Efficient Estimation of Word Representations in Vector Space

Paper Review

Fred Jeong

January 31, 2025

Overview

Background

A **language model** is a model that determines the probability of a given sequence of words occurring in a sentence by analysing natural language data using statistical and probabilistic techniques.

NLP problems such as machine translation, question answering (QA), sentiment analysis utilise LM techniques.

Today is Friday. Tomorrow is (   ).

One-hot encoding

Previous works treat words as **atomic units**.

- **Similarities** between words are not considered.
- Simple, but each word vector does not contain any **relevant information**.

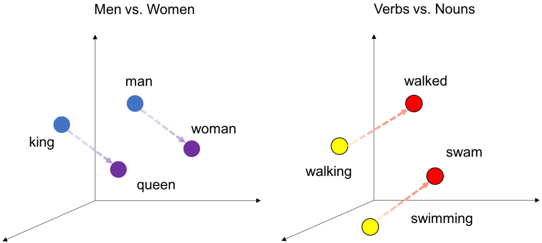
Suppose we have three words: apple, orange, grape.
One-hot encoding expresses each word as a unit vector, whose size is the same as the vocabulary size:

$$\begin{bmatrix} \text{apple} & \text{orange} & \text{grape} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Distributed Representations

Distributional hypothesis states that words with similar meanings tend to appear in similar contexts.

- We can find and compute similarities between words.
- Algebraic operations can be performed.



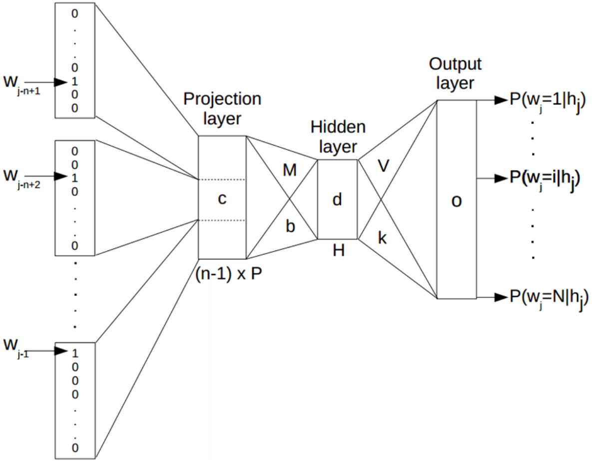
Distributed Representations

$$\text{Vector}(\text{Seoul}) - \text{Vector}(\text{Korea}) = \text{Vector}(\text{Tokyo}) - \text{Vector}(\text{Japan})$$

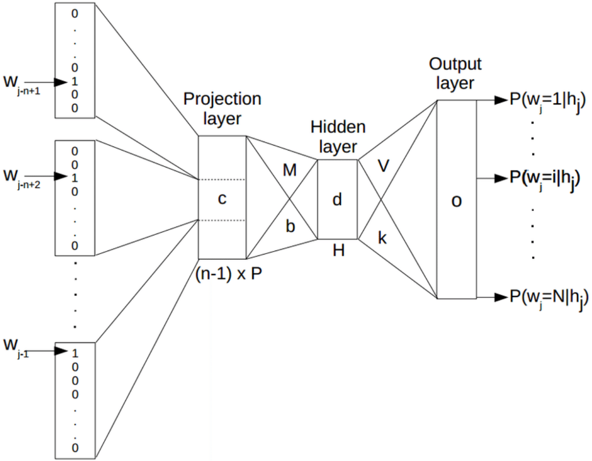
Feedforward Neural Net Language Model (NNLM)

Notation

- N: number of inputs (previous words)
- D: projected dimension (embedding dimension)
- H: hidden layer size
- V: Vocabulary size



Feedforward Neural Net Language Model (NNLM)



Dimensions

- Input layer: $N \times V$
- Projection matrix: $V \times D$
- Projection layer: $N \times D \rightarrow N \cdot D$
- Hidden layer: $N \cdot D \rightarrow V$

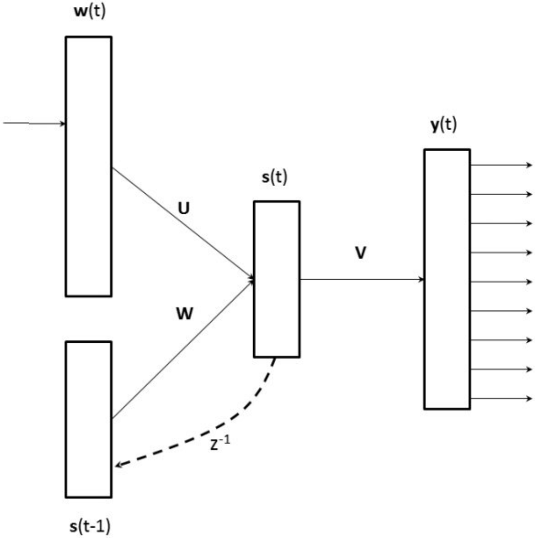
Feedforward Neural Net Language Model (NNLM)

- Need to fix the number of inputs.
- Does not consider any future words.
- Computational complexity:

$$Q = N \times D + N \times D \times H + H \times V$$

- Too slow!

Recurrent Neural Net Language Model (RNNLM)



Recurrent Neural Net Language Model (RNNLM)

- **No need to specify context length**.
- Hidden layer is recurrently connected to itself.
- The model can form **short term memory**.
- Computational complexity:

$$Q = H \times H + H \times V$$

- Still slow!

Goals of the Paper

- Most of the complexity is caused by the non-linear hidden layer in the model.
- There is a trade-off between accuracy and computational complexity.

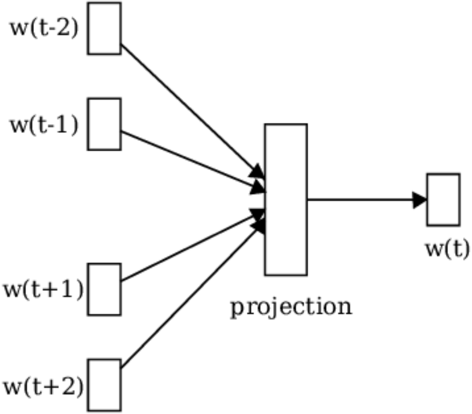
Goals of the Paper

- Learn high-quality word vectors from huge datasets.
- Let words have **multiple degrees of similarity** (**semantically and syntactically**).
  - "apple" and "orange" are **semantically** similar.
  - "ran" and "slept" are **syntactically** similar.
- Maximise accuracy of vector operations while minimising computational complexity.

Continuous Bag-of-Words Model

Dimensions

- Input layer: $N \times V$
- Projection matrix: $V \times D$
- Projection layer: $N \times D \rightarrow D$
- Weight matrix: $D \times V$
- Output: V



Continuous Bag-of-Words Model

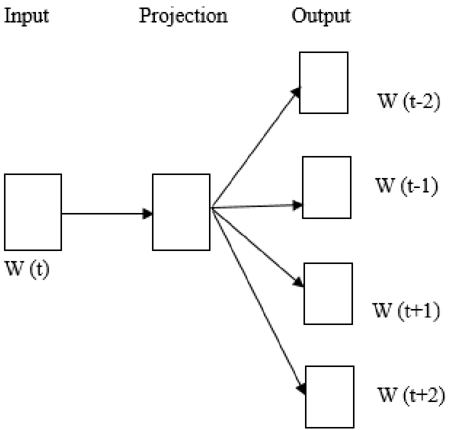
- Predict the current word **based on the context**.
- Input: word vectors of context words
- Output: probabilities of all words in the vocabulary appearing at the current position.

...you should have somehow **realised** what you gotta do...

Continuous Skip-gram Model

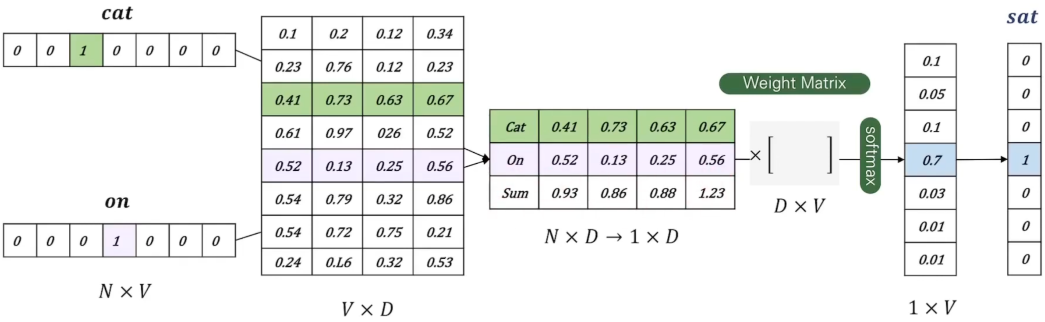
Dimensions

- Input layer: V
- Projection matrix: $V \times D$
- Projection layer: D
- Weight matrix: $D \times V$
- Output: V



Example

- ✓ "The fat cat sat on the mat"
  - $window\ size = 1$



Complexity

Neural network based models tend to have high levels of computational complexity due to the existence of **hidden layers**.

Continuous-bag-of-words model

$$Q = N \times D + D \times \log_2(V)$$

Continuous skip-gram model

$$Q = C \times (D + D \times \log_2(V))$$

Frame Title

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

- Using more data and higher dimensional word vectors improve accuracy.
- Adding more dimensions or adding more training data provides diminishing improvements.
- We have to increase both vector dimensionality and the amount of the training data together.

Experiments

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

Experiments

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

Learned Relationships

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Conclusion

Pros

- It is possible to train high quality word vectors using very simple model architectures.
- Thanks to lower computational complexity, it is possible to compute very accurate high dimensional word vectors from a much larger dataset.

Cons

- Out-of-vocabulary problem is not resolved. $\rightarrow$ FastText
- Dependent on frequencies of words.

The End