Bag of Words vs Word2Vec vs N-Gram

Machine Learning 2024-25 Course Activity

Furno Francesco - francesco.furno@studenti.unipd.it - 2139507

January 20, 2025

Introduction

In this activity we will compare three common models used in Natural Language Processing (NLP): the Bag of Words model, the Word2Vec model and the N-Gram model.

Bag of Words

The Bag of Words model is a simple model in NLP that represents text as a bag of words, disregarding grammar and word order. The Bag of Words model is based on the assumption that the order of words in a text does not matter and that the presence of words in a text is more important than their order.

It represents a text as a vector of word counts, where each element in the vector corresponds to the frequency of a word in the text.

For example, the sentence *"The pen is on the table"* could be represented as
$BoW = \{\text{The: 2, pen: 1, is: 1, on: 1, table: 1}\}$
assuming a vector of 8-dimensionality, it corresponds to
$BoW = [0, 2, 1, 1, 0, 1, 0, 1]$ in the Bag of Words model, where each element corresponds to the frequency of a word in the sentence.

Another possible representation for Bag of Words is the binary representation, where each element in the vector is 1 if the word is present in the text and 0 otherwise (one hot encoding). This representation is useful when the frequency of words is not important, but only their presence in the text. However, one hot encoded vectors are orthogonal, so they do not capture any relationship between words. In order to capture relationships between words, we can use word embeddings, such as Word2Vec.

Pros

- Simplicity: The Bag of Words model is simple and easy to implement
- Flexibility: The Bag of Words model can be used with different types of text data, such as documents, tweets, and reviews

Cons

- Sparsity: the representation is sparse, with many zero entries due to the high dimensionality of the vocabulary.
- No semantic relationships: it does not capture relationships between words, for example *"happy"* and *"joyful"* are treated as unrelated.
- Ignores word order: It loses all syntactic and positional information, which can be crucial for certain tasks.

Word2Vec

The Word2Vec model is a popular word embedding model that is used to represent words as dense vectors in a continuous vector space. Word embeddings are distributed representations of words that capture semantic and syntactic relationships between words.
Word2Vec is trained on large text corpora using neural networks to learn word embeddings that encode semantic relationships between words. The Word2Vec model is based on the distributional hypothesis, which states that words that occur in similar contexts tend to have similar meanings.

By using this concept of context Word2Vec can learn word embeddings in two different ways:

- CBOW (Continuous Bag Of Words) in which the network uses the context to predict a target word.
- Skip-gram in which it uses the target word to predict a target context.

Pros

- Captures semantic relationships: Word2Vec encodes complex semantic relationships between words, allowing for tasks such as analogy solving (e.g., "king" - "man" + "woman" = "queen").
- Dense representations: Word2Vec creates dense, low-dimensional vectors, reducing memory and computational costs for downstream tasks.
- Generalization: Word2Vec embeddings can represent unseen words or contexts effectively if the training corpus is large and diverse.
- Flexibility: Word2Vec embeddings are versatile and can be applied to many NLP tasks, such as similarity measurement, text classification, and clustering.

Cons

- Requires training: Generating embeddings requires substantial computational resources and a large, high-quality corpus.
- Loss of local information: Word2Vec does not explicitly retain information about word order or syntactic structure.
- Interpretability: Dense vectors are less intuitive to understand compared to sparse representations like Bag of Words.

N-Gram

The N-Gram model is a statistical language model that is used to predict the next word in a sequence of words. A statistical language model is a model that assigns probabilities to sequences of words.
N-Gram models define probabilities for sequences of words. There are different types of N-Gram models, such as:

- unigram: a model that predicts the next word based on the current word
- bigram: a model that predicts the next word based on the current word and the previous word
- trigram: a model that predicts the next word based on the current word and the two previous words

Note that here we are considering N-Gram models with words, but they can also be used with characters or other units of text.

The N-Gram model is based on the Markov assumption, which states that the probability of a word depends only on the previous N-1 words.

The probability of a word given the previous N-1 words is calculated using the chain rule of probability. The chain rule of probability states that the probability of a sequence of events is the product of the probabilities of each event in the sequence.
In the case of a trigram model, the probability of a word w_i given the two previous words is calculated as follows:

$$P(w_i) = \prod_{i=1}^{N} P(w_i \mid w_{i-2:i-1})$$

For example, with the trigram *"Today is hot"*:

$$P(\text{Today is hot}) = P(\text{Today}) \cdot P(\text{is}|\text{Today}) \cdot P(\text{hot}|\text{Today is})$$

Pros

- Simplicity: N-Gram models are simple and easy to implement
- Words order: N-Gram models preserve the order of words in a text
- N-grams work well across different languages without requiring significant adaptation.

Cons

- Generalization: N-Gram models fail to capture relationships between words that are not adjacent in the text
- Sparsity: N-Gram models suffer from the sparsity problem. As more text is analyzed, the dimensionality of the feature space increases, causing the frequency of each N-Gram to decrease. Additionally, many entries in the feature space are zero, as most word combinations in the vocabulary do not commonly occur together in typical text.
- Complexity increases with the order of the model: quadratic for bigrams, cubic for trigrams, etc.

Comparison

Applications and Tasks

- Text classification: Bag of Words and Word2Vec are commonly used, but Word2Vec often outperforms due to its semantic capabilities.
- Semantic similarity: Word2Vec excels by capturing relationships between words.
- Autocomplete: N-grams are highly effective for predicting the next word based on local sequences.

Similarities and Differences

- Purpose: All three models are used to represent text data in a numerical format that can be used for NLP tasks.
- Semantic patterns: Word2Vec captures semantic relationships between words, while Bag of Words and N-Gram focus on word frequency and order respectively.

- Representation: Bag of Words represents text as a vector of word counts, Word2Vec represents words as dense vectors, and N-Gram models represent sequences of words as probabilities.
- Sparsity: Bag of Words and N-Gram models suffer from sparsity due to the high dimensionality of the feature space, while Word2Vec does not have this issue.
- Order: Bag of Words ignores word order, N-Gram models (with $N > 1$) preserve word order, and Word2Vec captures semantic relationships between words regardless of order.
- Training: Bag of Words and N-Gram models are based on counting word occurrences in a corpus, while Word2Vec is trained using neural networks to learn word embeddings.

Conclusion

The choice of model depends on the specific problem:

- Bag of Words $\rightarrow$ for simple tasks requiring word frequency analysis or as a baseline.
- N-gram $\rightarrow$ for tasks requiring local context and patterns.
- Word2Vec $\rightarrow$ to capture semantic relationships and create generalizable representations.