I have heard a lot of talk about Word2Vec being fundamental to NLP, so I decided to start the review out of curiosity about 'calculation of language'. This techreview will focus on understanding the basic conceptual parts of Word2Vec.

[Introduction – Operate words like vectors]

After selecting 'Word2Vec' as the subject of tech review, I tried inputting a few words as a test on a site[1] that can check the similarity by performing vector calculations with Korean words, and I was able to derive the word results as shown in the equation below.

e.g.
Korea – Seoul + Washington = USA
Basketball – Micheal Jordan + Lionel Messi = Soccer

How is it possible to do arithmetic with words? Curious about the vectorization of words, I researched various data along with the data provided by the class. The vectorization of words is explained by the concept of 'Word Embedding', and I try to understand and organize it with Word2vec.

* Word Embedding : In natural language processing (NLP), word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.[2]

[Main 1 – Word2Vec's principle : Distributed Representation]

Representation is philosophy of mind, cognitive psychology, neuroscience, and cognitive science, is a hypothetical internal cognitive symbol that represents external reality.[3] In other words, it can be understood simply as the concept of recognizing abstraction.
Among them, Distributed Representation is the concept that words appearing in similar contexts have similar semantics. In vector notation for Distributed Representation, it is essential to understand one-hot vector, and the explanation is as follows. One-hot vector is a group of bits among which the legal combinations of values are only those with a single high(1) bit and all others low(0).[4]

e.g "Fruit"
Apple [1 0 0]
Peach [0 1 0]
Mango [0 0 1]

However, according to Distributed Representation, unlike the above method, the meaning of a word is expressed by 'dispersing and expressing the meaning of a word' in a lower dimension, rather than a method of separating the dimensionality in the higher dimension. Each dimension has a real value.

e.g Apple [0.2 0.5 0.1]

And using Distributed Representation makes it easy to calculate the similarity between word vectors. One of the learning methods for the calculation becomes Word2Vec.

---

1 word2vec.kr
2 https://en.wikipedia.org/wiki/Word_embedding
3 https://en.wikipedia.org/wiki/Mental_representation
4 https://en.wikipedia.org/wiki/One-hot

[Main 2 – Word2Vec's learning method : CBOW/Skip-gram]

There are two main learning methods of Word2Vec

- CBOW : Predict Word in the middle(center word) with input of Words around(context word)

- Skip-gram : Predict Words around(context word) with input of Word in the middle(center word)

In more detail, CBOW determines the range of how many surrounding words before and after to predict the center word. The range is called 'window', and if the size of the window is x, x in front of the center word and x in the back of the center word are used for a total of 2x. And for learning, the set of context word and center word is changed by moving one word side by side. This method is called 'sliding window'.

An example of a sliding window with a window size of 1 can be expressed as follows.

Example Source[5]

Training	Context word	Center(target) Word
1	[1,0,0,0,0], [0,0,1,0,0]	[0,1,0,0,0]
2	[0,1,0,0,0], [0,0,0,1,0]	[0,0,1,0,0]
3	[0,0,1,0,0], [0,0,0,0,1]	[0,0,0,1,0]
4	[0,0,0,1,0]	[0,0,0,0,1]

And in the case of skip-gram, various context words (output) are predicted based on the target word (input). Learning Skip-gram with the same example as CBOW above is as follows.

Training	Context word	Center(target) Word
1	i,natural	like
2	like, language	natural
3	natural, processing	language
4	language	processing

Because it learns words across various contexts, skip-gram performance is often better than CBOW. In other words, the two learning methods are schematically as follows.

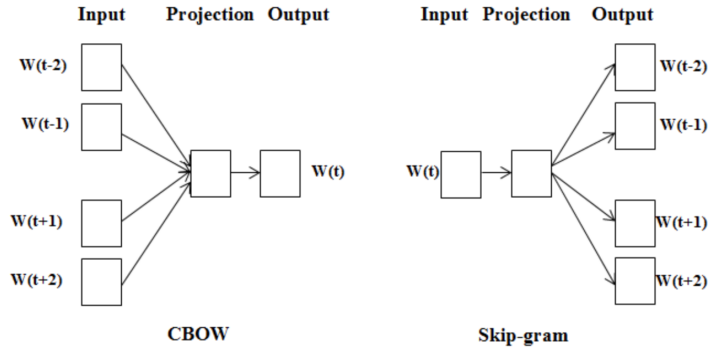

image source[6]

Both methods are optimal for formulating distributed presentations. It can be understood that the operation of Word2Vec is possible based on the two learning methods.

---

[5] https://thinkinfi.com/continuous-bag-of-words-cbow-multi-word-model-how-it-works/

[6] https://www.researchgate.net/figure/CBOW-and-Skip-gram-models-architecture-1_fig1_319954363

[Conclusion – Word2Vec's practice and Expected things]

If someone wants to learn Word2Vec in python, they can practice with code like the example below.

```
from gensim.models import Word2Vec
model = Word2Vec(X_train, size = #, window = #, min_count = #, workers = #, …)

result = model.wv.most_similar("word")
print(result)
```

e.g. word = 'dog' → result = [('puppy', 0.8571829385941), ('pet', 0.81128571957359), …]

In this way, a basic understanding of Word2Vec, which is the basis of natural language processing, was performed through techreview. I am sure that understanding of 'word counting' will be made a little easier and will become a cornerstone that can be done along with consideration of basic concepts rather than simply using language analysis tools in the future.