**BA820 Team 3 Final Deliverable - Topic Identification for AG News Corpus**

Members: Yuanshan Zhang, Yiming Wang, Jiayun Liu, Mengxin Zhao, Yahui Wen

**Problem Definition**

It's fair to say a large number of documents are unclassified in our day-to-day life, which can be a problem when we want to sort them later. For example, when a library seeks to reorganize its unclassified files from twenty years ago, the common approach is to label (classify) them one by one using manpower, at the cost of time, money, low productivity, and infeasible to scale.

**Motivation**

With this problem in mind, the team is interested in seeing if we can come up with a more sufficient way in terms of "labeling", like employing unsupervised machine learning techniques like NLP methods to auto-identify the similarity between documents, piling highly-similar ones together for further organization, along with integrating supervised machine learning skills and deep learning. We aim to explore a system that efficiently categorizes news/documents using headlines, returns with high precision, and can extract valuable insights and trends from the news. This would point to a possible direction of practical model building, and provide convenience and confidence to stakeholders in the news landscape.

**Data Source**

Archived from AG News corpus collected by Antonio Gull, covering news headlines, the dataset contains 30,000 training and 1,900 test examples for 14 classes. The source can be accessed [here](https://nlpprogress.com/english/text_classification.html).

**Data Analysis**

***Methods Application:***

The dataset contains 14 different categories, and we randomly selected 5,000 rows from four of them - Business, Entertainment, Sports, and Sci/Tech, to ensure the balance of samples.

First, we tokenized the text which allowed us to create a word cloud to display frequent words (Figure 1). Subsequently, text vectorization/embedding was performed, including Bag of Words, N-grams, TF-IDF, and GloVe (glove.6B.100d). We also used Word2Vec, which maps words into vector space to capture semantic information and combine them through simple averaging and TF-IDF weighted averaging. Additionally, Doc2Vec is used to make these vectorized data more intuitive. We visualized the word vectors of the first row, it helped us to understand and interpret the characteristics of the data and the behavior of the model. Right after vectorization, we attempted to apply unsupervised learning to cluster the dataset. We were able to visualize the clusters after PCA. There appear to be 2 clusters that are quite far from each other, but when we tried to plot the silhouette score for k=2-7, there is no indicator for an optimal number of clusters. We plotted the silhouette scores for up to 20 clusters, and the score shows an upward trend and it will get even higher if the number of clusters increases. To save computational space, we threw away that graph and proceeded with supervised learning by applying a variety of supervised learning models. First, we model our different vectorization results with a random forest model and compare their prediction accuracy. In addition, we applied Naive Bayes, Logistic Regression, XGBoost, and SVM. Finally, we tried to apply some deep learning algorithms, including transfer learning with LSTM and then ran the regular LSTM and simple RNN.

Methods related to association rules do not apply to our project because our data is specifically for NLP, so it is more suitable for applying different NLP methods rather than association rules.

***Results & insight:***

We used Bag of Words as a starting point, to observe the vocabulary of the corpus. It also allowed us to graph the frequency of each word (Figures 4 & 5). From the charts, we learned the distribution of word frequency, as well as the ratio between common and rare words. Out of 52,238 unique words in the corpus, there were only 996 words that appeared more than 100 times; further, there are words that have been used more than 4,000 times. We used a 20% test size and a random state of 42 to ensure the consistency of our results. The first model we tested was Random Forest, we used N-Gram, TF-IDF, Word2Vec average, Word2Vec TF-IDF weighted average, Doc2Vec, and GloVe. From the classification report, GloVe performed the best in precision, recall, f1-score, and accuracy. We also used Naive Bayes, Logistic Regression, XGBoost, and SVM on TF-IDF and GloVe. TF-IDF performed slightly better in Logistic Regression and SVM (highest accuracy across all models and methods) but was outperformed in all other models with bigger differences in scores. Interestingly, in all the methods and models we have tried, the classification for Sports news performed the best among all four categories.

Overall, the embedding method from GloVe in SVM performed the best. A possible explanation is that 94.55% of the words in the corpus appeared less than 30 times, and more than 80% of words appeared less than 5 times. Since we created our vectorization/embedding and trained the models on a limited amount of data, a pre-trained NLP model would work better.

***Deep Learning - Sequential Modeling:***

The fundamental technique that today’s LLMs (Large Language Models) are built upon is the RNN (Recurrent Neural Network). However, it is hard for RNN to keep track of early information due to gradient exploding/vanishing issues. This usually happens when the sequence length (i.e. time step) is greater than 100. LSTM (Long Short-Term Memory) partially fixes this problem with gate control, thereby being called Long Short-Term Memory. To investigate the differences between these deep learning techniques, we built 2 LSTMs using with transfer learning method (i.e. W2V) and embedding method (i.e. embedding is performed on the embedding layer) and 1 RNN(Figure 6). Preprocessing procedures are as follows:

1. *Embed each token into a fixed-size vector (n\_dim<<n\_token)*
2. *For each document, arrange vectors to form sequence*
3. *Pad sequences so that all documents have the same length*
4. *One-hot encode the target*

On one hand, LSTM with embedding has 5 million more parameters to train during embedding, leading to a longer training process. On the other hand, LSTM with W2V converges faster because vectors are initialized by a pre-trained W2V model. However, LSTM performs better on the validation set than LSTM with W2V since embedding without pre-trained models fits deep learning models better into the linguistic context specific to the task rather than into a general linguistic context, which can also explain the overfitting issue of LSTM with embedding layer.

The sequence length after padding is 90, and after the convolution layer and max pooling layer, the length is reduced to 21 (less than 100). Therefore, RNN should be a good choice for this task without running into serious gradient exploding/vanishing issues. From the training result, we can observe that the validation accuracy of RNN is close to LSTM, and both of them tend to overfit(Figure 7).

CNN (Convolutional Neural Network) has proved its efficiency in combined with sequential modeling. In our case, we used 128 1D 5×100 (5 is the dimension along a sequence, 100 equals the embedding dimension) convolutional kernels and max pooling of size 4 to extract features, thereby reducing from 90-time steps to 21-time steps, but increased vector dimensions from 100 features/time step to 128 features/time step. Our major observation of applying the convolutional and max pooling layer is that this approach somehow managed to maintain the spatial relationships of embedded vectors without destroying it like traditional dimension reduction or feature transformation techniques do(Figure 8).

**Challenges**

*1.Change dataset*

Initially, we utilized the food-recipe review dataset. However, after model construction, we encountered issues with the data: the 'best\_score' didn't correspond well with positive/negative reviews, which means we cannot use it as a label to test our model. Besides, the distribution of the data is very uneven over time. Hence, we switched to the AG News dataset, which provided category labels for test.

*2.Pure text dataset*

Since this dataset lacked numerical values, conducting exploratory data analysis (EDA) became challenging. Eventually, we opted to visualize tokenized word frequency distributions and create word clouds for EDA.

*3.Colab crashing*

Another significant challenge we encountered was the limited computational space available on Google Colab. After we combined everything in one notebook, it could not execute to the end without crashing. During the process of model execution, we tried to avoid storing embedding data frames as variables, but there was zero improvement in RAM usage. We minimized the frequency of storing outputs into variables to reduce computational cost and optimized the cell block for Word2Vec TF-IDF weighted average embedding for significantly faster processing speed.

**Conclusion**

In conclusion, our project has thoroughly explored various vectorization methods and classification approaches to identify the most effective model for text classification tasks. Through experimentation with different vectorization techniques, including self-trained, pre-trained models as well as diverse classification methods, we have found that the TF-IDF combined with SVM gave the best-performed model, achieving an accuracy of 0.82. Additionally, our experiments with the GloVe model, trained by Stanford University, have demonstrated a comparable accuracy of 0.81 with a vector dimension of only 100. These findings prove that as a pre-trained model, GloVe can understand the meaning of text data very well. Furthermore, we conducted comparisons between traditional models and deep learning models such as RNN and LSTM. GloVe outperformed deep learning models. Possible reasons can be: First, the sample size is too small for deep learning models to fathom the context as well as GloVe; second, the context might not be very complex such that TF-IDF works well; third, we did not design a sophisticated network structure for the deep learning model and using convolution and pooling layers to extract features reduced the information we have.

Turning to commercial applications, our outcome of text classification techniques can be utilized in various domains. Analyzing news titles and descriptions facilitates sentiment analysis, trend analysis, and risk assessment, enabling businesses to gain insights into public perception, identify emerging industry trends, and anticipate potential threats or crises. These capabilities empower organizations to make informed decisions and strategic adjustments, enhancing their competitive edge and mitigating risks effectively.

**Appendix**

Link to Github:

<https://github.com/olaf-ys/Ba820-Group3>

Link to Contribution Sheet:

<https://docs.google.com/document/d/1T2J-Wv7fYu-UBfxRJ37Oq95XRNkvZ_oqfxYqSefowXQ/edit?usp=sharing>

Figure 1:

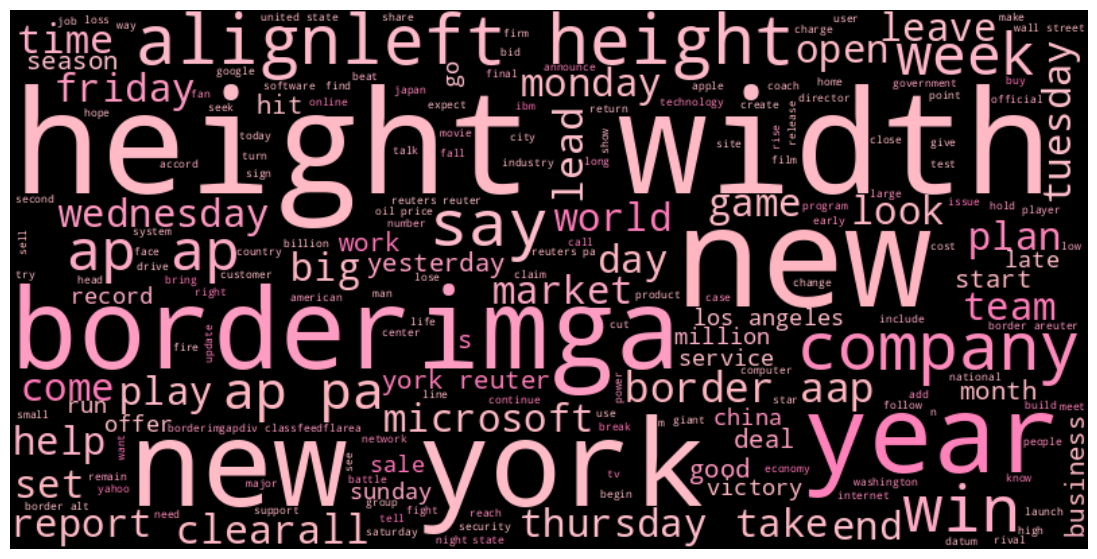
****

Figure 2:

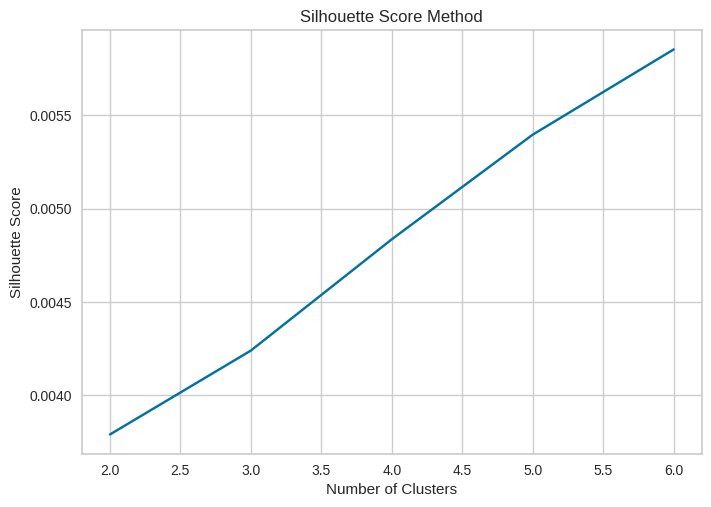


Figure 3:

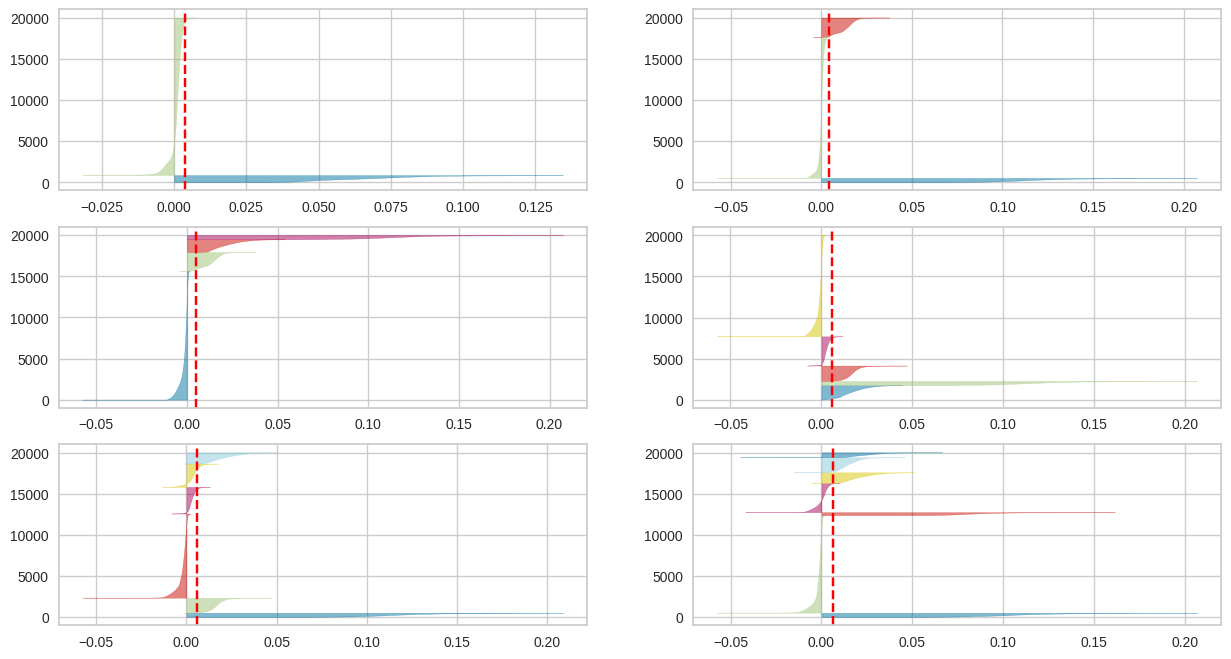
****

Figure 4:

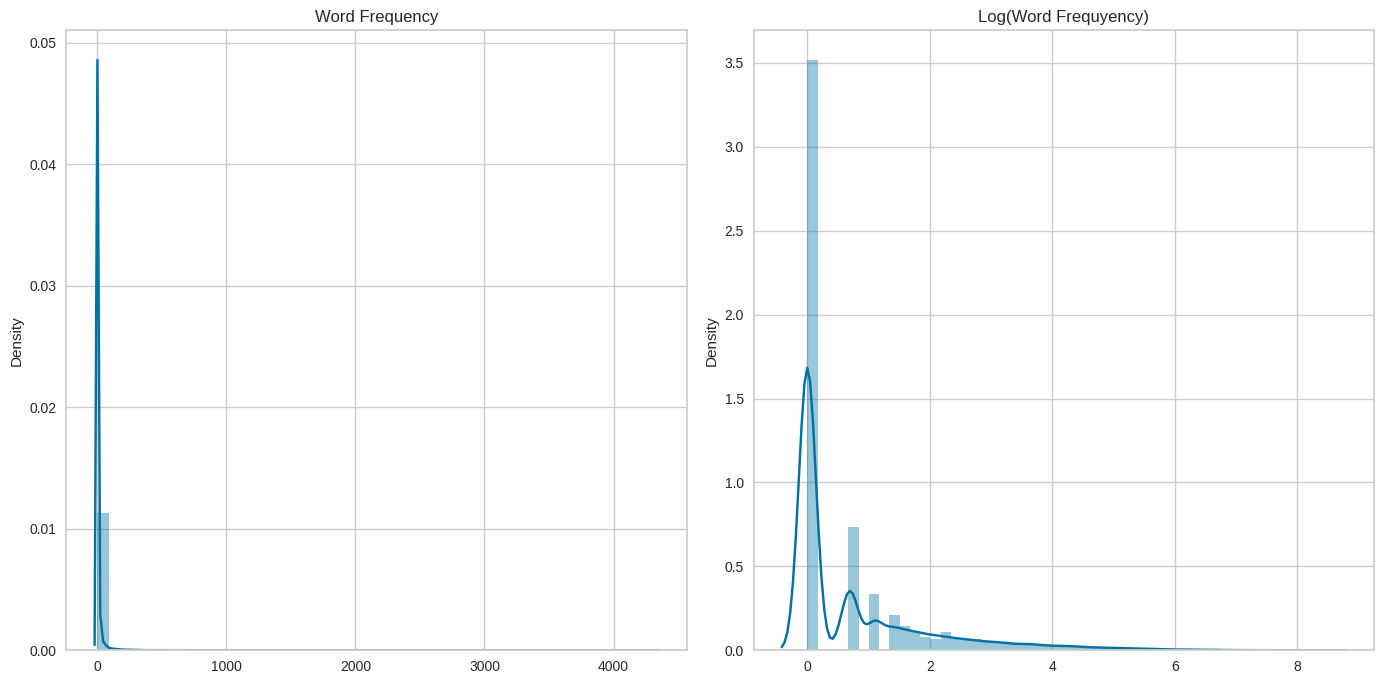


Figure 5:

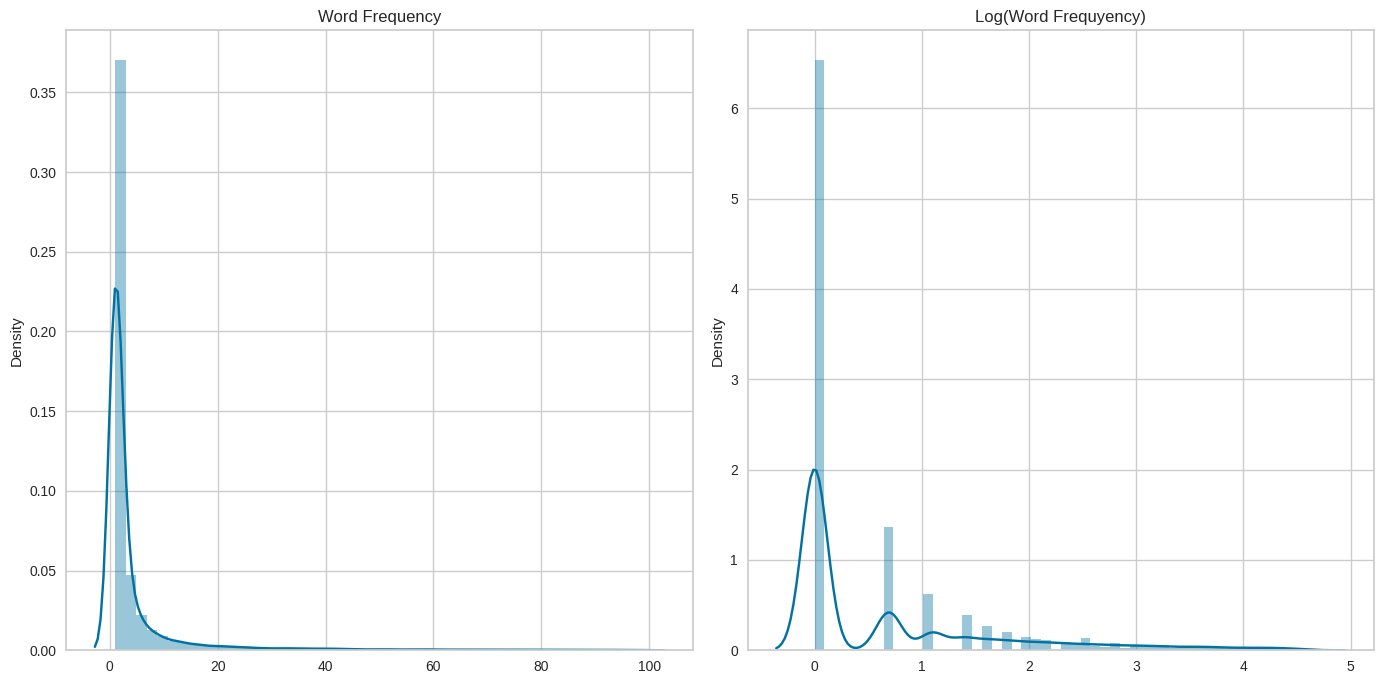
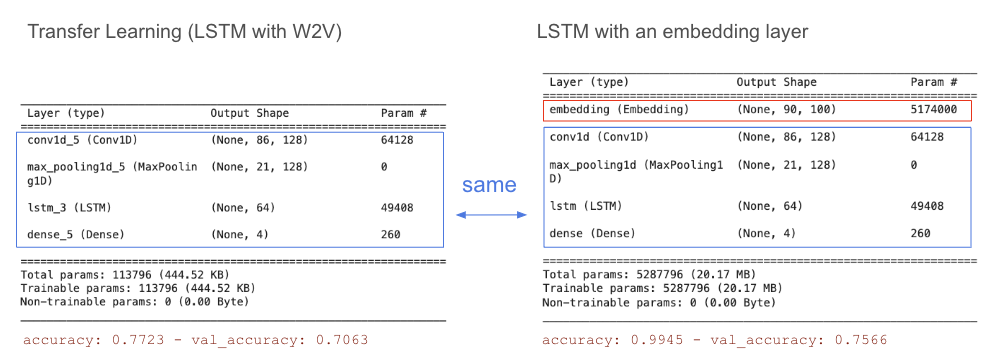
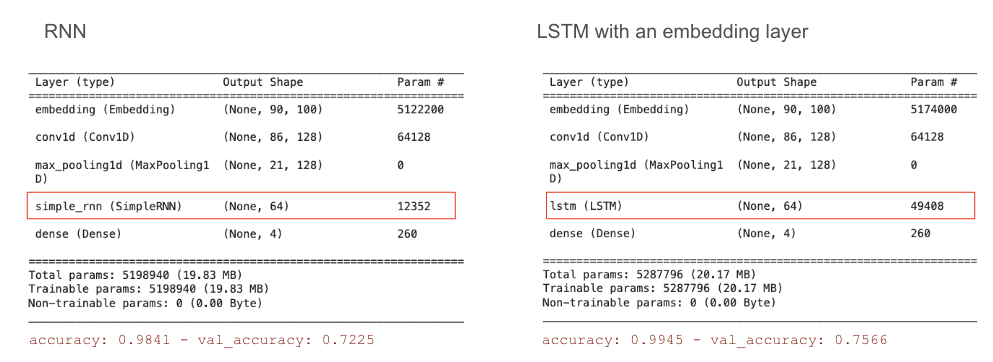
****

Figure 6:



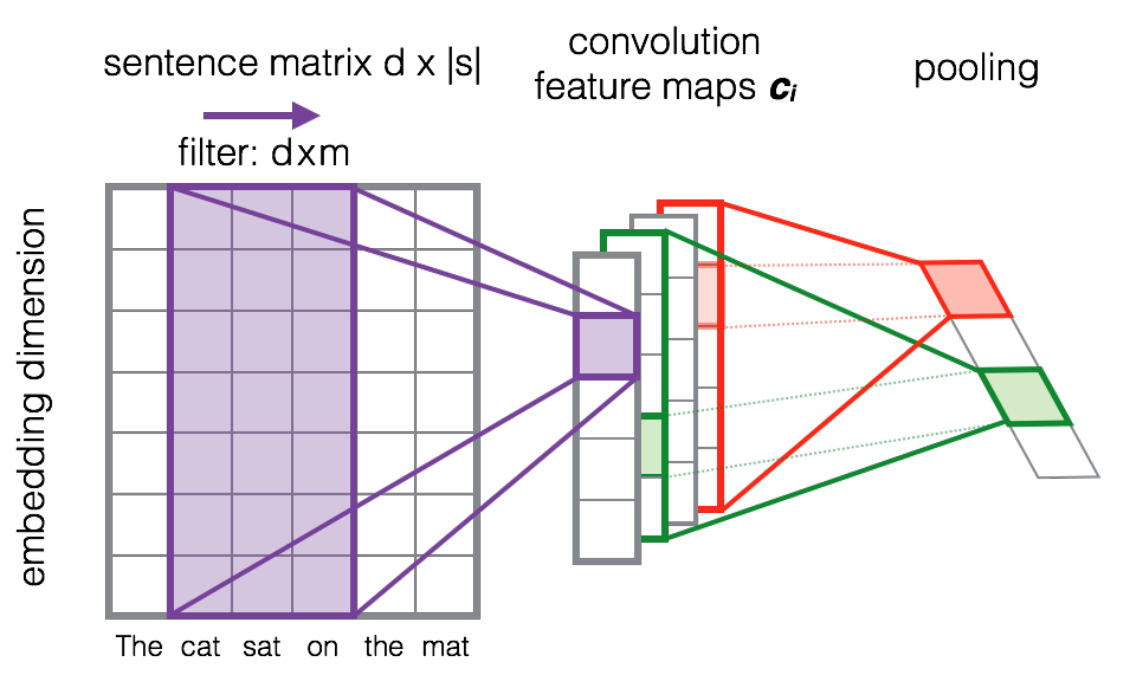
Structural design of LSTM with W2V and LSTM with an embedding layer are similar except for the embedding layer.

Figure 7:



Structural design of RNN and LSTM with an embedding layer are similar except for their hidden layers.

Figure 8:



Contribution Sheet:

|  |  |  |
| --- | --- | --- |
| Member | Contribution | Percentage |
| Yuanshan Zhang | * Set up GitHub repository and write instructions on how to coordinate on the Github * Connect to the data source * Convert text to csv * Clean and preprocess text * Vectorize the text using: tf-idf, bow, w2v, d2v * Document level representation of w2v by averaging and tf-idf weighted averaging * Use 1D Convolution and Max Pooling to generate feature map * Apply LSTM with embedding layer * Apply LSTM using W2V * Apply RNN | 32% |
| Yiming Wang | * Composed motivation * Applied clustering models * Dataset skepticism and issue explanation * Combined second notebook, optimized computational space * Created word frequency plot, counted proportion of rare words and determine min\_count * Added some markdown, analysis, and conclusion | 17% |
| Jiayun Liu | * Generated Data Dictionary during Recipe Review phase * Organized and complied codes together, and documented markdown during Recipe Review phase * Ran GloVe during AG-News phase * Evaluate the performance of GloVe during AG-News phase * Checked cosine similarity within GloVe during AG-News phase | 17% |
| Mengxin Zhao | * Data preprocessing and data cleaning * EDA, Word Cloud of AG News Dataset * Conclusion and Challenge part in report * Slides integration * Classification models: SVM, NaiveBayes, XGBoost, Logistic Regression on TF-IDF vectors * PCA, 3-d plot, K-means and Silhouette plot on TF-IDF. * Report integration | 17% |
| Yahui Wen | * Divided ‘best\_score’ into 5 bins * Processed data about time * EDA: Drew bar plots to see the distribution of reviews * Apply and tune n-gram to do the vectorization * Apply RandomForest Classification to test ngram * Apply SVM on TF-IDF vector matrix * Contribute to slides, documents and notebook markdown | 17% |