Natural language processing

About me

- Full name → Pooria Rahimi
- Student number → 99521289

Theory questions

1. Answer the following questions :
    - Explain the difference between one-hot encoding and word embeddings.
        - ▼ Answer :
            1. **One-hot Encoding :**
                - One-hot encoding is a method used to represent categorical data, including words in text.
                - In one-hot encoding, each word in a vocabulary is represented by a binary vector, where each dimension corresponds to a unique word in the vocabulary.
                - The length of the vector is equal to the size of the vocabulary, and only one element (dimension) in the vector is 1 (hot) while all others are 0 (cold), indicating the presence or absence of a particular word.
                - One-hot encoding treats each word as independent of every other word, resulting in a sparse representation where similarity between words is not captured.
                - The display of words is in the form of a binary sparse vector, where only one cell has a value other than 0, and its value is also 1. This type of display of words It is easy to implement, but it loses the inner meaning of the word in the sentences. As a result, the meaning of the sentence is lost.
            2. **Word Embeddings :**
                - Unlike One-hot encoding, this type of display also considers the meaning of the sentence in such a way that words with similar effect or meaning within a sentence, It gives similar values in one or more special features.
                - Word embeddings are dense, low-dimensional representations of words in a continuous vector space, where each word is mapped to a vector of real numbers.
                - Unlike one-hot encoding, word embeddings capture semantic relationships between words. Similar words are represented by similar vectors, and relationships such as similarity, analogy, and context are encoded in the vector space.
                - Word embeddings are learned from large corpora of text using techniques like Word2Vec, GloVe, or FastText, where words with similar contexts are mapped to nearby points in the embedding space.
                - These embeddings are trained in such a way that semantically similar words have similar vector representations, allowing for more nuanced understanding and manipulation of textual data.

            Key Differences :

            > **Representation :** One-hot encoding represents words as high-dimensional, sparse binary vectors, while word embeddings represent words as dense, low-dimensional real-valued vectors.

            > **Semantic Information :** Word embeddings capture semantic relationships between words, while one-hot encoding does not. Words with similar meanings have similar representations in the embedding space, facilitating tasks like semantic similarity, analogy detection, and context understanding.

            > **Dimensionality :** One-hot encoding results in high-dimensional sparse representations (size of the vocabulary), whereas word embeddings typically have lower dimensionality (e.g., 100 to 300 dimensions) and dense representations.

            > **Efficiency :** Word embeddings are more efficient in terms of memory and computational resources compared to one-hot encoding, especially for large vocabularies.
    - Briefly explain how the GloVe algorithm generates word embeddings.
        - ▼ Answer :
            1. **Co-occurrence matrix :** GloVe first creates a large matrix of co-occurrence, which indicates the number of repetitions of each word with another word in the corpus.
            2. **Matrix factorization :** Then the co-occurrence matrix is factorized to become a matrix with less dimensions. that each row is a display vector It can be from any word.
            3. **Semantic relationships :** The key idea is that these embeddings are designed in such a way that their dot product is multiplied by the logarithm. The probability of words is co-occurrence. The geometric relationship between the embedding vector of words is related to their semantic relationship.

            > Overall, GloVe generates word embeddings by iteratively optimizing word vectors to capture meaningful relationships between words based on their co-occurrence statistics in the text corpus.
    - Briefly explain how the Word2Vec algorithm generates word embeddings.

▼ Answer :

> 💡 Word2Vec is a popular algorithm used to generate word embeddings, which are dense vector representations of words in a continuous vector space. There are two primary approaches to Word2Vec: Continuous Bag of Words (CBOW) and Skip-gram.

> 💡 Word2Vec is a two-layer neural network that processes text to vectorize words.

1. **Continuous Bag of Words (CBOW):**
    - **Input and Output :** The input of this model is a text and its output is a series of feature vectors that express the words inside that text.
    - **Vectorization :** This algorithm vectorizes the words so that they can be understood by the computer. We can check the similarity between words by performing mathematical operations.
    - **Contextual Information :** Word2Vec creates vectors that express the characteristics of words in the form of distributed numbers. This algorithm performs this work without human intervention and in a self-supervised manner.
    - **Semantic Relationships :** Given enough data, this algorithm correctly constructs the meaning of words based on previous events.
    - **CBOW :** This model creates the target word based on its context. context includes a series of words before and after the word.
2. **Skip-gram :**
    - Skip-gram, on the other hand, predicts the context words given a target word.
    - It takes a target word as input and tries to predict the surrounding context words within a certain window.
    - The input layer consists of a one-hot encoded vector representing the target word, which is projected onto a hidden layer.
    - The hidden layer is then connected to multiple output nodes, each representing a context word. Similar to CBOW, the output layer uses softmax to make predictions.
    - During training, the model adjusts the weights to maximize the probability of predicting context words given the target word.
    - To obtain negative samples, it randomly samples other words in the vocabulary.
    - It uses the learned weights as embedding.

> Both CBOW and Skip-gram are trained using large corpora of text data through techniques like stochastic gradient descent. Once trained, the hidden layer weights (the embeddings) are used as representations of words in the continuous vector space, capturing semantic relationships between words. Words with similar meanings or contexts tend to have similar vector representations, allowing for various natural language processing tasks such as sentiment analysis, language translation, and text classification.

- How are words with multiple meanings controlled in word embeddings and what are the challenges do they produce ?

  ▼ Answer :

  - Old embedding methods such as GloVe and Word2Vec create a vector for each word without its context. (static embedding) that is The same words have the same vector, which causes the loss of semantic precision. However, today new methods such as ELMo and BERT are an embedding They create depending on the context. (contextual embedding)

  ### Challenges posed by words with multiple meanings in word embeddings include :

  - **Semantic Ambiguity :** Polysemous words introduce ambiguity into word embeddings, making it challenging for models to accurately capture the intended meaning of a word in a given context.
  - **Interference between Senses :** In traditional word embeddings, different senses of a polysemous word are often represented using the same vector, leading to interference between the senses and potentially affecting downstream tasks such as language understanding or generation.
  - **Data Sparsity :** Disambiguating the senses of polysemous words requires labeled data or external knowledge sources, which may be scarce or expensive to obtain, particularly for low-resource languages or specialized domains.
  - **Context Sensitivity :** Even with contextualized embeddings, accurately disambiguating the senses of polysemous words can be challenging, especially in cases where the context is ambiguous or the meanings of the word senses overlap closely.

  > 💡 For example, love and hate are possible have similar embedding because they are used in the same context.

- Word embeddings require all words to be present in the training set. How do you deal with words out of vocabulary? A method of production word embedding Suggest words that were not present in the training data.

  ▼ Answer :

  - **Nearest Neighbors :** Use a nearest neighbor approach to find similar words that are present in the training data and use their embeddings as a proxy for the OOV word.(use context)
  - **Fallback Strategies :** You can assign a default or placeholder vector representation for OOV words. This vector could be randomly initialized or set to zeros. However, this approach doesn't capture any semantic information about the OOV words and may not be suitable for downstream tasks.
  - **Character-Level Models :** Train embeddings at the character level instead of the word level. This way, even if a word is OOV, its character-level embeddings can still be utilized to represent it.
  - **Special tokens :** You can replace these words with a representative like UNK and use its embedding. To produce embedding It is also possible to replace words whose frequency is less than 10, for example, with UNK during training and make an embedding corresponding to it.
  - **Backoff to Character N-grams :** If a word is OOV, you can try to represent it using embeddings of its character n-grams or morphemes. This method can capture some semantic and syntactic information even for OOV words.

2. Write the co-occurrence matrix of the text below. Consider window size 2.

```
I love computer science and I love NLP even more.
```

▼ Answer :

1. First, we specify the following contexts :

> (I,love),(love,computer),(computer,science),(science,and),(and,I),(I,love),(love,NLP),(NLP,even),(even,more)

2. Now, based on the size of the window, which is equal to 2, and according to the existing contexts, we create the following matrix :

	I	love	computer	science	and	NLP	even	more	Cw
I	0	2	1	1	1	1	0	0	6
love	2	0	1	1	1	1	1	0	7
computer	1	1	0	1	1	0	0	0	4
science	1	1	1	0	1	0	0	0	4
and	1	1	1	1	0	0	0	0	4
NLP	1	1	0	0	0	0	1	1	4
even	0	1	0	0	0	1	0	1	3
more	0	0	0	0	0	1	1	0	2
C(context)	6	7	4	4	4	4	3	2	

💡 Then, to continue and obtain the probability, the values obtained in the matrix can be divided by the number of contexts, where the number of contexts is equal to 9, then the final probability can be obtained using the following formula. But this part is not the purpose of the question.

$$\mathrm{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Good luck😍