# SP25 IS567 Text Mining

# Assignment 5: Word Embeddings and Neural Networks

­­­

Due Date: 4/11/2025, 11:59PM CT

**Goals:**

The goal of this assignment is to help you gain hands-on experience with word embeddings and convolutional neural networks (CNNs). You will use a tweet sentiment classification dataset (same as the dataset in assignments 1-3). In this assignment, you will:

* Create your own word embeddings, then compare them with pretrained GloVe embeddings.
* Train a CNN with the pretrained GloVe embedings.

**Deliverables:**

The submission must include **a narrative document as a PDF file** and **the original script/program that you implemented to produce the results** and should be uploaded to Canvas.In your narrative document, you can include screenshots and figures as necessary to answer the questions. **If you include screenshots and figures, make sure that they are readable in a PDF file that is printed on paper.** Make your writing informative!

**Task 1: Create word embeddings**

* Import the dataset (*train.txt and test.txt*) provided on the Canvas assignment page (<https://canvas.illinois.edu/courses/52836/assignments/1166358>).
  + Load the training and test set into df\_train and df\_test respectively.
  + Convert the ratings to negative, neutral, and positive to make it a three-way classification for later tasks, as you did in assignment 3.
    - Group labels {-2: moderately negative emotional state can be inferred, -3: very negative emotional state can be inferred} -> negative, {-1: slightly negative emotional state can be inferred, 0: neutral or mixed mental state can be inferred, 1: slightly positive emotional state can be inferred} -> neutral, {2: moderately positive emotional state can be inferred, 3: very positive emotional state can be inferred} -> positive
* Convert df\_train – “Tweet” column and df\_test – “Tweet” column to lowercase.
* Create a new dataframe df\_all by concatenating df\_train and df\_test.
* Using the text in df\_all – “Tweet” column, create word2Vec embeddings (first with CBOW, then with skip-gram) based on the text from df\_all – “Tweet” column.
  + - Use NLTK to tokenize the “Tweet” text.
    - Train word embeddings using word2vec. Consult course material and additional information in the following link to support training with CBOW or skip-gram: <https://radimrehurek.com/gensim/models/word2vec.html>.
  + Use the embeddings created by different methods to find the top 20 terms most similar to "people", "smile", "amazing", and "time".

**Task 2: Use pretrained word embeddings**

* Load the pretrained GloVe model “glove.6B.100d” provided on the Canvas assignment page (<https://canvas.illinois.edu/courses/52836/assignments/1166358>).The model is pretrained on a much larger dataset with around 6B tokens. The dimension of the embedding vectors is 100.
* Use the GloVe embeddings to find the top 20 terms most similar to "people", "smile", "amazing", and "time".
* Discuss the differences between the terms generated by the pretrained GloVe embeddings and the embeddings you trained in Task 1.

**Task 3: Classification with embeddings**

* Classification training input: df\_train “Tweet” column;

training labels: df\_train “Intensity Class” column;

test input: df\_test “Tweet” column;

test labels: df\_test “Intensity Class” column.

* Use the token embeddings, which are generated by the embedding models you created (experiment with different kinds of embeddings you created: word2vec – skip-gram and word2vec-CBOW), as features to train logistic regression classifiers.
* Use the token embeddings generated by the pretrained GloVe model as features to train another logistic regression classifier.
* Compare the performance of different logistic regression models created in the previous two steps. Discuss your observations.
* Compare the models that use embeddings in this assignment and the models created in assignment 3 using count-based features. Discuss your observations.

**Task 4: Build a Convolutional Neural Network (CNN) classifier with embeddings**

* You can run “Tweets\_Emotion\_CNN.ipynb” (<https://canvas.illinois.edu/courses/52836/assignments/1166358>) on your local machine and train the model using a CPU. Alternatively, you can run it on a GPU by using Google Colab, which will speed up the training process.
* If you plan to use Google Colab, open <https://colab.research.google.com> and sign up for a Google Colab account or use your existing account. Open “Tweets\_Emotion\_CNN.ipynb” on the Canvas assignment page using Google Colab.

Change your Runtime setup to GPU: click on Runtime → Change Runtime Type → click “T4 GPU” option as shown in the screenshot below. T4 is the only choice for the free plans. More powerful GPUs (e.g., A100 GPU) require paying for an upgrade, but you should be able to do the assignment using T4 GPU.

A screenshot of a computer program

Description automatically generated

* Start from the beginning of the notebook, run each section of code, and answer the following questions. Make sure to run the code following the order in the notebook and do not skip any section.
* Explain the TextDataset function. In particular:
  + What are the inputs and outputs?
  + What does the function do?
  + What is the vocabulary size?
* Explain the CNNModel function. In particular:
  + What are the inputs and outputs?
  + What does the function do?
  + Explain the lines

“conv1\_out = F.relu(self.conv1(x))

conv2\_out = F.relu(self.conv2(conv1\_out))

conv3\_out = F.relu(self.conv3(conv2\_out))”

* + Explain the lines

“pooled = F.max\_pool1d(conv3\_out, conv3\_out.size(2)).squeeze(2)

out = self.dropout(pooled)

out = self.fc(out)”

* Explain the create\_glove\_matrix function. In particular:
  + What are the inputs and outputs?
  + What does the function do?
  + Do you find any connection between create\_glove\_matrix function and TextDataset class?
* Explain the train\_and\_eval function. In particular:
  + What are the inputs?
  + What does the function do?
* Comparing the classification performance by CNN and by other machine learning models created in task 3, what do you observe?

**Bonus: Try to improve your results**

Now that you have a vanilla CNN text classification model (Tweets\_Emotion\_CNN.ipynb), you can explore alternative approaches to enhance its performance. For example, you can use different optimizers (Adagrad, SGD, AdamW, instead of Adam) and different pretrained word embeddings (e.g., fastText). When comparing the results of the vanilla CNN model with your alternative approach, which one yields better results? Why?