Exploration of word embedding methods and considerations about informativeness

Eddie Conti
Faculty of Mathematics and Computer Science
Universitat de Barcelona
Barcelona, 08007
`econtico8@alumnes.ub.edu`

Abstract

This study addresses the challenge of representing words or texts as real-valued vectors, commonly known as word embeddings. We delve into various techniques employed for this purpose, specifically focusing on Bag of Words, TF-IDF Word2Vec, GloVe and Latent Semantic Analysis (LSA). Some of them rely on context and word interaction, as GloVe and Word2Vec, while others prioritize, as TF-IDF, the relevance of individual words. A key point is dimensionality reduction, therefore we analyze LSA method which provides a tool to achieve it. Finally, we present theoretical considerations to assess the informativeness of an embedding for text classification in the context of a preprocessed vocabulary.

1 Word embedding methods: an overview of the main techniques

In this first section we delve our attention to the main embedding methods. Word embedding is a representation of a word, or a set of word, typically using a real-valued vector that encodes some features of it. The representing vector has tens or hundreds of dimensions, which means that we are embedding words in a "large" dimensional space. By means of that, we expect that words that are closer in the vector space are similar in meaning. Consequently, the main idea is to be able to define a metric on the set of words.

1.1 Bag of Words

It is a simple and flexible technique to extract features from a text. Essentially, a bag of words is a representation of text that describes the occurrence of words within a document. First of all, the document must be tokenized into sentences and every sentence tokenized into words; then all the punctuation and stop word[1] is removed and all the words are converted in lower case; finally each word is scored. The scoring part is the most delicate part of the process: usually only the unique words are taken and the word is scored by marking the presence in the sentence with 1 or 0 or by counting how many times it appears. To be more clear, let us consider the following example:

"This is a first sentence"

"This is a second example of sentence"

The unique words, or vocabulary, is {"This","is", "a", "first", "sentence", "second", "example", "of" }. Hence, the vector representing the first sentence is

$$[1, 1, 1, 1, 1, 0, 0, 0].$$

The main problems with this simple technique are the dimensionality of the embedding and the lack of attention to global context. In the next pages we present methods that solve these problems: stemming algorithms, dimensionality reduction and Neural Network-based Methods.

[1]In simple terms, any word that does not add information. Examples of stop words are "the", "is", "and", "or".

1.2 Term frequency-inverse document frequency (TF-IDF)

TF-IDF is a common machine learning algorithm used for word embedding for text. The main goal of this technique is to measure the importance of a word in a document or in various documents. The method is divided in two parts: the former, TF, is the relative frequency of a term t in document d. In mathematical terms it is defined as

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{1}$$

where $f_{t,d}$ is the number of times the term t appears in document d, over the total amount of words in document d. Even though it is the most common way of defining TF, there are other functions that can be used:

$$tf(t,d) = f_{t,d}; \qquad tf(t,d) = \log(1 + f_{t,d}); \qquad tf(t,d) = \frac{1}{2} + \frac{1}{2}\frac{f_{t,d}}{\max_{t' \in d} f_{t',d}}.$$

The latter, IDF, measure how much information the word provides:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}, \tag{2}$$

where N is the total number of documents, while the quotient represents the number of documents containing that word amon the set of documents D. As a consequence, the fraction in (2) is always greater or equal than 1, and approaches to 0 when the word is present in all the documents. We can add 1 to prevent division by zero (for example if the vocabulary contains words that do not appear in any document):

$$idf(t,D) = \log \frac{1+N}{1+|\{d \in D : t \in d\}|}.$$

Finally, combining (1) and (2) we define tf-idf as it follows:

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D).$$

The vector representing the word is therefore, assuming $D = \cup_{i=1}^{n} d_i$,

$$[tf(t,d_1) \cdot idf(t,D), \ldots, tf(t,d_n) \cdot idf(t,D)].$$

A word is high weighted by tf-idf, if it has a high frequency in a document and a low document frequency in the set of documents. This means that tf-idf tends to devalue words that are common in all the documents, since they provide no additional information. Although this strategy provides a better insight into the general structure of words, the challenges persist on two fronts: words being treated as distinct entities and the high dimensionality of the embedding.

1.3 Word2Vec

Word2Vec is a neural network-based method for generating vector representations of words that capture their semantic meaning and relationships. Word2Vec has two neural network-based variants: Continuous Bag of Words (CBOW) and Skip-gram.
The CBOW model takes a window of surrounding words as input and tries to predict the target word in the center of the window. The model is trained on a large text dataset and learns to predict the target word based on the patterns it observes in the input data.
First of all each word is represented with one hot encoding, namely a vector of the form

$$[0, \ldots, 0, 1, 0, \ldots, 0],$$

with length the cardinality of the vocabulary V, that will be denoted $|V|$. We decide in advance dimensionality of the embedding n, so that each word will be a vector in $\mathbb{R}^n$. Now, since we want to take into consideration the context, the input will be a vector with some 1 depending on the window. For example in the sentence "She is a great dancer" if we want to predict the word "is" with window 1, then we have to deal just with the word "She" and "a" since we have to consider words with distance at most 1 from the center "is". The input, therefore, will be $[1, 0, 1, 0, 0]$. The aim is to predict the word "is", therefore we would like the output to be $[0, 1, 0, 0, 0]$.
We are now ready to present the architecture: we have the input layer in which we have a vector (usually represented as more vector to highlight the words used) of dimension the cardinality of the

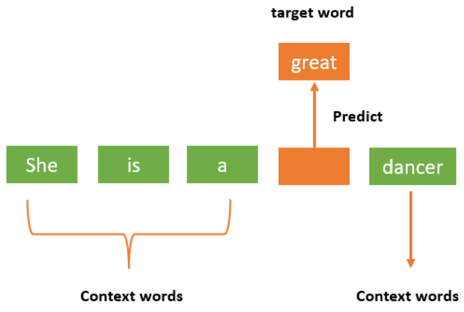

Figure 1: Visual interpretation of input, target and window

vocabulary, an hidden layer with n neurons used to represent the word, an output layer with the same size of the vocabulary. The final result is obtained using a Softmax function:

$$P(\omega_a|\omega_b) = \frac{\exp(\omega_a^T\omega_b)}{\sum_{j\in V} exp(\omega_j^T\omega_b)}.$$

where ω_a, ω_b represents two words and we are evaluating the probability of word a given word b. The key idea is that the product $\omega_a^T\omega_b$ encodes the correlation between the two words and so we are weighting this correlation over all the possible correlation with word ω_b. In the case of context words, since we want the probability of a word given a set of words, we have to take into consideration

$$\prod_{-m\leq j\leq m} P(\omega_a|\omega_{a+j})$$

if we use a window of size m. In simple terms, if we aim at representing the $j-$th word in the vocabulary $w_j = [0,\ldots,0,1,0,\ldots,0]$, the output will be a vector of the form $[x_1,\ldots,x_{|V|}]$ that we want as close as possible to w_j. Hence we train the model in order to achieve this goal and adjust the entries of the output. The function we aim at maximizing is

$$\prod_{t=1}^{|V|}\prod_{-m\leq j\leq m} P(\omega_t|\omega_{t+j}).$$

For modelling reason and in order to switch to a minimization problem, we minimize

$$-\sum_{t=1}^{|V|}\sum_{-m\leq j\leq m} \log(P(\omega_t|\omega_{t+j})).$$

On the other hand, Skip-Gram operates in the opposite direction since given a word as input it predicts words in a certain range before and after the current word (i.e. the context). Although the idea behind and the architecture is the same, empirical tests showed that Skip-Gram works well with a small amount of the training data and better represents rare words, while CBOW is faster and has a better accuracy for frequent words.
Nevertheless, the main problem with these methods is the inability to capture the statistical information at a global level since they are not considering the count or co-occurrence of words (how frequently two words appear together) in a global context, instead they work "locally".

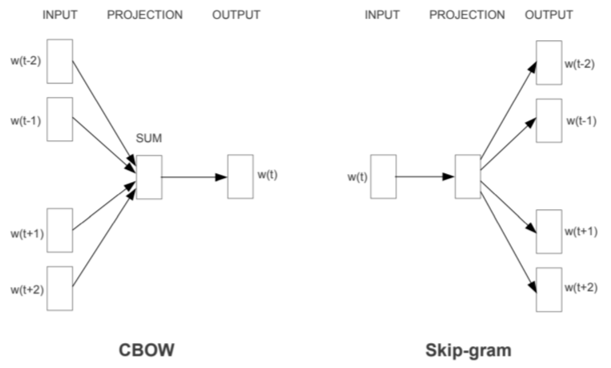

Figure 2: CBOW and Skip-Gram architectures

1.4 GloVe

GloVe is a co-occurrence-based model that also captures the global context by using conditional probability. Before proceeding we have to define some notation. Let the matrix of word-word co-occurrence counts be denoted by X, whose entries X_{ij} tabulate the number of times word j occurs in the context of word i. Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of word i. Finally, let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that word j appear in the context of word i.
Let us introduce the following example to explain the idea behind the model.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

As can be seen "solid" is more related to the word "ice" than "steam" and therefore ratio is 8.9. On the other hand, "ice" and "steam" are not related to "fashion", but if we consider the standalone probability values for $P(fashion|ice)$ or $P(fashion|steam)$, it will not help us to infer the lack of relationship. Therefore, the idea is to consider the ratio of the co-occurrence probabilities. Let us deepen the insight introducing the mathematics of this model: we define F as the function that model the ratio of probabilities relationship

$$F(\omega_i, \omega_j, \tilde{\omega_k}) = \frac{P_{ik}}{P_{jk}},$$

where ω_i, ω_j are the context words, $\tilde{\omega_k}$ is the out of context word all of them in vectorial form and P_{ik}, P_{jk} are the probabilities of the co-occurrence. At this point, we have on LHS a function F whose arguments are vectors, while on the other side we have a scalar. According to the original paper [1] in order to convert vectors in scalars we consider the following dot product

$$F((\omega_i - \omega_j)^T \cdot \tilde{\omega_k}) = \frac{P_{ik}}{P_{jk}}. \tag{3}$$

By doing that we are still working with the same words but now in the LHS function F has a scalar argument which itself is related to a similarity measure. Indeed, the dot product of two vectors contains the $\cos(\theta)$ which is a measure of how dependent two vectors are. Finally, we consider the term $(\omega_i - \omega_j)$ because we can perform the dot product with two vectors and this vector difference encodes the analogy between the two words.
Now, the distinction between a word and a context word is arbitrary. This symmetry property can be restored asking the function F to be an homomorphism between the group $(\mathbb{R}, +)$ and the group $(\mathbb{R}_{>0}, \times)$, such that $F(a + b) = F(a)F(b)$. In simple terms, an homomorphism establish

the equivalence of two algebraic structures at the cost of adapting the operation. For example the function $f(x) = e^x$ defines an homomorphism between $(\mathbb{R}, +)$ and the group $(\mathbb{R}_{>0}, \times)$: indeed

$$f(x + y) = e^{x+y} = e^x \times e^y = f(x) \times f(y)$$

and so it transform $+$ into $\times$ by using the exponentiation. In our scenario, by (3)

$$F((\omega_i - \omega_j)^T \cdot \tilde{\omega_k}) = F((\omega_i^T \tilde{\omega_k} - \omega_j^T \cdot \tilde{\omega_k}) = \frac{F(\omega_i^T \tilde{\omega_k})}{F(\omega_j^T \tilde{\omega_k})} \tag{4}$$

Hence, combining (3) and (4) yields to

$$F(\omega_i^T \tilde{\omega_k}) = P_{ik} = \frac{X_{ik}}{X_i}. \tag{5}$$

From (4) we observe that a solution is $F = exp$ since it is an homomorphism and satisfies $F(a-b) = F(a)/F(b)$. Therefore, from (5) we obtain

$$\omega_i^T \tilde{\omega_k} = \log(P_{ik}) = \log(X_{ik}) - \log(X_i).$$

The latter equation, shows the expected symmetry if not for the $\log(X_i)$ because by changing the roles we would have $\log(X_{ik}) - \log(X_k)$. However, this term is independent of k so it can be absorbed into a bias b_i for ω_i. Finally, adding an additional bias $\tilde{b}_k$ for $\tilde{\omega_k}$ restores the symmetry

$$\omega_i^T \tilde{\omega_k} + b_i + \tilde{b}_k = \log(X_{ik}).$$

We are now ready to define the cost function

$$J = \sum_{i,j}^{|V|} f(X_{ij})[\omega_i^T \tilde{\omega_j} + b_i + \tilde{b}_j - \log(X_{ij})]^2$$

where the term in square brackets is similar to the linear regression term $(y - \hat{y})^2$ and the weighting term $f(X_{ij})$ is a fancy term to add freedom degree to the model. Indeed, by changing f we obtain a different loss function, so it is possible to choose f adapting to the data.
Furthermore, f should obey some rules as it should vanish as $x \to 0$ fast enough that the $\lim_{x\to 0} f(x) \log(x)^2$ is finite and $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

1.5 Dimensionality reduction: Latent Semantic Analysis and Stemming

Latent Semantic Analysis is a technique that aims to describe relationships between a set of documents and the terms they contain by generating a set of concepts related to the documents and terms. The starting point is a document-term matrix (typically the one derived from TF-IDF), and the idea is to perform rank reduction to decrease the dimensionality of the embedding. In light of Eckart–Young theorem, given a matrix M, the SVD factorization allows us to obtain the optimal approximation, in terms of the Frobenius norm, represented by a matrix $\tilde{M}$ of rank r. This implies that $\tilde{M}$ provides the most accurate approximation of rank r for the given matrix M.
Hence, if $M = U\Sigma V^T$, then $\tilde{M} = U\tilde{\Sigma}V^T$, where $\tilde{\Sigma}$ consists of the r largest singular values (the others are replaced with 0).
Let X be a matrix where the element x_{ij} describes the occurrence of term i in document j. A word is represented by a row of the matrix

$$w_i = [x_{i,1}, \dots, x_{i,n}].$$

If we consider the SVD factorization of X of rank r we obtain that

$$X_r = U\tilde{\Sigma}V^T, \tag{6}$$

with $\tilde{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0)$. As a consequence of (6), the word w_i is now represented by a vector with r entries. These new dimensions do not relate to any comprehensible concepts, but they are related to the relationship between documents and words. To understand this, since $X = U\Sigma V^T$ then $XX^T = U\Sigma^2 U^T$ and $X^T X = V\Sigma^2 V^T$. Matrices XX^T and $X^T X$ are diagonalizable, therefore U is made of the eigenvectors of XX^T, V of the eigenvectors of $X^T X$ and the singular values are the square root of the eigenvalues of XX^T. Furthermore, the dot product $w_i^T \cdot w_j$ represents the correlation between the terms over the set of documents and all this quantities are encoded in XX^T. Similarly, the matrix $X^T X$ contains the dot products between all the document vectors, giving their correlation over the terms. To sum up, from (6) the new word representation is a linear combination of entries of U,V and singular values, hence its entries encodes the connection among documents and words.

In the context of dimensionality reduction but shifted to the preprocessing part, stemming algorithms aim at reducing words into a root/base word. For instance, the word "appearing", "appearance", "appeared" can be stemmed to "appear". One of the most important technique is the Porter stemming algorithm which works for English text. In simple terms, each word is represented in the form

$$[C](VC)^m[V]$$

where V and C respectively stand for a list of one or more consecutive vowels/consonants, while m is the measure of the words and represents VC repeated m times. The term $[C]$ and $[V]$ are optional. From here we define a list of rules of the form $(condition)$ $S_1 \to S_2$ so that if the stem before S_1 satisfies the given condition, S_1 is replaced by S_2. For example,

$$(m > 1) \text{ 'ement'} \to \text{' '}$$

means that we replace 'ement' with nothing for a word followed by 'ement'. The rule applied to word "replacement" will give as output "replace". Then the idea is after defining a list of rules, to apply all of the to the words and collect the results. It is remarkable that stemming may produce a non-word as the root form.

2 Assessing the informativeness of an embedding

In this section we aim to define the notion of informativeness of an embedding for text classification. As the previous pages highlighted, once we choose the model to transform words into vectors, we end up with a matrix representation, that can be sparse or dense, of the text. In rigorous terms, if we have a vocabulary V of size $|V| = m$, the embedding is a function

$$E\colon V \to \mathcal{M}_{\mathbb{R}}(m, n)$$

where $\mathcal{M}_{\mathbb{R}}(m, n)$ is the space of real matrices of dimension $m \times n$. Thereby, $E(\omega) \in \mathbb{R}^n$, namely a word is represented by a row vector with n entries. We assume that the vocabulary has been preprocessed according to the dataset. The operation can include techniques as lemmatization that works similarly to stemming techniques but it takes into account the context of the word or stemming algorithms.
If we let A the matrix representing the vocabulary, how can we define the goodness of the embedding? Since our aim is to classify text, we expect the extracted features to naturally incorporate a word classification. In simple terms some entries should have a relevant weight for some words, while others have negligible weight. Let us assume, without loss of generality, that the entries $a_{ij} \geq 0$. In this way, given a word

$$w_i = [a_{i,1}, \dots, a_{i,n}],$$

then we expect that there exists a subset of indexes $S \subset \{1, \dots, n\}$ such that $a_{i,j} <<$ if $j \in \{1, \dots, n\} \setminus S$. Also if the n entries are actually encoding the structure of the vocabulary, similar word should be represented by the same weights. As a consequence, if we collect similar words together and their relevant weights, the matrix should present a block structure. In mathematical terms, there exist permutation matrices P,Q, such that PAQ has a block form:

$$PAQ = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_l \end{bmatrix}$$

where l is the number of blocks and the rest of entries are less that a fixed value $\epsilon > 0$. This structure does not hold for every vocabulary: if for example all the words share the same meaning, all of them can be represented by a a few entries, and so the structure of the matrix is

$$\left[\begin{array}{c|c} A_1 & A_2 \end{array}\right]$$

where entries in A_2 are small. However, for a large and general text this is unlikely. Geometrically this diagonal block structure means that the n axis transform group of words in "almost orthogonal" groups of vector and so they both cluster and separate the words.
The definition of informativeness should take into consideration the following:

- a very low number of blocks means that we have not captured the structure of the data;
- too many blocks may be misleading and stand for overfitting for a general text;
- the trade-off between m and n.

Hence, we can define a set of functions $f(l, m, n)$ that meet the requirements by asking. We can split the function $f(l, m, n)$ in two parts:

$$f(l, m, n) = h(l, n) + g(m, n),$$

where h is related to the blocks while g related to the trade-off between m and n and both function are positive. An example of $h(l, n)$ could be a function that if $l/n \approx 0$ then this means that we do not have enough blocks and so $h(l, n)$ should be small, the same when $l/n \approx 1$ since we have too many blocks. One example of this functions is

$$h(l,n) = \begin{cases} \alpha \cdot exp(-\frac{1}{1-(2l/n-1)^2}) & \text{if } l/n \in (0,1), \\ 0 & \text{otherwise.} \end{cases}$$

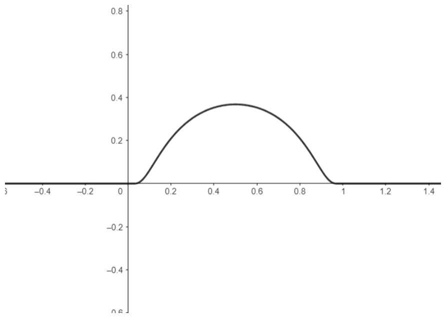

Figure 3: Graph of $h(l, n)$ for $\alpha = 1$

The same function can be used for $g(m, n)$, with a new parameter β, in the sense that when $n/m \approx 0$ then we are using a few features to describe the set and so we are not getting enough information, while $n/m \approx 1$ means that we are using all the dataset to describe the dataset itself. In this way, we aim at model the trade-off between m and n. It is evident that there exist an infinite number of functions observing the three requirements and the "right function" (even the literature shows that there is no unique and general way to define the informativeness of and embedding) to assess the the goodness of the embedding depends on the data and the purposes of classification. Another example of function is

$$h(l,n) = \begin{cases} \alpha \cdot cos(\pi(l/n - 1/2)) & \text{if } l/n \in (0,1), \\ 0 & \text{otherwise.} \end{cases}$$

By adjusting the α parameter in $h(l, n)$ and the similar β parameter in $g(m, n)$ we can give more value to some of the requirements. The general function does not exist but it is possible to use this kind of each (and any other that meet the requirements) in order to obtain a general idea of the informativeness of the embedding for text classification.

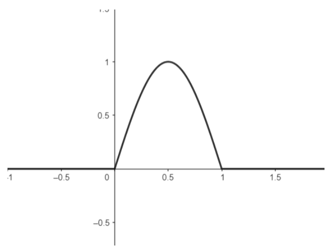

Figure 4: Graph of $h(l, n)$ for $\alpha = 1$

References

[1] Jeffrey P., Richard S., Christopher D. M. (2014) *GloVe: Global Vectors for Word Representation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[2] Tomas M., Kai C., Greg C., Jeffrey D. (2013) *Efficient Estimation of Word Representations in Vector Space*, ICLR Workshop Papers.

[3] Berger A., Lafferty J. (1999) *Information retrieval as statistical translation*, Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval, pages 222–229.

[4] Mohammad T. P., Jose C.-C. (2021) *Embeddings in Natural Language Processing Theory and Advances in Vector Representation of Meaning*, Morgan & Claypool publishers

[5] Austin J. B., Tingting M. S. A., John Y. G. (2017) *Quantifying the Informativeness of Similarity Measurements*, Journal of Machine Learning Research, pages 1-61