Convolutional Neural Networks and Word-Embedded for Text Classifier

CSCE689 NLP:: Instructor: Ruihong Huang

427000469 Ya-Chuan Hsu
825008857 Chia-wei Chang

Outline

- Motivation
- Project Description
- Background Introduction
- Design and Implementation
- Testing Results and Evaluation
- Conclusion

Motivation

It has become a common practice for any kinds of reviews to have a major impact on the decision of the potential customers of behavior.

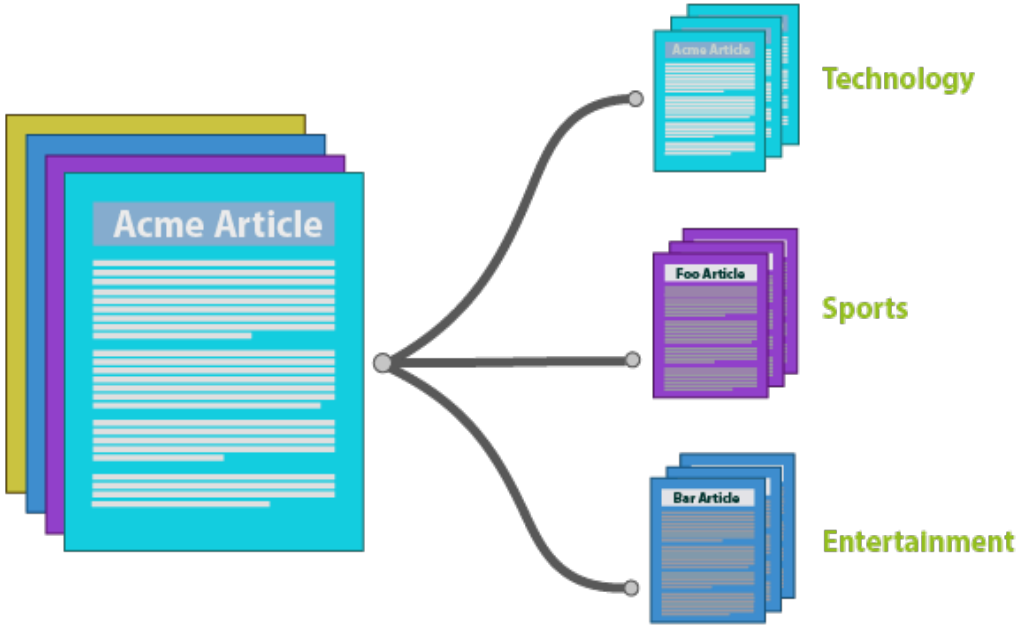
Digitization has changed the way we process and analyze information. There is an exponential increase in online availability of information reviews.

Text classification is a smart classification of text into categories. And, using machine learning to automate these tasks, just makes the whole process super-fast and efficient.

Also, text documents are one of the richest sources of data for businesses: whether in the shape of customer support tickets, emails, technical documents, user reviews or news articles, they all contain valuable information that can be used to automate slow manual processes, better understand users, or find valuable insights.

Motivation cont.

We want to push our project beyond existing text classifier model and gain hand-on experience with Deep Learning at the same time. By solving news categorization task with CNN and word embeddings.



Project Description

Our first goal was to adopt the Convolution Neural Network (CNN) [1] with or without pre-trained word embedded on imdb movie review data and evaluate with the baseline Naive Bayes Classifier. This help us prove that our CNN algorithm works

Secondly, we developed model focus on extracting the sentiment from the 20 Newsgroups dataset so as to find the accurate category of the input news.

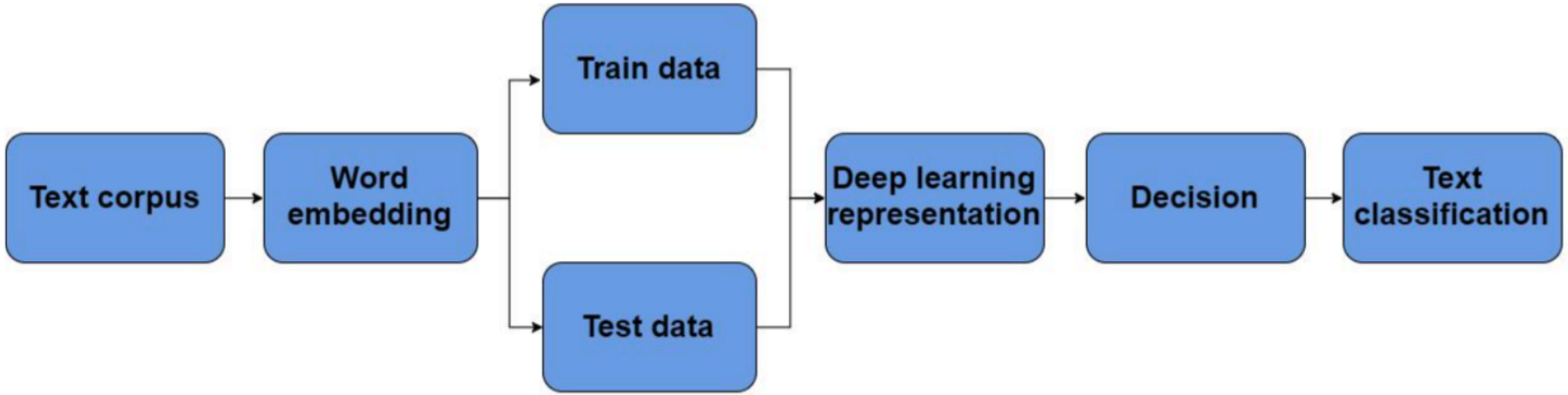
The idea is to create, analyze and report information fast while enhancing user experience.

The main method that we would use in this paper for creating a sentiment analyzer will be the CNN and among all classifier,

We also analyzed the performance accuracy from different representation of input dataset such as Bag of Words and Word2Vec in our experiment in this paper.

System

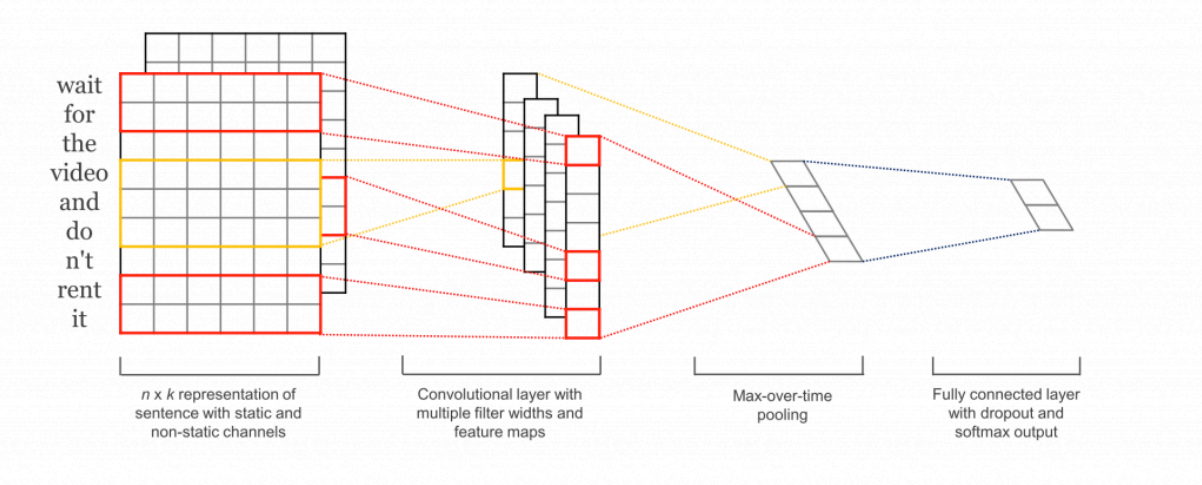
We design our system architecture as the following graphic.



Convolutional Neural Network (CNN)

CNN is widely used in image recognition task.

When using CNN in NLP, we construct the sentences that we want to apply deep learning on as a matrix. Each row is a vector that represents a word. As we slide over rows of the matrix with our filters, the height of the filter matrix is the number of words that is computed.



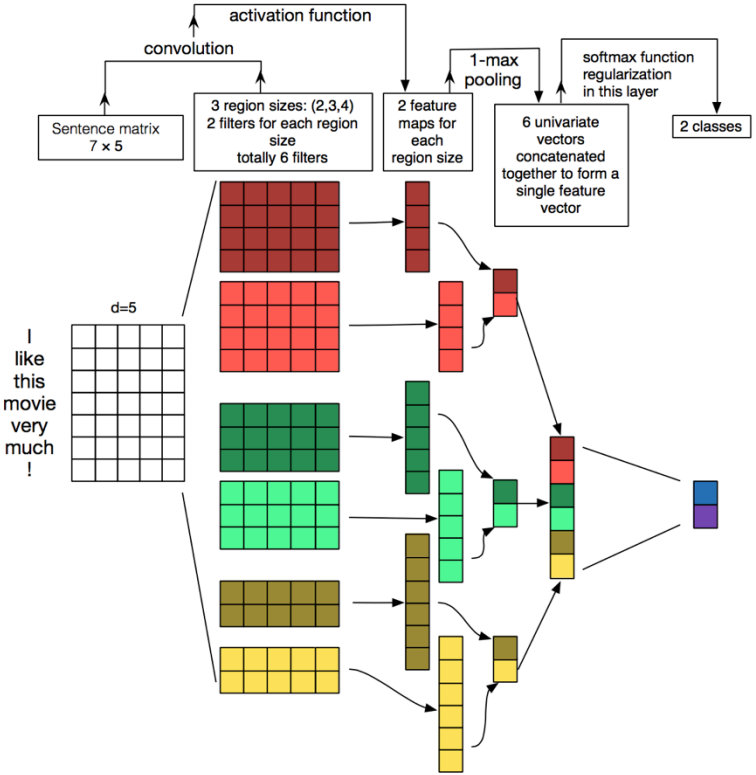
CNN with NLP

The first layers in the CNN would embed words into low-dimensional vectors.

The second layer would then perform convolutions over the embedded word vectors using multiple filter sizes.

After performing convolutions, we do a max-poll to the results to turn them into a long feature vector.

Finally, we add dropout regularization and classify the results using a softmax layer.



TensorFlow implementing CNN

- Convolution Layer

- Max-Pooling Layer

```
# Create a convolution + maxpool layer for each filter size
pooled_outputs = []
for i, filter_size in enumerate(filter_sizes):
    with tf.name_scope("conv-maxpool-%s" % filter_size):
        # Convolution Layer
        filter_shape = [filter_size, embedding_size, 1, num_filters]
        W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name="W")
        b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name="b")
        conv = tf.nn.conv2d(
            self.embedded_chars_expanded,
            W,
            strides=[1, 1, 1, 1],
            padding="VALID",
            name="conv")
        # Apply nonlinearity
        h = tf.nn.relu(tf.nn.bias_add(conv, b), name="relu")
        # Maxpooling over the outputs
        pooled = tf.nn.max_pool(
            h,
            ksize=[1, sequence_length - filter_size + 1, 1, 1],
            strides=[1, 1, 1, 1],
            padding='VALID',
            name="pool")
        pooled_outputs.append(pooled)

# Combine all the pooled features
num_filters_total = num_filters * len(filter_sizes)
self.h_pool = tf.concat(pooled_outputs, 3)
self.h_pool_flat = tf.reshape(self.h_pool, [-1, num_filters_total])

# Add dropout
with tf.name_scope("dropout"):
    self.h_drop = tf.nn.dropout(self.h_pool_flat, self.dropout_keep_prob)
```

Word embedding(word2vec)

Word2vec is used to represent words in a continuous dense vector space where semantically similar words are mapped to nearby points

- CBOW predict the current word w0 given only Context C
- Skip-Gram predict words from C given w0

Skip-gram produces better word vectors for infrequent words
CBOW is faster by a factor of window size – more appropriate for larger corpora (better)

TensorFlow implementing CNN

- Embedding Layer for Word2Vec

```
def load_embedding_vectors_word2vec(vocabulary, filename, binary):
    # load embedding_vectors from the word2vec
    encoding = 'utf-8'
    with open(filename, "rb") as f:
        header = f.readline()
        vocab_size, vector_size = map(int, header.split())
        # initial matrix with random uniform
        embedding_vectors = np.random.uniform(-0.25, 0.25, (len(vocabulary), vector_size))
        if binary:
            binary_len = np.dtype('float32').itemsize * vector_size
            for line_no in range(vocab_size):
                word = []
                while True:
                    ch = f.read(1)
                    if ch == b' ':
                        break
                    if ch == b'':
                        raise EOFError("unexpected end of input; is count incorrect or file otherwise damaged?")
                    if ch != b'\n':
                        word.append(ch)
                word = str(b''.join(word), encoding=encoding, errors='strict')
                idx = vocabulary.get(word)
                if idx != 0:
                    embedding_vectors[idx] = np.fromstring(f.read(binary_len), dtype='float32')
                else:
                    f.seek(binary_len, 1)
        else:
            for line_no in range(vocab_size):
                line = f.readline()
                if line == b'':
                    raise EOFError("unexpected end of input; is count incorrect or file otherwise damaged?")
                parts = str(line.rstrip(), encoding=encoding, errors='strict').split(" ")
                if len(parts) != vector_size + 1:
                    raise ValueError("invalid vector on line %s (is this really the text format?)" % (line_no))
                word, vector = parts[0], list(map('float32', parts[1:]))
                idx = vocabulary.get(word)
                if idx != 0:
                    embedding_vectors[idx] = vector
        f.close()
        return embedding_vectors
```

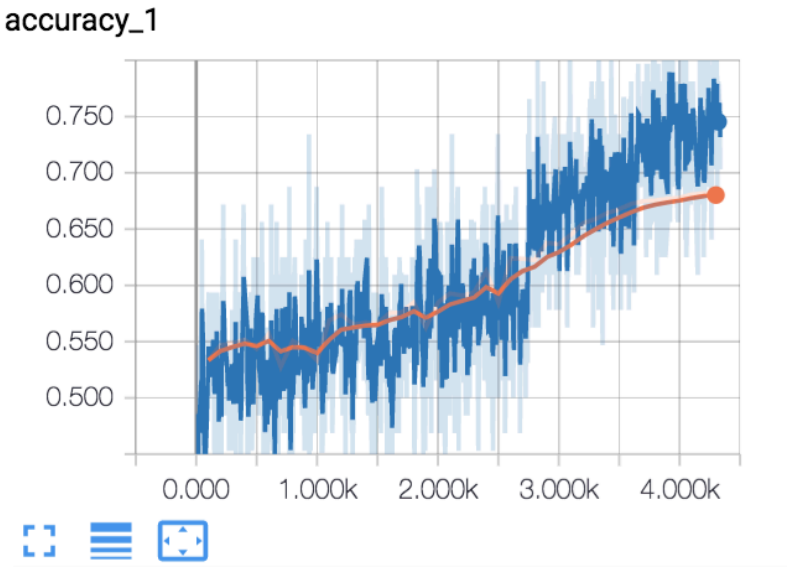
Results & Evaluation (IMDB Review)

Naive Bayes:

```
Accuracy: 0.791500
```

Convolution Neural Network:

Name	Smoothed	Value	Step	Time	Relative
dev	0.6802	0.6816	4.300k	Tue Apr 17, 11:03:17	13m 33s
train	0.7454	0.7500	4.318k	Tue Apr 17, 11:03:22	13m 55s



News Carectorization (NB)

```
$ python TextClassification.py                    (ruby-2.3.0) master 04ecf5a
Following newsgroups will be tested: ['alt.atheism', 'comp.graphics', 'sci.med',
 'soc.religion.christian']
Number of sentences: 2257
Naive Bayes correct prediction: 0.83
                    precision    recall  f1-score   support

       alt.atheism       0.97      0.60      0.74       319
     comp.graphics       0.96      0.89      0.92       389
           sci.med       0.97      0.81      0.88       396
soc.religion.christian       0.65      0.99      0.78       398

       avg / total       0.88      0.83      0.84      1502
```

News Carectorization (CNN)

```
Total number of test examples: 1502
Accuracy: 0.828895
                        precision    recall  f1-score   support

           alt.atheism       0.91      0.74      0.82       319
         comp.graphics       0.79      0.90      0.84       389
               sci.med       0.77      0.75      0.76       396
soc.religion.christian       0.87      0.91      0.89       398

           avg / total       0.83      0.83      0.83      1502

[[236  12  37  34]
 [  2 349  34   4]
 [ 20  65 296  15]
 [  2  13  19 364]]
```

News Carectorization (CNN+Word2Vec)

```
Total number of test examples: 1502
Accuracy: 0.930093
                        precision    recall  f1-score   support

             alt.atheism       0.97      0.82      0.89       319
           comp.graphics       0.90      0.99      0.95       389
                 sci.med       0.98      0.90      0.94       396
soc.religion.christian       0.88      0.98      0.93       398

             avg / total       0.93      0.93      0.93      1502

[[263   7   5  44]
 [  0 387   1   1]
 [  5  30 355   6]
 [  2   4   0 392]]
```

News Carectorization (SVM) using sklearn

- Code:

```
""" Support Vector Machine (SVM) classifier"""
svm_clf = Pipeline([('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', SGDClassifier(loss='hinge', penalty='l2', alpha=1e-3, n_iter=   5, random_state=42)),
])
svm_clf.fit(twenty_train.data, twenty_train.target)
joblib.dump(svm_clf, "svm_20newsgroup.pkl", compress=9)
""" Predict the test dataset using Naive Bayes"""
predicted = svm_clf.predict(docs_test)
print('SVM correct prediction: {:4.2f}'.format(np.mean(predicted == twenty_test.target)))
print(metrics.classification_report(twenty_test.target, predicted, target_names=twenty_test.target_names))
```

- Results:

```
SVM correct prediction: 0.91
                        precision    recall  f1-score   support

           alt.atheism       0.95      0.81      0.87       319
         comp.graphics       0.88      0.97      0.92       389
               sci.med       0.94      0.90      0.92       396
soc.religion.christian       0.90      0.95      0.93       398

           avg / total       0.92      0.91      0.91      1502
```

News Carectorization (SVM) using sklearn

SVM correct prediction: 0.91

	precision	recall	f1-score	support
alt.atheism	0.95	0.81	0.87	319
comp.graphics	0.88	0.97	0.92	389
sci.med	0.94	0.90	0.92	396
soc.religion.christian	0.90	0.95	0.93	398
avg / total	0.92	0.91	0.91	1502

Comparisons for News Carectorization

	Naive Bayes	Support Vector Machine	Convolution Neural Network (CNN)	CNN with Word2Vec
Accuracy	0.83	0.91	0.83	0.93
Training Time	O(1)	O(1)	>= 4 hours	>= 2 hours

Conclusion

1.Pos and Neg two categories:

In classifying we conclude that Naive bayes out perform CNN in such a simple task. The accuracy of NB is incredibly good, since words like great, richly, awesome, and pathetic, and awful and ridiculously are very informative cues. However, when we move to more complex task, the generative model NB decrease dramatically.

2. News categorization (4 group):

The down side of this CNN higher accuracy comparing to Bayes or SVM is the training time. Neural Network needs several hours or more for the training to achieve this accuracy, where SVM just needs a second to train the same dataset with respectable accuracy.

Future Work

We can further do analysis with Recurrent Neural Networks (RNN).

RNN is a way to use neural networks over a sequence of data which is perfect in our scenario.

Some data is senseless without context. A sentence, for example, is made up of a series of data points known as words. Each word holds meaning to us with or without context, but a sequence of words can encode a larger, more complex idea.

IDENTIFYING DUPLICATE QUESTION PAIRS ON QUORA

SHIVA KUMAR PENTYALA

ARJUN SURESH

CONTENTS

- Problem definition
- Motivation
- Approaches
- Experiments and Results
- Conclusion

PROBLEM DEFINITION

- Predict whether two questions on quora have the same meaning (intent).
- "What is the difference between IT and computer science engineering?“ and "What are the differences between computer science and computer engineering?“ aren’t duplicate.
- "What is a GDP?“ and "What is the significance of GDP?“ are duplicates.
- Compare the performances of shallow models and deep models in this respect.

MOTIVATION

- Quora is a place to gain and share knowledge—about anything.
- Over 100 million people visit Quora every month. So, chances are that similarly worded questions exist.
- This leads to two major hassles:
  - Seekers may need to visit multiple questions to get the best answers.
  - Writers may need to provide answers for multiple questions.
- Canonical questions provide better experience to both the groups.

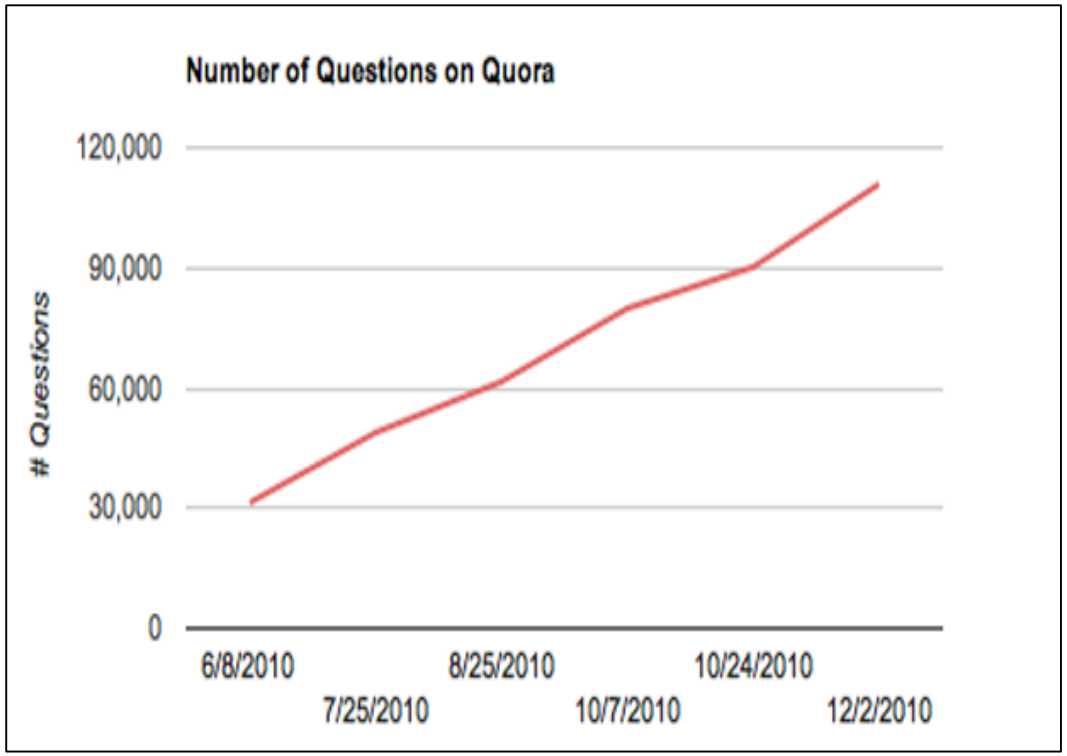


Fig: Increasing number of questions on quora

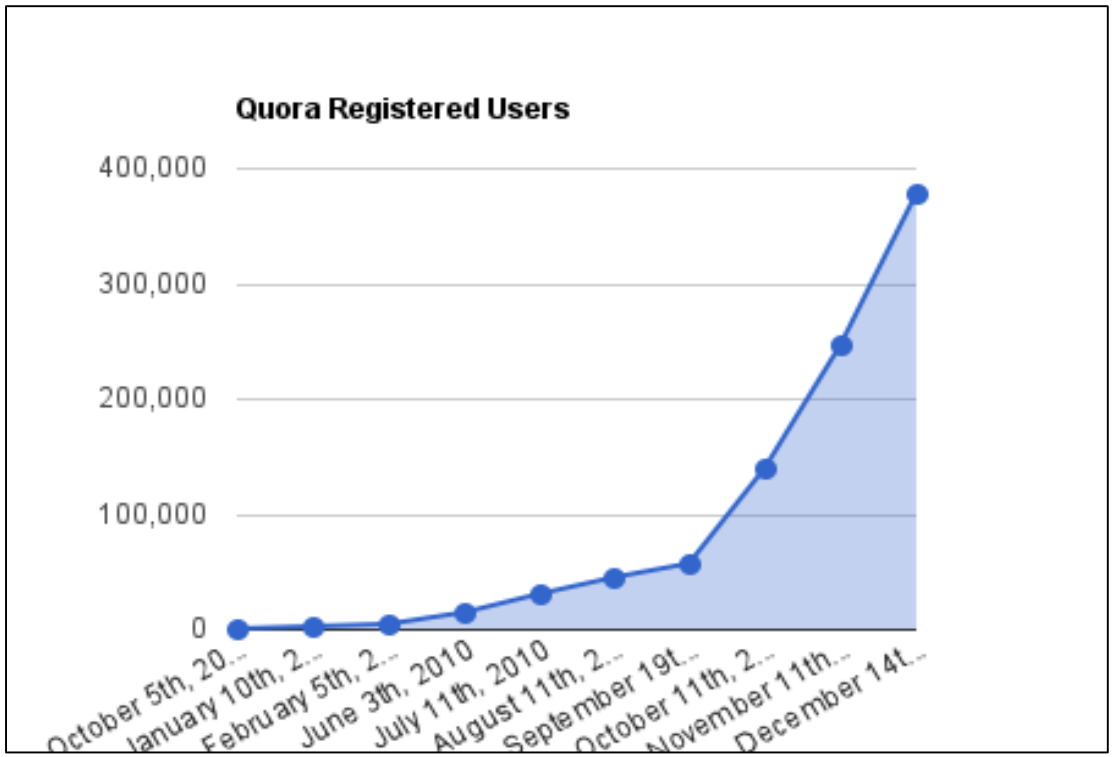


Fig: Increasing number of quora users

APPROACH

- Analyzed the data and collected some vital statistics.
- An iteration of data cleaning and preprocessing.
- Extract key features
- Apply the features on shallow models and deep learning algorithms
  - Shallow learning – Shallow learning is the traditional approach to machine learning including SVM, decision trees, Naive Bayes, Logisitic Regression, Neural Networks comprising of 1-2 layers et al.
  - Deep learning - Deep learning is a class of machine learning algorithms that use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Eg – Neural networks having multiple hidden layers.
- Compare the performances of the 2 categories of algorithms.

Data Analysis

The data provided for training is from the public dataset from quora. There are 404352 question pairs, each specified with he following fields in a tab-separated format.

- id: unique identifier for the question pair (unused)
- qid1: unique identifier for the first question (unused)
- qid2: unique identifier for the second question (unused)
- question1: full unicode text of the first question
- question2: full unicode text of the second question
- is duplicate: label 1 if questions are duplicates, 0 otherwise

We observed that roughly 37%, or 149302 question pairs of the entire set are duplicates and thus represent the positive class.

Data preprocessing

- Removed around 2413 question pairs from the dataset out of which 1157 were deemed to be duplicates.
- Removed non-alphabetical characters in words, like changing e-mail to email, “can’t” to “cannot”, “’ve” to “have”, removing non-Unicode characters et al.
- Expanded popular abbreviations like U.S. to United States, capitalized country names
- Resolved some typos of commonly used words like “latop” to “laptop”, “syllabus” to “syllabus”, “admission” to “admission” et al.
- For Deep Network, the raw questions, represented as single dimensional vectors of vocabulary indexes, are converted into pre-trained word embeddings using GloVe in the embedding.

Features

- POS Tags
  - The number of matching nouns between the 2 questions to be matched.
  - In short texts, it is very much evident that nouns, particularly proper nouns would be repeated across similar questions.
- Word match
  - Number of words matched between the 2 questions.
  - Filter out the stop words first. Used NLTK’s corpus to get the common stopwords.

- TF-IDF (Term Frequency-Inverse Document Frequency)
  - This feature is used to reflect the importance of a word with respect to the corpus.
  - TF-IDF related features that we used are:
    - TF-IDF Sum
    - TF-IDF Mean
    - TF-IDF Length
    - TF-Idf Word share
- Named Entity Recognition
  - Used StanfordNER Tagger to obtain important entites like ‘PERSON’, ‘LOCATION’, ‘ORGANIZATION’ et al.
  - Matched the entities across the 2 questions to get a ratio.

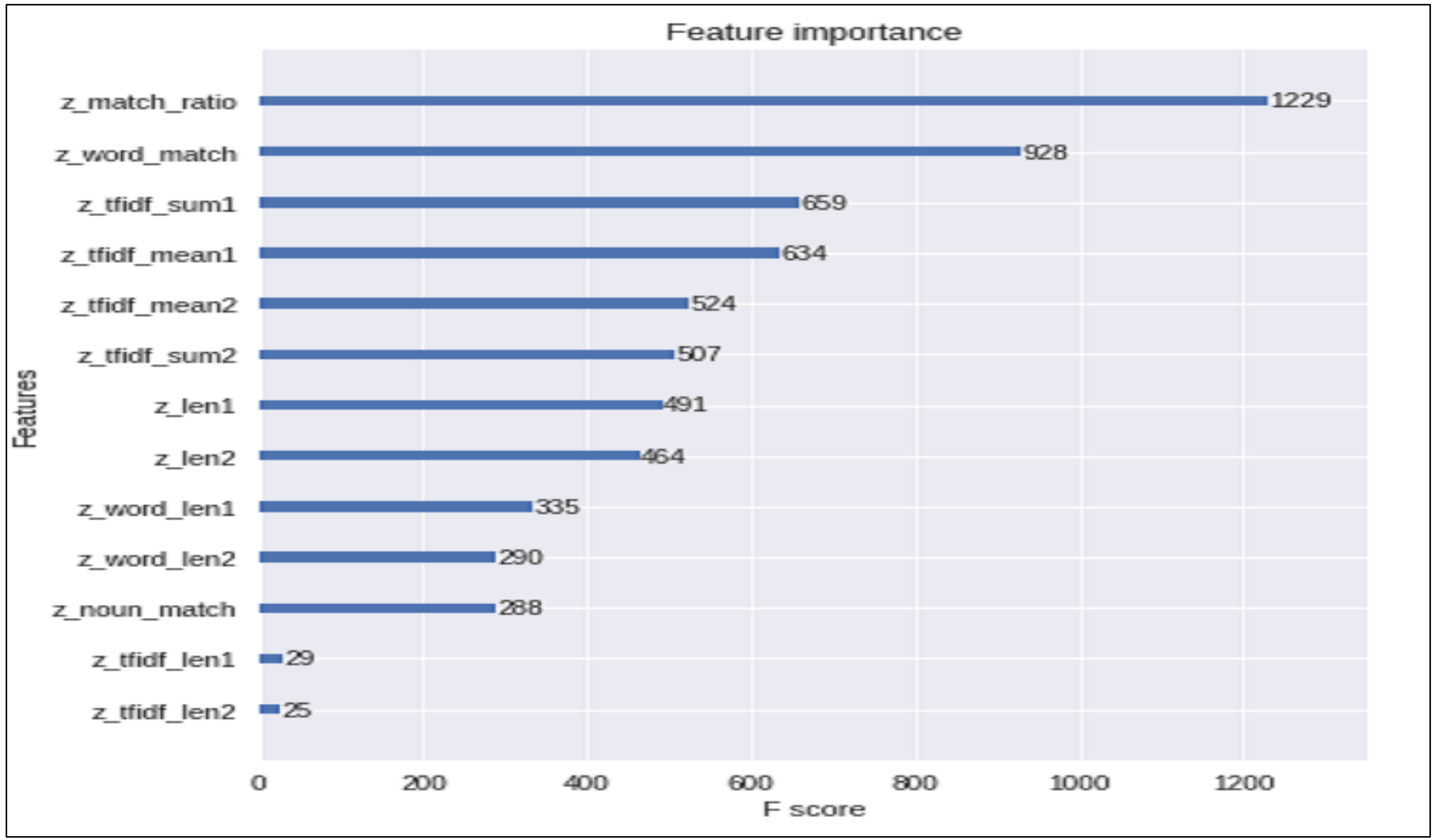


Fig: Feature importance diagram

Shallow learning algorithms

- Naïve Bayes
  - Probabilistic machine learning algorithms with the primary assumption that the features have to be independent of each other.
  - In this project, we used a multinomial Naïve Bayes algorithm.

- SVM
  - This is a supervised learning algorithm used for 2-class classification.
  - Strives to create a decision boundary with maximum margin from either classes.
  - We have used a nonlinear SVM with the optimal cost parameter estimated as 1000.

- XGBoost (Gradient boosting)
  - This algorithm produces a classification model based on the concept of ensemble learning.
  - Multiple weaker models like decision trees are used to gradually build the model in a stage-like fashion.
  - Our model uses a binary logistic objective function and log loss as the evaluation criteria.

Deep learning algorithms

- Mainly used Saimese network architecture and constructed 2 different models one using MaLASTM and LSTM, CNN.
- Siamese networks are networks that have two or more identical sub-networks in them.
- Siamese networks seem to perform well on similarity tasks.
- Generally used for tasks like sentence semantic similarity, recognizing forged signatures and many more.

- Baseline model
  - This model uses w-shingling to compare the similarity between the two questions using only their texts.
  - In shingling, we represent each question as a bag of shingles or n-grams with n being from 1 to 4.
  - The bags for each question are then compared using the Jaccard similarity coefficient:

    $J(S(q1); S(q2)) = S(q1) \setminus S(q2) = S(q1) [S(q2)$

    Where S(q) is the set of shingles for the given question
  - This score is then compared against a threshold, with any coefficient above the threshold being declared a duplicate.
  - Using binary search over the space of possible threshold values T in [0, 1], we obtained a best value for T of 0.13 on the training data.

EXPERIMENTS AND RESULTS

- Shallow models
  - The full dataset was used comprising of 404,352 question pairs for all the 3 classifiers – Multinomial Naive Bayes, XGBoost and SVM.
  - Crossvalidation technique was used to get the test sample points, with the split being 0.2 (80% of the dataset for training and 20% for testing).
  - Accuracy was used as the evaluation criteria.

Model	Accuracy
Naïve Bayes	0.808
XGBoost (Gradient boosting)	0.825
SVM	0.835

- Deep models
  - MaLSTM
    - Ran on a system which has a GTX 1070.
    - It is giving an 86 % accuracy on the validation data.
    - The preparation and training took about 21 hours.
  - LSTM,CNN based Siamese network
    - The results obtained are tabulated below:

	Baseline	CNN	LSTM	Hybrid
Accuracy	0.6657	0.8027	0.8107	0.8105
Recall	0.5151	0.7102	0.6862	0.7804
Precision	0.7297	0.7349	0.8441	0.7994
F1	0.6039	0.7223	0.7570	0.7446

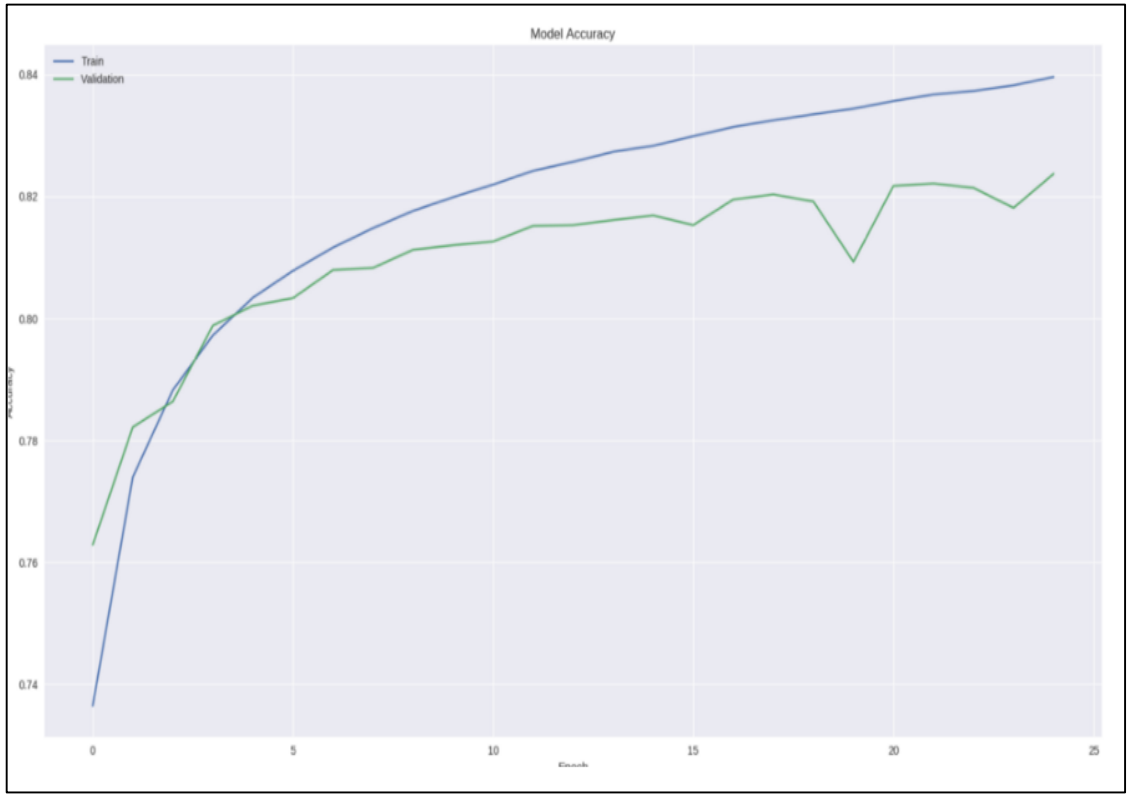


Fig. Accuracy vs Epochs.

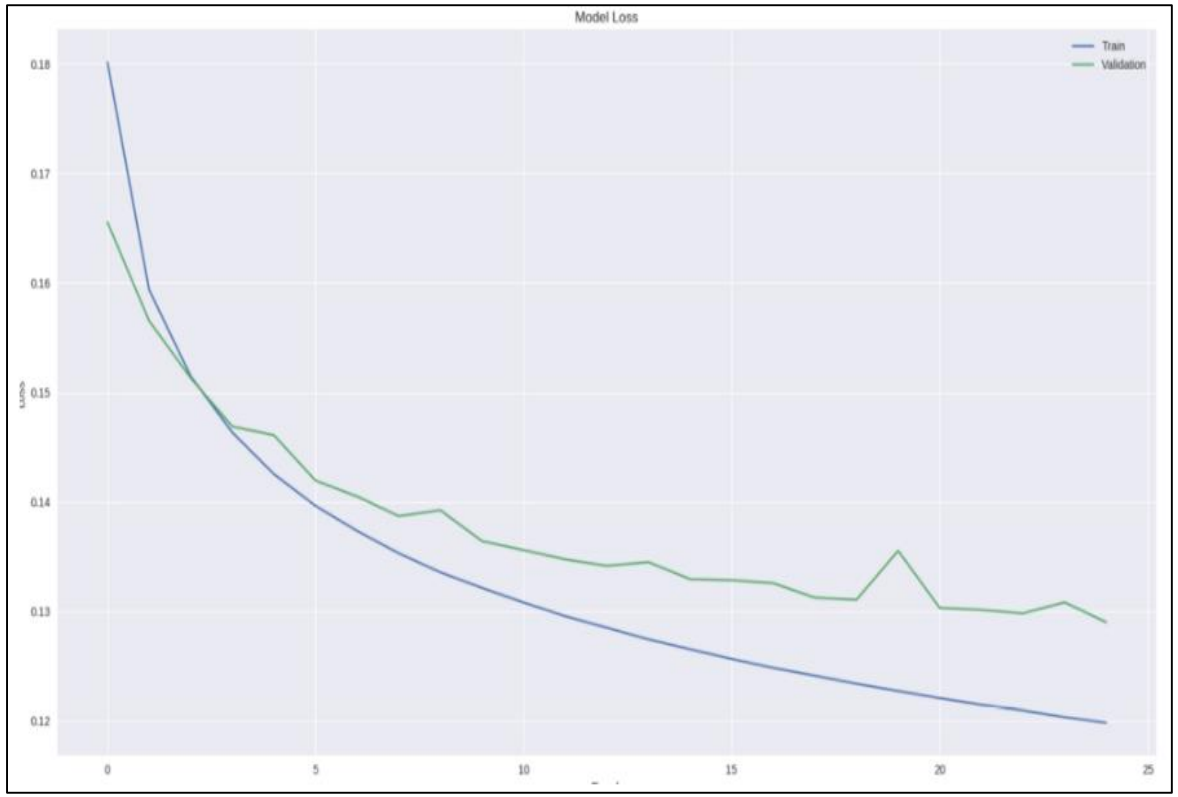


Fig. Loss vs Epochs.

CONCLUSION

- We found that important features extracted from the shallow models makes sense.
- Overall, Deep learning methods clearly outperform the shallow /baseline methods for the task of duplicate question pair detection.
- The Siamese network architecture achieved a 7 percent performance gain over a simpler multilayer perceptron architectures, and a 14 percent gain over the shingling method.
- The LSTM method had comparable and slightly better performance than the CNN, but was much faster to train.
- A hybrid model combining both the LSTM and the CNN together produced no noticeable performance gain, but was much slower to train overall.

Q&A

Semantic Relation Extraction and Classification in Scientific Papers

Class Project:

Chao Wan

CSCE 689 SPRING 2018

Introduction & Motivation

2.5 million Scientific Papers published each year

<1% high quality papers among all papers.

New Terms -Many papers come up with new terms that is hard to distinguish
-Relations between terms are complex

Consider - what if we could know the relations between entities
- Can we identify high quality papers from low onees?

Data Source

Semeval 2018 Task 7

- **350** abstracts from scientific papers
- **6** categories include Usage, PART_WHOLE, MODEL-FEATURE, RESULT, COMPARE, TOPIC
- **1,228** relations annotated manually

Data Source

Semeval 2018 Task 7

- <abstract>
- <entity id="H01-1001.1">Oral communication</entity>
- is ubiquitous and carries important information yet it is also time consuming to document. Given the development of
- <entity id="H01-1001.2">storage media and networks</entity>
- one could just record and store a
- <entity id="H01-1001.3">conversation</entity>
- for documentation. The question is, however, how an interesting information piece would be found in a
- <entity id="H01-1001.4">large database</entity>.

Data Source

Semeval 2018 Task 7

```
USAGE(H01-1001.5,H01-1001.7,REVERSE)
USAGE(H01-1001.9,H01-1001.10)
PART_WHOLE(H01-1001.14,H01-1001.15,REVERSE)
MODEL-FEATURE(H01-1017.4,H01-1017.5)
PART_WHOLE(H01-1041.3,H01-1041.4,REVERSE)
USAGE(H01-1041.8,H01-1041.9)
MODEL-FEATURE(H01-1041.10,H01-1041.11,REVERSE)
USAGE(H01-1041.14,H01-1041.15,REVERSE)
USAGE(H01-1042.1,H01-1042.3)
MODEL-FEATURE(H01-1042.10,H01-1042.11)
PART_WHOLE(H01-1049.3,H01-1049.4,REVERSE)
RESULT(H01-1058.2,H01-1058.4)
RESULT(H01-1058.9,H01-1058.10)
RESULT(H01-1058.13,H01-1058.14,REVERSE)
USAGE(H01-1058.16,H01-1058.18,REVERSE)
RESULT(H01-1058.19,H01-1058.20)
MODEL-FEATURE(H01-1058.24,H01-1058.25,REVERSE)
MODEL-FEATURE(H01-1058.27,H01-1058.28,REVERSE)
```

Preprocessing

Get the plain text with entity id

H01-1001.1 is ubiquitous and carries important information yet it is also time consuming to document. Given the development of H01-1001.2 one could just record and store a H01-1001.3 for documentation. The question is, however, how an interesting information piece would be found in a H01-1001.4

Represent each word as word vector

0	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , ...
1	float64	(20, 100, 50)	array([[[0.68047 , -0.039263 , 0.30186 , ..., -0.073297 , ...
2	float64	(20, 100, 50)	array([[[0.27724 , 0.88469 , -0.26247 , ..., -0.51611 , 0.023163, ...
3	float64	(20, 100, 50)	array([[[4.1800e-01, 2.4968e-01, -4.1242e-01, ..., -1.8411e-01, ...
4	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , 0. , ...
5	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , ...
6	float64	(20, 100, 50)	array([[[0.53074 , 0.40117 , -0.40785 , ..., 0.28762 , ...
7	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , 0. , ...
8	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , -0.027273, ...
9	float64	(20, 100, 50)	array([[[0.418 , 0.24968 , -0.41242 , ..., -0.18411 , ...
10	float64	(20, 100, 50)	array([[[0.1011 , -0.16566 , 0.22035 , ..., -0.49441 , 0.043538, ...
11	float64	(20, 100, 50)	array([[[0.53074 , 0.40117 , -0.40785 , ..., 0.28762 , ...
12	float64	(20, 100, 50)	array([[[0.57387 , -0.32729 , 0.070521, ..., 0.48759 , -0.18439 , ...
13	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , 0. , ...
14	float64	(20, 100, 50)	array([[[0.57387 , -0.32729 , 0.070521, ..., 0.48759 , -0.18439 , ...
15	float64	(20, 100, 50)	array([[[0.53074 , 0.40117 , -0.40785 , ..., 0.28762 , 0.1444 , ...
16	float64	(20, 100, 50)	array([[[0.21016 , -1.0903 , -0.23658 , ..., 0.77024 , 0.49909 , ...
17	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , -0.027273, ...
18	float64	(20, 100, 50)	array([[[5.7387e-01, -3.2729e-01, 7.0521e-02, ..., 4.8759e-01, ...
19	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , -0.027273, ...
20	float64	(20, 100, 50)	array([[[0.57387 , -0.32729 , 0.070521, ..., 0.48759 , -0.18439 , ...

Get the Entity representation

0	1
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0

Pr

Get the pl

H01-100
informatic
documen
one coul
documen
interestin
H01-100

Represent

0	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , ...
1	float64	(20, 100, 50)	array([[[0.68047 , -0.039263 , 0.30186 , ..., -0.073297 , ...
2	float64	(20, 100, 50)	array([[[0.27724 , 0.88469 , -0.26247 , ..., -0.51611 , 0.023163, ...
3	float64	(20, 100, 50)	array([[[4.1800e-01, 2.4968e-01, -4.1242e-01, ..., -1.8411e-01, ...
4	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , 0. , ...
5	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , ...
6	float64	(20, 100, 50)	array([[[0.53074 , 0.40117 , -0.40785 , ..., 0.28762 , ...
7	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , 0. , ...
8	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , -0.027273, ...
9	float64	(20, 100, 50)	array([[[0.418 , 0.24968 , -0.41242 , ..., -0.18411 , ...
10	float64	(20, 100, 50)	array([[[0.1011 , -0.16566 , 0.22035 , ..., -0.49441 , 0.043538, ...
11	float64	(20, 100, 50)	array([[[0.53074 , 0.40117 , -0.40785 , ..., 0.28762 , ...
12	float64	(20, 100, 50)	array([[[0.57387 , -0.32729 , 0.070521, ..., 0.48759 , -0.18439 , ...
13	float64	(20, 100, 50)	array([[[0. , 0. , 0. , ..., 0. , 0. , ...
14	float64	(20, 100, 50)	array([[[0.57387 , -0.32729 , 0.070521, ..., 0.48759 , -0.18439 , ...
15	float64	(20, 100, 50)	array([[[0.53074 , 0.40117 , -0.40785 , ..., 0.28762 , 0.1444 , ...
16	float64	(20, 100, 50)	array([[[0.21016 , -1.0903 , -0.23658 , ..., 0.77024 , 0.49909 , ...
17	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , -0.027273, ...
18	float64	(20, 100, 50)	array([[[5.7387e-01, -3.2729e-01, 7.0521e-02, ..., 4.8759e-01, ...
19	float64	(20, 100, 50)	array([[[0.33042 , 0.24995 , -0.60874 , ..., -0.50703 , -0.027273, ...
20	float64	(20, 100, 50)	array([[[0.57387 , -0.32729 , 0.070521, ..., 0.48759 , -0.18439 , ...
21	float64	(20, 100, 50)	array([[[0.57387 , -0.32729 , 0.070521, ..., 0.48759 , -0.18439 , ...

RNN

- Information persistence
- Hard to handle long context
- Gradient Vanishing, Gradient Exploding

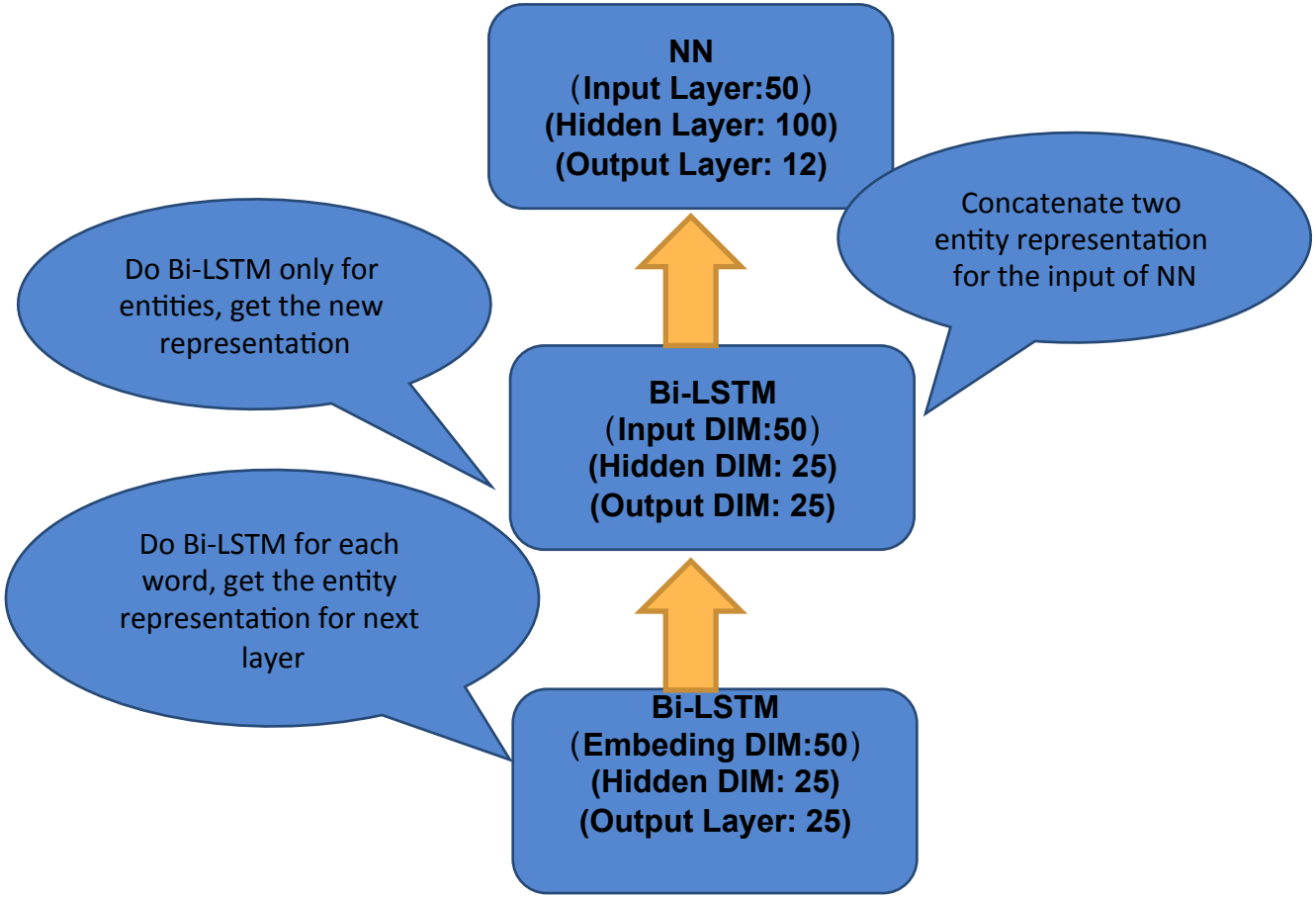
LSTM & Bi-LSTM

- Could also capture context information
- Handle long context
- Avoid Gradient Vanishing but still have problem of Gradient Exploding

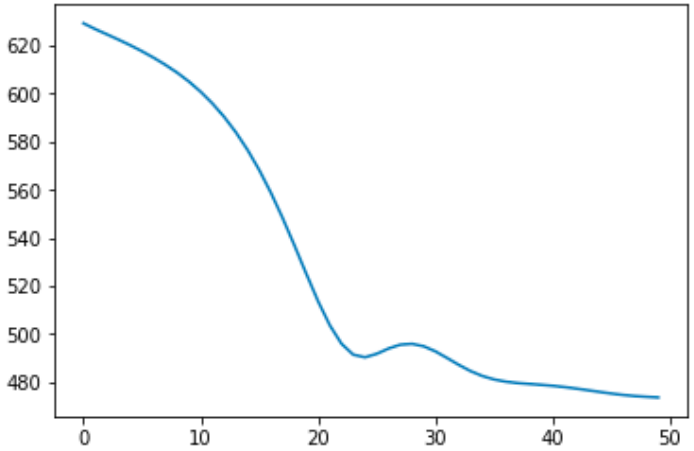
Optimization

- SVD?
- ADAM?
- Need dropout?

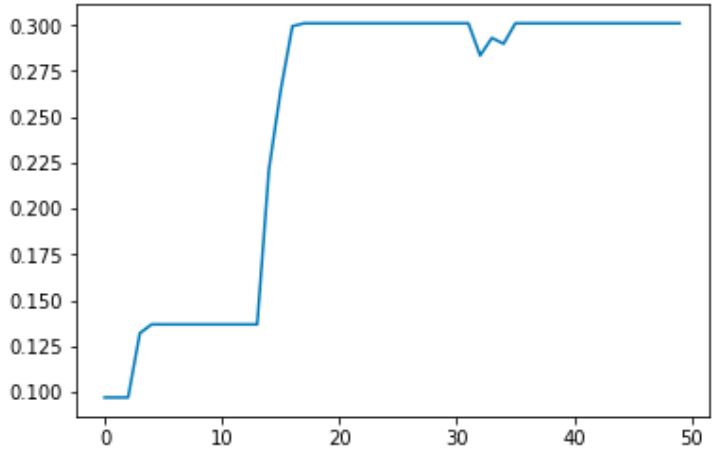
Classification Model



Result



CrossEntropy Loss Curve



Accuracy Curve

Analysis

- Bi-LSTM is not perfect, it could “Forget” important information or “Remember” the irrelevant one
- Treat entities as zero vector is not appropriate.
- Not enough data

Future Work

- Improve the accuracy of relation extraction, maybe change the model
- Could recognize entities rather than use the annotated ones.
- More data, more training epoch

Thank You

Conversational Chatbot

Aditya Gurjar
Shashwat Kumar

Introduction

Chatbots are AI agents which communicate and collaborate with human users through text messaging using a natural language, say English, to accomplish specific tasks.

- Are already being used to mine data in websites and answer queries

  Eg : Recommending best booking for a trip

Dataset

We make use of a dataset called DailyDialog

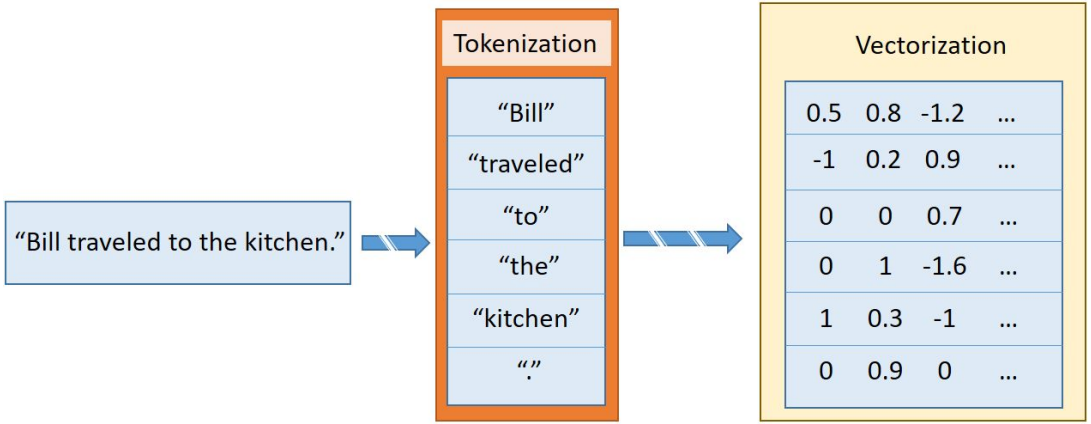
It consists of several conversations. Each conversation holds a context and is between two humans. Every other sentence is spoken by a different user.

Our model is trained by considering a set of two sentences, one as trigger, other as a response to it. This response becomes the trigger in our next set of sentences.

Preprocessing

Preprocessing of data involves three parts:

- Tokenization
- Padding
- Word embeddings



Model Architecture

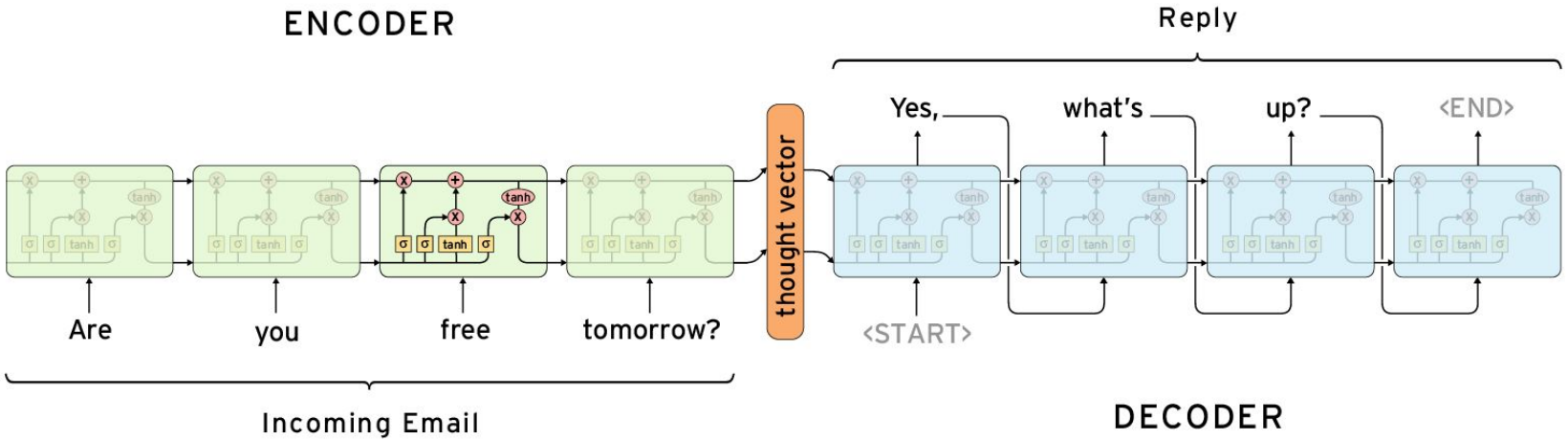
We use end to end, sequence to sequence models to generate the response.

There are two LSTM networks - an encoder and a decoder.

The input to the encoder is the query sentence. We then get the hidden state of the last time step of the encoder and feed it to the decoder as its initial hidden state.

The decoder takes in this hidden state, which acts as the context for the decoder to predict the next word in the sequence and generate a response.

Sequence to Sequence



Thank you!

Context Aware Chatbot

Pritish Uplavikar

Introduction

Chatbots are AI agents which communicate and collaborate with human users through text messaging using a natural language, say English, to accomplish specific tasks.

- Are already being used to mine data in websites and answer queries

  Eg : Recommending best booking for a trip

Dataset

We make use of a dataset called DailyDialog

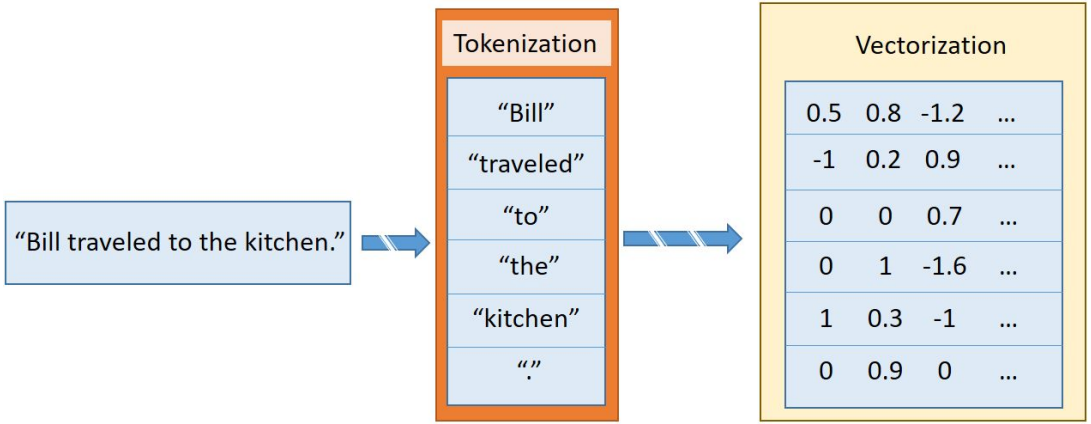
It consists of several conversations. Each conversation holds a context and is between two humans. Every other sentence is spoken by a different user.

Our model is trained by considering a set of two sentences, one as trigger, other as a response to it. This response becomes the trigger in our next set of sentences.

Preprocessing

Preprocessing of data involves three parts:

- Tokenization
- Padding
- Word embeddings



Model Architecture

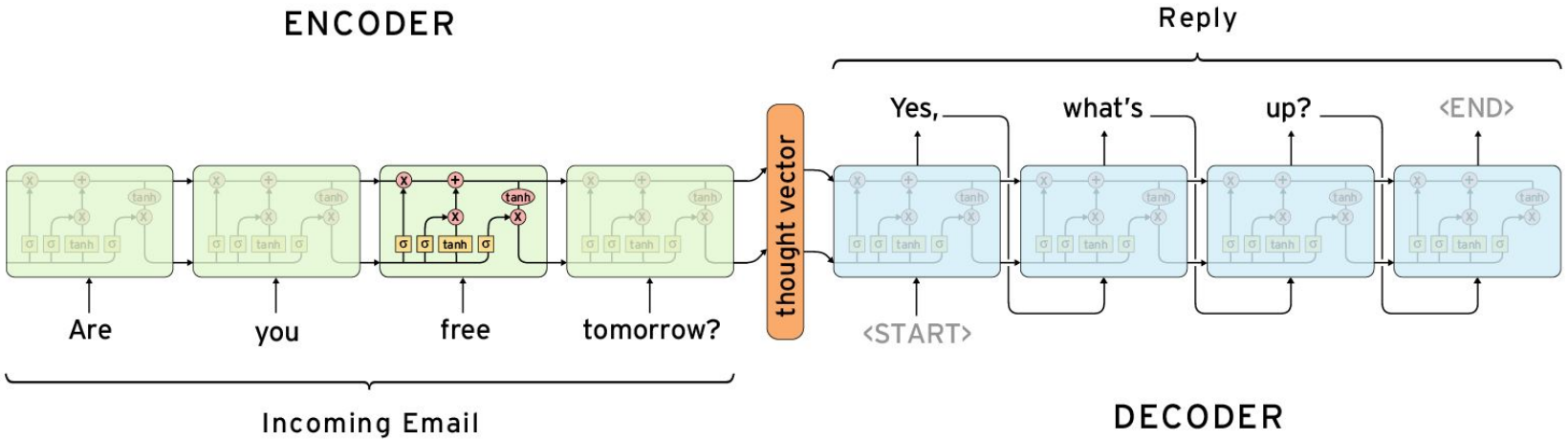
We use end to end, sequence to sequence models to generate the response.

There are two LSTM networks - an encoder and a decoder.

The input to the encoder is the query sentence. We then get the hidden state of the last time step of the encoder and feed it to the decoder as its initial hidden state.

The decoder takes in this hidden state, which acts as the context for the decoder to predict the next word in the sequence and generate a response.

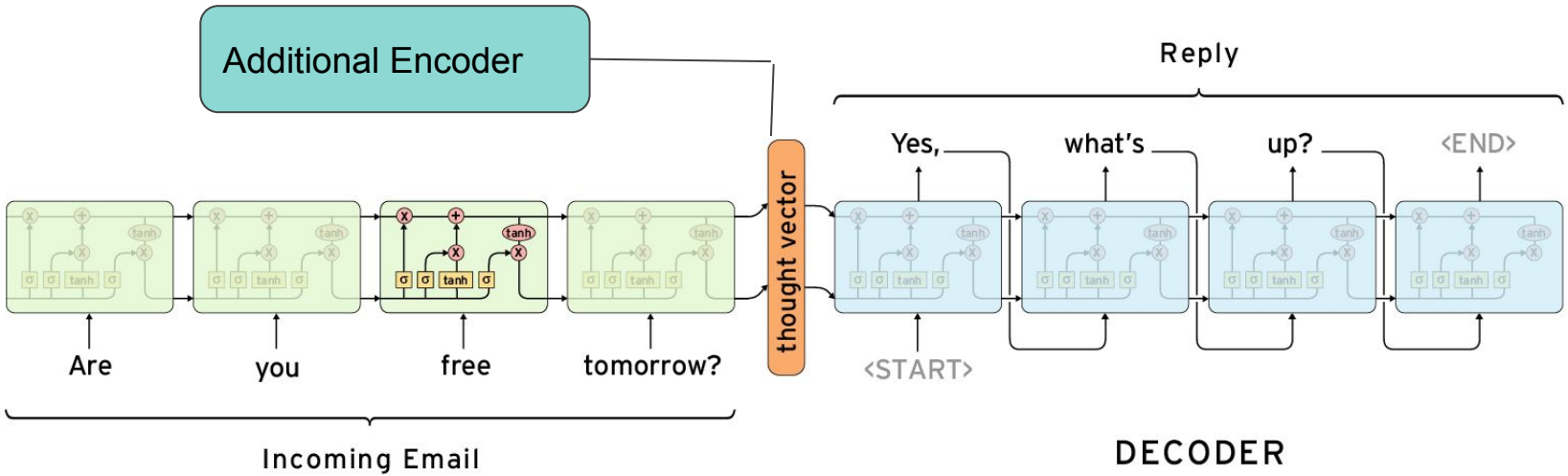
Sequence to Sequence



Adding context to the chatbot

- An additional encoder network is used to provide the context of conversation
- This encoder reads upto 3 sentences previous to the current conversation
- Each sentence is treated as a sequence and a hidden vector is computed for them with this encoder.
- This hidden state is concatenated with the hidden state of the current conversation and used by decoder to give a response

Adding context to the chatbot



Thank you!

Natural Language Processing

RNN Based article caption generator

Palash Parmar
626008848

Abhinav Ratna
227001723

Overview

- Problem Description
- Introduction
- Motivation
- Dataset and Preprocessing
- Methodology
- Evaluation and Analysis
- Results
- Conclusion
- Bottlenecks
- Future Work
- References

// Problem Description

- Generate a condensed meaningful summary from an input article
- Article is a sequence of N words and the summary generated is a sequence of M words where M < N
- Generated summary should preserve information content and overall meaning
- Arduous and time consuming task for Human beings to summarize large documents of text

Introduction

- Text summarization is a task of generating a short summary consisting of a few sentences that capture salient ideas of a text

- Recurrent neural networks based designs very effective for various transductional tasks such as Machine translation, speech recognition etc

- Implemented model comprised of encoder decoder RNN with LSTM units and attention mechanism

// Motivation

- Millions of news articles being published in a day by multiple sources with catchy headlines
- People generally make judgments based on reading the headline and skip the story
- Imperative to have a headline which captures prime motive of the story and provides useful and meaningful content
- Easier task for machines but painstaking for human beings

// Dataset and Preprocessing

- Trained model on “SIgnal Media News Articles Dataset”
- Dataset consists of one million news articles from multiple publishers with clear delineated headline and text
- Headline and text are lowercased and tokenized with punctuation removed from words
- Kept only the first paragraph of text

Dataset and Preprocessing

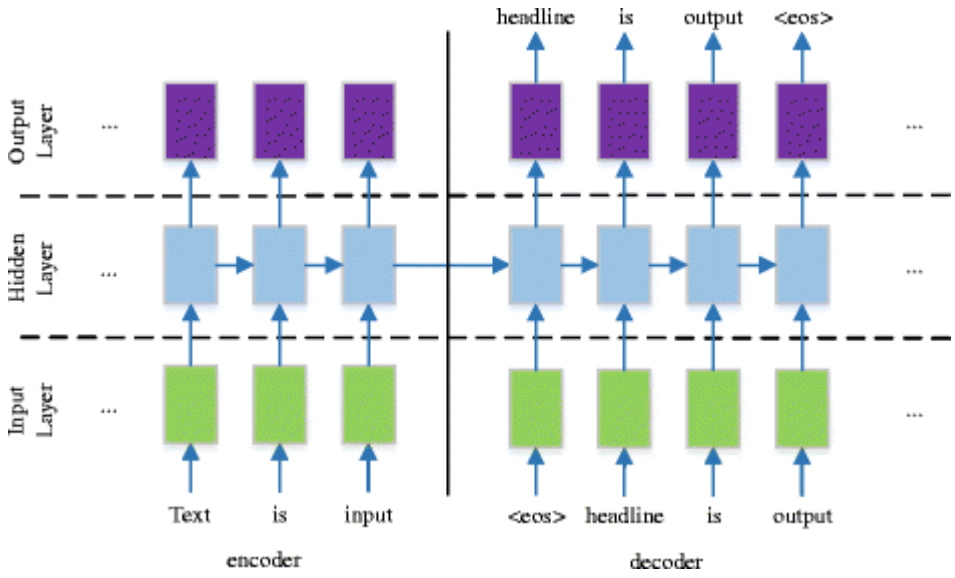
- End of Sequence (EOS) added to both headline and text
- Articles with missing headline or text or having headline tokens more than 25 and text tokens more than 50 are filtered out
- All rare words are replaced with <unk> symbol keeping only the top 40000 most frequent occurring words

// Methodology

- Architecture consists of two parts:
- Encoder and Decoder

- Both encoder and decoder are recurrent neural networks

- Used LSTM cell units
- for encoder and decoder
-
- Known as Sequence
- to Sequence Model



Encoding text

- Each word passes through embedding layer which transforms the word into distributed representation
- Embedded word vectors are then fed sequentially as an input to Encoder (LSTM Cell)
- Information flows from left to right and each word vector is learned according to not only current input but also all previous words
- When the sentence is completely read, encoder generates an output and a hidden state

// Decoding text

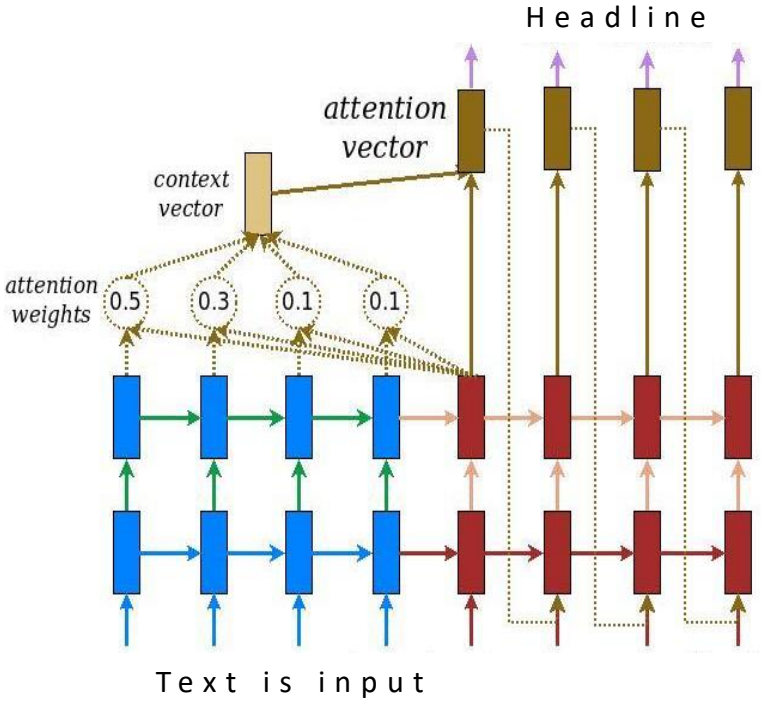
- Decoder (LSTM Cell) takes as input the hidden layers generated after feeding in the last word of the input text
- End-of-Sequence (EOS) symbol is fed in as input first using an embedded layer
- Decoder generates the text summary using a softmax layer and attention mechanism
- Trained by teacher forcing (a mode that takes previous cell's output as current input)

Attention Mechanism

- Alleviates the need of encoding the full source sentence into a fixed-length vector
- Helps network remember certain aspects of the input better, including names and numbers
- Used when outputting each word in the decoder
- For each output word, it computes weight over each of the input word which determines how much attention should be paid to that input word

Attention Mechanism

- Weights sum up to 1
- Used to compute a weighted average (context) of the last hidden layers generated after processing each of the input words
- Context is then input into the softmax layer along with the last hidden layer from the current step of decoding

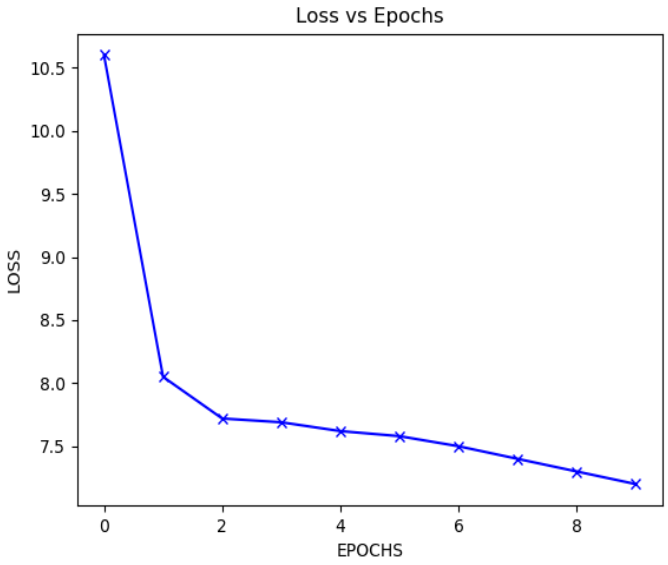


// Evaluation and Analysis

- Measured the performance of model in two ways:
  - Training and test loss
  - BLEU (an algorithm for evaluating the quality of text) metrics over test examples

- Average BLEU Score for 400 test examples over 10 epochs -> 0.06

- Performance on test examples worse than training examples



// Results

Article	Actual Headline	Generated Headline
'We're too broke to go on the beat': Police chiefs warn ministers that cuts mean they can't do their job as they threaten to stop street patrols and say they won't be able to protect public from rioters or terrorists?	Police chiefs warn ministers Were too broke to go on the beat	Police chiefs say not able to protect public warn ministers
What have you been listening to this year ? If you want to find out using cold , hard evidence , then Spotify 's new Year in Music tool will tell you .	Spotify Will Make You Smarter for Your App	The 10 Most Popular Songs Of All Time
NHS patients to be given option of travelling to Calais for surgical procedures NHS patients in Kent are set to be offered the choice of travelling to Calais for surgical treatments, local health commissioners have confirmed.	NHS patients to be given option of travelling to Calais for surgical procedures	Patients travelling Calais

// Conclusion

- Model successfully generates concise summary for first 50 words of articles (taken as input)
- Valid and grammatically correct summary most of the times
- Attention mechanism in the model makes it easier to study the function of network
- Network learns to detect linguistic phenomena, such as verbs, objects and subjects of a verb, ends of noun phrases, names, prepositions, negations etc.

Bottlenecks

- Dataset Preprocessing
  - Many training examples contain the headline which does not summarize the text at all or properly
  - Many examples have headline in the coded form such as “asn-rocr-2stld-writethru-sut“
- Model Training
  - Took 3 days to train the model on Nvidia Tesla K80 GPU for 200 iterations

// Future Work

- Using other type of attention mechanism called “Complex Attention”
- Implementation of Bidirectional RNN which is seen to be working better with attention model
- Use smaller dataset with better feature extraction to reduce the training time
- Use of GRU cell (faster than LSTM) to further reduce the model training time

References

- Ian Goodfellow, Aaron Courville, and Yoshua Bengio. Deep learning. Book in preparation for MIT Press,2015.

- Bing L, Li P, Liao Y et al (2015) Abstractive multi-document summarization via phrase selection and merging[J]. arXiv preprint arXiv:1506.01597.

- Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. CoRR , abs/1506.03099, 2015

- Cao Z, Li W, Li S et al (2016) Attsum: joint learning of focusing and summarization with neural attention[J]. arXiv preprint arXiv:1604.00125

- Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions.CoRR, abs/1412.2306, 2014.

Questions?