NAME: SHIVAPRAKASH BALASUBRAMANIAN **EMAIL ADDRESS**: sbalas22@ncsu.edu
NAME: SHREYA SABU **EMAIL ADDRESS:** ssabu2@ncsu.edu

PROJECT REPORT

TITLE

Sentence Ordering and Coherence Modeling using Neural Networks

DESCRIPTION

Modeling the structure of coherent texts is a key NLP problem. A well-written text has a particular high-level logical and topical structure. The actual word and sentence choices and their transitions come together to convey the purpose of the text. Our primary goal is to build models that can learn the structure by arranging a given set of sentences to make coherent text.

Many tasks like Multi-document Summarization and retrieval based Question Answering involve extracting information from multiple documents and organizing it into a coherent summary. Since the relative ordering of sentences from different sources can be unclear, being able to automatically evaluate a particular order is essential.

RELATED WORK

The ordering problem is traditionally formulated as a binary classification task: Given a reference paragraph and its permuted version, identifying the more coherent one. There have been pre existing methods [3,4] that capture local coherence by modelling patterns of entity distribution. Sentences are represented as syntactic roles of the entities in the document and features from the entity grid are used to train the SVM. Another approach is to model the **internal structure** in documents using syntax features. For the 2 methods, the feature extraction is decoupled from the target downstream tasks. This can limit the model's capacity to learn task specific features.

Sentence ordering is a task where the correct order of sentence is found, from some arbitrary ordering. We do not assume the availability of a set of candidate orderings to choose from and instead find a good ordering from all possible permutations of the sentences. RNNs and sequence to sequence networks are used to generate context from the current sentence. One possible result is the inclusion of ordering as a feature in NLP and NLU tasks would in general improve the performance of any model.

PROPOSED APPROACH

1. To solve sentence ordering, the sentences have to be represented as embeddings which can be achieved in the following ways [the best way has to be evaluated for the scenario]:
    a. ELMO encoding
    b. One Hot Encoding
2. Perform Sequence prediction on the sentences using LSTM and Multi-Layer PErceptron algorithm.

IMPLEMENTATION:

The first main task was to collect the data on which we would try to solve the ordering problem. We decided on 2 different sets of data:

1. arXiv Paper Abstracts: https://www.kaggle.com/neelshah18/arxivdataset
2. NIPS Paper Abstracts: https://www.kaggle.com/benhamner/nips-papers?select=papers.csv

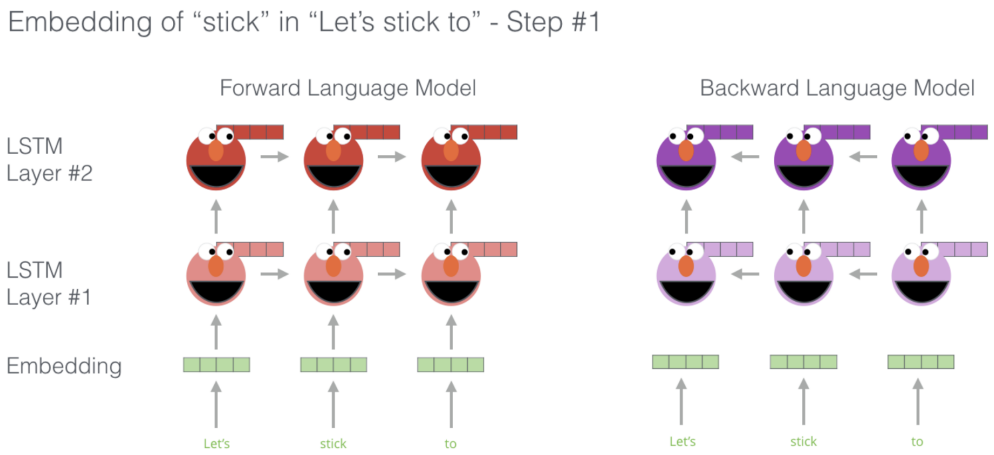
ArXiv has served the public and research communities by providing open access to scholarly articles. We used a free, open pipeline on Kaggle to the machine-readable arXiv dataset: a repository of 1.7 million articles, with relevant features such as article titles, authors, categories, abstracts are present [5]. Our main point of interest is the summary articles where we process the sentences and evaluate different permutations of the same. The hope is to empower new use cases that can lead to the exploration of richer machine learning techniques that combine multi-modal features towards applications like trend analysis, paper recommender engines, category prediction, co-citation networks, knowledge graph construction and semantic search interfaces.

To solve the problem of Sentence ordering we first had to generate sentence level embeddings and we tried out the 2 following approaches:

- ELMO Embeddings: Technique to generate word level embeddings. The sentence level embeddings were generated from the word embeddings and produced a final sentence embedding which was fed along with the others to the sorting Network. In this way the sentence embedding is learned during training. We used ELMO since it assigns different word vector representations for the same word as it also takes the context in which the word occurs into consideration.

- One Hot Encoding: Rather than using any pretrained embedding, we also tried experimenting with directly using raw data.

Text preprocessing and text normalization techniques such as lemmatization was used to prepare the words in the texts for further processing. Example : Words like 'Want', 'Wants', 'Wanted', 'Wanting' will be converted to it's common root form 'Want'. Such a mapping is useful to normalize the texts. Lemmatization will be done using the WordNetLemmatizer function provided by the nltk stem python library package.



We then produce the word embeddings using ELMO. ELMO is a NLP framework that is used for representation of word embeddings. In ELMO, the same word can have different word vectors under different contexts as the ELMo vector assigned to a token or word is actually a function of the entire sentence containing that word. Since ELMO also considers the context in which the word occurs, this is what sets ELMO apart from traditional word embeddings such as word2vec and GLoVe. To create those embeddings ELMO uses a bi-directional LSTM. Using the word embedding an embedding matrix is generated comprising the list of all words and their corresponding embeddings. Word embedding is the vector representation of words. Several pre-trained word embedding models are available. For the purpose of our project we have used ELMO.

We used the LSTM and MLP (Multi-layer Perceptron algorithm) networks to rank the sentences within a text -

- LSTM models are used for the sequential data. LSTM are better for sequential data then traditional neural networks. There is dependency between the words and the sentences in a text. Thus the sequence matters. By training our model using LSTM, we will predict this sentence ordering. The embedding layer is used to represent the words. And then after adding the LSTM layer, we flatten the output.
- We also experimented using the Multi-layer Perceptron algorithm to rank or order the sentences within a text by taking the embedding layer used to represent the words as input. A multi-layer percep-tron (MLP) is a deep, artificial neural network. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with even only one hidden layer are capable of approximating any continuous function. Multilayer perceptrons are often applied to supervised learning problems.

INNOVATION:

One hot encoding is allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories). In this case, a one-hot encoding can be applied to the integer representation. We thought trying this method to train the network may improve the performance of the model. Analogously, we plan to use the same to represent the order of sentences. For eg, a position 3 order in a paragraph of 4 sentences is represented as 0010.

This would be something new for use to work on. We plan to use LSTMs which are basically RNNs that capture the essence of the data processed till that point, i.e. the result of the previous part is fed as an input to the next layer. LSTMs have a memory component and it memorizes words in long sentences or paragraphs and helps capture the context.

EVALUATION METRICS:

Accuracy is the simplest way to check the performance of the model. For every sentence the predicted positioning of the sentence is checked against the actual position. Since accuracy takes a huge hit even if one of the sentences is off position creating a domino effect, another metric, Kendall Tau measure can be used.

Kendall Tau is a measure of rank correlation: the similarity of the orderings of the data when ranked by each of the quantities. The Kendall correlation between two variables will be high when observations have a similar rank between the two variables, and low when observations have a dissimilar rank between the two variables. We will primarily use rank correlation to evaluate the model.

CHALLENGES:

- Lack of baseline data to test the primary experiment
- The complexity of the coherence comprehension depends on the nature of the text and also the number of lines in the paragraph
- Existing approaches focus mostly on defining and using sophisticated features to capture the cross-sentence argumentation logic and syntactic relationships. This makes it difficult to scale methods to different datasets.
- argumentation semantics and cross sentence syntax are very hard to formalize and require domain specific work for every dataset.

RESULTS and CONCLUSION:

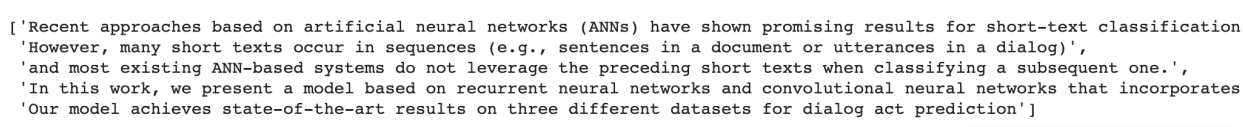
Github Code: https://github.com/shiva96b/CoherenceModel

Compare models and write paragraphs about how shitty one model is compared to another.

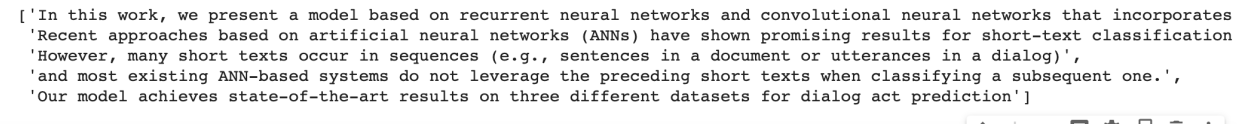
The arXiV dataset was run against 2 models, first a simple sequence to sequence neural networks and then an LSTM model that we referred to in the paper. We tried to implement the bert universal encoder but decided to go with elmo as it had a faster training time. The results of the same are given below:

	Accuracy	Kendall Tau Measure
Sequence to Sequence	0.13	0.33
LSTM	0.27	0.49

An example of a good shuffle ordering:



```
['Recent approaches based on artificial neural networks (ANNs) have shown promising results for short-text classification
 'However, many short texts occur in sequences (e.g., sentences in a document or utterances in a dialog)',
 'and most existing ANN-based systems do not leverage the preceding short texts when classifying a subsequent one.',
 'In this work, we present a model based on recurrent neural networks and convolutional neural networks that incorporates
 'Our model achieves state-of-the-art results on three different datasets for dialog act prediction']
```



```
['In this work, we present a model based on recurrent neural networks and convolutional neural networks that incorporates
 'Recent approaches based on artificial neural networks (ANNs) have shown promising results for short-text classification
 'However, many short texts occur in sequences (e.g., sentences in a document or utterances in a dialog)',
 'and most existing ANN-based systems do not leverage the preceding short texts when classifying a subsequent one.',
 'Our model achieves state-of-the-art results on three different datasets for dialog act prediction']
```

FUTURE WORK:

- More epochs and larger training dataset
- Better neural network architecture to capture the semantics
- Hyper-parameter Tuning to refine results

REFERENCES

1. https://arxiv.org/abs/1611.02654
2. https://www.kaggle.com/benhamner/nips-papers
3. https://people.csail.mit.edu/regina/my_papers/coherence.pdf
4. https://archiv.ub.uni-heidelberg.de/volltextserver/28281/
5. https://www.kaggle.com/Cornell-University/arxiv
6. http://jalammar.github.io/illustrated-bert/
7. https://medium.com/syncedreview/representations-for-language-from-word-embeddings-to-sentence-meanings-3fae81b2379a
8. https://www.aclweb.org/anthology/P18-1030/