity and methods to minimize the complexity while maximizing the accuracy, then introduces log-linear models CBOW and Skip-gram which learn the distributed representations of the words. To reduce the complexity the authors use Hierarchical Softmax, in which vocabulary is represented as Huffman binary tree (short binary codes for frequent words), thus lowering the complexity from a factor $O(V)$ to factor of $O(\log_2(V))$, where V is size of vocabulary.

Continuous Bag-of-Words Model (CBOW)

In this architecture N context words (half of them from history and half of them from future context) are projected onto the same position (vectors are added) and this projected vector is used to predict the current word. As the order of the context words has no influence in projection, this architecture is called bag-of-words model.

Continuous Skip-gram Model

The architecture is similar to CBOW, but here the current word which is input to a log-linear classifier, is used to predict the surrounding words within a certain range of the current word (history and future context words). It is also observed that increasing the range improves the quality of word vectors.

Insights

- The paper addresses that words can have different types of similarities and evaluated the word vectors on nine types of syntactic questions and five types of semantic questions. It is observed that the resulting word representations efficiently capture different syntactic and semantic linear regularities, e.g. $vector(\text{“}Paris\text{”}) - vector(\text{“}France\text{”}) + vector(\text{“}Italy\text{”})$ results in a vector that is closest (*Cosine Similarity distance*) to the vector representation of “$Rome$”. Likewise, $vector(\text{“}biggest\text{”}) - vector(\text{“}big\text{”}) + vector(\text{“}small\text{”})$ is closest (*Cosine Similarity*) to $vector(\text{“}smallest\text{”})$. The accuracy is reported as 60%.

- Earlier it was common practice to train word vectors on large corpus but with insufficient size

(50-100). The authors in this paper show that increasing both the training data size and dimensions of word vector improve the performance to a great extent.

- The comparative performance evaluation show, NNLM vectors perform better compared to the RNNLM vectors and CBOW model works better than the NNLM architecture on syntactic questions, but almost the same on semantic tasks. The Skip-gram model outperforms all other models on semantic part and on syntactic part, it performs better compared to NNLM and RNNLM models but a bit worse than the CBOW model.

- The paper also exhibits that a model trained on twice as much data over a single epoch, performs better or similar with significantly lesser training time, than model trained on the data over three epochs.

- Though Skip-gram and CBOW architectures are quite popular in use now a days, it has some disadvantages as well, e.g. the models does not capture the **morphological characteristics** of the words.

- These words representations also take context words from an window as inputs rather than entire sentence, thus ignoring the position of the word in a sentence, hence missing out the contextual meanings of polysemous words e.g. the word "get" could mean "procure", "become" or "understand" depending on context. All these different meanings of the word are represented by only a single word vector, which is an incomplete representation from linguistic perspective.

The second paper (Peters et al., 2018) introduces ELMo (Embeddings from Language Models) word representation which models both the linguistic context and complex characteristics (e.g. semantics and syntax) of a word. **The main contribution of this paper is to introduce word embeddings learned from large scale language models (ELMo representation) and incorporating ELMos in task specific models which established new state of the art across NLP problems.**

Embeddings from Language Models

Unlike other popular word vector representations ELMos are functions of entire sentence. As the authors explain in the paper, first the internal network states are obtained from **Bidirectional Language Models** pre-trained on large dataset, then **task specific linear combination of the internal states** are learned from downstream supervised NLP model; these linear combination of the internal states defines the task specific ELMo word representation.

Bidirectional Language Models

The authors used L-layer stacked Bidirectional LSTM, as language model. For any token t_k, there are $2L + 1$, token representations (r_k) from an L-layer biLM

$$r_k = \{x_k, \overrightarrow{h_{k,i}}, \overleftarrow{h_{k,i}} | i = 1, ..., L\} = \{h_{k,i} | i = 0, ..., L\} \tag{1}$$

Where $h_{k,0} = x_k$ is context-independent token layer representation , $h_{k,i} = [\overrightarrow{h_{k,i}}, \overleftarrow{h_{k,i}}]$ is context-dependent hidden state representations of k-th token at i-th layer (arrows denote corresponding forward and backward representations of bi-directional LSTM). In order to obtain token embeddings ($x_k, k = 1, ..., N$) for N tokens $(t_1, t_2, ..., t_N)$, each token is represented using character embeddings, these embeddings are processed through character n-gram convolutional layer followed by two highway layers. These token embeddings are fed as bi-LSTM inputs. Use of character embeddings ensures that the token representation picks up morphological features of words and also we can get valid token representation of out-of-vocabulary words.

Task Specific Linear Combination of Internal States

To use ELMo in a task, the biLM model weights are frozen, the vector representations r_k for each token t_k is passed to the task model, which learns a linear combinations of the representation as:

$$ELMo_k^{task} = \gamma^{task} \sum_{i=0}^{L} w_i^{task} h_{k,i} \tag{2}$$

where w^{task} are softmax normalized weights, that are learned during training the task specific models. γ^{task} allows the scaling of ELMo vector, which is important from optimization perspective.

The authors also observed fine-tuning the language model on domain specific data lowers the perplexity and improves the performance in downstream task.

Insights

The paper offered several important insights and observations:

- Integrating ELMo representations to existing baselines across set of six benchmark NLP tasks (Question Answering, Textual Entailment, Semantic Role Labelling (SRL), Coreference Resolution, Named Entity Recognition (NER), Sentiment Analysis) achieved a new state of the art performance.
- It is observed that including ELMos at both the input and output of the task specific baseline architecture improves the performance for Question Answering and Textual Entailment tasks, while highest performance is observed for SRL, when ELMo is added to the input layer.
- As activations of different layers of biLM have different distributions, incorporating layer normalization to each biLM layer before weighing them, is beneficial. Introducing dropout to ELMo and regularizing ELMo weights by adding $\lambda\|w\|^2$ to over all loss, improve the performance. It is also found that larger values of λ (e.g. λ = 1) reduce the weighing function to simple averaging over all layer representations (Eq. 2), while smaller values of λ allows the weights to vary.
- Experiment with Word Sense Disambiguation (WSD) shows that highest performance is attained when word representations are taken from second layer of 2-layer biLM, while for POS tagging task, accuracy is higher when representations are taken from the first layer. This implies, different layer of biLM captures different types of information; syntactic information is captured at lower layers while semantic information is better represented at the higher layers.
- ELMo representation also captures the contextual meaning of a word in a sentence besides syntax and semantics, which is shown through an example of a polysemous word ‘play’ in the paper.
- Integrating ELMo to task models improves the efficiency in terms of both dataset size and number of training epochs required to achieve a specified performance. For SRL task there is a 98% drop in number of parameter updates (epochs) to reach same performance level. ELMo-assisted models used dataset more efficiently; models without ELMo required much larger dataset than models with ELMo to achieve same performance.

Conclusion

The CBOW and Skip-gram model are very efficient in representing the words with high accuracy while maintaining the semantic and syntactic linear regularities, but they miss out some important word features such as morphology and contextual meanings (polysemous words), ELMo on the other hand, not only captures the complex semantic and syntactic characteristics but also the contextual meanings and uses different vectors for same word with different meanings depending on its context. The biLMs of ELMo are mainly use CNN over characters as token embeddings, however we can use the context-independent word2vec (Skip-gram or CBOW) as token-embedding inputs to ELMos. Though ELMos are more efficient than context-independent word representations, one disadvantage with the ELMo approach is, it’s complex

and time consuming and the end task still needs the language model on which the word vectors are trained on.

References

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.