MGTA 415 Homework 2

General Instructions and Tips

In HW1, we have tried bag-of-words with binary, frequency, and TF-IDFs as the representations of the documents, and used these feature vectors to train your text classifier.

For this HW2, we are going first to explore the usage of word embedding in text representations. Specifically, in Problem 1, we will work on the commonly used word embedding techniques – **Word2Vec and GloVe**, to encode words in a numeric format that is understandable and actionable by machine learning algorithms. This process is very similar to what we have done in HW1, and the only difference is that we are using different feature vectors.

In Problem 2 of HW2, we will try to work on some classical n-gram language models to consolidate your understanding. Most of the work could be done manually using paper and pens. You are, of course, welcome to implement the process in code to automate these calculations, but this is totally optional.

For the simplicity of grading, please prepare a Jupyter-Notebook for using a combination of code and markdown cells. Please **generate a PDF file with necessary intermediate results, comments, explanations/analyses for results, and answers to open-ended questions** based on the Jupyter-Notebook, and submit the PDF file onto Canvas, so the TA will have sufficient information to evaluate your work. These can be done in the same notebook with either markdown or code cells. Please submit your solution by the beginning of the Week 7 lecture, which is **on Feb 23**.

We will use the same dataset as HW1, **FinancialPhraseBank (FPB.csv)**. The dataset contain the sentiments for financial news headlines from the perspective of a retail investor. For text classification, please use the **Logistic Regression** classifier. You may find your work for HW1 is partly useful.

The classifier should be trained and tested on the FPB dataset. Shuffle the FPB data with random seed 42, and split it into **training, validation, and test splits, with an 80/10/10% ratio** (e.g., use `random_state` in `sklearn.model_selection.train_test_split`. You are supposed to train the model on the training set (e.g., using LogisticRegression), try different regularization setups and a few hyper-parameter choices (e.g., different C values), and check their performance on the validation set. According to their validation set performance, choose the best hyper-parameter combination. Following this best combination, train the LogisticRegression model on the training set again and then report its performance on the test set. You are encouraged to explore in more depth the scikit-learn API and try out various solver options.

Problem 1: Word Embeddings (20 points)

In this part, we are going first to explore the word embeddings — compare word2vec and GloVe qualitatively. And then, we will train a text classifier using the **100-dimensional** GloVe word vectors (i.e., "glove.6B.100d.txt" that we have worked on during the lecture). Compare and analyze their performances.

Let's start by experimenting with two ways of building the word vectors: GloVe and Word2Vec. These pre-trained word embeddings are publicly available. There are multiple ways to implement using the embeddings. One way is to directly download the file, load it up in python and process the file as a Python dictionary. Remember, you only need to use the 100-dimension file.

Or you can train the Word2Vec model yourself. Here we introduce **gensim** package. This is an open-source Python library for topic modeling, document indexing, and similarity retrieval with large corpora. You will get yourselves a dictionary mapping words to 100-dimension vectors. You will use the **FPB** headlines to build the vocabulary, and train the word2vec model.

a For each of the following words: `production, profit, acquisition, investment, job`, report the 10 most similar words based on word2vec and GloVe. Briefly discuss which method seems more reasonable to you or any other findings. **Note**: *This is an open-ended question. Feel free to propose new ideas.*

b The simplest way to construct sentence embedding from word embedding is to average the embedding of all words in a sentence, then you will get a 1D vector for each sentence. In this way, the order of words does not matter. Here we provide some pairs of sentences, calculate the cosine similarity of these sentences and give a brief discussion on whether these similarities are reasonable or not. What are the disadvantages of averaging word vectors for the document representation that you observe? Describe an idea to improve. **Note**: *This is an open-ended question. Feel free to propose new ideas.*

```
This movie is not bad and I would say I do enjoy it.
This movie is bad and I would say I do not enjoy it.

David is a cricket player and a opening batsman.
Leo is a cricket player too.  He is a batsman, baller and keeper.

I love horror movies.
Lights out is a horror movie.
```

c Use the above averaging word embedding method, and construct the feature matrix for the dataset with GloVe and word2vec word vectors, respectively. Feed the text classifier with a feature matrix, and re-run the text classification pipeline. Compare and analyze their performance.

Problem 2: N-gram Language Models (10 points)

a Here we provide a few sentences. You will use these to fit a unigram language model. Report the $P(w)$ for all possible words and punctuation. **Note**: *You do not need to write code. This is mostly a mathematics question.*

```
How about some major emerging economies?
Elephants are the largest living terrestrial animals in the world.
Weather derivatives are tradable commodities that protect business owners from future changes in the weather.
Most Sunday papers have comics, which children enjoy.
Da Vinci was brilliant in multiple fields!
```

b Now you have a unigram language model trained from the corpus. Use the model to calculate the probability of each following sentence.

```
She'd quickly look through the Greek story
Older children don't like being lectured at.
Intangible assets are of growing importance in the emerging knowledge-based economy.
```

c As you can observe, *zero* probability exists. How would you deal with zero probabilities? Are they reasonable? Is there any way to overcome this issue? Make it an open-ended discussion. **Note**: *This is an open-ended question. Feel free to propose new ideas.*