

An Overview of Natural Language Processing

From Word2Vec To GPT

Jaihua Yen

March 10, 2023

QNAP

Table of contents



1. Introduction
2. TF-IDF
3. Word2Vec
4. Transformer
5. BERT
6. Conclusion

Introduction



What is natural language processing



Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that focuses on the interaction between computers and humans using natural language. The goal of NLP is to enable computers to understand, interpret, and generate human language.  

NLP involves the use of techniques from computer science, linguistics, and machine learning to process, analyze, and generate natural language. Some common applications of NLP include sentiment analysis, language translation, text classification, chatbots, and speech recognition.

NLP is a rapidly evolving field with new developments and advancements being made regularly. As computers become better at processing language, the potential applications for NLP continue to expand, making it an important field of study in both industry and academia.

- Natural Language Processing (NLP) is now maturely developed in many tasks. For example, we can classify documents into several categories for as to search files easily. Moreover, we can generate text by generative AI such as GPT series and ChatGPT.
- In general, NLP is used in the following several scenarios:
 1. Machine Translation
 2. Document Classification
 3. ChatBot
 4. Text Semantic Analysis
 5. Text Generation

TF-IDF

Word2Vec

Issues in Word Representation

- We cannot gain information of relation between words if those words in text is represented as one-hot encoding.
- The one-hot encoding would be very sparse if there exists large amount of unique words.

For example, we set hotel and motel as two different representative of one-hot encoding:

$$\text{motel} = [0,0,0,0,1,0,0,0,0]$$

$$\text{hotel} = [0,0,0,0,0,1,0,0,0]$$

We have no idea how motel and hotel are relate to each other while those two vectors are orthogonal! That's why we introduce word embedding approach to tackle these problems. [1]

Word Embedding

Vocabulary:
Man, woman, boy,
girl, prince,
princess, queen,
king, monarch



	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets
a 1x9 vector
representation

Try to build a lower dimensional embedding

Vocabulary:
Man, woman, boy,
girl, prince,
princess, queen,
king, monarch



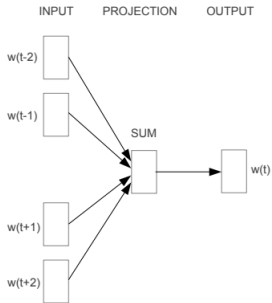
	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy	0	1	0
Girl	1	1	0
Prince	0	1	1
Princess	1	1	1
Queen	1	0	1
King	0	0	1
Monarch	0.5	0.5	1

Each word gets a
1x3 vector

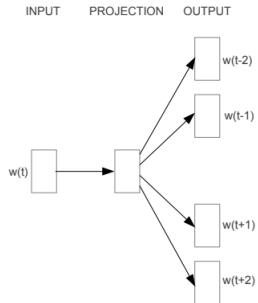
Similar words...
similar vectors

[@shane_a_lynn](#) | [@TeamEdgeTier](#)

[Image Source]

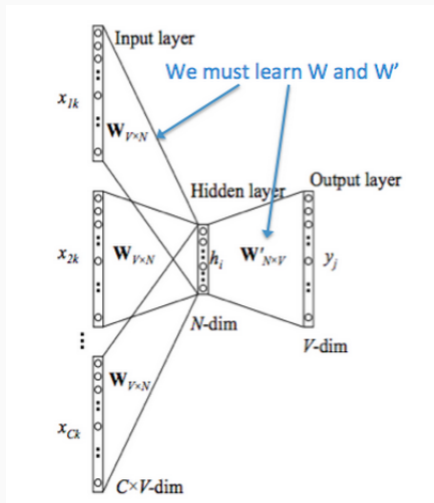


CBOW



Skip-gram

CBOW



[Image Source]

Use the probability $P(y_j | x_{1k}, x_{2k}, \dots, x_{Ck})$ to learn the weight matrix W !

Transformer

BERT

Conclusion

Questions?



T. Mikolov et al.

Efficient estimation of word representations in vector space.

arXiv preprint arXiv, 1301.3781, 2013.