NLP
Assignment N.1

December 9, 2024

Bar, Alon	Ben Tzvi, Nehoray	Baron, Lia
205476013	206389538	036635803

1 Understanding word2vec

a. Equation (1) uses the softmax function. Recall you should in HW0 that softmax is invariant to constant offset in the input, i.e that for any input vector x and any constant c, $softmax(x) = softmax(x + c)$

Ans:

Let $softmax(x_i + c) = \frac{e^{x_i+c}}{\sum_{j=1}^{N} e^{x_j+c}}$

$\frac{e^{x_i+c}}{\sum_{j=1}^{N} e^{x_j+c}} = \frac{e^{x_i}e^{c}}{\sum_{j=1}^{N} e^{x_j}e^{c}}$ factoring out e^c

$\frac{e^{x_i}e^{c}}{\sum_{j=1}^{N} e^{x_j}e^{c}} = \frac{e^{x_i}e^{c}}{e^{c}\sum_{j=1}^{N} e^{x_j}}$

$\frac{e^{x_i}e^{c}}{e^{c}\sum_{j=1}^{N} e^{x_j}} = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$ canceling e^c in numerator and denominator

$\frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} = softmax(x_i)$ By definition

b. Verify you understand why the naıve softmax loss given in Equation (2) is the same as the cross-entropy loss between y and $\hat{y}$, i.e., show that

$-\sum_{w\in W} y_w log(\hat{y_w}) = -log(\hat{y_o})$

Ans: We will divide W into two parts, $w \in U, w \in V$

y is a one-hot vector with a 1 for the true outside word o , and 0 everywhere else.

For $w \in V$ w is a center word:

$y_w = 0 \rightarrow -\sum_{w\in V} y_w log(\hat{y_w}) = 0$

For $w \in U$ so w is outside word:

y is one hot vector so $\sum_{w\in V} y_w \rightarrow$ is canceled

$y_w = 1 \rightarrow -\sum_{w \in V} y_w log(\hat{y_w}) = -log(\hat{y_w})$
$\hat{y_w} = \hat{y_o} \rightarrow -log(\hat{y_w}) = -log(\hat{y_o})$

c. Recall that in class we showed that the partial derivative of $J_{\text{naïve-softmax}}(c, o, V, U)$ with respect to v_c is $u_o - \mathbb{E}_{o' \sim p(o'|c)}[u_{o'}]$. Now, compute the partial derivative of $J_{\text{naïve-softmax}}(c, o, V, U)$ with respect to each of the 'outside' word vectors, u_w's. There will be two cases: when $w = o$, the true 'outside' word vector, and when $w \neq o$, for all other words.

You should verify you understand why the answer is as follow:

$\frac{\partial}{\partial u_{w \neq o}} J_{\text{naïve-softmax}}(c, o, V, U) = \hat{y_w} v_c$

$\frac{\partial}{\partial u_{w = o}} J_{\text{naïve-softmax}}(c, o, V, U) = v_c(y^T \hat{y} - 1)$

Ans:

$\frac{\partial}{\partial u_{w \in V}} J_{\text{naïve-softmax}}(c, o, V, U) = \frac{\partial}{\partial u_{w \in V}} - log(P(O = o|C = c)$ (Eq1 from pdf)

$= \frac{\partial}{\partial u_{w \in V}} - log(\frac{exp(u_o^T v_c)}{\sum_{w \in W} exp(u_w^T v_c)})$ (Eq2 from pdf)

$= \frac{\partial}{\partial u_{w \in V}} - [log(exp(u_o^T v_c) - log(\sum_{w \in W} exp(u_w^T v_c)]$

$(\log\left(\frac{f(x)}{g(x)}\right) = \log(f(x)) - \log(g(x)))$

$= \frac{\partial}{\partial u_{w \in V}} - [u_o^T v_c - log(\sum_{w \in W} exp(u_w^T v_c)]$ $(\log(exp(f(x)) = f(x))$

$= \frac{v_c exp(u_o^T v_c)}{\sum_{w \in W} exp(u_w^T v_c)} - \frac{\partial}{\partial u_{w \in V}}(u_o^T v_c)$ $(\frac{d}{dx}\log(f(x)) = \frac{f'(x)}{f(x)}), \frac{d}{dx} e^{f(x)} = e^{f(x)} \cdot f'(x)$

$= v_c \hat{y_w} - \frac{\partial}{\partial u_{w \in V}}(u_o^T v_c) =$ (Eq1 from pdf)

$\hat{y_w} v_c$ when $w \neq o$ $(w \neq o \rightarrow (u_o^T v_c)$ is constant $\rightarrow \frac{\partial}{\partial u_{w \in V}}(u_o^T v_c) = 0)$

$\hat{y_w} v_c - v_c$ when $w = o$ $(w = o \rightarrow (u_o^T v_c) du = v_c \rightarrow \frac{\partial}{\partial u_{w \in V}}(u_o^T v_c) = v_c)$

$= v_c(y^T \hat{y} - 1)$ $(y^T \hat{y} = \hat{y_w}$ y is a one-hot vector)

d. Here, J(c,w_{t+j},V,U) represents an arbitrary loss term for the center word c = w_t and outside word w_{t+j}. J(c,w_{t+j},V ,U) could be $J_{\text{naive-softmax}}(c, w_{t+j}, V, U)$ or $J_{\text{neg-sample}}(c, w_{t+j}, V, U)$, depending on your implementation.

Write down three partial derivatives:

(i) $\frac{\partial J_{\text{skip-gram}}(c, w_{t-m}, \ldots, w_{t+m}, V, U)}{\partial U}$

(ii) $\frac{\partial J_{\text{skip-gram}}(c, w_{t-m}, \ldots, w_{t+m}, V, U)}{\partial v_c}$

(iii) $\frac{\partial J_{\text{skip-gram}}(c,w_{t-m},...,w_{t+m},V,U)}{\partial v_w}$ when $w \neq c$

Write your answers in terms of $\frac{\partial J(v_c,w_{t+j},V,U)}{\partial U}$ and $\frac{\partial J(v_c,w_{t+j},V,U)}{\partial v_c}$. You should verify you understand why the answer is as follow:

(i) $\sum_{-m\leq j\leq m} \frac{\partial J(v_c,w_{t+j},V,U)}{\partial U}$

(ii) $\sum_{-m\leq j\leq m} \frac{\partial J(v_c,w_{t+j},V,U)}{\partial v_c}$

(iii) 0

Ans:

$J_{\text{skip-gram}}(c, w_{t-m}, ..., w_{t+m}, V, U) = \sum_{-m\leq j\leq m} J_{\text{skip-gram}}(c, w_{t+j}, V, U)$ (By def.)

(i) $\frac{\partial J_{\text{skip-gram}}(c,w_{t-m},...,w_{t+m},V,U)}{\partial U} = \frac{\partial}{\partial U} \sum_{-m\leq j\leq m} J_{\text{skip-gram}}(c, w_{t+j}, V, U)$

$= \sum_{-m\leq j\leq m} \frac{\partial J(v_c,w_{t+j},V,U)}{\partial U}$

(ii) $\frac{\partial J_{\text{skip-gram}}(c,w_{t-m},...,w_{t+m},V,U)}{\partial v_c} = \frac{\partial}{\partial v_c} \sum_{-m\leq j\leq m} J_{\text{skip-gram}}(c, w_{t+j}, V, U)$

$= \sum_{-m\leq j\leq m} \frac{\partial J(v_c,w_{t+j},V,U)}{\partial v_c}$

(ii) $\frac{\partial J_{\text{skip-gram}}(c,w_{t-m},...,w_{t+m},V,U)}{\partial v_{w\neq c}} = \frac{\partial}{\partial v_{w\neq c}} \sum_{-m\leq j\leq m} J_{\text{skip-gram}}(c, w_{t+j}, V, U)$

$= 0$ (it's not part of the term)

e. Try to explain in a few sentences why it is important to split each token representation into two - first for its being the center token, and second for it being an output token.

Ans:

It's important to split to two representations:

(i) to get the information from the word when it's the center word

(ii) to get the information from the word when it's the outside word

This split is crucial because words have asymmetric relationships.

For example, "money" frequently appears near "spend" when "spend" is the center word, but when "money" is the center word, it might more strongly relate to words like "save" or "bank" rather than "spend".

These different roles require different representations to capture these asymmetric relationships effectively.

f. Finally, try to explain the intuition behind this algorithm. That is, why we might expect this algorithm to lead the model end up representing two semantically similar tokens with two "close" vectors within the Euclidean space.

Ans: The loss function pushes word vectors to have higher dot products with their context words' vectors.

When two words share many context words (which often happens with semantically similar words), they're being pushed to have high dot products with the same vectors.

The dot product is a main component in both loss functions, and higher dot products mathematically result in vectors becoming similar to each other in the Euclidean space.

2 Implementing word2vec

After 40,000 iterations, the script will finish and a visualization for your word vectors will appear. It will also be saved as word vectors.png in your project directory. Include the plot in your homework write up, inside the pdf (not a separate file). Briefly explain what you notice in the plot. Are there any reasonable clusters/trends? Are the word vectors as good as you expected? If not, what do you think could make them better?

Ans: According to 1

Looking at the word vector visualization, several interesting patterns emerge.

The model appears to have learned some basic gender-related semantic relationships, as evidenced by the proximity of words like "woman," "man," "female". Similarly, sentiment-based relationships are visible, with positive words such as "amazing," "wonderful," "brilliant," and "enjoyable" clustering together.

The model also shows some understanding of topical relationships, as seen in the clustering of weather-related terms ("rain," "snow," "hail") and beverage terms ("tea" and "coffee").

However, the quality of these word vectors could be enhanced, as the clusters aren't as tight or distinct as one might expect, and some semantically related words are positioned farther apart than they should be. To improve these results, several approaches could be considered.

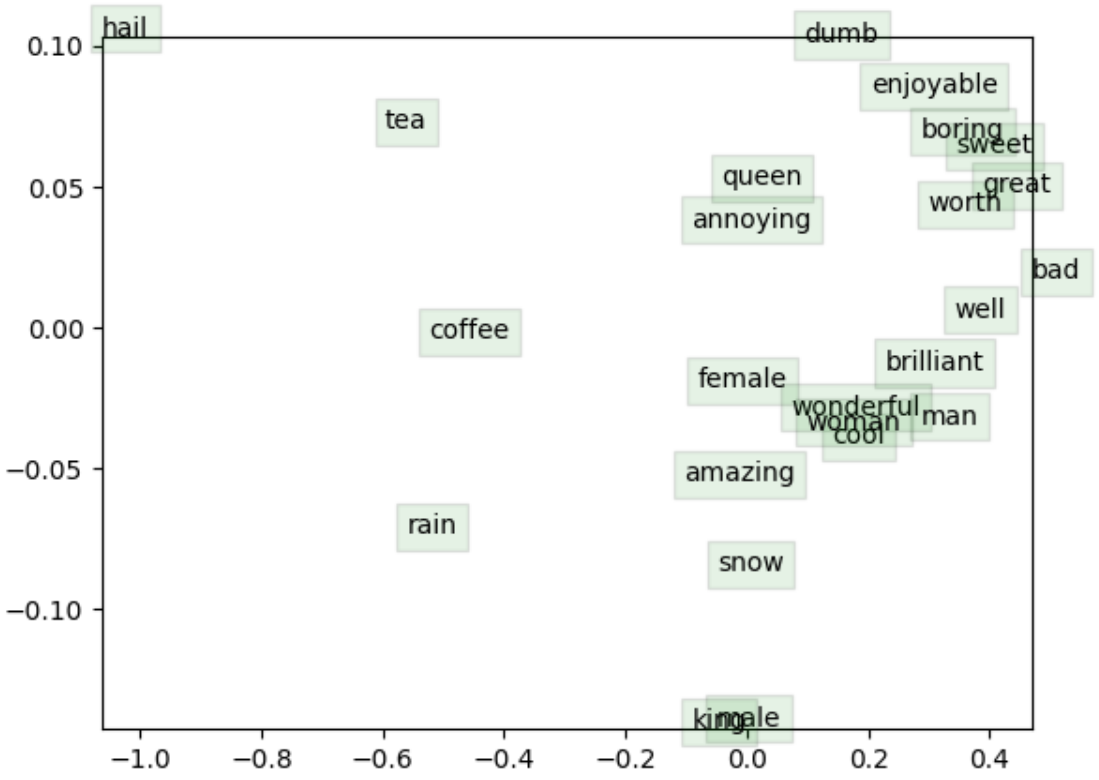

Figure 1: vectors results

Using a larger training dataset would likely lead to better semantic learning, while adjusting the context window size could help capture more relevant relationships. While the current vectors demonstrate some meaningful semantic patterns, there's significant potential for capturing these relationships more strongly with these modifications.

3 Implementing word2vec

a. Prove that if $\theta^* = argmax_\theta \mathcal{L}(\theta)$ then $p_{\theta^*}(o|c) = \frac{\#(c,o)}{\sum_{o'} \#(c,o')}$

Where # (c,o) = Number of co-occurrence of c and o in the corpus.

Hint 1: For a fixed c,o in the vocabulary, how many times does the term p_θ(o |c) appear in L(θ)

Hint 2: Use Lagrange multipliers

Ans:

$\theta^* = argmax_\theta \mathcal{L}(\theta)$ then $\theta^* = argmax_\theta log(\mathcal{L}(\theta))$ (log is monotonic function)

$log(\mathcal{L}(\theta)) = \sum_{t=1}^{T} \sum_{-m \le j \le m} log(p_\theta(w_{t+j}|w_t))$ (Def.)

$\sum_{t=1}^{T} \sum_{-m \le j \le m} log(p_\theta(w_{t+j}|w_t)) = \sum_{c \in V} \sum_{o \in U} \#(c,o) log(p_\theta(o|c))$

(Going throw the corpus and summing it is as counting the number of co-occurrence)

We will take $\sum_{o \in U} p(o|c') = 1$ (Def. of probability) as constraint for Lagrange

$h(\theta) = \sum_{c \in V} \sum_{o \in U} \#(c,o) log(p_\theta(o|c)) + \sum_{c \in V} \lambda_c (1 - \sum_{o' \in U} p(o'|c))$

($h(x) = f(x) + \lambda g(x)$ By Lagrange multipliers)

Deriving h(θ) by specific $p(o|c)$ to find the max:

$\frac{dh}{dp(o|c)} = \frac{\#(c,o)}{p_\theta(o|c)} - \lambda_c = 0$

(1) $p_\theta(o|c) = \frac{\#(c,o)}{\lambda_c}$ $(+\lambda, /\lambda, *p(o|c))$

(2) $\sum_{o' \in U} p(o'|c) = 1 \rightarrow \sum_{o' \in U} \frac{\#(c,o')}{\lambda_c} = 1 \rightarrow \lambda_c = \sum_{o' \in U} \#(c,o')$

$p_\theta(o|c) = \frac{\#(c,o)}{\lambda_c} = \frac{\#(c,o)}{\sum_{o' \in U} \#(c,o')}$ (1) & (2)

b. Let's assume each word is represented by a single scalar (real number). Prove that there is a corpus over a vocabulary of no more than 4 words, where reaching the

optimum solution is impossible. You can assume that a corpus is a list of sentences, such as "aa","bb","cc",...,. For the given corpus: aa,aa,aa,ab,ab,ac. We have:
$p(a|a) = 0.5, p(b|a) = 1/3, p(c|a) = 1/6$

Ans:

The vocabulary is a, b, c, d

The corpus is aa, ab, ac, ba

p(d|a) should be 0.

$p(d|a) = \frac{exp(u_o^T v_c)}{\sum_{w \in W}(u_w^T v_c)}\ exp(u_o^T v_c) > 0 \rightarrow p(d|a) > 0 \rightarrow p(d|a) \neq 0$

4 Paraphrase Detection

Consider the following model for paraphrase detection:
p(The pair is a paraphrase$|x_1, x_2) = \sigma(relu(x_1)^T relu(x_2))$
where relu(x) = max(0,x)

a. In this model, what is the maximal accuracy on a dataset where the ratio of positive to negative examples is 1:2?

Ans:

$relu(x_1), relu(x_2) \geq 0$ (definition of relu)

$relu(x_1)^T(relu(x_2) \geq 0$ (positive @ positive)

$\sigma(relu(x_1)^T relu(x_2)) \geq 0.5$ $(\sigma(x) = \frac{1}{1+e^{-x}}, x \geq 0 \rightarrow e^{-x} \leq 1)$

So this model never predict that $p < 0.5 \rightarrow$ so it always predict True

Ratio of positive to negative is 1:2 $\rightarrow$ p(The pair is a paraphrase$|x_1, x_2) = 1/3$

Accuracy $= \frac{TN+TP}{Total}, \frac{TP}{total} = \frac{1}{3}, TN = 0 \rightarrow Accuracy = \frac{1}{3}$

b. Suggest a simple fix for the problem.

Ans:

We will use bias term to fix the problem $\sigma(relu(x_1)^T relu(x_2) - b)$

We could remove the relu but the model will be more simple and might not capture enough complexity of this task.

We could use Leaky relu but there will be inductive bias towards positive predictions.

We could use at the end of the model instead of sigmoid $ffn(relu(x_1)^T relu(x_2))$ but it's more complicated

c. What evaluation metric should you use to evaluate the success of a model on this imbalanced dataset? Please explain why. (Consider: Accuracy, precision, Recall, ROC-AUC, AUC-PR, confusion matrix. Select one metric that will enable you to compare multiple models.)

Ans:

We should use **AUC-PR** to evaluate the success

Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$ Won't be good because prediction of always negative will give pretty good results (2/3)

Precision: $\frac{TP}{TP+FP}$ Focus on positive prediction which are less common in the data

Recall: $\frac{TP}{TP+FN}$ Ignores FP so predicting always positive will lead to mistaken results

ROC-AUC: $\int_0^1 \mathbf{TPR}(t) \cdot \mathbf{FPR}(t)\, dt$ This mathematical property makes it less informative for imbalanced datasets

AUC-PR: $\int_0^1 \mathbf{Precision}(R)\, dR$ The baseline depends on positive frequency but it capture imbalanced data

Confusion matrix: $\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$ Is very good for detailed analyses but it require human in the loop

5 TF-IDF

a. Now you're going apply TF-IDF on a case that might have been a real use-case for Question Answering systems. First, run the cells of code which construct the corpus, and define the query. Then, add code of your own that uses the TF-IDS algorithm you already implemented (you may edit/add new functions of course), in order to retrieve the most relevant sequences out of the corpus, to the query: What is the capital of New Zealand?. As a final answer, please print the 3 highest scored sentences out of the corpus

Ans:

New Zealand's capital city is Wellington, and its most populous city is Auckland Score: 0.7053

The South Island is the largest landmass of New Zealand Score: 0.5166

Elizabeth II is the queen of New Zealand and thus the head of state Score: 0.4207

6 Topic modeling

Question number 1:

a. What are the metrics used to evaluate Topic modeling (name at least two, not including perplexity)? Please explain each of them (Feel free to use the web to extend your knowledge)

Ans:

C_v **NPMI**: (Normalized Pointwise Mutual Information) measures the statistical association between words by computing the normalized pointwise mutual information. It calculates the probability of word co-occurrence compared to their individual probabilities, normalized to a range between -1 and 1. NPMI is computed by dividing the pointwise mutual information (logarithm of joint probability divided by individual word probabilities) by the negative logarithm of the joint probability. Positive values indicate words that appear together more frequently than expected by chance, while negative values suggest words that rarely co-occur.

C_v **Topic coherence**: each topic word is compared with the set of all topics. A boolean sliding window of size 110 is used to assess whether two words co-occur. Then, the confirmation measure consists of direct and indirect confirmations. For all N most probable words per topic, a word vector of size N is created in which each cell contains the NPMI between that word and word i. Then, all the word vectors in a topic are aggregated into one big topic vector. The average of all the cosine similarities between each topic word and its topic vector is used to calculate the score.

b. What benchmark datasets exist for this task? (name at least 3)

Ans:

• 20 Newsgroups Dataset: This dataset comprises approximately 20,000 newsgroup documents across 20 different categories, making it suitable for assessing the performance of topic models in distinguishing between various topics.

• Reuters-21578 Dataset: Containing thousands of news documents categorized into multiple topics, this dataset is widely used for evaluating topic modeling and text classification algo- rithms.

• OCTIS Benchmark Datasets: OCTIS (Optimizing and Comparing Topic models Is Simple) provides a collection of datasets specifically designed for topic modeling evaluation, including subsets of Wikipedia and other corpora.

Question number 3:

a. Now, we will evaluate the model performance on the benchmark dataset. We will use the gensim package for topic modeling evaluation. Complete and run the code in "Evaluation using Gensim" section. Note, you need to fill in the coherence metrics names (same as Q1.a). Verify the string format match the expected input to CoherenceModel according to the library documention. Write in the pdf the results: the model, the evaluation metrics names and their values

Ans:

The model: alexman83/BERTopic 20newsgroups base

The evaluation metrics and values:

c_v: 0.53

NPMI: -0.06

Question number 4:

a. Can you think of some aspect of the task that these metrics fail to evaluate, but human-evaluator may consider? Write your thoughts in the pdf

Ans:

Human evaluators value a model's ability to generate a diverse set of topics that cover different themes without significant overlap. Metrics like C_V and NPMI primarily focus on individual topic coherence and may not adequately assess the overall diversity of topics produced by the model.

Humans can discern whether topics are too broad or too specific, affecting their usefulness. Automated metrics may not effectively measure the granularity of topics.

Question number 5:

a. (Bonus:) If you are responsible for evaluating a new topic-modeling solution, how would you approach this task?

Ans:

Normalize NPMI and c_V to 0-1 values.

Composite score = $\beta_1 N(NPMI) + \beta_2 N(C_v)$

If you prioritize statistical co-occurrence, assign a higher weight to NPMI.

If semantic coherence is more critical, give more weight to C_V.

A balanced approach often involves setting both weights equally

Human Evaluation - Complement the composite score with qualitative assessments from domain experts to ensure the topics are meaningful and relevant.

Based on feedback, adjust the weights and normalization methods as necessary to better align the composite score with human judgments.