Technology Review
CS 425
Jackson DeDobbelaere
11/6/2020

Word2Vec: A Brief Overview and Discussion

Word2Vec is a family of related models that are utilized to produce word embeddings. The models are very shallow, meaning that they operate in a very trivial manner and do not employ sorts of machine learning to gain "human intelligence". For most text analysis tasks, this is sufficient especially since deeper models tend to hit the mark sometimes but miss the mark other times. Thus, deep models cannot be used reliably in an unsupervised manner in a way that Word2Vec models can be.

Word2Vec is more of a framework, rather than a specific algorithm for performing all sorts of comparisons, filtration, sorting, etc. The main concept is to break certain passages of texts, whether that passage of text be a sentence, review, paragraph, page, etc., and represent all of the words as vectors that can then be analyzed for a variety of purposes in text analysis. One common use case is to take the similarity between two-word vectors to and rank them against other passages of text to discover commonalities and related passages of text. However, due to Word2Vec being a broad framework to represent passages of text, many things are left open-ended for implementers to decide and choose which parameters suit their needs the best. For example, the similarity method is left open-ended; however, a common comparison method is the cosine similarity function.

The power of the Word2Vec framework really comes from the fundamental concept of being able to represent abstract one-dimensional passages of text in a higher dimensional space that can be analyzed mathematically. As we have learned from recent years, data powers everything and being able to harness the power of data and use it in a useful way if crucial to many of todays business goals. Word2Vec is no exception as we need a way to gather business intelligence about similarities, rankings, and sentiments to better drive our user experiences, customer engagement, and/or overall business strategies.

In addition to providing this basic framework of representing a passage of text, Word2Vec provides optimizations for encoding this vector of words in reduced storage space. A naïve storage implementation would map words one to one and take up storage space that is O(N), where N is the number of words in the vocabulary. Word2Vec optimizes mapping and storing closely related vectors (based on similarity method) together to reduce storage and cluster similar passages of text together.

Knowing this embedding is what we would like to achieve, Word2Vec allows us to generate this embedding in an optimized way by using a two-level neural network. There are two main models used to help generate this embedding. The first one is the Skip Gram and the second one is the Common Bag of Words, or CBOW for short.

The CBOW approach uses the same naïve approach discussed above, utilizing the storage space that is proportional to the size of the vocabulary. This is often referred to as one-hot encoding.

The main goal of the CBOW approach is to take in a word and the context of the word and try to predict the word from the context. Within the CBOW, we also have the word which ultimately serves as the ground truth to compare against. Because we can compare vectors of words against one another, we can predict an output word from the context and measure the difference between the output from the model and the actual word. More specifically, to get a deeper understanding of how the model actually predicts an output word: it uses the word vectors of all the words in the context and combines them into a N-dimensional hidden layer whereas all the words in the context are represented by matrices that are N by the size of the vocabulary. Word2Vec can optimize this by only storing the values that are 1 on the 0-1 vectors per a word and thus, perform computations on sparce matrices.

Another approach is the Skip-Gram model. The Skip Gram model takes almost the complete backwards approach and tries to generate many probability distributions given the actual word and some word from the context, Therefore, the model generates a probability for every word in the context and the differences between contexts can be compared again as every word in a context is made up of a vector. This model uses backpropagation to learn. Backpropagation is very popular and widely used in industry. It efficiently computes the gradient to minimize the loss function and use those resulting parameters in the next iteration.

When designing systems in the real-world it is worth considering the tradeoffs of using the CBOW approach vs the Skip-Gram model. There is no clear winner as each model can server different purposes. The CBOW approach is more efficient than the Skip Gram approach; but this is because the CBOW approach is lossy when just considering words individually rather than keeping the context around them. CBOW may be sufficient and fast for some tasks, whereas other in-depth tasks may be better suited for the Skip-Gram approach.

All in all, Word2Vec is an extremely powerful framework for relating passages of words to words within it, other passages of words, and/or other words all together. It is an extremely powerful tool that has been implemented in backend of many speech recognition products such as Siri. Siri, for example, can now predict with high accuracy what one or many words may have been given the context of what else was said. The Microsoft translator on Teams goes back and changes previous words because of techniques like Word2Vec in their live captioning feature. Word2Vec enables computer scientists to treat words and passages of words as they are represented in a multi-dimensional space and operate on them just like any other numerical data to discover, capture, and operationalize business insights to improve products and ultimately create a better world for humans all thanks for the Word2Vec framework. Many extensions have added onto these models and hopefully more intuitive extensions of these models, or even new models themselves, will come in the future.

Jackson DeDobbelaere

Works Cited

Alammar, J. (2019, March 27). *The Illustrated Word2Vec*. Retrieved from http://jalammar.github.io/illustrated-word2vec/

Karani, D. (2018, September 1). *Introduction to Word Embedding and Word2Vec*. Retrieved from Towards Data Science: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

Rong, X. (2016). *word2vec Parameter Learning Explained.*