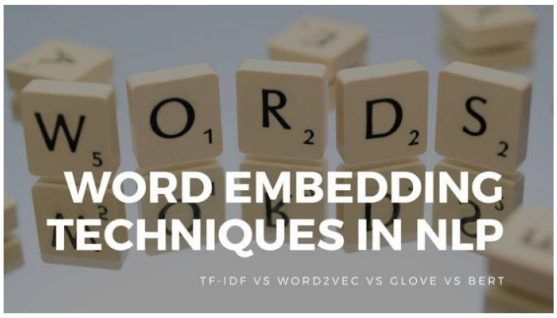
The Ultimate Guide To Different Word Embedding Techniques In NLP

A machine can only understand numbers. As a result, converting text to numbers, called embedding text, is an actively researched topic. In this article, we review different word embedding techniques for converting text into vectors.

comments

By Neeraj Agarwal, a founder of Algoscale.



> *"You shall know a word by the company it keeps!"* — John Rupert Firth

Wouldn't it be incredible if computers could start understanding Shakespeare? Or write fiction like J.K Rowling? This was unimaginable a few years back. Recent advancements in Natural Language Processing (NLP) and Natural Language Generation (NLG) have skyrocketed the ability of computers to better understand text-based content.

To understand and generate text, NLP-powered systems must be able to recognize words, grammar, and a whole lot of language nuances. For computers, this is easier said than done because they can only comprehend numbers.

To bridge the gap, NLP experts developed a technique called *word embeddings* that convert words into their numerical representations. Once converted, NLP algorithms can easily digest these learned representations to process textual information.

Word embeddings map the words as real-valued numerical vectors. It does so by tokenizing each word in a sequence (or sentence) and converting them into a vector space. Word embeddings aim to capture the semantic meaning of words in a sequence of text. It assigns similar numerical representations to words that have similar meanings.

Let's have a look at some of the most promising word embedding techniques in NLP.

1. TF-IDF — Term Frequency-Inverse Document Frequency TF-IDF is a machine learning (ML) algorithm based on a statistical measure of finding the relevance of words in the text. The text can be in the form of a document or various documents (corpus). It is a combination of two metrics: Term Frequency (TF) and Inverse Document Frequency (IDF).

TF score is based on the frequency of words in a document. Words are counted for their number of occurrences in the documents. TF is calculated by dividing the number of occurrences of a word (i) by the total number (N) of words in the document (j).

$$TF(i) = \log(\text{frequency}(i,j)) / \log(N(j))$$

IDF score calculates the rarity of the words. It is important because TF gives more weightage to words that occur more frequently. However, words that are rarely used in the corpus may hold significant information. IDF captures this information. It can be calculated by dividing the total number (N) of documents (d) by the number of documents containing the word (i).

$$IDF(i) = \log(N(d) / \text{frequency}(d,i))$$

The *log* is taken in the above formulas to dampen the effect of large scores for TF and IDF. The final TF-IDF score is calculated by multiplying TF and IDF scores.

TF-IDF algorithm is used in solving simpler ML and NLP problems. It is better used for information retrieval, keyword extraction, stop words (like 'a', 'the', 'are', 'is') removal, and basic text analysis. It cannot capture the semantic meaning of words in a sequence efficiently.

2. Word2Vec — Capturing Semantic Information Developed by Tomas Mikolov and other researchers at Google in 2013, Word2Vec is a word embedding technique for solving advanced NLP problems. It can iterate over a large corpus of text to learn associations or dependencies among words.

Word2Vec finds similarities among words by using the *cosine similarity* metric. If the cosine

Search KDnuggets...

Latest News

Genetic Algorithm Key Terms, Explained
Python: The programming language of machine learning
Must-haves on Your Data Science Resume
Learn MLOps with This Free Course
How Activation Functions Work in Deep Learning
A Beginner's Guide to Q Learning

Top Posts Last Week

1 The Complete Collection of Data Science Books – Part 2
2 Decision Tree Algorithm, Explained
3 Data Science Projects That Will Land You The Job in 2022
4 The 6 Python Machine Learning Tools Every Data Scientist Should Know About
5 Naïve Bayes Algorithm: Everything You Need to Know

More Recent Posts

A Beginner's Guide to Q Learning
Five Signs of an Effective Data Science Manager
Discover a pioneering career in AI
Machine Learning Is Not Like Your Brain Part 3: Fundamental Ar...
Top Industries and Employers Hiring Data Scientists in 2022
KDnuggets News, June 1: The Complete Collection of Data Scienc...
KDnuggets Top Posts for April 2022: 15 Python Coding Interview...
21 Cheat Sheets for Data Science Interviews
Top 18 Data Science Groups on LinkedIn
Metadata Store for Production ML!

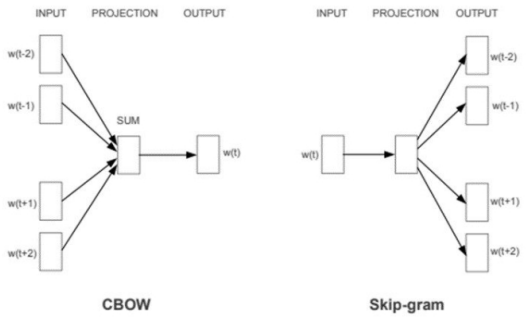
Related Posts

An Introductory Guide to NLP for Data Scientists with 7 Common Techniques

angle is 1, that means words are overlapping. If the cosine angle is 90, that means words are independent or hold no contextual similarity. It assigns similar vector representations to similar words.

Word2Vec offers two neural network-based variants: Continuous Bag of Words (CBOW) and Skip-gram. In CBOW, the neural network model takes various words as input and predicts the target word that is closely related to the context of the input words. On the other hand, the Skip-gram architecture takes one word as input and predicts its closely related context words.

CBOW is quick and finds better numerical representations for frequent words, while Skip Gram can efficiently represent rare words. Word2Vec models are good at capturing semantic relationships among words. For example, the relationship between a country and its capital, like Paris is the capital of France and Berlin is the capital of Germany. It is best suited for performing semantic analysis, which has application in recommendation systems and knowledge discovery.



CBOW & Skip-gram architectures. Image Source: Word2Vec paper.

3. GloVe — Global Vectors for Word Representation Developed by Jeffery Pennington and other researchers at Stanford, GloVe extends the work of Word2Vec to capture global contextual information in a text corpus by calculating a global word-word co-occurrence matrix.

Word2Vec only captures the local context of words. During training, it only considers neighboring words to capture the context. GloVe considers the entire corpus and creates a large matrix that can capture the co-occurrence of words within the corpus.

GloVe combines the advantages of two-word vector learning methods: matrix factorization like latent semantic analysis (LSA) and local context window method like Skip-gram. The GloVe technique has a simpler least square cost or error function that reduces the computational cost of training the model. The resulting word embeddings are different and improved.

GloVe performs significantly better in word analogy and named entity recognition problems. It is better than Word2Vec in some tasks and competes in others. However, both techniques are good at capturing semantic information within a corpus.

GloVe word vectors capturing words with similar semantics. Image Source: Stanford GloVe.

4. BERT — Bidirectional Encoder Representations from Transformers Introduced by Google in 2019, BERT belongs to a class of NLP-based language algorithms known as transformers. BERT is a massive pre-trained deeply bidirectional encoder-based transformer model that comes in two variants. BERT-Base has 110 million parameters, and BERT-Large has 340 million parameters.

For generating word embeddings, BERT relies on an *attention mechanism*. It generates high-quality context-aware or contextualized word embeddings. During the training process, embeddings are refined by passing through each BERT encoder layer. For each word, the attention mechanism captures word associations based on the words on the left and the words on the right. Word embeddings are also positionally encoded to keep track of the pattern or position of each word in a sentence.

BERT is more advanced than any of the techniques discussed above. It creates better word embeddings as the model is pre-trained on massive word corpus and Wikipedia datasets. BERT can be improved by fine-tuning the embeddings on task-specific datasets.

Though, BERT is most suited for language translation tasks. It has been optimized for many other applications and domains.

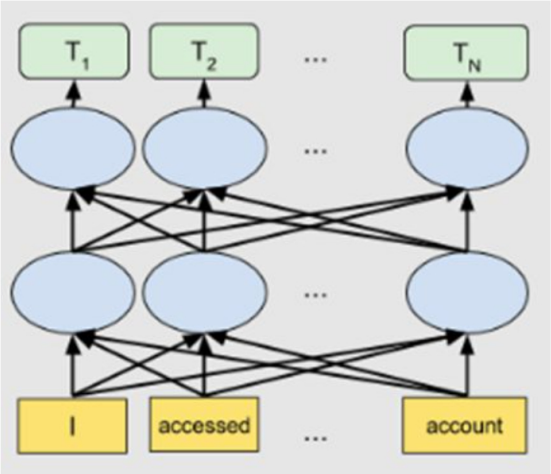
BERT is changing the NLP landscape

Beyond Explainability: A Practical Guide to Managing Risks in Machine...

Word Embedding Fairness Evaluation

Roadmap to Natural Language Processing (NLP)

Understanding NLP and Topic Modeling Part 1



Bidirectional BERT architecture. Image Source: Google AI Blog.

Concluding Thoughts

With advancements in NLP, word embedding techniques are also improving. There are many NLP tasks that don't require advanced embedding techniques. Many can perform equally well with simple word embedding techniques. The selection of a word embedding technique must be based on careful experimentations and task-specific requirements. Fine-tuning the word embedding models can improve the accuracy significantly.

In this article, we have given a high-level overview of various word embedding algorithms. Let's summarize them below:

Word Embedding Technique	Main Characteristics	Use cases
TF-IDF	Statistical method to capture the relevance of words w.r.t the corpus of text. It does not capture semantic word associations.	Better for information retrieval and keyword extraction in documents.
Word2Vec	Neural network-based CBOW and Skip-gram architectures, better at capturing semantic information.	Useful in semantic analysis task.
GloVe	Matrix factorization based on global word-word co-occurrence. It solves the local context limitations of Word2Vec.	Better at word analogy and named-entity recognition tasks. Comparable results with Word2Vec in some semantic analysis tasks while better in others.
BERT	Transformer-based attention mechanism to capture high-quality contextual information.	Language translation, question-answering system. Deployed in Google Search engine to understand search queries.

References

1. https://www.ibm.com/cloud/learn/natural-language-processing
2. https://www.techopedia.com/definition/33012/natural-language-generation-nlg
3. https://arxiv.org/abs/1301.3781
4. https://www.machinelearningplus.com/nlp/cosine-similarity/
5. https://www.ibm.com/cloud/learn/neural-networks
6. https://www.expert.ai/blog/natural-language-process-semantic-analysis-definition/
7. https://nlp.stanford.edu/pubs/glove.pdf
8. https://medium.com/swlh/co-occurrence-matrix-9cacc5dd396e
9. https://blog.marketmuse.com/glossary/latent-semantic-analysis-definition/
10. https://www.investopedia.com/terms/l/least-squares-method.asp
11. https://www.expert.ai/blog/entity-extraction-work/
12. https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html
13. https://arxiv.org/abs/1810.04805
14. https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
15. https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/

Bio: Neeraj is a founder of Algoscale, a data consulting company covering data engineering, applied AI, data science, and product engineering. He has over 9 years of experience in the field and has helped a wide range of organizations from start-ups to Fortune 100 companies ingest and store enormous amounts of raw data in order to translate it into actionable insights for better decision-making and faster business value.

Related:

- Roadmap to Natural Language Processing (NLP)
- An Introductory Guide to NLP for Data Scientists with 7 Common Techniques
- Word Embeddings in NLP and its Applications

More On This Topic

- Text Encoding: A Review
- KDnuggets™ News 19:n39, Oct 16: Key Ideas in Document Embedding; The...
- Top Stories, Oct 7-13: 10 Free Top Notch Natural Language Processing...
- Where NLP is heading
- KDnuggets™ News 20:n31, Aug 12: Data Science Skills: Have vs Want:...
- Three Methods of Data Pre-Processing for Text Classification

<= Previous post

Next post =>