AI539 Natural Language Processing with Deep Learning – Homework 1

Word Vectors: Distributed Representations of Words

Ameyassh Nagarajan

0 Written Responses

▶ TASK 1.1 [2pt] Implement the `tokenize` function in `Vocabulary.py` that processes a string into an array of strings corresponding to tokens. You are free to implement any tokenization schema that eliminates punctuation. If you want to try out lemmitization, the `nltk` package may be helpful. In your writeup for this question, include a description of what you implemented and include an example input-output pair.

▶ TASK 1.1 Answer For the `tokenize` function I have used the several inbuilt functions in python. To remove the punctuations I have used the following code:

```
        new_text = text.translate(str.maketrans('','',string.punctuation))

```

The tokens are then stored in a list named `tokens` using the following code:

```
        tokens = new_text.lower().split()

```

▶ TASK 1.3 [5pt] Implement the `make_vocab_charts` function in `Vocabulary.py` to produce Token Frequency Distribution and Cumulative Fraction Covered charts like those above for your tokenizer and vocabulary cutoff heuristic. We recommend `matplotlib` for this. Afterwards, running `build_freq_vectors.py` will generate these plots. In your write-up for this question, include these plots and briefly describe the cutoff heuristic you implemented and explain your rational for setting it.

For these figures I have implemented a minimum frequency of 50.
Using a cutoff frequency of 50 allows to ignore words like 'the', 'is', 'and' which might occur frequently. This cuttoff also allows to reduce dimensions which might result in making the model more computationally efficient.

▶ TASK 2.1 [5pt] What are the minimum and maximum values of PMI (Eq. ??)? If two tokens have a positive PMI, what does that imply about their relationship? If they have a PMI of zero? What about if the PMI is negative? Based on your answers, what is an intuition for the use of PPMI?

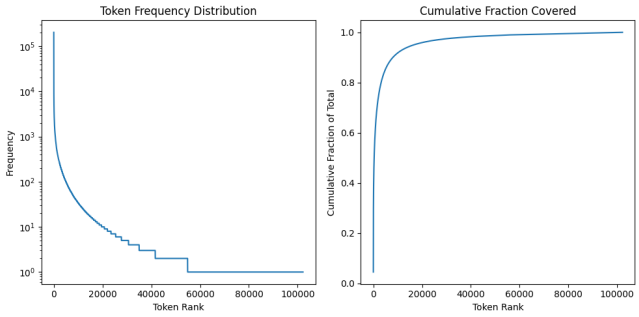

Figure 1: Cumulative and frequencies

The maximum value of PMI will be positive infinity and the minimum value will be negative infinity. The maximum value is achieved when two words always appear together, and similarly the minimum value is achieved when two words never appear together.
If two token have a positive PMI this indicates that they occur together frequently as opposed to when they have a negative PMI, they do not appear frequently. When two words have a PMI value of 0 this implies that the two words are independent of each other.
The intiuition for using PPMI is that using words that occur together helps in tasks like semantic analysis in Natural Language Processing.

▶ TASK 2.2 [2pt] Implement the `compute_cooccurrence_matrix` function in `build_freq_vectors.py` which takes a list of article overviews and a vocabulary and produces a co-occurrence matrix C. It is up to the student to define what a context is. Note that looping in Python is quite slow such that unoptimized versions of this function can take quite a long time. Feel free to save the result while you are developing to reduce time in future runs (see `numpy.save`/`numpy.load`). In your writeup for this task, describe how you defined your context.

I have defined the context using a window size of 2. The window size allows the program to refer to two words at the same time. This allows the model to understand what words a single word is being used with frequently. The code for my context definition is shown below:

```
def compute_cooccurrence_matrix(corpus, vocab):
  window_size = 2
  vocab_size = len(vocab.word2idx)
  C = np.zeros((vocab_size, vocab_size), dtype=int)
  word_2_indx = vocab.word2idx

  for s in corpus:
    words= vocab.tokenize(s)
    indx = [word_2_indx.get(word,word_2_indx['UNK']) for word in words]
    for center_indx, center_word in enumerate(indx):
      start = max(0,center_indx - vocab_size)
      end = min(len(indx), center_indx + vocab_size + 1)

      for context in range(start, end, window_size):
        if context != center_indx:
          context_word = indx[context]
          if context_word is not None:
            C[center_word,context_word] += 1

  return C

```

▶ TASK 2.4 [10pt] It has all led up to this! Run `build_freq_vectors.py` and examine the visualized word vectors. You should have semantically meaningful clusters in your plot. Zoom in and identify three such clusters. In your write-up, include an image of each cluster and what category you think it is representing. Also include the plot as a whole.

The full plot can be seen in this figure below and three clusters of words can be seen below:

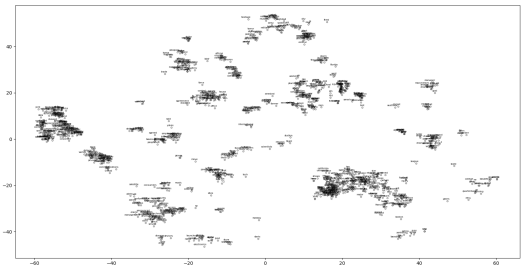

Figure 2: Word Vectors

Figure 3: Cluster representing words related to legislation

Figure 4: Words related to a normal conversation

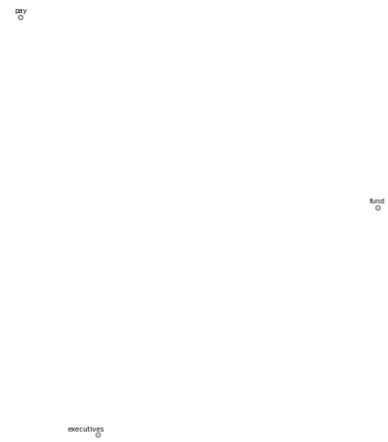

Figure 5: Cluster of words related to executives

▶ TASK 3.1 [10pts] Derive the gradient of the objective J_B in Eq.5 with respect to the model parameters w_i, $\tilde{w}_j$, b_i, and $\tilde{b}_j$. That is, write the expression for $\nabla_{w_i} J$, $\nabla_{\tilde{w}_j} J$, $\nabla_{b_i} J$, and $\nabla_{\tilde{b}_j} J$. Note that parameters corresponding to words not in B will have zero gradient.

Given the objective function

$$J_B = \sum_{(i,j)\in B} f(C_{ij}) \left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log C_{ij} \right)^2,$$

the gradients with respect to the model parameters are as follows:

- Gradient with respect to $\mathbf{w}_i$:

$$\nabla_{\mathbf{w}_i} J_B = \sum_{j:(i,j)\in B} 2f(C_{ij}) \left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log C_{ij} \right) \tilde{\mathbf{w}}_j$$

- Gradient with respect to $\tilde{\mathbf{w}}_j$:

$$\nabla_{\tilde{\mathbf{w}}_j} J_B = \sum_{i:(i,j)\in B} 2f(C_{ij}) \left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log C_{ij} \right) \mathbf{w}_i$$

- Gradient with respect to b_i:

$$\nabla_{b_i} J_B = \sum_{j:(i,j)\in B} 2f(C_{ij}) \left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log C_{ij} \right)$$

- Gradient with respect to $\tilde{b}_j$:

$$\nabla_{\tilde{b}_j} J_B = \sum_{i:(i,j)\in B} 2f(C_{ij}) \left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log C_{ij} \right)$$

▶ TASK 3.3 [5pts] Run `build_glove_vectors.py` to learn GloVe vectors and visualize them with TSNE! In your write-up, describe how the loss behaved during training (how stably it decreased, what it converged to, etc). Also include the TSNE plot. If everything has been done correctly, you should observe similar clustering behavior as in Task 2.3.

During the training the loss started at about 90% during the first few epochs and started decreasing as the number of epochs increased. After about 50-60 epochs the loss converged at about 25 to 24%. The TSNE plot can be seen in the figure 6 below:

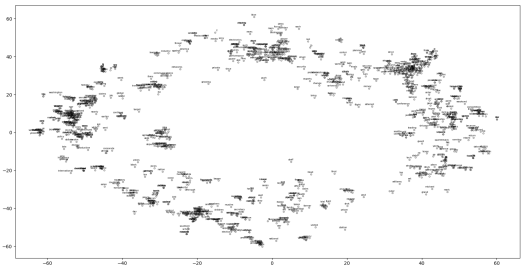

Figure 6: TSNE Plot

▶ TASK 4.1 [3pts] Use the `most_similar` function to find three additional analogies that work. In your response, provide the analogies in the compact `a:b::c:?` form, the model's list of outputs, and why you consider this output to satisfy the analogy.

▶ TASK 4.2 [3pts] Use the `most_similar` function to find three analogies that did not work. In your response to this question, provide the analogies in the compact `a:b::c:?` form, the model's list of outputs, and why you consider this output to not satisfy the analogy.

Some examples where the `most_similar` function works are mentioned below

analogy : night : sleep :: day :?

The function `most_similar` provides the following potential analogs with their corresponding similarity scores:

naps : 0.45,
doze : 0.438,
nap : 0.434,
restful_sleep : 0.433.

analogy : pen : book :: tablet :?

The function `most_similar` provides the following potential analogs with their corresponding similarity scores:

ebook : 0.524,
e_reader : 0.521,
eBook : 0.511.

analogy : tiger : jungle :: fish :?

The function `most_similar` provides the following potential analogs with their corresponding similarity scores:

nearshore_reefs0.499 :
mangrove_swamps : 0.496
sandy_bottoms : 0.494

Some examples where the function `most_similar` does not work:

man : house :: monkey :?

The function `most_similar` provides the following potential analogs with their corresponding similarity scores:

houses : 0.458,
monkeys : 0.418,
rabbit_hutch : 0.415.

The function `most_similar` provides the following potential analogs with their corresponding similarity scores:

music : relaxing :: disturbance :?

relaxation : 0.485,
Relaxing : 0.469,
relax : 0.449,
disturbances : 0.44.

apple : healthy :: ice_cream :?

The function `most_similar` provides the following potential analogs with their corresponding similarity scores:

healthier : 0.491,
heathy : 0.483,
eating_nutritious_foods : 0.451.

▶ TASK 4.3 [2pts] Use the `most_similar` function to find two additional cases of bias based on gender, politics, religion, ethnicity, or nationality. In your response, provide the analogies in the compact `a:b::c:?` form, the model's list of outputs for both `a:b::c:?` and `c:b::a:?`, and why you consider this output to be biased based on the model's two responses.

▶ TASK 4.4 [2pts] Why might these biases exist in `word2vec` and what are some potential consequences that might result if `word2vec` were used in a live system?

Examples where the function `most_similar` shows bias can be seen here

Consider the analogy:

$$\text{man} : \text{earns} :: \text{woman} :?$$

$$\begin{aligned} \text{receives} &: 0.633, \\ \text{garners} &: 0.597, \\ \text{earn} &: 0.566. \end{aligned}$$

These results may include bias related to gender roles that are present in the model's training corpus.

Consider the analogy:

$$\text{man} : \text{handsome} :: \text{woman} :?$$

$$\begin{aligned} \text{shapely} &: 0.556, \\ \text{comely} &: 0.534, \\ \text{bosomy} &: 0.524, \\ \text{curvacious} &: 0.518. \end{aligned}$$

These results highlights bias in describing the physical features of a man and a woman.

Consider the analogy:

$$\text{dress} : \text{good} :: \text{burkha} :?$$

$$\begin{aligned} \text{bad} &: 0.499, \\ \text{decent} &: 0.494, \\ \text{excellent} &: 0.484. \end{aligned}$$

These results reflect how there is a bias based on religion.

One of the reasons why these biases might exist in word2vec is mainly because of the kind of data it trains on and the associations made in the data for particular words. As word2vec uses context to learn word association, the way in which these words are used plays a role in the bias that is inherited by the model. Potential issues that might arise as a result of this bias is the reinforcement of stereotypes, biases can reinforce stereotypes which might result in limited visibility to underrepresented individuals. Bias might also lead to flaws in decision making as bias might influence a live systems decision making which might lead to potential unfair treatement.