IJREAT International Journal of Research in Engineering& Advanced Technology, Volume 8, Issue 4, Aug - Sep, 2020
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

A PRIMER ON WORD EMBEDDING

[1]Satvika, [2]Dr. Vikas Thada, [3]Dr. Jaswinder Singh

1PhD Scholar, CSE Deptt., Amity University, Gurugram, Haryana, India, Satvika16oct@gmail.com
2Assistant Professor, CSE Deptt., Guru Jambhehwar University, Hisar, Haryana, India, vthada@ggn.amity.edu
3Associate Professor, CSE Deptt., Amity University, Gurugram, Haryana, India, jaswinder_singh_2k@rediffmail.com

Abstract—The current research on the topic Machine Learning and especially the domain of Natural Language Processing has gained much popularity in the modern era.One such framework for attaining NLP tasks is word embedding, which represent data as vectors i.e. real numbers rather than words of natural language because neural networks do not understand them naturally. Word embeddings try to capture both syntactic and semantic information of words and capture relationships according to context and morphology. This paper reviews each word embedding technique available in the contemporary world ranging from traditional embeddings based on frequency of terms to pre-trained embeddings like prediction-based embeddings. The goal of this paper is to present the myriad methods available for word embedding, classify their working patterns, also identifying their pros and cons for working on text classification and detect their hegemony over the traditional methods of NLP.

Keywords—CBoW; Deep Learning; fastText; GloVe; Machine Learning; Natural Language Processing(NLP), Skip-Gram; Word Embeddings.

I. Introduction

The modern world is called the "data age" because data is the new tool or weapon for achieving laurels in the business or corporate era. The correct and updated data (or information) is the most prominent thing required by the commercial houses either small or big to gain a candid advantage over its competitors. Nowadays, more and more companies are investing capital (human resource, money and time) in data collection, processing and analytics. This process involves enormous data harvesting or warehousing from numerous sources whether social media or traditional methods, and then processing this whole data so that data collected from various sources is in a common format. It is quite vital as unstructured data is estimated to be approximately 70% of the total data accumulated; which must be brought to same format to apply mathematical or statistical functions.

These functions help in analyzing the data and discovering the hidden patterns or the information. By doing so, the companies get a real insight on customer behavior, buying pattern and potential customer identification for peculiar products and services; which is mandatory in the contemporary world to boost the productivity and revenue. There are myriad mechanical or algorithmic paradigms available for customers' opinion mining such as Rule-based approach, Machine Learning and Deep Learning, etc. which try to learn the exact sentiments behind customers' reviews and predict them accurately[1].

Rule-based approaches are based on defining a bunch of rules for identifying the intensity and emotions that a word expresses[2].These rules must be logical, intuitive for a particular domain under consideration and should also take into account the sarcasm and satire. The chief concern in rule-based approaches is their static behavior i.e. the rules are static and do not change rapidly with time, which make them obsolete soon. However, it must be considered that data is too dynamic these days and hidden patterns contained in data change quickly also[3]. Another imperative hindrance in rule-based systems is the huge amount of hard work and testing involved, which makes it sluggish. Also, rule-based systems give excellent results in a narrow domain, but behave poorly when it comes to generalization[4]. Due to these quandaries, the rule-based approach got obsolescent and machine learning algorithms surfaced which is used widely today for data analytics.

Machine learning is the state-of-art branch of Artificial Intelligence, which uses statistical models and algorithms to attain a precise task using inferences and patterns present in the data itself rather than explicit directions from humans[5]. The machines (or computers in this case) learn by gaining knowledge from past know-hows and then try to practice it in the unknown but related domains. There are copious categories of machine learning available chiefly supervised, unsupervised, reinforcement, semi-supervised, self-supervised, multi-instance, inductive, deductive, transductive learning, etc, which are used specifically for particular tasks.

The machine learning approach varies its underlying algorithm according to the varying data and hence more dynamic than rule-based approach[6]. The usual problems solved by machine learning are classification, clustering, dimensionality reduction, anomaly detection, etc. and some of the noteworthy machine learning algorithms to handle these hitches are Naïve Bayes, Support Vector Machine, K-Nearest Neighbor, Decision Trees, etc. Overall, it can be said that machine learning is very adaptive which learns patterns from data and therefore can be applied to plethora of applications like natural language processing, spam detection, sentiment analysis, recommendation systems, genre (multi-class) classification, stock prediction, etc[7].

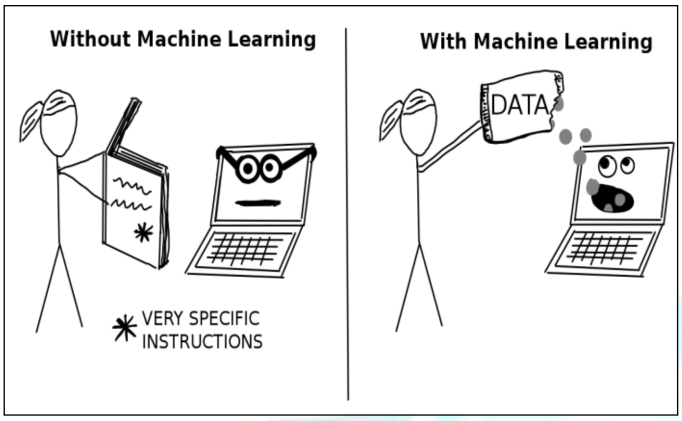


Fig. 1: How machine learning works[8]

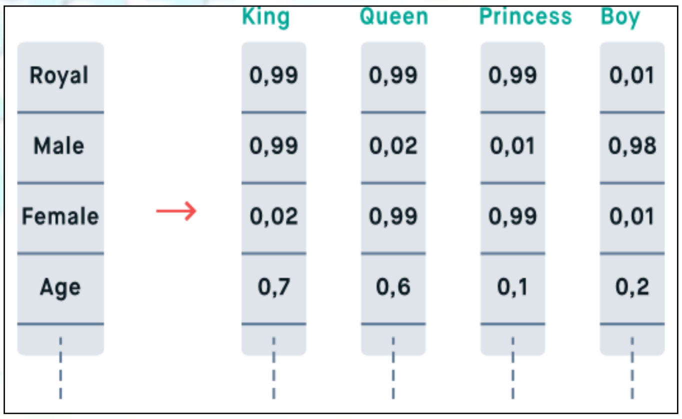
Last but not the least is the Deep learning methodology, which is nothing but group of algorithms and models that entailmanifold processing layers. Deep learning is the newest branch of Artificial Intelligence which uses artificial neural networks (similar to neural network present in human brain) for the process of learning and predicting[9]. Therefore, it can be defined as the science of making machines intelligent my making them learn by example, just like the humans do. It is usually an unsupervised kind of learning which can absorb information from either structured or even unstructured data; hence it is practically more useful than its counterparts.

There are plenty of applications of deep learning like natural language processing, self-driving cars, healthcare, financial fraud detection, fake news detection, virtual assistants (like Siri, Cortana & Jarvis) etc. The next section discusses the need of word embeddings. Section III details about the numerous types of word embeddings. Section IV gives an insight on the traditional word embeddings and segment V deliberates the pre-trained word embeddings. Finally, the next section specifies the conclusion for the article.

II. Word Embedding and its Need

Machine learning as well as Deep learning basically uses the neural networks and its myriad variations like Convolution Neural Networks, Recurrent Neural Networks, etc. to handle the perplex problems especially in the domain of text processing. Nevertheless, the key point here is that the neural networks cannot directly use the natural languages like English or Hindi because neural nets do not understand the words or phrases; instead they use numerical data. This is where word embedding comes into play i.e. represent the textual information from languages into statistics.

In short, word embeddings can be explained as the numerical representation of a documenttexts' semantic meaning[10]. These vectors actually provide the relation between various words or phrases of the documenti.e. the words having similar meaning have closer vector values, which establish their closeness in the linguistics. For example, the word "male" is proximate to "King" and "Boy" and quite far from "Queen" and "Princess", as shown in figure 2.



	King	Queen	Princess	Boy
Royal	0,99	0,99	0,99	0,01
Male	0,99	0,02	0,01	0,98
Female →	0,02	0,99	0,99	0,01
Age	0,7	0,6	0,1	0,2

Figure 2: Example of Word Embeddings[11]

It is essential to understand that word embedding is not the mere translation of texts into numbers rather it conveys the semiology meanings of the words. The distance between vectors represent the similarity between myriad words[12]. In brief, it can be concluded that computers are incapable of understanding natural language's words and that's why word embeddings are required. Also, encoding phrases into numeric form can make mathematical functions especially matrix operations successfully operate for NLP tasks.

Word embeddings can be categorized broadly into two categories namely Frequency-based and Prediction-based. As the name suggests, Frequency-based embeddings take into account the frequency or number of times a word occurs in the document to find its relevance. It is also called count-based embeddings and its principal point is to give weights according to the occurrence and also the context[13].Frequency based word embeddings can again be classified into: Count Vector, TF-IDF Vector and Co-Occurrence. The Prediction-based embeddings fundamentally consists of two major approaches: Continuous Bag-of-Words (CBOW) and Skip-Gram that come together to form the Word2Vec which is the prominent state-of-art model of word embedding. Each of the above will be discussed in detail in the following sections.

III. Count-Vectorization

Count Vectorizing or One-Hot Encoding (OHE) is the most basic word embedding that works on a simple binary principle and produces a high-dimensional sparse matrix. Firstly it creates a bag-of-words (vocabulary) from all the given text or corpus, which is defined as the assemblage of similar text. Secondly, it counts the occurrence of each word in the document existing in the corpus. The final output of count vectorizing is a sparse matrix with dimensions D*T; where D denotes number of documents and T signifies number of dissimilar words in the vocabulary[14]. This can be understood by the following example:

Document 1 (D1): "The dog ate the cat".

Document 2 (D2): "The lion can eat a cat".

As displayed above, there are two documents i.e. D=2 and total 8different words thus making T=8 in the corpus: {"the", "dog", "ate", "cat", "lion", "can", "eat", "a" }. Hence, count vectorizing encoding for the above corpus should be:

Table 1: Example of Count Vectorization

	the	dog	ate	cat	lion	can	eat	a
D1	2	1	1	1	0	0	0	0
D2	1	0	0	1	1	1	1	1

The matrix generated is self-explanatory: if a word appears in a corresponding document, it is given the value "1", else given "0". If a word appears multiple times, it can be given that frequency; just like "the" word is coming two times in the D1, hence value of "the" is marked as 2 for D1. As the words "lion", "can", "eat" and "a" are absent in D1, hence they are assigned "0" for D1. Similarly, the words "dog" and "ate" are not given in D2 and marked "0".Each cell in the matrix corresponds to one document and one precise word in the corpus. Usually a corpus contains 1000s of sentences with 10s of words, thus producing a lot of words which are not occurring in most of the documents. Thus the topmost pickle in this embedding is the occurrence of cells with loads of 0's, hence called the sparsity matrix with high dimensions. Nonetheless, these dimensions or features can be diminished, so that visualization is possible because of lower dimensions[15]. One way of doing so is to eliminate commonly occurring words also called "stopwords" from the corpus. This whole process of generating table is called tokenization i.e. identifying all the individual words present in the corpus.

IV. TF-IDF Vectorization

The full abbreviation of TF-IDF is "Term Frequency – Inverse Document Frequency". The phrase "Term Frequency" refers to the incidence of a term in a document divided by the entire amount of documents; while the term "Inverse Document Frequency" denotes the logarithmic value (to the base 10) of whole number of documents divided upon the magnitude of documents a specific term is appearing in.

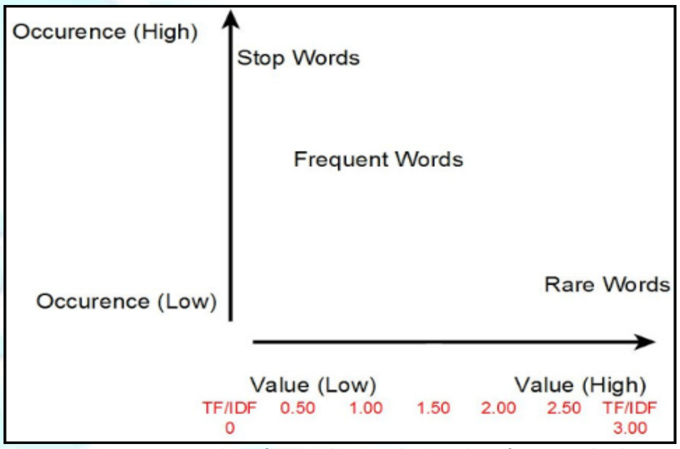


Figure 3: TF-IDF values for words according to their frequency[17]

The product of TF and IDF values for a word is known as its TF-IDF weight; its higher value signify rarity of a term and vice-versa[16]. TF-IDF vectorization is the embedding which removes the problem in count vectorizing which simply records the occurrence of a particular word in the document irrespective of its significance[18]. Actually the commonly used words like "a", "an", "the", "this", "that", "is", "am", "are", etc. are most likely to appear in the documents and their frequency does not mean they are strongly imperative, instead they are least important. Hence, TF-IDF tries to eliminate these frequent words by assigning them lower weights and try to include more noteworthy terms by empowering them with more weight values[19]. For example, when a 1000-word document on cryptography comprises the word "cipher" 145 times, the TF value for this word is 145/1000 i.e. 0.145. Also consider if the term "cipher" is contained in 40 documents out of total 1000 documents in the corpus, then IDF value will be calculated by formula $\log_{10}(1000/40) \rightarrow \log10(25) \rightarrow 1.398$. Having calculated both TF and IDF values, the final TF-IDF value can be deliberated by their multiplication:

TF-IDF for word "cipher" is 0.145 * 1.398 i.e. 0.203.

The advantage of TF-IDF can be taken by running a TF-IDF calculation on every word of the corpus and choosing the terms with higher TF-IDF values. Correspondingly these selected terms can be again be aligned according to their search volumes on the web and lastly selecting the terms with greater search volumes as they make more sense according to the users. One downside of TF-IDF vectorization is that it also produces a high dimensional representation that may not confine the actual semantic relationship between words.

V. Co-Occurrence Vectorization

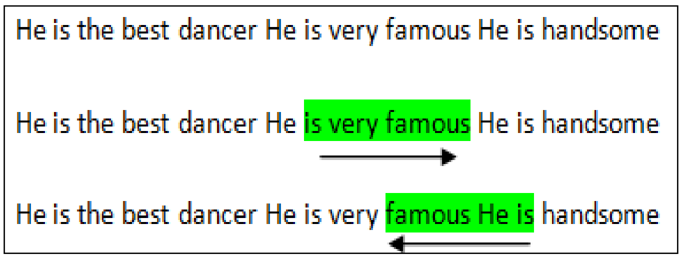
As the name suggests, it is a matrix that suggests how some words tend to occur jointly and are probably used in the similar context. It is a colossal matrix (even bigger than the one-hot encoding matrix) that is comparable in size to the whole corpus and uses positive whole numbers to mark the presence of co-occurrence of two words and integer value “0” to mark absence of co-incidence. If some terms are coming multiple times together, the presence is denoted by that frequency[20]. To check the co-occurrence of words, a context window size must be decided, which indicates how many words before and after must be scanned for a particular term. To explain this, let’s take an example:

Corpus: “He is the best dancer. He is very famous. He is smart”. This corpus has 3 sentences and let’s assumes the size of context window is 2 i.e. each word will be scanned 2 words both before and after a given word. The co-occurrence matrix for the above corpus should be the following:

Table II. Example of Co-Occurrence Matrix

	He	is	the	best	dancer	very	famous	smart
He	0	3	1	1	1	1	1	1
is	3	0	1	1	1	1	2	1
the	1	1	0	1	1	0	0	0
best	1	1	1	0	1	0	0	0
dancer	1	1	1	1	0	0	0	0
very	1	1	0	0	0	0	1	0
famous	1	2	0	0	0	1	0	0
smart	1	1	0	0	0	0	0	0

The explanation is quite easy; first of all every word in the given corpus does not occur with itself, hence the cell at intersection of word with itself is marked “0”, like “He” does not occur with “He” in any sentence of vocabulary. The occurrence of “He” and “is” is coming in all three sentence, hence marked “3” for intersection of “He-is” and also for “is-He”. Similarly, the occurrence of “is-famous” is stated as “2” coz in 2nd sentence, “famous” term is coming in context window of “is” of 2nd sentence as per forward propagation and also comes in context window of “is” of 3rd sentence according to the backward scanning. Also, check some words do not happen together like the pair of “famous-smart” as the context window is only 2 words long, hence they are marked “0” for both “famous-smart” and “smart-famous”.


He is the best dancer He is very famous He is handsome

He is the best dancer He is very famous He is handsome

He is the best dancer He is very famous He is handsome

Figure 4: The Co-Occurrence for “is-famous” word pair

Although the biggest benefit of Co-occurrence vectorization over Count vectorization and TF-IDF is that is preserves the semantic relationship between terms and also comparatively faster[21]. However, the focal drawback is its enormous size; but the modern tools like Hadoop can handle this by factorizing the matrix.

VI. Continuous Bag of Words (CBoW) Model

Bag of Words (BoW) can be defined as the means of mining features from text to employ in machine learning or deep learning algorithms. BoW basically involves two entities: a dictionary of recognized words and a way to quantifythe presence of these words. CBoW model is one of the fundamental approaches based on BoW and used for word embeddings. It is generally recognized as a learning model that predicts a term by its context and that context maybe a single or multiple words (usual case). Henceforth for every word, ‘n’ words before and after it is considered to check the semantic association among words. It is based on neural networks with hidden layer/s and works on a simple norm that milieu of a word/phrase can be branded by the neighboring words[22].The dimension of the hidden layer and the output layer should remain same, but the input layer dimensions can be altered along with the activation function of the hidden layer. Figure 5 shows an example of CBOW model:

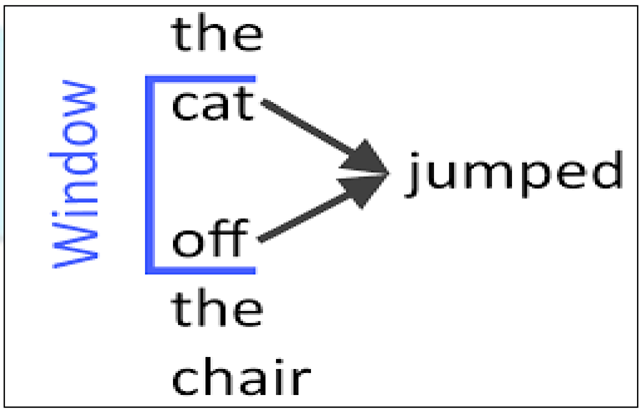


Figure 5: Prediction using CBOW Model[23]

As depicted in the above example the surrounding words: “the”, “cat”, “off”, “the” and “chair” are able to predict the central word “jumped” to complete the sentence. Here the term “window” refers to the context window that determines how many words afore and later a given term. There are several advantages of CBoW like chiefly diminutive memory usage and being probabilistic in nature comparative to the traditional frequency-based embeddings. The main disadvantages of CBoW are sometimes average prediction for a word. It happens due to the hidden layer's output which is the average of word vectors related to context terms at the input layer[24]. For example, "Windows" can be both a graphical operating system by Microsoft or opening in a wall.

VII. Skip-Gram Model

The skip gram model or continuous skip gram model is an alternate to the CBoW model with a small twist. Instead of using the surrounding words to unearth the middle word, it learns an embedding by predicting the surrounding words with the help of a given central word. It can be said that skip gram architecture is exact reverse of CBoW but keeps the topology same. Nonetheless it can be imagined that skip gram is analogous to 1-context architecture of CBoW[25].

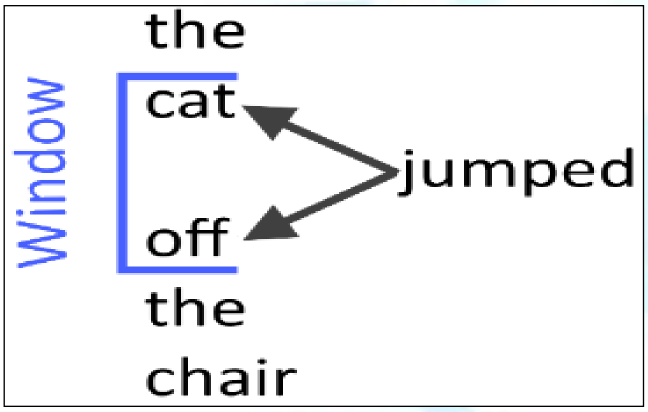


Figure 6: Prediction using Skip-Gram Model[23]

As shown above, skip gram approach predicts the neighboring words using the current word i.e. each central word is given input to a classifier with constant projection layer and predicts the nearbywords within a particular range previous to and next to the current word. The training purpose is to learn word vector illustration that is superior at forecasting the close by words. The most important plus of using skip-gram is capturing multiple semantics for each word, like foretelling both graphical operating system by Microsoft and opening in a wall for "Windows". Another chief boon of skip gram model is its association with negative sampling, which outperforms its other competitors[26].

VIII. word2Vec

Word2Vec can be defined as the Google's magic wand for word embedding, which has made the process easy and simplified. Word2Vec is based on the amalgamation of CBoW and Skip-Gram models (the shallow neural network architectures) to map word(s) to the target term(s). It can also be stated as the successor of the neural probabilistic model, which acquires embedding by attaining classification or modelling. Word2Vec was published in a research paper by Tomas Mikolov, Kai Chen, Greg Corrado & Jeffrey Dean (all Google employees) in September 2013[27].It is word embedding powerhouse by Google that trained on circa 100 billion words taken from Google News Data with nearly 300 dimensions. Now Word2Vec can be said as the state-of-art word embedding standard as it produces generalized results and that too with lesser dimensions. It is based on simple working principle that using trivial two-layer neural network for training of rebuilding linguistic backgrounds of the word[28]. The input of Word2Vec is usually an enormous text corpus and produces "vector space" of huge dimensions, which means assigning a corresponding vector in the space to each exclusive word. At that moment, words of corpus that share mutual context are located adjacent to the each other in vector space and are hence called word vectors[29]. The metric used here is "cosine similarity" or "Euclidean Distance" which discovers the likeness between myriad words or documents; the smaller the cosine angle, closer the documents. Cosine similarity can be defined as the amount of cosine of the angle between two non-zero vectors in a three-dimensional space[30].

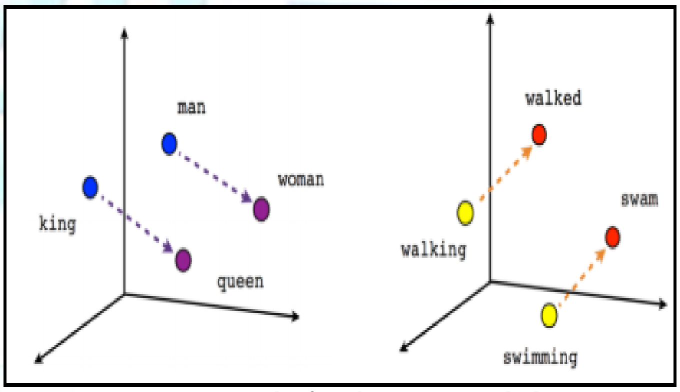


Figure 7: Example of words in vector space[31]

As Word2Vec works on two alike yet opposite architectures, which both have their own pros and cons. On one hand, CBoW is way quicker and produces enhanced word presentations for more recurrent terms. On the other hand, Skip gram is better suited for rare words and can also work fine with trivial vocabulary[32]. A major parameter of Word2Vec is sub-sampling i.e. eliminating high frequency words above a definite threshold maybe subsampled to upsurge the training swiftness. Another imperative factor is dimensionality whose ideal value lies between 100 and 1000 because word embedding quality escalates with increased dimensionality. However the most prominent parameter for Word2Vec is its training algorithm: hierarchical softmax and/or negative sampling being the most eminent ones. The hierarchical softmax technique uses a Huffman tree to diminish calculation and boost the conditional log-likelihood, thus making it work superior for the rare words. Besides, the negative sampling uses the sampled negative instances and minimalize their log-likelihood for attaining better results for commonly used words and vectors with lower dimensions[33].

As stated above, negative sampling with continuous skip-gram is far superior method for embedding, but how does it work? The answer is quite simple: instead of just using positive samples, add some negative samples too. As the ultimate aim of every embedding algorithm is to better predict the context words according to a particular word, hence each model takes input and output words and its output score forecasts whether these words are neighbors or not in the vector space. If the words are neighbor, the target is set as “1”, else it is given the value “0”. A desirable model may generate 100% correctness by giving all 1s but will be unable to learn and generalize. Hence, the negative samples must be added to the dataset for 0s i.e. randomly designated words that are not neighbors as the output terms. The negative sampling increases the efficiency by slightly decreasing the speed, thus maximizing the result. This can be easily understood by an example: if a bag contains 6 red balls, 12 white balls and 2 black balls, then simple sampling has probability of picking red balls is 6/20 or 0.3, black ball is 0.1 and white ball is 0.6. The main shortcoming here is that high frequency data will be picked (because of higher probability), which typically is of lesser value. Hence adjusted sampling is done i.e. reducing probability by 3/4th rule, hence reducing white ball probability to 0.45 and probabilities of red balls and black balls will be enhanced. The appropriate number of negative samples varies between 5 and 20, although 2 to 5 negative samples are also sufficient[34].

IX. GLOVE

GloVe stands for Global Vectors for word representations, which can be stated as the extension of Word2Vec, but yielding even better results.It was developed by Jeffrey Pennington, Richard Socher and Christopher D. Manning in 2014 as an unsupervised learning algorithm for gaining vector illustration for words at Stanford[35]. The chief addition to the arsenal here is the inclusion of global statistics and exclusion of window feature for local context to generate improved embeddings. The training here is done on the non-zero values of the global word co-occurrence matrix, that depicts how rarely or frequently two terms co-occur. The representations also show fascinating linear substructures of word vector space.

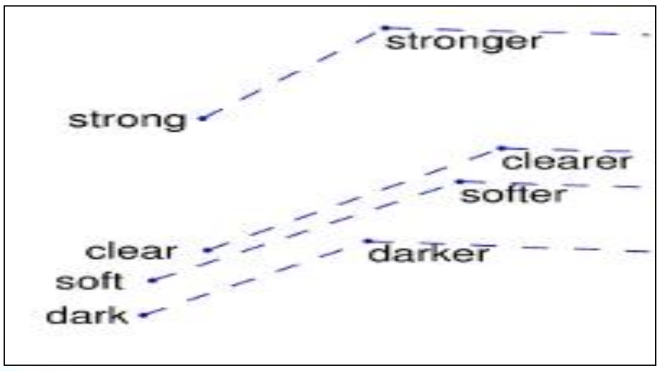


Figure 8: GloVe showing comparative-superlative terms[36]

These pre-trained word vectors can be downloaded online and it comes in different shapes and sizes. The smallest one contains 6 billion tokens with 50, 100, 200 or 300 dimensions, which is 822 MB in size. The subsequent file contains 42 billion tokens with 300 dimensions of 1.75 GB size. The largest file is trained on 840 billion tokens with 300 dimensions and of size 2.03 GB size. Additional feather in its cap is a pre-trained file for Twitter data that contains 2 billion tokens of 25, 50, 100 & 200 dimensions with 1.42 GB size.

X. FASTTEXT

fastText by Facebook is the free and open source yet lightweight word embedding library to create supervised or unsupervised algorithms that are generally used for text representation and classification. fastText is based on two research papers written by Tomas Mikolov et. al. in 2016 and now it offers three models with 1 or 2 million word vectors and 300 dimensions[37]. The maximum model size of fastText is closely 8.2 GB in size.

The chief features of fastText are its pre-trained models for nearly 294 languages, working with general hardware with ease and even compatibility to work on cellphones[38]. However, a striking characteristic of fastText is its ability to yield vectors for even the unknown vocabulary, misspelled words, two words merged into one another. This is achieved by splitting entire terms into a bag of n-gram letters and these subwords can be merged independently to create new words, as shown in figure 8. The best results of fastText can be achieved by using skip-gram architecture in conjunction with negative sampling. As explained in the previous section, negative samples are random samples from the corpora and the sub-words will be used as positive samples. Also, fastText does not support GPU till now and works only on the CPU[39].

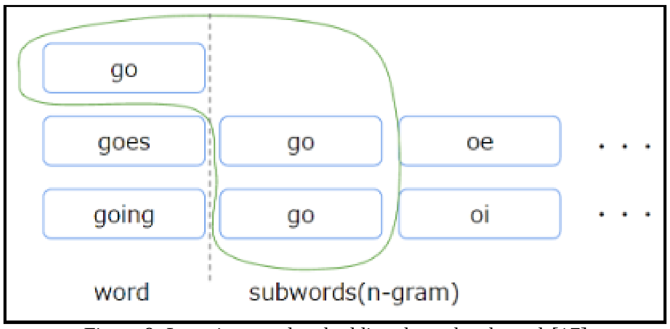


Figure 9: Learning word embedding through subwords[17]

XI. Poincare Embeddings

Poincare Embeddings by Facebook is a novel research idea given by Maximilian Nickel and Douwe Kiela in 2017 that focuses on finding the hierarchical word representations[40]. To trace any term's hierarchical properties, Poincare uses a hyperbolic geometry, where the distance is used to express likeness and associations. The hyperbolic space is preferred over the Euclidean distance because it is unremitting form of a tree and hence superior to comprehend the organizational property of learning. Due to the above elements, the embeddings generated by hyperbolic space are more generalized and better for illustration. The key aspect here is that reduced dimensions are used here to create the hierarchical information, which was proved when they generated WordNet[41].On the downside, Poincare embeddings are less proficient for modelling the perplex patterns due to their dimension limitation.

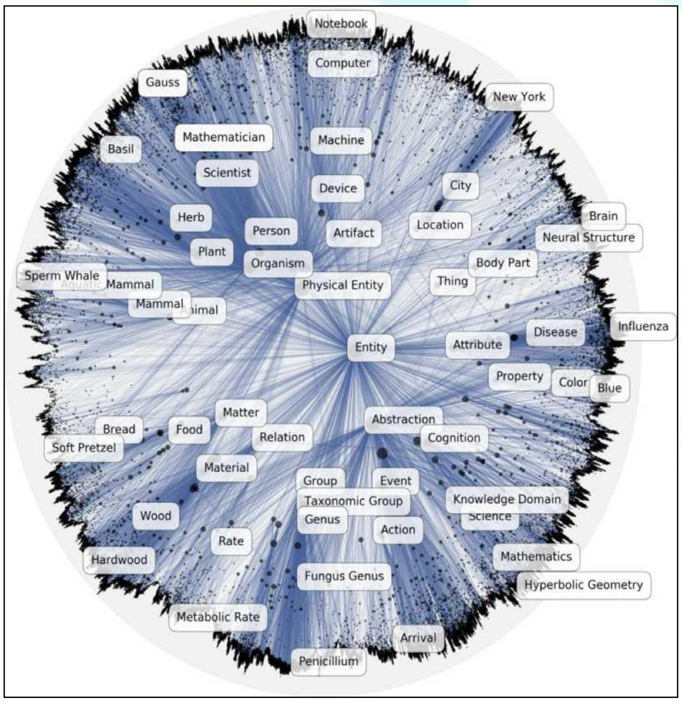


Figure 10: Poincare Embeddings' WordNet[41]

X. ELMo

Embeddings for Language Models (ELMo) is designed to overcome the problems encountered by the Word2Vec and GloVe embeddings. It is defined as the deep contextualized term representation that looks at the whole sentence, then try to gain knowledge about its syntax and semantics afore allocating embeddings to each word[42]. Similar to fastText, ELMo also concentrates on character level, thus exploiting on subwords and gaining the ability to even identifying the previously unidentified words. ELMo is based on bi-directional Long Short-Term Memory(deep neural networks)also called Deep Bi-directional Language Model (biLM) that provides enhanced understanding of language and modelling that can envisage succeeding word in a sentence. A major disadvantage of ELMo is that its final yield is a blend of its inner layers' output[43].

XI. Probabilistic fastText

It is the most novel type of word embedding introduced by Ben Athiwaratkun, Andrew Gordon Wilson and Amina Anandkumar in June 2018[44]. It is named after fastText and works similar too but with additional responsibilities like capturing ambiguous information, getting subwords organization and most importantly the words that are spelled the same but have diverse meanings. For example: the word "fair" could mean a festival or unbiased or even appearance of a person.Hence to understand in which context any particular term is being, Probabilistic fastText uses Gaussian mixture models are used for word representation instead of vectors[45]. Moreover, Probabilistic fastText outclasses the traditional dictionary-based embeddings and fastText in performance both for English language and other foreign languages.

XII. Creating Own Embeddings

It is also possible for a novel or expert data scientist to generate his/ her own word embeddings in case he/ she is not satisfied with the pre-trained embeddings available like GloVe, ELMo or fastText, etc. The procedure is quite simple, which starts with building a vocabulary list of all the distinctive words and their frequency. Then, a dictionary should be prepared, where each distinct word will be matched with a unique id. Then, a model must be defined by specifying myriad parameters like number of dimensions, training algorithm, least count of words to consider while model training and window (utmost space between target word and its close by words). An imperative point of consideration here is that default number of all these parameters is also set. Subsequently, this model can be saved either in ASCII or binary format for future use and finally it can be supply this data in the neural networks to produce word embeddings[46].

XIII. Conclusion

This research paper discusses the most vigorous subject matter in the domain of sentiment analysis and that is word embeddings. In comparison to the conventional embeddings available earlier, some novel embedding methods have surfaced and taken the world with storm.This has caused havoc for the beginners or new-fangled scholarswho are trying to learn and finding which out of plethora of word embedding to use in which situation. This paper is intended to simplify the chaos and present each available embedding technique in as simple way possible describing in detail the working principle, benefits and detriments and also ability to work on numerous natural languages.Finally, it can be concluded that no word embedding method can be confirmed as the best method of all because their performances vary with the dataset; thus one behave well with a dataset and may yield poor results for another.

REFERENCES

[1] M. S. Neethu and R. Rajasree, "Sentiment analysis in twitter using machine learning techniques",in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT*, July 2013, pp. 1-5.
[2] Y. Yin, F. Wei, L. Dong, K. Xu, M. Zhang, andM. Zhou, "Unsupervised word and dependency path embeddings for aspect term extraction", *arXiv preprint arXiv:1605.07843*, 2016.
[3] A. Collomb, C. Costea, D. Joyeux, O. Hasan, O., & Brunie, L,"A study and comparison of sentiment analysis methods for reputation evaluation", *Rapport de recherche RR-LIRIS-2014-002*, 2014.
[4] L. K. W Tan, J. C. Na, Y. L. Theng, and K. Chang, "Phrase-level sentiment polarity classification using rule-based typed dependencies and additional complex phrases consideration", *Journal of Computer Science and Technology*, vol. *27, no.* 3,pp. 650-666, 2012.
[5] S. S. Shwartz, andS. B. David, "*Understanding machine learning: From theory to algorithms",* Cambridge university press, 2014.
[6] W. Y. Zou, R. Socher, D. Cer, and C. D. Manning, "Bilingual word embeddings for phrase-based machine translation",in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Oct. 2013, pp. 1393-1398.
[7] S. Das, A. Dey, A. Pal, N. Roy, "Applications of artificial intelligence in machine learning: review and prospect", *International Journal of Computer Applications*, vol. 115, no. 9, 2015.
[8]C. Molnar. (2020) Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. Retrieved from https://christophm.github.io/interpretable-ml-book/index.html
[9] L. Deng and Y. Liu, *Deep learning in Natural Language Processing*. Springer, 2018.
[10] Y. Li, and T. Yang, "Word embedding for understanding natural language: a survey",in *Guide to Big Data Applications*, pp. 83-104, Springer, Cham, 2018.
[11] Online Article. (2020) Text Embedding Beta. Retrieved from https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/text-embedding
[12] Y. Zhao, S. Huang, X. Dai, J. Zhang, and J. Chen, "Learning word embeddings from dependency relations",in *2014 international conference on asian language processing (IALP)*, pp. 123-127, 2014.
[13] M. Kusner, Y. Sun, N. Kolkin, N and K. Weinberger, "From word embeddings to document distances",in *International conference on machine learning*, pp. 957-966, 2015.
[14] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment embeddings with applications to sentiment analysis", *IEEE transactions on knowledge and data Engineering*, vol. 28, no. 2, pp. 496-509, 2015.
[15] Y. Chen, B. Perozzi, R. Al-Rfou, andS. Skiena, "The expressive power of word embeddings", *arXiv preprint arXiv:1301.3226*, 2013.
[16] C. D. Boom, S. V. Canneyt, T. Demeester, and B. Dhoedt, "Representation learning for very short texts using weighted word embedding aggregation", *Pattern Recognition Letters*, *80*, pp. 150-156, 2016.
[17]H. Heidenreich, (2018), Introduction to Word Embeddings. Retrieved from http://hunterheidenreich.com/blog/intro-to-word-embeddings/
[18] J. H. Paik, "A novel TF-IDF weighting scheme for effective ranking",in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 343-352, 2013.
[19] B. G. Gebre, M. Zampieri, P. Wittenburg, and T. Heskes, "Improving native language identification with tf-idf weighting",in *the 8th NAACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA8)*, pp. 216-223, 2013.
[20] A. Gebejes, and R. Huertas, "Texture characterization based on grey-level co-occurrence matrix", *databases*, vol. 9, no. 10, 2013.
[21] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, "Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence", in *Proceedings of the 10th ACM conference on recommender systems*, pp. 59-66, 2016.
[22] Q. Luo, W. Xu, and J. Guo, "A Study on the CBOW Model's Overfitting and Stability",in *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, pp. 9-12, 2014.
[23] Online Article. (2016) Word Embeddings for Natural Language Processing. Retrieved from http://deep-solutions.net/blog/WordEmbeddings.html
[24] G. Guo, S. Ouyang, F. Yuan, and X. Wang, "Approximating word ranking and negative sampling for word embedding", International Joint Conferences on Artificial Intelligence Organization, 2018.
[25] L. Yang, X. Chen, Z. Liu, and M. Sun, "Improving word representations with document labels", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 863-870, 2017.
[26]Y. Goldberg, and O. Levy,. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method", *arXiv preprint arXiv:1402.3722*, 2014.
[27] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", in *Advances in neural information processing systems*, pp. 3111-3119, 2013.
[28] K. Yoshioka, and H. Dozono, "The classification of the documents based on word2vec and 2-layer self organizing maps", *International Journal of Machine Learning and Computing*, vol. 8, no. 3, pp. 252-255, 2018.
[29] T. Mikolov, W. T. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations",in *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 746-751, 2013.
[30] G. Sidorov, A. Gelbukh, H. G. Adorno, and D. Pinto, "Soft similarity and soft cosine measure: Similarity of features in vector space model", *Computación y Sistemas*, vol. 18, no. 3, 491-504, 2014.
[31] Online Article. (2020). The Synthesis Project: Materials Word Embeddings. Retrieved from https://www.synthesisproject.org/resources
[32] O. Levy, and Y. Goldberg, "Dependency-based word embeddings",in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 302-308, 2014.
[33] X. Rong, "word2vec parameter learning explained", *arXiv preprint arXiv:1411.2738*, 2014.
[34] C. Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning",in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2840-2848, 2017.
[35] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation",in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532-1543, 2014.
[36]J. Pennington, R. Socher, C. D. Manning. (2014) Glove: Global vectors for word representation. Retrieved from https://nlp.stanford.edu/projects/glove/
[37] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext. zip: Compressing text classification models", *arXiv preprint arXiv:1612.03651*, 2016.
[38]I. Santos, N. Nedjah, andL. M. Mourelle, "Sentiment analysis using convolutional neural network with fastText embeddings",in 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), pp. 1-5, 2017.

[39] A. Moffat, and J. Zobel, "Self-indexing inverted files for fast text retrieval", *ACM Transactions on Information Systems (TOIS)*, vol. 14. no. 4, pp. 349-379, 1996.
[40] M. Nickel, and D. Kiela, "Poincaré embeddings for learning hierarchical representations", in *Advances in neural information processing systems*, pp. 6338-6347, 2017.
[41] C. Saedi, A. Branco, J. Rodrigues, and J. Silva, "Wordnet embeddings",in *Proceedings of the third workshop on representation learning for NLP*, pp. 122-131, 2018.
[42] V. Jha, (2018) Insight into Hierarchical Representations through Poincaré Embedding. Retrieved online https://www.techleer.com/articles/478-insight-into-hierarchical-representations-through-poincare-embedding/
[43] I. Chami, A. Wolf, F. Sala, and C. Ré, "Low-dimensional knowledge graph embeddings via hyperbolic rotations",in *Graph Representation Learning NeurIPS 2019 Workshop*, 2019
[44] B. Athiwaratkun, A. G. Wilson, and A. Anandkumar, "Probabilistic fasttext for multi-sense word embeddings", *arXiv preprint arXiv:1806.02901*, 2018
[45] J. Choi, and S. W. Lee, "Improving FastText with inverse document frequency of subwords", *Pattern Recognition Letters*, 2020.
[46] J. Brownlee, "Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems", Machine Learning Mastery, pp. 133-143, 2017.