УДК 004.04, 004.8

Лосєва Є. А.[1], студ.

Огляд представлення слів у обробці природної мови

Київський національний університет імені Тараса Шевченка, Україна, 01601, м.Київ, вул. Володимирська, 64/13
e-mail: bettyloreley@gmail.com

Y.A. Losieva[1], stud.

Representation of Words in Natural Language Processing: A Survey

Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska Street, Kyiv, Ukraine, 01601

e-mail: bettyloreley@gmail.com

В роботі розглянуто декілька сучасних типів векторного представлення слів в природній обробці мови. Описано три основні типи векторного представлення слова, а саме: статичне векторне представлення слова, застосування глибоких нейронних мереж для представлення слів та динамічне представлення слова, що базується на контексті. Це дуже актуальна і дуже затребувана область досліджень в природній обробці мови, обчислювальній лінгвістиці та штучного інтелекту загалом. Запропоновано розглянути декілька різних моделей для векторного представлення слова (або вбудовування слова), від найпростішої (як подання тексту, що описує виникнення слів у документі та розуміння взаємозв'язку між парою слів), до багатошарової нейронної мережі та глибоких двонаправленних трансформаторів для розуміння мови. Опис моделей подано хронологічно відносно появи. Описано вдосконалення щодо попередніх моделей, як переваги, так і недоліки представлених моделей і в яких випадках або завданнях краще використовувати ту чи іншу модель.

Ключові слова: штучний інтелект, природна обробка мови, комп'ютерна лінгвістика, векторне представлення слів.

The article is devoted to research to the state-of-art vector representation of words in natural language processing. Three main types of vector representation of a word are described, namely: static word embeddings, use of deep neural networks for word representation and dynamic) word embeddings based on the context of the text. This is a very actual and much-demanded area in natural language processing, computational linguistics and artificial intelligence at all. Proposed to consider several different models for vector representation of the word (or word embeddings), from the simplest (as a representation of text that describes the occurrence of words within a document or learning the relationship between a pair of words) to the multilayered neural networks and deep bidirectional transformers for language understanding, are described chronologically in relation to the appearance of models. Improvements regarding previous models are described, both the advantages and disadvantages of the presented models and in which cases or tasks it is better to use one or another model.

Key Words: artificial intelligence, natural language processing, computational linguistics, word embeddings.

Статтю представила к.ф.-м.н., доц. Розора І. В.

Introduction

Representation of a word is one of the fundamental tasks of natural language processing. On how much efficiently we present a vector representation of a word depends how effectively the neural model we constructed will work.

There are many different approaches, both in complexity and effectiveness for the vector representation of a word. Perhaps one of the simplest and most well-known approaches is one-hot encoding and bag-of-words model. But unfortunately how simple they are, how much they are not effective to use for neural models.

Now very often used approaches distributed word or sentence representation. Some of them we will consider in this article. We will not describe the neural representations of words, because these were already compared by Milajevs [1].

© Є. А. Лосєва, 2019

The purpose of this article is to provide a more complete picture of the vector representation of a word from the simplest models to state-of-art.

Static Word Embeddings

Some systems and methods of natural language processing (NLP) consider words as atomic units — there is no concept of similarity between words because they are presented as they are presented in the form of a dictionary with indices. There are several good reasons for this choice — simplicity, reliability, and the observation that simple models trained on huge amounts of data outperform complex systems trained on fewer data.

However, simple methods have their limits in many tasks.

The vector of the word «Queen» almost coincides with the vector obtained as a result of the following calculation of the vectors of the word: *vector («King») - vector («Man») + vector («Woman»)*. Those. the word «King» refers to the word «Queen» in the same way that the word «Man» refers to the word «Woman» (Figure 1).

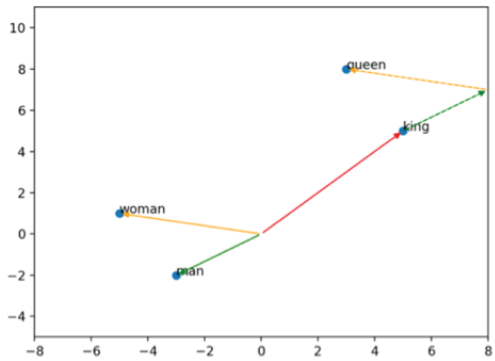

Figure 1. Word vector calculation

Bag-of-words model. A bag-of-words, or BoW for short, is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of the known words

Bag of words models encode every word in the vocabulary as one-hot-encoded vector i.e. for the vocabulary of size $|V|$, each word is represented by a $|V|$ dimensional sparse vector with 1 at the index corresponding to the word and 0 at every other index. As vocabulary may potentially run into millions, a bag of word models faces scalability challenges.

Bag of word models does not respect the semantics of the word. For example, the words *«Car»* and *«Automobile»* are often used in the same context. However, the vectors corresponding to these words are orthogonal in the bag of words model. The problem becomes more serious while modelling sentences. For example, *«Buy used cars»* and *«Purchase old automobiles»* are represented by orthogonal vectors in the bag-of-words model.

While modelling phrases using bag-of-words the order of words in the phrase is not respected. For example, *«This is good»* and *«Is this good?»* have exactly the same vector representation.

Continuous bag of words (CBOW). Continuous bag of words model was described by Mikolov [2]. The architecture of this model is meant for learning the relationship between a pair of words. The architecture of this model attempts to predict the current target word (centre word) based on the words in the original context (surrounding words) (Figure 2).

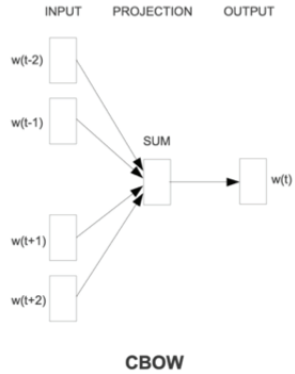

Figure 2. The CBOW architecture predicts the current word based on the context

The approach is to treat {*«The», «cat», «over», «the», «puddle»*} as a context and from these words, be able to predict or generate the centre word *«jumped»*.

But this model has its drawbacks:

- Continuous bag of words takes the average value of the context of the word (when calculating latent activation). For example, *«Applet»* can be either a fruit or a company, but CBOW takes an average from both contexts and places it between the cluster for fruits and companies.

- Continuous bag of words training from scratch can last forever if it is not properly optimised.

Skip-gram Model. The architecture of Skip-gram model is somewhat similar to continuous bag of words. It just flips CBOW architecture upside down. The purpose of a Skip-gram is to predict the context of a given word (Figure 3). It is an effective method for studying high-quality vector representations of words from a large amount of unstructured text data [3].

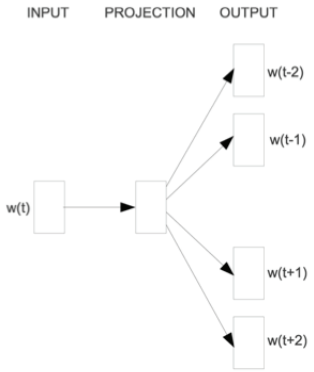

Figure 3. The Skip-gram predicts surrounding words given the current word

Each current word is used as input for a logarithmic classifier with a continuous projection layer and predicts words in a certain range (before and after the current word). Increasing the range improves the quality of the resulting word vectors, but also increases the complexity of the calculations. More distant words are usually less associated with the current word than those that are close to it, so they give less weight to distant words, selecting fewer examples of these words from case studies.

This model has several significant advantages:

- Skip-gram model can capture two semantics for a single word. i.e it will have two vector representations of Apple. One for the company and other for the fruit.
- Skip-gram with negative sub-sampling outperforms every other method generally.

Word2Vec. The combination of the two previously described models, namely the continuous bag of words and ыkip-gram, is a model developed by Google under the name Word2Vec.

The Word2Vec is a two-layer neural network with an input layer, one linear hidden layer called a projection layer, and a softmax output layer. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. The input and output layers are dictionary sizes. The size of the projection layer directly depends on the dimension of the created word vectors.

The purpose and usefulness of Word2vec is to group the vectors of similar words together in vectorspace. That is, it detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words.

Word2vec is similar to an autoencoder, encoding each word in a vector, but rather than training against the input words through reconstruction, as a restricted Boltzmann machine does, word2vec trains words against other words that neighbour them in the input corpus.

FastText. The core of FastText relies on the Continuous Bag of Words (CBOW) model for word representation and a hierarchical classifier to speed up training. FastText replaces the objective of predicting a word with predicting a category [4]. These single-layer models train incredibly fast and can scale very well. Also, FastText replaces the softmax over labels with a hierarchical softmax. Here each node represents a label. This reduces computation as we don't need to compute all labels probabilities. The limited number of parameters reduces training time.

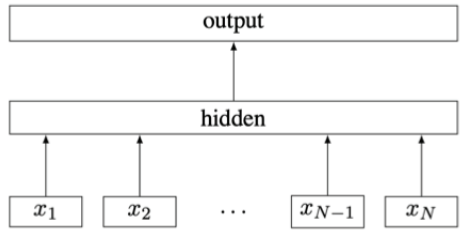

Figure 4. Model architecture of FastText for a sentence with N ngram features $x_1, \ldots, x_N$.

In contrast to word embedding technologies that are taking words as an atomic unit, FastText takes the morphology of words into account, i. e. each word consists of multiple characters and the combination of characters. In Word2Vec, for example, each word has its own distinct vector. FastText, however, represents a word by its id within a vocabulary and its n-grams of characters (Figure 4). Using n-grams of size 3

would result in a representation of the word *«apple»* as follows, *«<ap», «app», «ppl», «ple», «le>»* [5].

This approach outperformed both Word2Vec variants Skip-gram and CBOW in all word similarity datasets except for the English one. The languages with a lot of grammatical cases, like German or Russian, benefit from this method.

GloVe: Global Vectors for Word Representation. Many methods for generating word embeddings vectors are based on statistics on the contextual words surrounding them and are trained on them separately. The way it predicts surrounding words is by maximizing the probability of a context word occurring given a centre word by performing a dynamic logistic regression.

The creators of GloVe [6] illustrate that the ratio of the co-occurrence probabilities of two words (rather than their co-occurrence probabilities themselves) is what contains information and so look to encode this information as vector differences (Figure 5).

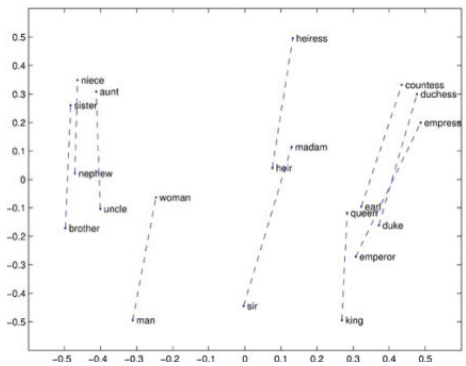

Figure 5. Linear substructures of the word in vector space

The similarity metrics used for nearest neighbour evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, a man may be regarded as similar to a woman in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

In order to capture in a quantitative way the nuance necessary to distinguish man from woman, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two-word vectors. The GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words.T he model utilizes the main benefit of count data — the ability to capture global statistics — while simultaneously capturing the meaningful linear substructures prevalent in recent log-bilinear prediction-based methods like Word2Vec. In the case of GloVe, the counts matrix is preprocessed by normalizing the counts and log-smoothing them. Compared to Word2Vec, GloVe allows for parallel implementation, which means that it's easier to train over more data. This method surpasses the results of other methods, such as CBOW, Skip-Gram in tasks such as word analogies, word similarity and recognition of named entities.

But like all existing models, this method is not perfect and has its drawbacks:

- This model uses a lot of memory: the fastest way to construct a term-cooccurrence matrix is to keep it in RAM as a hash map and perform cooccurrence increments in a global manner
- Sometimes quite sensitive to the initial learning rate

Deep Neural Networks for Word Representations

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. DNNs work well whenever large labelled training sets are available. DNNs are powerful because they can perform arbitrary parallel computation for a modest number of steps. So, while neural networks are related to conventional statistical models, they learn an intricate computation.

Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. It is a significant limitation since many important problems are best expressed with sequences whose lengths are not known a-priori.

Sequence-to-Sequence (seq2seq). Sequence-to-Sequence is a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. This method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector [7].

The model consists of 3 parts: encoder, intermediate (encoder) vector and decoder (Figure 6).

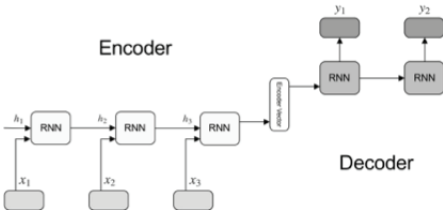

Figure 6. Encoder-Decoder Sequence-to-Sequence model

The encoder is s stack of several recurrent units (LSTM or GRU cells for better performance) where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Encoder vector is the final hidden state produced from the encoder part of the model. This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions. It acts as the initial hidden state of the decoder part of the model.

The decoder is a stack of several recurrent units where each predicts an output at a time step. Each recurrent unit accepts a hidden state from the previous unit and produces and output as well as its own hidden state. The previous hidden state is used to compute the next one.

The power of this model lies in the fact that it can map sequences of different lengths to each other. The inputs and outputs are not correlated and their lengths can differ.

A sequence to sequence model lies behind numerous systems. For instance, seq2seq model used in tasks like machine translation [8] (Figure 7), speech recognition [9], video captioning [10], etc.

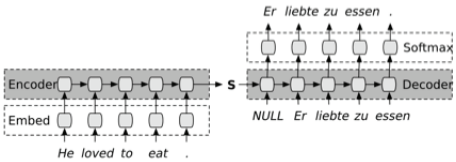

Figure 7. Sequence-to-Sequence model for machine translation

Contextualized (Dynamic) Word Embeddings

There are two existing strategies for applying pre-trained language representations to down- stream tasks: feature-based and fine-tuning. The feature-based approach, such as ELMo [11], uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning all pre- trained parameters.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Unlike previous language representation models, BERT is designed to pre-train deep bidirectional representations from the unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications [12].

There are two steps in this framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. The training is done by masking a few words (~15% of the words) in a sentence and tasking the model to predict the masked words. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labelled data from the downstream tasks.

BERT's model architecture is a multi-layer bidirectional Transformer encoder.

To make BERT handle a variety of down-stream tasks, input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g., ⟨ Question, Answer ⟩) in one token sequence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings (Figure 8).

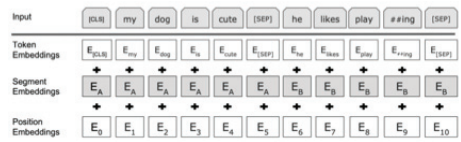

Figure 8. BERT input representation

Deep bidirectional architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP.

Conclusions

This article represents three types of word embedding: static word embeddings, deep neural networks for word representations and contextualized (dynamic) word embeddings.

Static Word Embeddings fail to capture polysemy. They generate the same embedding for the same word in different contexts. Static Word Embeddings could only leverage off the vector outputs from unsupervised models for downstream tasks — not the unsupervised models themselves.They were mostly shallow models to begin with and were often discarded after training (e.g. word2vec, Glove) .

The state-of-art results at the moment showed by contextualized (dynamic) word embeddings. Contextualized words embeddings aim at capturing word semantics in different contexts to address the issue of polysemous and the context-dependent nature of words. The output of Contextualized (Dynamic) Word Embedding training is the trained model and vectors — not just vectors.

Traditional word vectors are shallow representations (a single layer of weights, known as embeddings). They only incorporate previous knowledge in the first layer of the model. The rest of the network still needs to be trained from scratch for a new target task. They fail to capture higher-level information that might be even more useful. Word embeddings are useful in only capturing semantic meanings of words but we also need to understand higher-level concepts like anaphora, long-term dependencies, agreement, negation, and many more.

Список використаних джерел

1. *D.Milajevs* Evaluating Neural Word Representations in Tensor-Based Compositional Settings / D.Milajevs, D. Kartsaklis, M. Sadrzadeh, M. Purver. // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2014. – С. p. 708–719.
2. *T.Mikolov* Efficient Estimation of Word Representations in Vector Space / T.Mikolov, K. Chen, G. Corrado, J. Dean. – 2013.
3. *T.Mikolov* Distributed Representations of Words and Phrases and their Compositionality / [T. Mikolov, I. Sutskever, K. Chen та ін.]. – 2013.
4. *A.Joulin* Bag of Tricks for Efficient Text Classification / A.Joulin, E. Grave, P. Bojanowski, T. Mikolov. – 2016.
5. *P.Bojanowski* Enriching Word Vectors with Subword Information / P.Bojanowski, E. Grave, A. Joulin, T. Mikolov. – 2017.
6. *J.Pennington* GloVe: Global Vectors for Word Representation / J. Pennington, R. Socher, C. Manning. // Association for Computational Linguistics. – 2014. – С. 1532–1543.
7. *I.Sutskever* Sequence to Sequence Learning with Neural Networks / I. Sutskever, O. Vinyals, Q. Le. – 2014.
8. *O.Vinyals* Order Matters: Sequence to sequence for sets / O. Vinyals, S. Bengio, M. Kudlur. // ICLR 2016. – 2016.
9. *R.Prabhavalkar* A Comparison of Sequence-to-Sequence Models for Speech Recognition / [R. Prabhavalkar, K. Rao, T. Sainath та ін.]. // ISCA. – 2017. – С. 939–943..
10. *S.Venugopalan* Sequence to Sequence – Video to Text / [S. Venugopalan, M. Rohrbach, J. Donahue та ін.]. // Computer Vision Foundation. – 2015. – С. 4534–4542.
11. *M.Peters* Deep contextualized word representations / [M. Peters, M. Neumann, M. Iyyer та ін.]. – 2018.
12. *J.Devlin* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J.Devlin, M. Chang, K. Lee, K. Toutanova. – 2019.

References

1. MILAJEVS D., KARTSAKLIS D., SADRZADEH M., PURVER M. (2014) *Evaluating Neural Word Representations in Tensor-Based Compositional Settings*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 708–719.
2. MIKOLOV T., CHEN K., CORRADO G., DEAN J. (2013) *Efficient Estimation of Word Representations in Vector Space*.
3. MIKOLOV T., SUTSKEVER I., CHEN K. (2013) *Distributed Representations of Words and Phrases and their Compositionality*.
4. JOULIN A., GRAVE E., BOJANOWSKI P., MIKOLOV T. (2016) *Bag of Tricks for Efficient Text Classification*.
5. BOJANOWSKI P., GRAVE E., JOULIN A., MIKOLOV T. (2017) *Enriching Word Vectors with Subword Information*.
6. PENNINGTON J., SOCHER R., MANNING C. (2014) *GloVe: Global Vectors for Word Representation*, Association for Computational Linguistics, pp. 1532–1543.
7. SUTSKEVER I., VINYALS O., LE Q. (2014) *Sequence to Sequence Learning with Neural Networks*.
8. VINIALS O., BENGIO S., KUDLUR M. (2016) *Order Matters: Sequence to sequence for sets*, ICLR 2016.
9. PRABHAVALKAR R., et al. (2017) *A Comparison of Sequence-to-Sequence Models for Speech Recognition,* ISCA, pp. 939–943.
10. VENUGOPALAN S. et al. (2015) *Sequence to Sequence – Video to Text*, Computer Vision Foundation, pp. 4534–4542.
11. PETERS M. et al. (2018) *Deep contextualized word representations.*
12. DEVLIN J., CHANG M., LEE K., TOUTANOVA K. (2019) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Надійшла до редколегії 20.12.2018