Natural Language Processing Workshop 2

Computer Engineering Department
Sharif University of Technology

Spring 2023

Outline

Word2Vec

- Bag-of-Words
- Continuous Bag-of-Words (CBoW)
- Skip-Gram

N-gram

fastText

Word2Vec

What?

- A family of model architectures and optimizations that can be used to learn word embeddings from large datasets.

  a. **Continuous Bag-of-Words Model**

  b. **Skip-gram Model**

BoW & CBoW

Bag of Words

What?

- The most commonly used method of text classification
- The (frequency of) occurrence of each word is used as a feature for training a classifier

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Continuous Bag of Words

What?

- To obtain a meaning for each word, a **fake task** should be defined
- Consider the following incomplete sentence

S := I prefer to travel by … rather than cars.

- By using which one of the words flowers, airplanes, or lions should we fill in the above sentence? The most **probable** one!
- What if a sentence is too long? Use a **window**.

$$\underset{w \in \{\text{flower, airplane, lion}\}}{argmax} \; P(w_i = w | w_{i-l}, w_{i-l+1}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+l-1}, w_{i+l})$$

Continuous Bag of Words

- Given a window of words of a length 2m+1

$$x_{-m}, \cdots, x_{-1}\ x_0 x_1, \cdots, x_m$$

- Define a probabilistic model for predicting the middle word

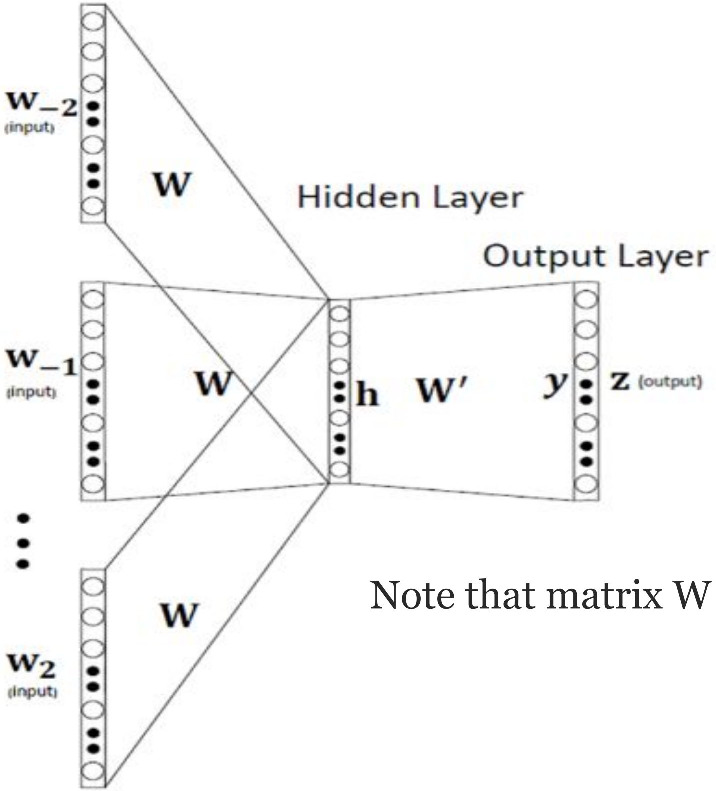
$$P(x_0 \mid x_{-m}, \cdots, x_{-1}, x_1, \cdots, x_m)$$

- Train the model by minimizing loss over the dataset (cont.)

Continuous Bag of Words

- The classification task:
    1. Input: context words
    2. Output: The **center** word (one-hot vector)
- Notation:
    1. n: the embedding dimension
    2. V: vocabulary
    3. $\mathcal{V}$: a matrix of size n*|V|
    4. $\mathcal{W}$:a matrix of size |V|*n

Continuous Bag of Words



Note that matrix W is **shared**.

Continuous Bag of Words

How?

- Map all the context words into the n dimensional space using V

$$v_{x_{-m}}, \cdots, v_{x_{-1}}, v_{x_1}, \cdots, v_{x_m}$$

- Average these vectors to get a context vector

$$\hat{v} = \frac{1}{2m} \sum_{i=-m, i\neq 0}^{m} v_{x_i}$$

- Use this to compute a score vector for the output $score = \mathcal{W}\hat{v}$
- Use the score to compute probability via softmax

$$P(x_0 = \cdot \mid context) = softmax(\mathcal{W}\hat{v})$$

Continuous Bag of Words

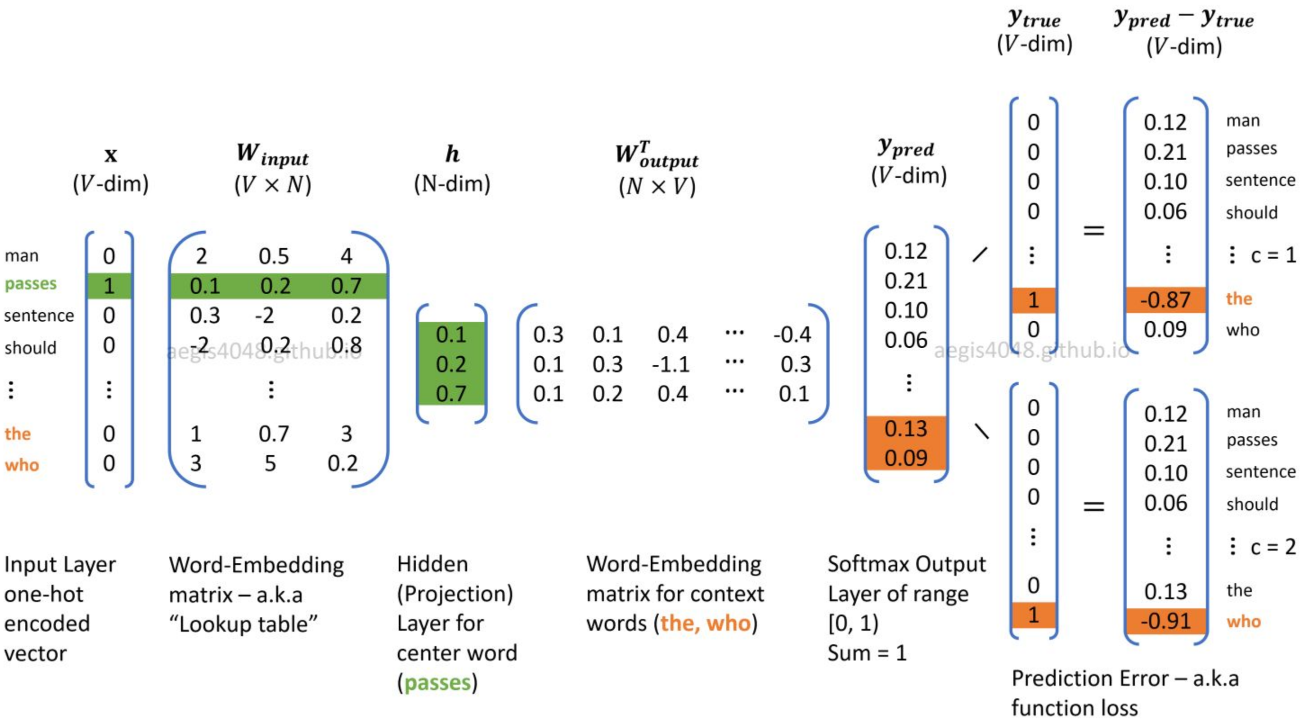
Objective?

$$
\begin{aligned}
\text{minimize } J &= -\log P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}) \\
&= -\log P(u_c | \hat{v}) \\
&= -\log \frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}
$$

Skip-Gram

Skip-Gram

It is similar to CBoW and just the fake task is **inverted**.



Skip-Gram

- Given a window of words of a length 2m+1

$$x_{-m}, \cdots, x_{-1}\ x_0 x_1, \cdots, x_m$$

- Define a probabilistic model for predicting the middle word

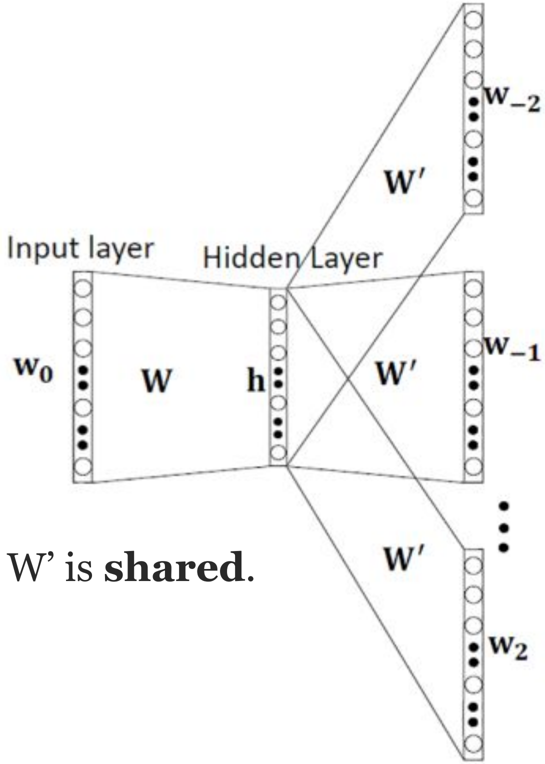
$$P(x_{context} \mid x_0)$$

- Train the model by minimizing loss over the dataset (cont.)

Skip-Gram

- The classification task:
  1. Input: The **center** word (one-hot vector)
  2. Output: context words
- Notation:
  1. n: the embedding dimension
  2. V: vocabulary
  3. $\mathcal{V}$: a matrix of size n*|V|
  4. $\mathcal{W}$:a matrix of size |V|*n

Skip-Gram



Note that matrix W’ is **shared**.

Skip-Gram

How?

- Map all the context words into the n dimensional space using W.
  - We get an n dimensional vector $w = \mathcal{W} x_0$
- For the i-th context position, compute the score for a word occupying that position as:

$$v_i = \mathcal{V} w$$

- Normalize the score for each position to get a probability

$$P(x_i = \cdot \mid x_0) = \text{softmax}(v_i)$$

Skip-Gram

Objective?

$$
\begin{aligned}
\text{minimize } J &= -\log P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | w_c) \\
&= -\log \prod_{j=0, j \neq m}^{2m} P(w_{c-m+j} | w_c) \\
&= -\log \prod_{j=0, j \neq m}^{2m} P(u_{c-m+j} | v_c) \\
&= -\log \prod_{j=0, j \neq m}^{2m} \frac{\exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|} \exp(u_k^T v_c)} \\
&= -\sum_{j=0, j \neq m}^{2m} u_{c-m+j}^T v_c + 2m \log \sum_{k=1}^{|V|} \exp(u_k^T v_c)
\end{aligned}
$$

N-Gram

N-Gram

What? chain rule

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1}) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

$$P(X_1 \ldots X_n) = P(X_1)P(X_2|X_1)P(X_3|X_{1:2})\ldots P(X_n|X_{1:n-1}) = \prod_{k=1}^{n} P(X_k|X_{1:k-1})$$

Example

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

How? Markov assumption

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

fastText

fastText

What?

- An open-source library designed to help build scalable solutions for text representation and classification (By Facebook AI Research (FAIR))

Why?

- Word2Vec faces the problem of Out of vocabulary (OOV)
- By using a distinct vector representation for each word, the Word2Vec model ignores the internal structure of words

fastText

How does it works?

- Creation of word embeddings
- Text Classification

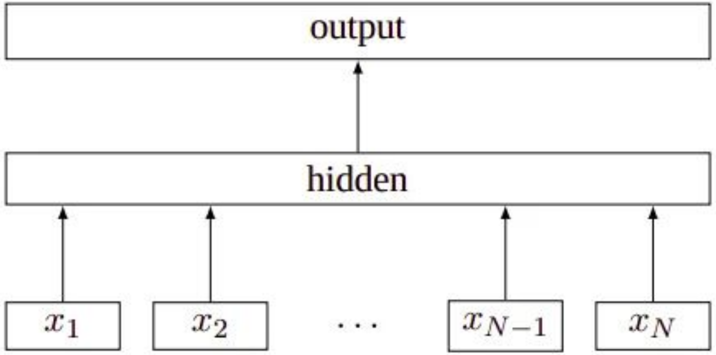
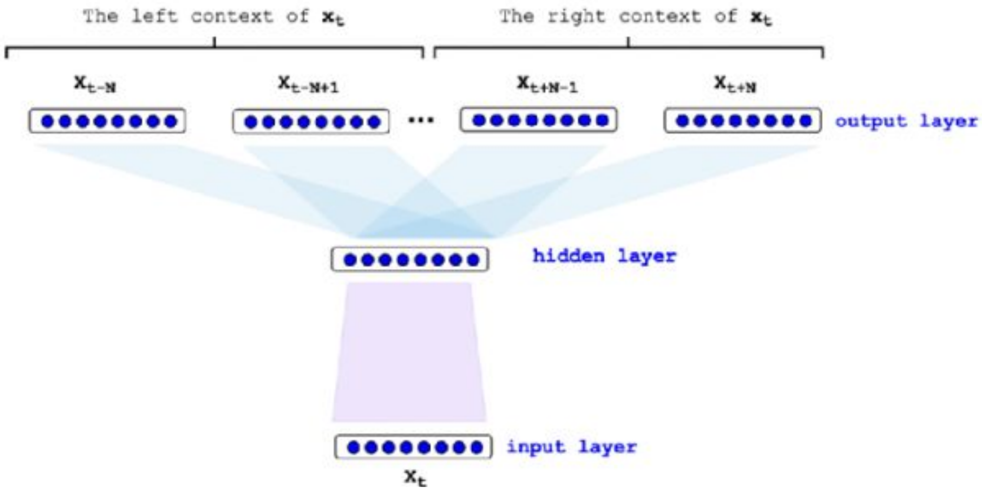


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

fastText



$$\sum_{t=1}^{M} \sum_{c \in [t-N, t+N]} \log p(w_c \mid w_t) \longrightarrow \sum_{t=1}^{T} \left[\sum_{c \in [t-N, t+N]} -\log\left(1 + e^{-s(w_t,\ w_c)}\right) - \sum_{w_r \in \mathcal{N}_{t,c}} \log\left(1 + e^{s(w_t,\ w_r)}\right) \right]$$

$$p(w_c \mid w_t; \theta) = \frac{e^{v_c \cdot v_t}}{\sum_{c' \in C} e^{v_{c'} \cdot v_t}}$$

$$s(w_t, w_c) = \sum_{x \in \mathcal{S}_{w_t}} v_x^{\top} v_C$$

fastText

e(where) =

e(<wh)
+ e(whe)
+ e(her)
+ e(ere)
+ e(re>)

3-grams

+ e(<whe)
+ e(wher)
+ e(here)
+ e(ere>)

4-grams

+ e(<wher)
+ e(where)
+ e(here>)

5-grams

+ e(<where)
+ e(where>)

6-grams

+ e(<where>)

word

e(*) = embedding for *

fastText vs. Word2Vec

- Word2Vec works on the word level, while fastText works on the character n-grams.
- Word2Vec cannot provide embeddings for out-of-vocabulary words, while fastText can provide embeddings for OOV words.
- FastText can provide better embeddings for morphologically rich languages compared to word2vec.
- FastText uses the hierarchical classifier to train the model; hence it is **faster** than word2vec.

HandsOn Coding

https://drive.google.com/file/d/1a9cvG_cSes4YWvi8QLZqEa
hZOPDd96Q_/view?usp=sharing

Questions?