MO434 - Deep Learning
Text Representation

Alexandre Xavier Falcão

Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

Text Representation

We are interested in text representations for machine learning.

- ML algorithms require numeric representations (feature vectors, embeddings) as input.

Text Representation

We are interested in text representations for machine learning.

- ML algorithms require numeric representations (feature vectors, embeddings) as input.
- In this context, feature engineering plays a crucial role to convert text into numbers.

Text Representation

We are interested in text representations for machine learning.

- ML algorithms require numeric representations (feature vectors, embeddings) as input.
- In this context, feature engineering plays a crucial role to convert text into numbers.
- Even if you use a model that learns features from textual data, you still need to understand the concepts behind this process.

Text Representation

We are interested in text representations for machine learning.

- ML algorithms require numeric representations (feature vectors, embeddings) as input.
- In this context, feature engineering plays a crucial role to convert text into numbers.
- Even if you use a model that learns features from textual data, you still need to understand the concepts behind this process.
- This module then covers some popular feature engineering techniques.

Agenda

- Concept of vector space model.
- Bag of words, Bag of N-Grams and TF-IDF.
- Word2Vec: CBOW and Skip-Gram.
- GloVe and FastText.

Vector space model

Let $\mathcal{D}$ be a set of documents (corpus). One can create a vector space (vocabulary) $VS = \{W_1, W_2, \ldots, W_n\}$ as the set of all words present in all documents from $\mathcal{D}$.

Vector space model

Let $\mathcal{D}$ be a set of documents (corpus). One can create a vector space (vocabulary) $VS = \{W_1, W_2, \ldots, W_n\}$ as the set of all words present in all documents from $\mathcal{D}$.

A document D (text with one or more sentences) can then be numerically represented by a weight vector $D = (w_1, w_2, \ldots, w_n)$ where w_i is a value related to the presence of the word W_i in document D.

Vector space model

Let $\mathcal{D}$ be a set of documents (corpus). One can create a vector space (vocabulary) $VS = \{W_1, W_2, \ldots, W_n\}$ as the set of all words present in all documents from $\mathcal{D}$.

A document D (text with one or more sentences) can then be numerically represented by a weight vector $D = (w_1, w_2, \ldots, w_n)$ where w_i is a value related to the presence of the word W_i in document D.

For instance, $w_i \in \{0, 1\}$ may indicate absence/presence of W_i in D (one-hot key) or w_i may indicate the number of occurrences (frequency) of W_i in D.

Vector space model

Let $\mathcal{D}$ be a set of documents (corpus). One can create a vector space (vocabulary) $VS = \{W_1, W_2, \ldots, W_n\}$ as the set of all words present in all documents from $\mathcal{D}$.

A document D (text with one or more sentences) can then be numerically represented by a weight vector $D = (w_1, w_2, \ldots, w_n)$ where w_i is a value related to the presence of the word W_i in document D.

For instance, $w_i \in \{0, 1\}$ may indicate absence/presence of W_i in D (one-hot key) or w_i may indicate the number of occurrences (frequency) of W_i in D.

This concept is adopted in feature engineering techniques.

- Bag of words – it creates a sparse feature vector with the number w_i of occurrences of the word (term) W_i in D.

Traditional feature engineering techniques

- Bag of words – it creates a sparse feature vector with the number w_i of occurrences of the word (term) W_i in D.

- Bag of N-Grams – it creates a sparse feature vector with the frequency of sequences with N words in D.

Traditional feature engineering techniques

- Bag of words – it creates a sparse feature vector with the number w_i of occurrences of the word (term) W_i in D.

- Bag of N-Grams – it creates a sparse feature vector with the frequency of sequences with N words in D.

- TF-IDF – it accounts for relevant words that are not very frequent in the documents.

$$
\begin{aligned}
w_i &= tf(W_i) \times idf(W_i), \\
idf(W_i) &= 1 + \log \frac{|\mathcal{D}|}{1 + df(W_i)},
\end{aligned}
$$

  where $tf(W_i)$ is the term frequency of W_i in D (w_i in BOW), $idf(W_i)$ is the inverse document frequency with $df(W_i)$ being the number of documents in $\mathcal{D}$ in which W_i occurs.

Traditional feature engineering techniques

Let's see the following notebook with traditional feature engineering techniques (TRADITIONAL FEATURE ENGINEERING).

Advanced feature engineering techniques

Traditional techniques generate long and sparse feature vectors. Advanced methods can create dense embeddings, considerably shorter, by exploring unsupervised neural networks and capturing contextual and semantic similarity. We will see the following examples.

Advanced feature engineering techniques

Traditional techniques generate long and sparse feature vectors. Advanced methods can create dense embeddings, considerably shorter, by exploring unsupervised neural networks and capturing contextual and semantic similarity. We will see the following examples.

- Word2Vec – it is divided into two strategies:
  - Skip-Gram and
  - Continous-Bag-Of-Words (CBOW).
- GloVe.
- FastText.

Let $W_{i-k}, W_{i-k+1}, \ldots, W_{i+k-1}, W_{i+k}$ be the surrounding words of a given central word W_i within an observation window of size $2k+1$ in a document D of our vocabulary $\mathcal{D}$.

Let $W_{i-k}, W_{i-k+1}, \ldots, W_{i+k-1}, W_{i+k}$ be the surrounding words of a given central word W_i within an observation window of size $2k+1$ in a document D of our vocabulary $\mathcal{D}$.

For instance, $D =$"the brown fox jumped over the lazy dog", $W_i =$"fox", $k = 1$, $W_{i-1} =$"brown" and $W_{i+1}=$"jumped".

Let $W_{i-k}, W_{i-k+1}, \ldots, W_{i+k-1}, W_{i+k}$ be the surrounding words of a given central word W_i within an observation window of size $2k+1$ in a document D of our vocabulary $\mathcal{D}$.

For instance, $D =$"the brown fox jumped over the lazy dog", $W_i =$"fox", $k = 1$, $W_{i-1} =$"brown" and $W_{i+1}=$"jumped".

A Skip-Gram model learns to predict the surrounding words, brown and jumped, from the input word, fox.

Let $W_{i-k}, W_{i-k+1}, \ldots, W_{i+k-1}, W_{i+k}$ be the surrounding words of a given central word W_i within an observation window of size $2k+1$ in a document D of our vocabulary $\mathcal{D}$.

For instance, $D =$"the brown fox jumped over the lazy dog", $W_i =$"fox", $k = 1$, $W_{i-1} =$"brown" and $W_{i+1}=$"jumped".

A Skip-Gram model learns to predict the surrounding words, brown and jumped, from the input word, fox.

By sliding that window along all documents, a neural network is trained with $(\mathit{source}, \mathit{target})$ pairs: (W_i, W_{i-1}) and (W_i, W_{i+1}).

Word2Vec

Let $W_{i-k}, W_{i-k+1}, \ldots, W_{i+k-1}, W_{i+k}$ be the surrounding words of a given central word W_i within an observation window of size $2k+1$ in a document D of our vocabulary $\mathcal{D}$.

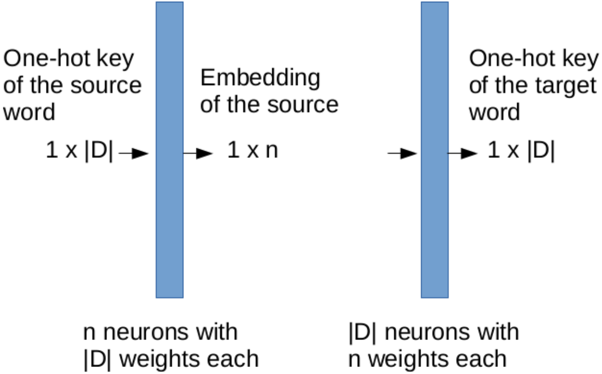
For instance, $D =$"the brown fox jumped over the lazy dog", $W_i =$"fox", $k = 1$, $W_{i-1} =$"brown" and $W_{i+1}=$"jumped".

A Skip-Gram model learns to predict the surrounding words, brown and jumped, from the input word, fox.

By sliding that window along all documents, a neural network is trained with $(source, target)$ pairs: (W_i, W_{i-1}) and (W_i, W_{i+1}).

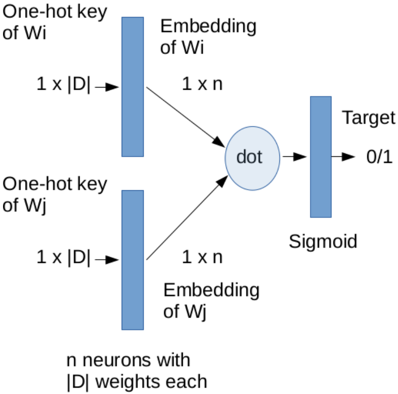
A CBOW model learns to predict W_i from the input $[W_{i-1}, W_{i+1}]$ – i.e., it predicts the central word from the surrounding ones.

Skip-Gram: the input is the one-hot key of a central word (source) W_i and a hidden layer with n neurons with no activation transforms it into an embedding $1 \times n$ for W_i, while the output layer with $|\mathcal{D}|$ neurons and softmax creates the one-hot key of the surrounding word used as target.

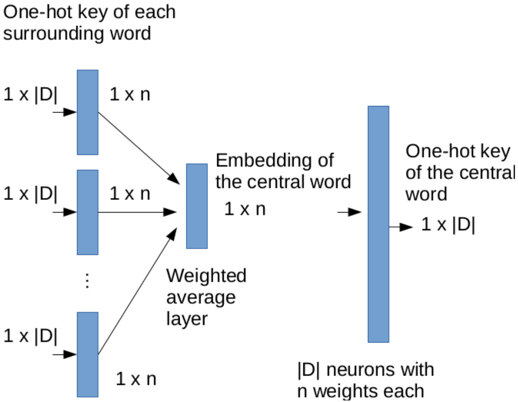


Skip-Gram model

Skip-Gram: another possibility is to input pairs $[W_i, W_j]$ of words with a target label equal to 1, when W_j is a surrounding word of W_i, and 0, otherwise. The one-hot keys of W_i and W_j pass through one hidden layer each and the inner product between their $1 \times n$ embeddings passes through a sigmoid to estimate the target.



CBOW: The one-hot keys of all surrounding words pass through a hidden layer with no activation each, creating one embedding per surrounding word. Those embeddings are averaged, creating an embedding $1 \times n$ for W_i and the output layer with softmax transforms it into the one-hot key of W_i.



GloVe

- Note that Skip-Gram and CBOW explore context (sequence of surrounding words) and semantics, when they relate surrounding words with a central word.

GloVe

- Note that Skip-Gram and CBOW explore context (sequence of surrounding words) and semantics, when they relate surrounding words with a central word.

- Glove first creates a huge word-context co-occurrence matrix WC consisting of (word,context) pairs, in which the elements store the frequency a word occurs with the context (one or all surrounding words).

GloVe

- Note that Skip-Gram and CBOW explore context (sequence of surrounding words) and semantics, when they relate surrounding words with a central word.

- Glove first creates a huge word-context co-occurrence matrix WC consisting of (word,context) pairs, in which the elements store the frequency a word occurs with the context (one or all surrounding words).

- The idea is to apply matrix factorization to compute $WC = WF \times FC$, where WF is a word-feature matrix and FC is a feature-context matrix.

- Note that Skip-Gram and CBOW explore context (sequence of surrounding words) and semantics, when they relate surrounding words with a central word.

- Glove first creates a huge word-context co-occurrence matrix WC consisting of (word,context) pairs, in which the elements store the frequency a word occurs with the context (one or all surrounding words).

- The idea is to apply matrix factorization to compute $WC = WF \times FC$, where WF is a word-feature matrix and FC is a feature-context matrix.

- The SGD algorithm is used to minimize the error and, finally, WF provides the embeddings for all words in $\mathcal{D}$.

FastText

- FastText uses Word2Vec models, but it adds to the representation of a word the representation of its n-grams.

FastText

- FastText uses Word2Vec models, but it adds to the representation of a word the representation of its n-grams.

- For instance, for $n = 3$, the word $< where >$ is represented by itself and its subwords $< wh, whe, her, ere, re >$.

FastText

- FastText uses Word2Vec models, but it adds to the representation of a word the representation of its n-grams.
- For instance, for $n = 3$, the word $< where >$ is represented by itself and its subwords $< wh, whe, her, ere, re >$.
- The boundary symbols "$<$" and "$>$" are used to distinguish the word $< her >$ from the subword "her" in $< wh, whe, her, ere, re >$.

FastText

- FastText uses Word2Vec models, but it adds to the representation of a word the representation of its n-grams.

- For instance, for $n = 3$, the word $< where >$ is represented by itself and its subwords $< wh, whe, her, ere, re >$.

- The boundary symbols "$<$" and "$>$" are used to distinguish the word $< her >$ from the subword "*her*" in $< wh, whe, her, ere, re >$.

- This helps preserve the meaning of shorter words that may show up as n-grams of other words. Inherently, this also allows you to capture meaning for suffixes/prefixes.

FastText

- FastText uses Word2Vec models, but it adds to the representation of a word the representation of its n-grams.

- For instance, for $n = 3$, the word $< where >$ is represented by itself and its subwords $< wh, whe, her, ere, re >$.

- The boundary symbols "$<$" and "$>$" are used to distinguish the word $< her >$ from the subword "her" in $< wh, whe, her, ere, re >$.

- This helps preserve the meaning of shorter words that may show up as n-grams of other words. Inherently, this also allows you to capture meaning for suffixes/prefixes.

Finally, advanced feature engineering techniques are illustrated in (ADVANCED FEATURE ENGINEERING).