Imperial College Business School

Machine Learning and Finance -Coursework-

Due: 06/06/2022

Instructions:

- The code should be presented as a python file or a jupyter notebook.
- The theoretical questions can be answered in the jupyter notebook or in a separate pdf.
- The following table shows the marks for each of the 30 questions across 4 sections.

Section	Question	Number of marks
Introducing the word2vec approach	1	2
Preprocessing the Data	2	3
	3	4
	4	3
	5	3
Preparing the Training Dataset	6	4
	7	3
	8	4
	9	3
	10	3
	11	4
	12	3
	13	3
	14	3
	15	3
	16	3
	17	2
Learning the Word Vectors	18	2
	19	3
	20	3
	21	3
	22	3
	23	3
	24	3
	25	3
	26	10
	27	3
	28	3
	29	4
	30	4

Notations:

- $\mathcal{M}_{n,p}(\mathbb{R})$ is the space of the matrices composed of n rows and p columns.
- The gradient of a function $f : \theta \in \mathbb{R}^D \mapsto \mathbb{R}$ at $\theta \in \mathbb{R}^D$ is denoted as follows $\nabla_\theta f(\theta) = \left(\frac{\partial f}{\partial \theta_1}(\theta), \dots, \frac{\partial f}{\partial \theta_D}(\theta)\right)$
- Convention:
  - The rows $(A_i)_{1 \leq i \leq n}$ of a matrix $A = \begin{pmatrix} - & A_1 & - \\ \vdots & \vdots & \vdots \\ - & A_n & - \end{pmatrix} \in \mathcal{M}_{n,p}(\mathbb{R})$ are considered $\mathcal{M}_{p,1}(\mathbb{R})$ matrices.
  - The columns $(B_j)_{1 \leq j \leq p}$ of a matrix $B = \begin{pmatrix} | & \dots & | \\ B_1 & \dots & B_p \\ | & \dots & | \end{pmatrix} \in \mathcal{M}_{n,p}(\mathbb{R})$ are considered $\mathcal{M}_{n,1}(\mathbb{R})$ matrices.

Presentation of the Coursework:

The objective of this coursework is to implement the **Word2vec** approach. The model was introduced in Mikolov et al. 2013. It is one of the most successful ideas for learning an embedding matrix from a corpus. The model captures some linguistic patterns between word vectors and performs well on the word analogy task. For instance, the embedding vectors learned using the word2vec approach have the following property: $e_{\text{France}} - e_{\text{Paris}} \approx e_{\text{England}} - e_{\text{London}}$ (where e_X stands for the embedding of the word X).

The coursework is subdivided into four parts:

- In section 1, we introduce the concept of word embedding and the word2vec approach.
- In section 2, the objective is to load the dataset and perform the first processing steps in order to get the corpus.
- In section 3, we use the processed corpus to create a dataset for a binary classification problem.
- In section 4, we create an embedding matrix by learning the parameters of a shallow neural network trained for the binary classification task.

1 Introducing the word2vec approach

The objective of the coursework is to train a model on a corpus of training sentences in order to represent words in a D-dimensional space. We would like to encode the similarity between the words in the embedding vectors themselves.

Question 1: Explain why this notion of similarity is not encoded in the one hot vector representation of words.

Several methods have been used to create word embeddings. The most popular ones rely on the intuition that *a word's meaning is given by the words that frequently appear close-by.*

For instance, we have introduced in Programming Session 5 the GloVe approach, a popular method used to learn low-dimensional word representations by using **matrix factorization** methods on a matrix of word-word **co-occurence** statistics.

In this programming session, we are going to introduce the **word2vec approach**, which represents the tokens as parameters of a shallow neural network predicting a word's context given the world itself.

Similarly to what we have done in the GloVe approach, we will rely on the intuition that a word is defined by the context words.

Figure 1 is an illustration of the concept of neighbors. The center word "economy" is represented in red, and the neighbors for a context size of 3 are boxed. The "context size" is the number of words we consider on either side of a center word as neighbors.

If there are at least 3 words on the left of a center word, we can extract the first 3 left neighbors. If there are fewer than 3 words, we only extract the ones found within the document. For the right neighbors, the same principle applies.

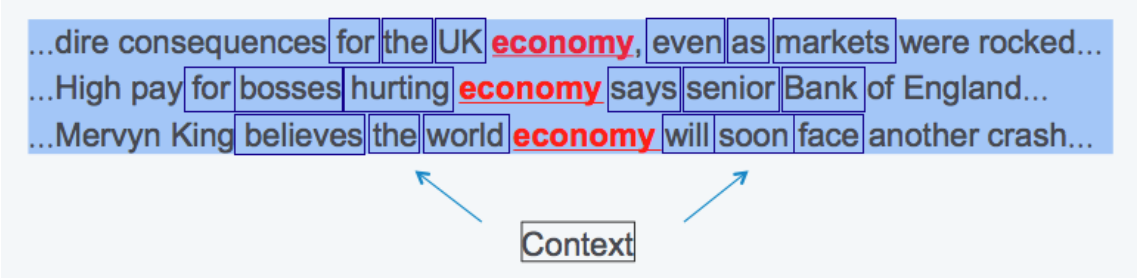

Figure 1: Context words

2 Preprocessing the data

The **data** folder contains a csv file named **RedditNews.csv**[1].

In the *RedditNews.csv* file are stored historical news headlines from Reddit WorldNews Channel, ranked by reddit users' votes, and only the top 25 headlines are considered for a single date.

You will find two colomns:

- The first column is for the "date".
- The second column is for the "News". As all the news are ranked from top to bottom, there are only 25 lines for each date.

Question 2: Load the data from the csv file, create a list of all the news.

[1]Source: Sun, J. (2016, August) Daily News for Stock Market Prediction, Version 1. Retrieved [26 may 2020] from https://www.kaggle.com/aaron7sun/stocknews.

Question 3: Preprocess the data by transforming the list of sentences into a list of sequences of integers, via a dictionary that maps the words to integers.

Question 4: For each sentence, add a specific index for the token "< sos >" (start of sequence) at the beginning of each sequence and an index for the token "< eos >" (end of sequence) at the end of each sequence.

The resulting list of lists of integers is called **sequences**.

The processed corpus is represented in figure 2. It is composed of several document $(\mathcal{D}_{1\leq i\leq N})$.

Each document $(\mathcal{D}_i) = (w_i^1, \ldots, w_i^{n_i})$ is a list of indices of length n_i.

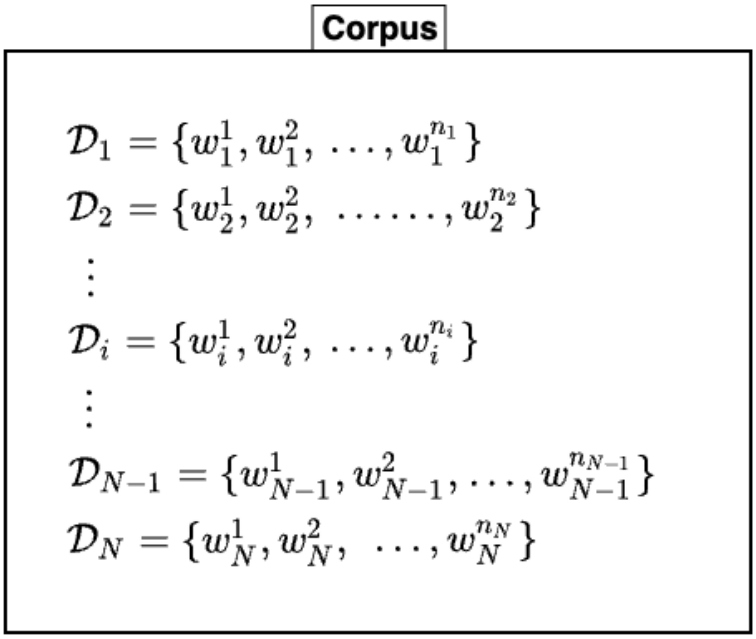

Figure 2: Corpus

Question 5: Filter the documents by only keeping the ones such that $n_i \geq 6$.

3 Preparing the training dataset for a binary classification problem from the processed corpus

3.1 Introducing the positive and negative batch associated with a true center word

Let us consider a document $(\mathcal{D}_i) = (w_i^1, \ldots, w_i^{n_i})$ such that $n_i \geq 6$.

Although the elements of the documents $(\mathcal{D}_i)$ are integers from $\{0, \ldots, V-1\}$, we will refer to them as "words" in the section 3.1.

Let us consider a **context size** of 3 words.

Figure 3 represents the context words associated with a center word w_i^t in a document $(\mathcal{D}_i)$.

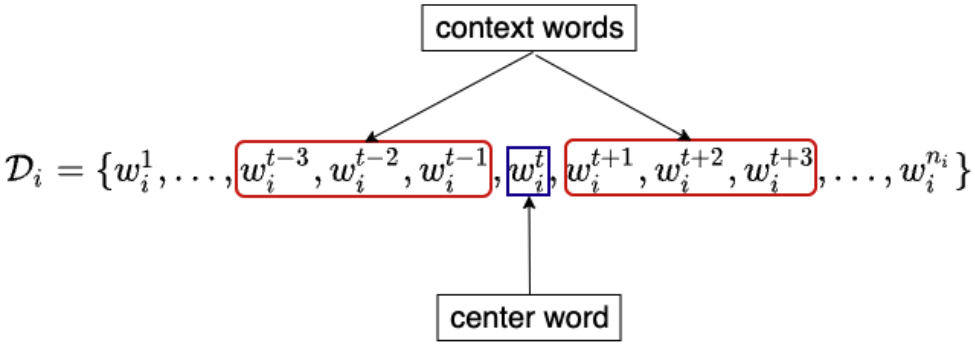

Figure 3: Context words associated with the center word w_i^t

Consequently, each center word is associated with 3, 4, 5 or 6 context words (a maximum of 3 on the left and a maximum of 3 on the right).

Let us suppose that the position of the center word w_i^t allows the existence of 3 context words on the left and 3 context words on the right. We can then define 6 *true couples* of the form (true center word, context word). In the example of figure 3, the 6 *true couples* associated with the center word w_i^t are (w_i^t, w_i^{t-3}), (w_i^t, w_i^{t-2}), (w_i^t, w_i^{t-1}), (w_i^t, w_i^{t+1}), (w_i^t, w_i^{t+2}) and (w_i^t, w_i^{t+3}).

These couples constitute the **positive batch associated with the center word** w_i^t, denoted $\left(\mathcal{B}_+^{w_i^t}\right)$.

In the word2vec approach, for each true center word w_i^t, we need to artificially create the same number of *fake couples* (fake center word, context word). It can be done by keeping the context words associated with the center word w_i^t untouched, and randomly select a fake center word from the vocabulary.

In the original paper, the fake center word, denoted f_i^t, is sampled from the **negative sampling distribution**.

For each word $w \in \{0, \ldots, V-1\}$, let $N(w)$ be the number of times the word w appears in the corpus.

We define the **negative sampling distribution** n as follows:

$$\forall w \in \{0, \ldots, V-1\} \quad n(w) = \frac{N(w)^{0.75}}{\sum_{w'=0}^{V-1} N(w')^{0.75}} \tag{3.1}$$

Question 6: Create a function that takes as arguments :

- **sequences : the list of list of integers composing the corpus.**
- **V : the vocabulary size**

The function should output the list $[n(w)$ **for** $w \in \{0, \ldots, V-1\}]$

Question 7: Use the function defined in the previous question to sample fake center words from the negative sampling distribution.

(Hint: You can use the following reference Link to random choice using numpy).

As a result, we end up with the same number of *fake couples*. The 6 *fake couples* associated with the fake center word f_i^t are (f_i^t, w_i^{t-3}), (f_i^t, w_i^{t-2}), (f_i^t, w_i^{t-1}), (f_i^t, w_i^{t+1}), (f_i^t, w_i^{t+2}) and (f_i^t, w_i^{t+3}).

These couples constitute the **negative batch associated with the center word** w_i^t, denoted $\left(\mathcal{B}_-^{w_i^t}\right)$.

Figure 4 summarizes all the steps involved in creating the *true couples* and the *fake couples* associated with the center word w_i^t.

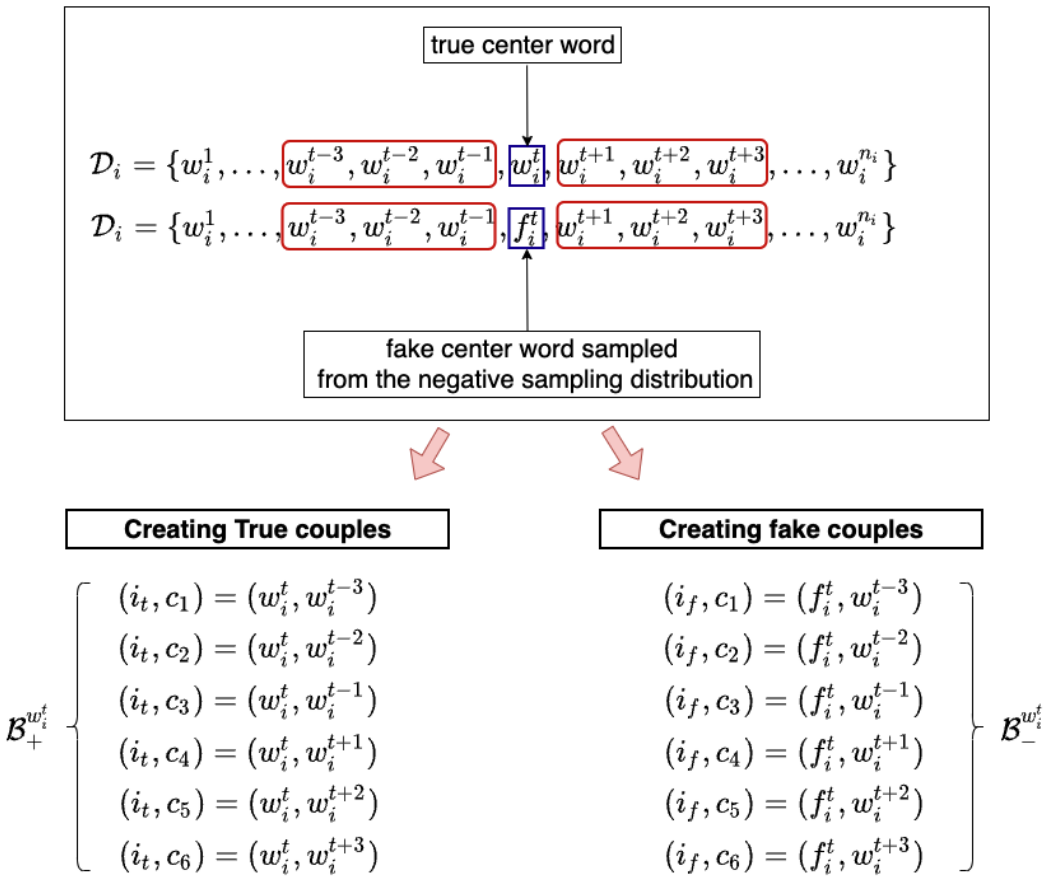

Figure 4: Creating the true couples and the fake couples associated with the true center word w_i^t

Question 8: Create a function which takes as argument:

- **position: an integer representing the position of the center word.**
- **sequence: a list of integers representing the document.**
- **contextsize: an integer representing the number of neighbors we want to consider on each side.**

The function should output the list of integers representing the context words.

An example, is represented in figure 5.

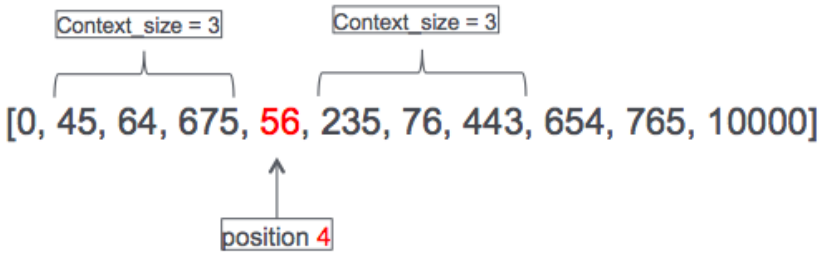

Figure 5: Getting the context words

- The position is equal to 4.
- The sequence is $[0, 45, 64, 675, 56, 235, 76, 443, 654, 765, 10000]$
- The contextsize is 3.
- The function should output the list $[45, 64, 675, 235, 76, 443]$

Question 9: As a sanity check, test the function defined in Question 8 on the document represented in figure 5.

3.2 A Small Example

Let us have a concrete example to illustrate the way we create the positive batch and the negative batch associated with a true center word.

Figure 6 represents a fake corpus composed of three documents.

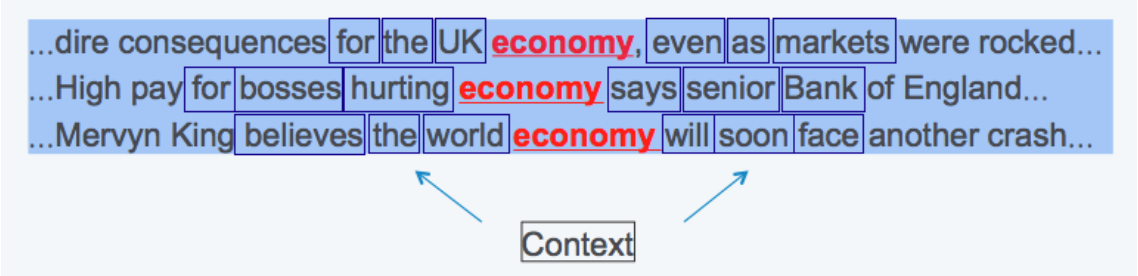

Figure 6: A Fake Corpus of 3 documents

Question 10: Create the fake corpus represented in figure 6.

Question 11: Create the word2idx dictionary (a dictionary mapping each token to a unique index) without using any libraries.

Question 12: Transform the corpus into a list of lists of integers.

We denote id_X the index of the word X in the word2idx dictionary.

Let us consider the true center word **economy** in the second document represented in figure 7.

Figure 7: Example of a center word in a document

The context words associated with the word **economy** are "for", "bosses", "hurting", "says", "senior" and "Bank".

Question 13: Use the function defined in Question 8 to get the indices of the context words "for", "bosses", "hurting", "says", "senior" and "Bank".

The positive batch associated with the center word **economy** is composed of the following *true couples*: $(\text{idx}_{\text{economy}}, \text{idx}_{\text{for}})$, $(\text{idx}_{\text{economy}}, \text{idx}_{\text{bosses}})$, $(\text{idx}_{\text{economy}}, \text{idx}_{\text{hurting}})$, $(\text{idx}_{\text{economy}}, \text{idx}_{\text{says}})$, $(\text{idx}_{\text{economy}}, \text{idx}_{\text{senior}})$ and $(\text{idx}_{\text{economy}}, \text{idx}_{\text{Bank}})$.

Question 14: Use the function defined in Question 6 in order to sample the index of a fake center word.

Question 15: Reverse the word2idx dictionary to determine the fake center word from its index.

Let us suppose that the fake center word is **music**. The negative batch associated with the center word **economy** is composed of the following *fake couples*: $(\text{idx}_{\text{music}}, \text{idx}_{\text{for}})$, $(\text{idx}_{\text{music}}, \text{idx}_{\text{bosses}})$, $(\text{idx}_{\text{music}}, \text{idx}_{\text{hurting}})$, $(\text{idx}_{\text{music}}, \text{idx}_{\text{says}})$, $(\text{idx}_{\text{music}}, \text{idx}_{\text{senior}})$ and $(\text{idx}_{\text{music}}, \text{idx}_{\text{Bank}})$.

As shown in Figure 8, we associate each *true couple* with a label 1 and each *fake couple* with a label 0.

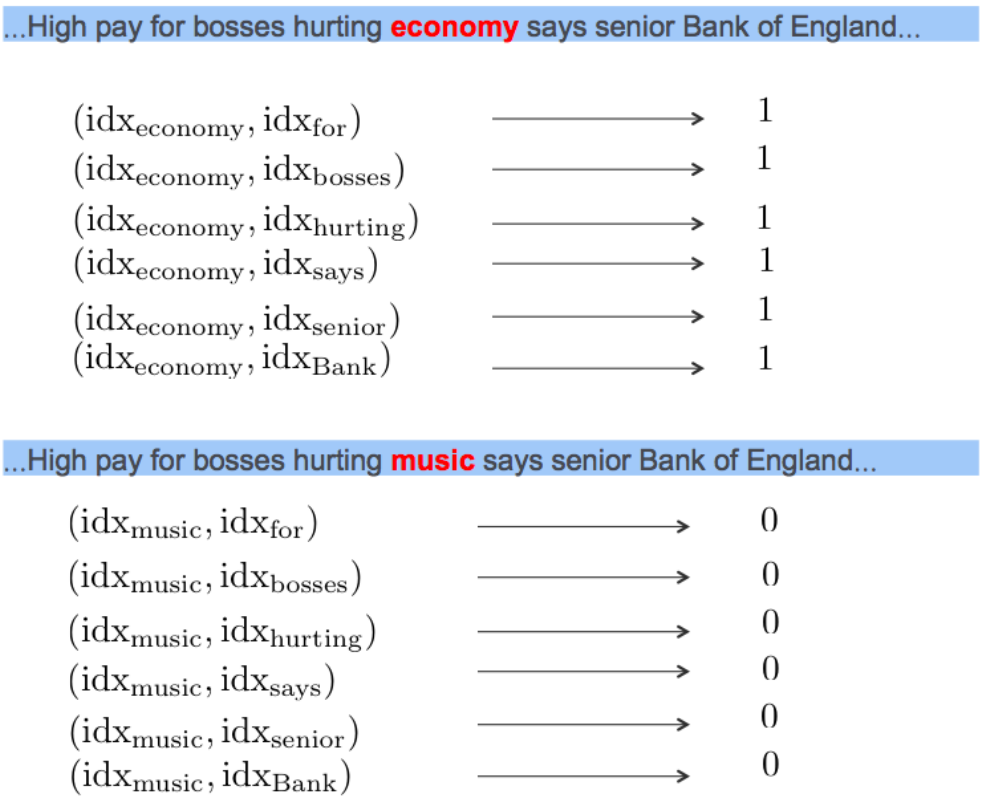

Figure 8: The positive and negative batch associated with the center word "economy" in the second document

3.3 Creating the training dataset

We have explained the way we create the positive and negative batches associated with a center word w_i^t in section 3.1.

In order to create the whole training dataset, we need to iterate the process by considering each word w_i^t in the

corpus for all $i \in \{1, \dots, N\}$ and for all $t \in \{1, \dots, n_i\}$ as a center word, and determine the positive batch $\left(\mathcal{B}_+^{w_i^t}\right)$ composed of the *true couples* associated with the center word w_i^t and the negative batch $\left(\mathcal{B}_-^{w_i^t}\right)$ composed of the *fake couples* associated with the fake center word f_i^t.

As shown in the example 3.2, Each element of the *true couples* is considered as a positive sample (with a label 1) and each element of the *fake couples* is considered as a negative sample (with a label 0).

Let N_T be the number of true couples (i.e, associated with a label 1).

i.e,

$$N_T = \sum_{i=1}^{N} \sum_{t=1}^{n_i} \text{Card}\left(\mathcal{B}_+^{w_i^t}\right)$$

Where Card (F) stands for the cardinality of F.

Question 16: What is the expression of N_T as a function of $(n_i)_{1 \leq n \leq N}$?.

Question 17: What is the number of fake couples (i.e, couples associated with a label 0)?.

4 Learning the Word Vectors

4.1 The Forward Propagation

We would like to use the shallow neural network represented in figure 9 in order to predict the context words from the center word. The hidden layer contains D neurons.

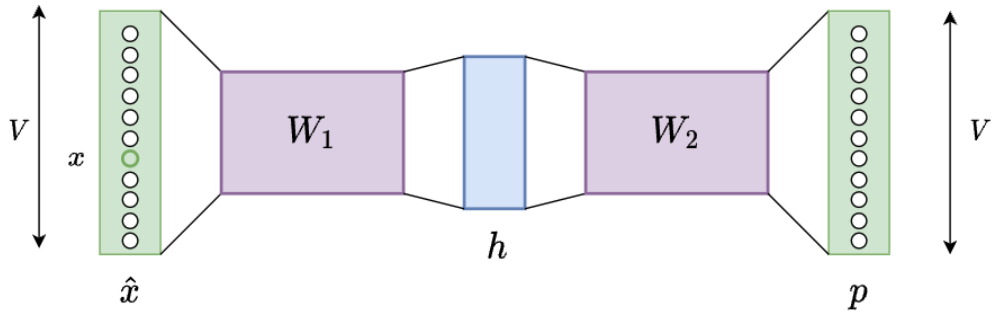

Figure 9: A Shallow Neural Network

Let us consider a center word of index $x \in \{0, \dots, V-1\}$ and $\hat{x} \in \{0,1\}^V$ the V-dimensional one hot vector associated with it.

A first linear transformation maps $\hat{x}$ to the D-dimensional vector h as follows:

$$h = W_1^T \hat{x}$$

A second transformation maps the hidden vector h to the V-dimensional vector $p = (p_1, \dots, p_V)$ as follows:

$$p = \sigma\left(W_2^T h\right) \quad \text{where } \sigma \text{ is the sigmoid activation function.}$$

We use the sigmoid activation function because we would like the o-th element of p (denoted p_o for $o \in \{0, \dots, V-1\}$) to represent the probability that the word of index o is in the **true** context of the word of index x.

In other words, p_o is the probability that the couple (x, o) is a *true couple*.

Question 18: What are the shapes of W_1 and W_2 ?

Let $W_1[0], W_1[1], \dots, W_1[V-1]$ be the rows of the matrix W_1 and $W_2[0], W_2[1], \dots, W_2[V-1]$ be the columns of the matrix W_2.

Question 19: Show that:

$$p_o = \sigma\left(W_1[x]^T W_2[o]\right)$$

Let us consider a center word w_i^t and the following positive and negative batches associated with w_i^t, each containing $K = 6$ couples:

- The positive batch $\left(\mathcal{B}_+^{w_i^t}\right) = \{(i_t, c_1), \dots, (i_t, c_K)\}$ composed of the *true couples* associated with a label 1.
- The negative batch $\left(\mathcal{B}_-^{w_i^t}\right) = \{(i_f, c_1), \dots, (i_f, c_K)\}$ composed of the *fake couples* associated with a label 0.

4.1.1 The Forward Propagation for the positive batch associated with the center word w_i^t

Figure 10 is a representation of the forward propagation of the one hot vector associated with the true center word i_t.

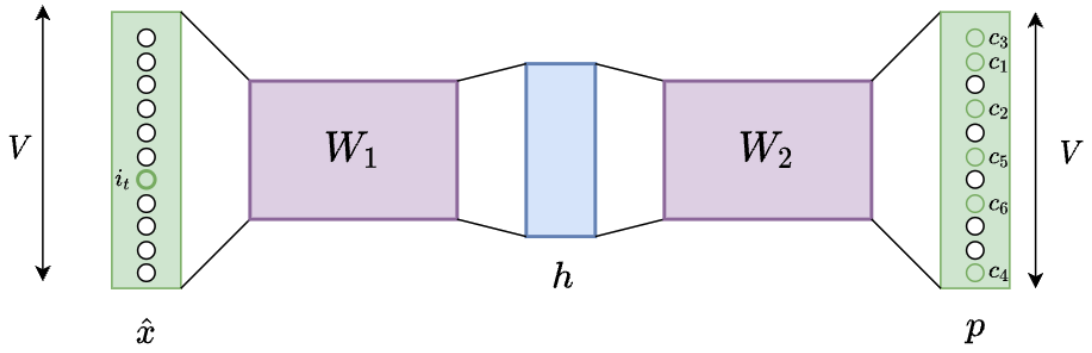

Figure 10: The forward propagation for the positive batch

We need to compare the predictions p_c for all $c \in \{c_1, \dots, c_K\}$ to the true targets 1.

Question 20: Explain why the loss associated with the positive batch $\left(\mathcal{B}_+^{w_i^t}\right)$ is the following:

$$J_+^{w_i^t} = -\frac{1}{K} \sum_{k=1}^{K} \log\left(\sigma\left(W_1[i_t]^T W_2[k]\right)\right)$$

Hint: Remember that if you have N training data for a binary classification problem, if p_n represent your

prediction for the sample n and t_n the associated target, then the loss function is the following:

$$J = -\frac{1}{N}\sum_{n=1}^{N}(t_n \log(p_n) + (1-t_n)\log(1-p_n))$$

4.1.2 The Forward Propagation for the negative batch associated with the center word w_i^t

Figure 11 is a representation of the forward propagation of the one hot vector associated with the fake center word i_f.

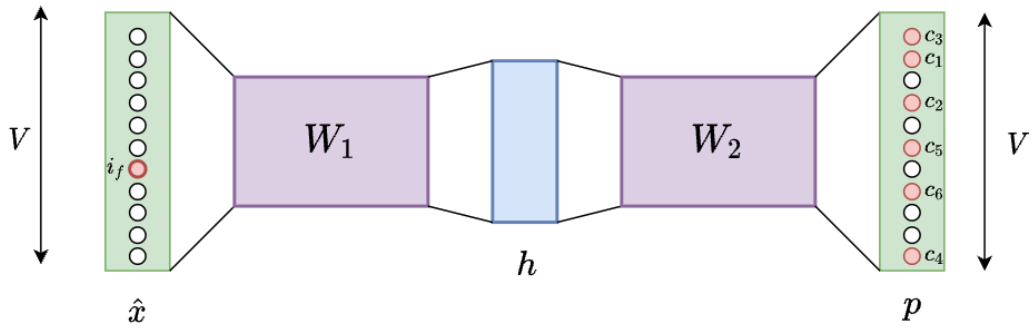

Figure 11: The forward propagation for the negative batch

We need to compare the predictions p_c for all $c \in \{c_1, \ldots, c_K\}$ to the true targets 0.

Question 21: Explain why the loss associated with the negative batch $\left(\mathcal{B}_-^{w_i^t}\right)$ is the following:

$$J_-^{w_i^t} = -\frac{1}{K}\sum_{k=1}^{K}\log\left(1-\sigma\left(W_1[i_f]^T W_2[k]\right)\right)$$

4.2 The Backward Propagation

We would like to update the parameters of the neural network $\theta = (W_1, W_2)$ after each batch using stochastic gradient descent.

4.2.1 Updating the parameters after the forward propagation associated with the positive batch

The forward propagation, calculated from the one hot vector of i_t, results in a V-dimensional prediction vector p.

Question 22: Which rows and columns of W_1 and W_2 are involved in the computation of p_c for all $c \in \{1, \ldots, K\}$?

(Hint: Use the result of Question 19)

In order to avoid calculating the gradients of the loss function $J_+^{w_i^t}$ with respect to all the parameters $\theta = (W_1, W_2)$. We are only going to update the gradients of the rows and columns of W_1 and W_2 involved in the forward propagation.

We get the following gradients:

$$\nabla_{W_1[i_t]} J_+^{w_i^t} = \frac{1}{K} \sum_{k=1}^{K} \left(\sigma(W_1[i_t]^T W_2[k]) - 1\right) W_2[k]$$
$$\nabla_{W_2[k]} J_+^{w_i^t} = \frac{1}{K} \left(\sigma(W_1[i_t]^T W_2[k]) - 1\right) W_1[i_t] \quad \text{for all} \quad k \in \{1, \dots, K\}$$

Let η be the learning rate.

Question 23: What are the update equations of the parameters $\theta = (W_1, W_2)$ after the forward propagation associated with the positive batch $\left(\mathcal{B}_+^{w_i^t}\right)$

4.2.2 Updating the parameters after the forward propagation associated with the negative batch

The forward propagation, calculated from the one hot vector of i_f, results in a V-dimensional prediction vector p.

Question 24: Which rows and columns of W_1 and W_2 are involved in the computation of p_c for all $c \in \{1, \dots, K\}$?

Question 25: What are the update equations of the parameters $\theta = (W_1, W_2)$ after the forward propagation associated with the negative batch $\left(\mathcal{B}_-^{w_i^t}\right)$

4.3 Learning the Embedding Matrix

Question 26: By combining all the previous steps, implement Algorithm 1

Algorithm 1 The Word2vec algorithm

Input: sequences (list of lists of integers), N_{epochs} (number of epochs)
Output: losses(list of losses associated with each epoch), W_1, W_2 (the trained parameters of the shallow neural networks)

Initialize the matrices W_1 and W_2 randomly.
for epoch in N_{epochs} **do**
    Initialise an empty list of losses: losses = []
    Initialise the loss associated with the whole corpus: $J = 0$
    for sequence $(\mathcal{D}_i)$ in sequences **do**
        for word w_i^t in sequence $(\mathcal{D}_i)$ **do**
            Get the true center word $i_t = w_i^t$
            Get the context words $\{c_1, \dots, c_K\}$
            Get the fake center word $i_f = f_i^t$
            Do one step of Stochastic Gradient Descent associated with the positive batch $\mathcal{B}_+^{w_i^t}$
            Calculate the loss $J_+^{w_i^t}$
            $J \leftarrow J + J_+^{w_i^t}$
            Do one step of Stochastic Gradient Descent associated with the negative batch $\mathcal{B}_-^{w_i^t}$
            Calculate the loss $J_-^{w_i^t}$
            $J \leftarrow J + J_-^{w_i^t}$
        end for
    end for
    Append the list of losses with the value J
end for

Question 27: Plot the list of losses from Algorithm 1

Question 28: Which embedding matrix can you build once the training process is finished ?

Question 29: Using an unsupervised learning algorithm of your choice, reduce the dimensionality of your embedding vectors into 2 dimensions and show a scatter plot of the reduced embedding vectors.

Hint: You can use a PCA for instance.

Question 30: Show an example of an analogy like $e_{\mathbf{France}} - e_{\mathbf{Paris}} \approx e_{\mathbf{England}} - e_{\mathbf{London}}$ from the corpus using your trained embedding vectors.