DOI: 10.37943/ELGD6408

Bokan Madina Erzhankyzy
2nd year master's student
bokanmadina98@gmail.com, orcid.org/0000-0003-0767-4094
Kazakh British Technical University, Kazakhstan

NEGATIVE-SAMPLING WORD-EMBEDDING METHOD

Abstract. One of the most famous authors of the method is Tomas Mikolov. His software and method of theoretical application are the major ones for our consideration today. It is better to pay attention that it is more mathematically oriented. The use of embedding models to turn KGs into vector space has become a well-known field of research. In recent years, a plethora of embedding learning approaches have been proposed in the literature. Many of these models rely on data already stored in the input KG. Following the closed world assumption, the knowledge not presented in the KG cannot be judged untrue; instead, it may only be labeled as unknown. On the other hand, embedding models, like most machine learning algorithms, require negative instances to learn embeddings efficiently. To deal with this, a variety of negative sample generating strategies have been developed. The author himself had more to do with mathematics, and his method concerns, first of all, a mathematical solution for a theoretical, and then a practical solution for creating this and the method we are analyzing. Dense vector word representations have lately gained popularity as fixed-length features for machine learning algorithms, and Mikolov's system is now widely used. We investigate one of its main components, Negative Sampling, and offer efficient distributed methods that allow us to scale to indicate and exclude the possibility of probability loss in a similar value. Furthermore, this method is laser-focused on a single action in the broad sense for processing the recognition of the above-mentioned vector or words. It is important to pay attention to mathematical theory and understand the importance of the neural network in this field.

Keywords: graphs, negative sampling, method of word embedding, Tomas Mikolov, train, mathematical basic theory, sequence, matrix, vectors.

Introduction

It is better to start with the fact that the development of computer technology, namely computers, began in the 20th century, in 1920. Since then, computers have evolved into full-fledged personal computers. Vivid examples that we can observe in the 1960s [1]. With the development of computer technology, there was an urgent need to enter texts, as computers became not only a means of calculus but also of text input and search. Embedding itself can be understood as a translation of the text into the language of the computer itself. Accordingly, there is only an open and closed gate for the processor architecture. That is the supply of current and its absence. Hence, the existence of 1 and 0 modes. The easiest way that can come to mind is the orientation of words by numbers. After all, combinations can be made up to 01 to the power of 26 to subtract 1. But this method is considered extremely inconvenient and takes up productivity. And with the existence of programming languages and explanations through the writing of certain formulas of action, and we are considering single actions that reduce the load on the constant computing power of the processor itself. This way, we find more acceptable ways for embedding. This paper presents the implementation of this method for embedding and the convenience of recognizing and using the formulas below in the

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 10, JUNE 2022

present tense. It's vital not to forget that the important formulas and people involved in the evaluation of the method will be impacted. It is critical to consider that the transition to this way of embedding uses the existing search engines such as Google and Yandex in their search engines, where the much-needed ability methods of word recognition the actual cell servers by the processor in place. In this article, we will answer these questions: What is this method? How is it applied and where? What formulas exist? Graphs and an illustrative demonstration of the translation of a word into a programming language that we understand are presented. These and many other questions that you would like to find answers to may be present in this article. [1, p. 25] The following information is to get acquainted with the method itself and its application in practice.

Acknowledgements

In the first part, it is worth considering conditional probabilities and these include words from the corpus of the letter w and the letter d. In this case, the probability of loss of $p(d/w; \theta)$ will be considered

1) $$\arg\max_{\theta} \Pi_{w \in T\,ext} [\, \Pi \; p(d/w\,;\theta)\,]$$

in the practical equation shown, D(w) is a set of contexts of the word w and its corresponding resulting probability. Generation that allows you to find out the subsequent possibilities. The alternative for this equation is tracking :

2) $$\arg\max_{\theta} \Pi_{(w,c) \in D} \; p\,(d/w;\theta)$$

In this case, D is a set of all available pairs of words from a given context that we extract from the selected text. This method is an alternative to the first one with high possibilities for accurate calculation since we have a basic set in the form of D, which allows us to perform an exponential calculation.

Main body

Meanwhile, Mikolov introduced a new negative sampling system that would be much more efficient based on probability and resource capacity. This method of negative sampling is based on the theories of the probability of loss or skip-gram. It is worth considering this formula below. [2, p. 20]

1) Here are a couple of (w, d) words from the context under consideration. Was it taken from the relevant database? It is worth denoting them as follows $(D = 1 \mid w, d)$ this will be the probability that $(w. d)$ obtained not mediocre from an array or corpus of words.
2) Then $p\,(D = 0 \mid w, d)$ will be the probability that (w, d) is not obtained from the data of the actual case itself. As before, we assume that there are parameters θ. governing the distribution $p\,(D = 1 \mid w, d)$.
3) Throughout the computational process, we believe that the words and contexts are taken from separate arrays or dictionaries so that the vector associated with the word "airplane" will be quite different from the vector associated with the context of the word "aircraft". This follows from the very logic of context separation, from the vector itself. Existing points in their places and the probability of their loss. Depending on the location of the points, we introduce variables into the corresponding formula, whereas vectors and their relationships may remain completely different in one case or another. One of the reasons for prompting this question is the separation of the word by its direct meaning

DOI: 10.37943/ELGD6408

and the context for comparison. Finding points and vectors. For example, an airplane is an aircraft and an airplane. Words have the same motivated context and vector. In this case, all the words will be shared in the probable context of D. It is rather possible to assign a low probability. In practical application, the theory itself will look like this: Π (aircraft | aircraft), which will subsequently lead to a low value of $V \cdot V$, where we can assume the impossibility. Such questions appear everywhere, and we proceed from the fact that we divide a word into a direct meaning and in context, and then we perform a matching operation.

Our real task in the future is to answer this question and build steps to improve this method to maximize the probability loss. Let's imagine the given in the formulas below. To demonstrate our actions. Maximizing our actions is necessary for the code to be viable and adapted to work in a negative sample.

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(D = 1 | w, d; \theta)$$

$$= \arg\max_{\theta} \log \prod_{(w,c)\in D} p(D = 1 | w, d; \theta)$$

$$= \arg\max_{\theta} \prod_{(w,c)\in D} \log p(D = 1 | w, d; \theta)$$

The rate in this equation $p(= 1 | d, w; \theta)$ the datum can be shown by referring to soft max

$$p(D = 1 | w, d; \theta) = 1 \,\%\, 1 + e^{-vc \cdot vw}$$

And the corresponding definition of the solution looks practical as follows:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log 1 \,\%\, 1 + e^{-vc \cdot vw}$$

In this case, it is indicative that the problem has a trivial solution, if we establish θ from here we can conclude that $p(D = 1 | w, c; \theta) = 1$ for a pair of each, otherwise we will proceed from (w, d). This is not difficult to achieve given the set value of θ and in such a way that $vc = vw$ and $vc \cdot vw = K$ for each subsequent standing vc, vw, where K is a sufficiently large number (in a practical sense, we can get a probability of 1, but only as $K \approx 40$. In a general sense, an indicative solution is also necessary for theoretical understanding. The loss of words with the practical and meaning of the vector can be very different and it is worth considering without fail. And to have a sufficient understanding of the indicator, a decision is needed regarding the probability itself. The loss of words with a similar meaning and in general words in general in context or vectorial basically has a direct relationship to the probability of loss. The practice in this case is quite easy to show. It is obligatory to have a tool or other mechanism that would not allow existing vectors to have the same value. Prohibiting similar combinations (w, d). A good way for this representation exists and it looks like this for existing pairs (w, d), for them $p(D = 1 | w, d; \theta)$ must have a low value for them. For those cases where pairs do not exist, we can achieve the goal by referring to the generation of a set of D' random pairs (w, d), we can assume that they are not true.(the name of this set "negative sampling" actually comes from a set of D' randomly selected negative examples). In this case, the goal of optimization is to select or use another language to generate negative possibilities.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 10, JUNE 2022

$$\arg\max_{\theta} \prod_{(w,d)\in D} p(D = 1|d, w; \theta) \prod_{(w,c)\in D'} p(D = 0|d, w; \theta)$$

$$= \arg\max_{\theta} \prod_{(w,d)\in D} p(D = 1|c, w; \theta) \prod_{(w,d)\in D'} (1 - p(D = 1|d, w; \theta))$$

$$= \arg\max_{\theta} \sum_{(w,d)\in D} \log p(D = 1|d, w; \theta) + \sum_{(w,d)\in D'} \log(1 - p(D = 1|w, d; \theta))$$

$$= \arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1 + e^{-vc \cdot vw}} + \sum_{(w,c)\in D'} \log\left(1 - \frac{1}{1 + e^{-vc \cdot vw}}\right)$$

$$= \arg\max_{\theta} \sum_{(w,d)\in D} \log \frac{1}{1 + e^{-vc \cdot vw}} + \sum_{(w,d)\in D'} \log\left(\frac{1}{1 + e^{-vc \cdot vw}}\right)$$

$$= \arg\max_{\theta} \sum_{(w,d)\in D} \log \frac{1}{1 + e^{-vc \cdot vw}} + \sum_{(w,d)\in D'} \log\left(\frac{1}{1 + e^{vc \cdot vw}}\right)$$

If we let to be $\sigma(x) = \frac{1}{1+e^{-x}}$ we will have answer like this:

$$\arg\max_{\theta} \sum_{(w,d)\in D} \log \frac{1}{1 + e^{-vc \cdot vw}} + \sum_{(w,d)\in D'} \log\left(\frac{1}{1 + e^{vc \cdot vw}}\right)$$

$$= \arg\max_{\theta} \sum_{(w,d)\in D} \log \sigma(vc \cdot vw) + \sum_{(w,d)\in D'} \log \sigma(-vc \cdot vw)$$

following a certain method of construction, there is a difference, and it is extremely clearer and easier to master, from Mikolov. The most striking difference we want to show you to readers is that we represent the goal in general for the corpus $D \cup D'$ at the same moment they represent it as follows in one example $(w, c) \in D$ and k examples $(w, cj) \in D'$ following a certain example of construction D' following the basic law of construction in the negative sampling method. [4, c.30]

In particular, a negative sample k Mikołów and other related scientists have created a methodical method to perform the task D' k times erected more than D, and for each of the existing $(w, d) \in D$, we create k samples $(w, d1), \ldots, (w, dk)$, where cj is shown later in relation to its distribution across anagramme, raised to a power is necessary to derive the indicator 3/4. This is equivalent to extracting samples from the given case (w, d) in D' from the unigram distribution $(w, d) \sim pw(w)\ p_{contexts}(c)\ ¾\ Z$, where pwords(w) and pcontexts(c) are the one-program distributions of words in this equation and the contexts of this represented word and, respectively, that Z is the normalization constant in the verbal sense in the equation. In the work of Mikolov and other scientists, each context is in a vector in the existing equation – this is the represented word (and all words are displayed as contexts for probability derivation), therefore we assume that $pcontext(x) = pwords(x) = count(x)\ \%\ |Text|$ [6,p.130]

5.1 Remarks on the topic and the corresponding definition of the solution look practical as follows:

1) The main significant difference in this case, from the skip-gram model described in the skip-gram topic, is that the main formulation in this section does not model p(d|w), but rather a specific model compatible with the distribution of w and d.
2) Thomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado and Jeffrey Dean. They are connected in the creation of works and the promotion of the theory of negative selection in the field of the probability of words falling out and the main works for search engines and individual tasks for the processor. In promotion with quite successful results in the field of neural information processing systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of the meeting held on December 5-8, 2013, Lake Tahoe, Nevada, USA States, pages 3111-3119, 2013.

DOI: 10.37943/ELGD6408

The transition from theory to an indicator of understandable practice.

Having understood the theoretical meaning and structure and solved the problem of the probability of falling out in a negative selection by example, we decided to show readers the main practical steps for understanding a negative selection in the simplest way below. For example, the word "Train" or any word. To show how the code works. Below is the code itself and the graph. As we can see, there are some sides of the code that we can use to make a word in computer language. Negative sampling: [7]

```
epochs = 10000
print_every = 1000#In skip-gram middle word becomes the input which predicts #surrounding
words(targets)
#Every time one_hot_auto() is called fresh batch is generatedmid_hot, sur_hot, labels =
one_hot_auto()
for i in range(epochs):#Forward prop to get hidden layer
    z_midl = fc_inp_word(torch.Tensor(mid_hot))
    z_sur = fc_targ_word(torch.Tensor(sur_hot))

#Initialize a 1d matrix of 0s to store dot products between each
#row of first hidden matrix embedding input with second hidden
#matrix embedding target words
#This score forms the basis for optimizationdot_u_v = torch.zeros(mid_hot.shape[0], 1)
for j in range(len(z_midl)):
    dot_u_v[j, :] = z_midl[j, :] @ z_sur[j, :]

#Sigmoid activation applied to dot products of vectors
desired_logits = dot_u_v
sig_logits = nn.Sigmoid()(desired_logits)

#Back prop and stepping
optimizer.zero_grad()
loss = criterion(sig_logits, torch.Tensor(labels).view(sig_logits.shape[0], 1))loss.backward()
optimizer.step()
if i % print_every == 0:
    print(loss.data)

#Scheduled one_hot_auto() to generate fresh random pairs
mid_hot, sur_hot, labels = one_hot_auto()#Prints losses
tensor(0.6029)
tensor(0.0674)
tensor(0.0146)
tensor(0.0060)
tensor(0.0029)
tensor(0.0022)
tensor(0.0011)
tensor(0.0008)
tensor(0.0003)
tensor(0.0003)
```

Thus, we find out that it is also possible for the processor to recognize words by single actions with the calculation of a certain probability of loss. In the future, it is worth considering that in the vector, if we are talking about a linear vector, there is a certain array of letters forming a verbal composition through a prepared sample. This saves resources and has only advantages.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 10, JUNE 2022

For example, processing an array is more than a single action, unlike processing by a processor with several actions to create its own calculation and refer not to existing letters. Thus, during negative processing, the processor runs in information referring to an existing array, which is stored in a certain way. [3, p.20] This greatly reduces the load not only on the processor, but also on the service itself. In this case, we are talking about the existing advantages.

Basically, we will allude to the method of selection with a square. A square in which the current supply and its absence are located on the left side. A certain sequence creates the probability of memorizing the most convenient ways of marking letters. 1A and 1B in this case contiguously create 1 "a" and 1 "b". Given the matrix, we can imagine a complete working software system for our own word recognition and the probability of a semantic hit on the graph. Next, it should be borne in mind that it is the negatively excluding probability that is the main meaning for a negative selection in embedding. The main idea is to eliminate duplicate values, which clutters up the code and loads it with unnecessary calculations. Having completely analyzed the topic, we can see that by eliminating such a scenario in advance, we can finally come to the most convenient computational method. [4, p.20]

Nevertheless, the general indicator is the very concept of an array from where the basic information for processing comes from. Without understanding the array, it is impossible to imagine a general semantic selection for words. Arrays can be quite different, but the classic way of the indicator in square brackets with the form of processing the prepared combinations mainly in the language itself can look like this [a, b, c, d, e, f, g, etc.]. Thus, we create a word by selection and exclusion in negative processing in the simplest way, which is the concept of embedding. Do not forget that the code is extremely large, however in this work we have examined the main examples for a complete understanding of the working state of the codes and the method of non-reactive selection. [5, p. 30]

The existing algorithm. An algorithm is a basic principle in programming and life. An algorithm is a certain sequence of actions. For example, when processing, first of all, we refer to an array or resource. After accessing the resource, we proceed to identify variables, for example, and from here we proceed from the negative selection method to prevent the repetition of certain probabilities D, which we presented above.

Conclusion

It is worth saying that the main analysis for understanding the general features of this article is given with the concepts of practical examples with references in the form of theoretical material. In general, the method of negative selection itself was mainly presented in 2013. [9] Speaking about the development of the neural network, do not forget about the mathematical basic theory of understanding the probability of word selection.[10] In this context, a deep theory is related to as neural network. It is also crucial to concentrate on the real application of the code in practice. We apply practical code referring to the developments. The method of negative selection serves to indicate the possibility of probability loss in a similar value and excludes it. Also, this method is deeply focused on a single action in the general sense for processing the recognition of the above-proposed vector or in a linguistic context. [12, p.10]

For an approximate understanding, we started calculating existing points in relation to each other and considered vectors. In the presented abstractness, we would talk about a square that could show us several adjacent points with similar meanings. We must exclude other existing points that are not related to our selection. Often speaking about this method, many forget that the main task of this negative selection method is to create certainty in the answer in the recognition itself.

DOI: 10.37943/ELGD6408

References

1. Shenron (April 2009). *History of programming development. U.S. history of computers and developing*. https://www.historyofthings.com/history-of-computers
2. word2vec and Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, pages 3111–3119, 2013. https://tensorflowkorea.files.wordpress.com/2017/03/cs224n-2017winter-notes-all.pdf
3. Embedding process and Negative-Sampling Word-Embedding Method by Tomas Mikolov. USA (Lake Tahoe, Nevada, 2013)
4. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, pages 3111–3119, 2013.
5. Han, L., Kashyap, A. L., Finin, T., Mayfield, J., & Weese, J. (2013, June). UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity* (pp. 44-52).
6. Walker, A. J. (1974). New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, *8*(10), 127-128. https://doi.org/10.1049/el:19740097
7. Mikolov, T. (2008). Language models for automatic speech recognition of czech lectures. *Proc. of Student EEICT*. https://spark.apache.org/docs/latest/mllibfeature-extraction.html.
8. Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*. https://doi.org/10.21437/interspeech.2014-564
9. Kiros, R., Zemel, R., & Salakhutdinov, R. R. (2014). A multiplicative model for learning distributed text-based attribute representations. *Advances in neural information processing systems*, *27*.
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.
11. Mikolov, T., Kopecky, J., Burget, L., & Glembek, O. (2009, April). Neural network based language models for highly inflective languages. In *2009 IEEE international conference on acoustics, speech and signal processing* (pp. 4725-4728). IEEE. https://doi.org/10.1109/icassp.2009.4960686
12. Grbovic, M., Djuric, N., Radosavljevic, V., Silvestri, F., & Bhamidipati, N. (2015, August). Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (pp. 383-392).