**M21AdvNLP: Advanced NLP** **Monsoon 2021**

# Assignment 1

Strict Deadline : 11-09-2020, 23:55 Hrs

*Instructor: Dr. Manish Shrivastava* *TA: Ananya Mukherjee, Manikanta Sai Nuthi*

## 1 General Instructions

1. The assignment can be implemented in Python.
2. Ensure that the submitted assignment is your original work. Please do not copy any part from any source including your friends, seniors, and/or the internet. If any such attempt is caught then serious actions including an F grade in the course is possible.
3. A single .zip file needs to be uploaded to the Moodle Course Portal.
4. Your grade will depend on the correctness of answers and output. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.
5. Late submissions will not be evaluated.

## 2 Problem Statement

Modern distributional semantic algorithms (also known as word embedding algorithms) can be found inside many intelligent systems dealing with natural language. The different types of word embeddings can be broadly classified into two categories:

**Frequency-based Embedding** There are generally three types of vectors that we encounter under this category: Count Vector, TF-IDF Vector, Co-Occurrence Matrix with a fixed context window.

**Prediction-based Embedding** There are generally two types of vectors that we encounter under this category: Continuous Bag of Words(CBOW), Skip-Gram.

The ultimate goal is to learn meaningful vectors (embeddings) forwards in natural language, such that semantically similar words have mathematically similar vectors. The current assignment aims to make you familiar with the above algorithms. In this assignment, you need to implement the following tasks :

1. Implement a Word embedding model and train your own word vectors using Co-Occurrence Matrix by applying Singular Value Decomposition (SVD). [10 marks]
2. Implement the Word2vec model and train your own word vectors using CBOW model. [10 marks]

**For each of the models, report the following for the model:**

- Display the top-10 word vectors for 5 different words (a combination of nouns, verbs, adjectives etc) using the above pre-trained models (1,2) using t-SNE (or such methods). [5 marks]
- What are the top 10 closest words for the word ‘camera’ in the embeddings generated by your program. Compare them against the pre-trained word2vec embeddings that you can download off the shelf (can use gensim). (1,2). [5 marks]

**Note on Terminology:** The terms ”word vectors” and ”word embeddings” are often used interchangeably. The term ”embedding” refers to the fact that we are encoding aspects of a word’s meaning in a lower dimensional space. As Wikipedia states, ”conceptually it involves a mathematical embedding from a space with one dimension per word to a continuous vector space with a much lower dimension”.

## 3 Training corpus

Please train your model on the following corpus:
`http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Electronics_5.json.gz`

## 4 Submission Format

Zip the following into one file and submit in the Moodle course portal:

1. Source Code
2. Pre-trained model and Embeddings generated
3. Report answering all the questions raised above.
4. Readme File :on how to execute the code, how to restore the pre-trained model. Any other information.

If the pre-trained models and embeddings cross the file size limits: upload them to a OneDrive/GDrive and share the links in the Readme File.

## 5 Grading

1. Evaluation will be individual and will be based on your viva, report, submitted code review.
2. In the slot you are expected to walk us through your code, explain your experiments, and report.

## 6 Queries

1. **Vocab size: Do we train for all the unique words that we see in the corpus?**
   You can place a limit: words with frequency less than 5 can be ignored. If you have enough compute train for as many words as possible.

2. **Do I need to ignore the stop words?**
   Usually stop words occur with very high frequency. You don't have to ignore the stop words. To avoid the effect of the words(stop words) which occur very frequently you can use sub sampling. This usually speeds up training.

3. **Do I need to use Lemmatisation before feeding the samples to the model?**
   No. For the sake of this assignment do no worry about Lemmatisation.

4. **I do not have enough compute. Can I reduce the vocab size ?**
   For reference, you can easily implement a model with vocab size of 4,00,000 words on a 8 GB RAM, decently sized CPU, home laptop. You can make use of Google Collab for implementing this, which offers greater ram and better CPUs.

5. **Can I submit my code in jupyter notebooks?**
   Yes , you can submit your code base in jupyter notebooks.

6. **Explain the stuff we have to report?**
   In section 2 we have asked you to report two aspects for each of the model.

   When we say "display" we are asking you to pick 10 nearest words for a particular word in terms of cosine similarity. After having picked the top 10 words plot them on 2-D plot. You can use t-sne(or such) to reduce dimensions.

   When we say compare against pre-trained word2vec embeddings :
   a) you can download any english pre-trained vectors that are available, and
   b) get the 10 closest words for "camera" in the embeddings generated by you.
   and compare the two outputs.

   You can use libraires like Scikit-learn, gensim to read vector files and pick nearest k words for a query word.

## 7 Reference Material

1. http://jalammar.github.io/illustrated-word2vec/

2. https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

3. https://cs224d.stanford.edu/lecture_notes/notes1.pdf