Unstructured Information Processing Project Report, Monsoon 2020
Automatic Essay Grading

Aastha Shah, Nandini Agrawal

Problem Statement and Significance:

The aim of this project is to build a mechanism for automatically assigning numeric grades to texts, such as essays. Such an automatic essay scoring system (AES) has multiple applications, including but not limited to test grading for standardized tests such as the GRE, TOEFL, as well as open source applications for students and writers to check the quality of their writing. As a large number of students are taking these standardised tests, it can be a very time consuming process for humans to grade these essays, hence an automatic essay grader can help speed up the process. Considering the variation in text and the subjectivity of scoring such text/ essays objectively (without having to form fixed policies or rubrics), it is helpful to use a supervised learning approach to train models to score essays similarly. In our approach, we use an LSTM-based model with GloVe embeddings and achieve an average quadratic weighted kappa score of around 0.71, as opposed to state of the art models that report ~0.76.

This task requires that the predicted scores from our model match human assigned/ label scores as closely as possible.

Dataset:

For this task, we use the Hewlett Foundation's Automated Essay Scoring dataset from Kaggle. This dataset provides essays for along with scores (integer values) provided by expert human graders. On average, each essay is 150 to 550 words in length from a range of areas and topics. In total, there are about 13000 essays. The essays are from a variety of topics but are largely similar in tone/formality, but they are divided into 8 datasets (based on different prompts and score ranges). Hence, for this task, though we train on the entire data, we get results on separate datasets and average them. The distribution of essays and their score ranges is as follows:

PROMPT ID	ESSAYS	SCORE RANGE
1	1783	2-12
2	1800	1-6
3	1726	0-3
4	1772	0-3
5	1805	0-4
6	1800	0-4
7	1569	0-30
8	723	0-60

<u>Performance/ evaluation metric:</u>

Since here the scores are integer values, metrics like MSE may not be the best options for evaluating the model. If treated as traditional classification, since there are so many possible scores/ classes, it becomes harder to rely on accuracy. For this purpose, the original Kaggle competition from the Hewlett Foundation provides their evaluation metric in the form of a *quadratic weighted kappa* (QWK) error metric. This metric varies from 0 to 1, with 0 being random agreement between the automatic grader and human grader, and 1 being complete agreement between the automatic grader and human grader. (A negative value is also possible if the similarity is less than that expected by chance.)

The QWK is calculated as follows:
There is a weight matrix **W** as follows:

$$\mathbf{W}_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

where i and j are human ratings and automated ratings respectively, and N is the number of essays in the dataset.

Then, another matrix $\mathbf{O}_{i,j}$ is created that contains the count of the number of essays graded i by the human and j by the model. Finally, an expected count matrix $\mathbf{E}_{i,j}$ is created using the outer product of the vector of human and model ratings, and normalized for the sums of $\mathbf{O}_{i,j}$ and $\mathbf{E}_{i,j}$ to be the same. Finally, the QWK is calculated as follows:

$$\kappa = 1 - \frac{\sum_{i,j} \mathbf{W}_{i,j}\mathbf{O}_{i,j}}{\sum_{i,j} \mathbf{W}_{i,j}\mathbf{E}_{i,j}}$$

Another way to look at this would be to view the kappa metric as one that r**emoves the probability of random similarity from the observed/total similarity** to give us a more robust metric. Another formula would be:

$$\kappa = (p_o - p_e)/(1 - p_e)$$

where p_0 represents probability of the observed agreement and p_e represents expected agreement when both annotators assign scores randomly.

<u>Common Approaches:</u>

Automatic essay scoring has been approached from many different angles, including statistical approaches with feature engineering as well as deep learning methods. Some of these approaches involved explicit feature engineering by looking at the text. For example, word length, phrase lengths, POS tags, style, syntax of the text, TF-IDF etc., have been used as features. Neural approaches largely do not use these, but some concatenate such features to word embeddings. Common approaches have been:

Statistical approaches:

- Latent Semantic Analysis & Probabilistic Latent Semantic Analysis: These are some the oldest approaches on the topic. These models involve understanding the semantics of the text. They try to determine the 'meaning' of words and passages. For an automated task, often the meaning of the labelled essays are compared with the meaning of test essays to determine the similarities between these. These often use approaches like word document matrices, and can often be similar to TF-IDF methods wherein the frequencies of word occurrences within and across documents is examined. They may also use other features like POS tags and punctuation in computing similarity. The accuracy of these methods is often determined by using the Spearman correlation (Kakkonen et. al., 2006).
- Bayesian Text Classification: This was used in the BETSY grader (Lawrence & Liang, 2002) to categorize text automatically into 'appropriate', 'partial', and 'inappropriate' responses using Bayesian statistics and conditional probabilities. This also involved feature selection like information gain/ entropy, number of occurrences etc., for updating probabilities.

Machine learning/ deep learning approaches:

- K Nearest Neighbors: Such clustering mechanisms have been used to categorize text. Words/phrases and other features may be used to represent essays, and values such as TF-IDF may be used for determining distances using cosine or other metrics. Then, essays are categorized into point-scales and tiers i.e., scores 1-3 may be category A, 4-6 may be category B and so on. The KNN classifier is used to assign categories (Jiang & Jin, 2016).
- Neural methods: RNNs are most often used due to the temporal nature of text data. These often use mechanisms to map essays to high dimensional spaces and use supervised learning using the human-assigned labels to train the model. Sometimes, CNNs may also be used. Within neural methods, there has been an increase in the use of word embeddings for better representation of vocabulary. Pre-trained embeddings like Word2Vec are also improving accuracies as opposed to previous attempts that had to use less scalable methods like POS tagging or even named entity recognition.

Though a lot of work is being done on other areas within automatic text grading (e.g., tone detection or work on other languages), a lot of them are also domain-specific or task-specific. **This given task is fairly general and does not necessarily aim to look at niche metrics of 'quality', but just writing sentence structure/ style**.

<u>Our Approach:</u>

In this project, we opt for a neural model: an LSTM-based model with a 'mean-over-time' mechanism, as inspired by Taghipour & Ng (2016), along with GloVe embeddings. This process involves preprocessing the data, encoding it using pre-trained word embeddings, and passing it through layers to finally average them, and give a single output in the range [0, 1]. We describe this in depth in the next section. We went with an LSTM model for the following reasons:

- We saw that the statistical approaches required a fair bit of feature engineering, which can often be domain specific. We did not want to risk generalizability by making the wrong features. We also have different datasets for 8 prompts, and this could pose problems with this approach.
- We also had enough data (~13000 essays with roughly 300 words each) for training a deep learning model. We also saw that a fairly shallow LSTM model was good enough if used with the right embeddings.
- We did not want to go with other neural approaches like just CNNs because we want to check the sentence structure/ style of the essays, which is more of a temporal task than a spatial task. Though we do use a CNN layer, it is an optional addition and not the main part of the model, as explained later.
- They help us preserve important information in the form of intermediate states and are more efficient when it comes to dealing with the vanishing gradient problem.
- The 'mean-over-time' mechanism used with the LSTM model made our training process much faster as compared to a simple LSTM model.

Implementation Details:

Preprocessing

Cleaning Text

Cleaned Essays - For initial preprocessing, after loading our data, we clean up the text of the essays by keeping a **consistent (lower) case** and **removing contracted words** (i.e., "should've" becomes "should have" and so on). This results in uniformity across essays and makes sure we don't treat variations of the same word as different words with different numerical representations/ embeddings. We also **remove most punctuations and special symbols like * or :** for cleaner text. We realize that this might cause a decrease in accuracy in terms of the model understanding the sentence structure/ style, but we find that our final results were only marginally improved on including these. (We keep common punctuations like commas.)

Named Entity Recognition Tags - Next, we tackle the named entity recognition tags that are provided in the data. In the original dataset, names such as people and locations are removed and replaced with named entity recognition tags i.e., names of people are replaced with @PERSON1 and names of locations are replaced with @LOCATION2 and so on. Though they have identifier numbers to indicate specific entities, since we are grading the essay on the writing, **we remove these numbers** during preprocessing (along with lowering the case).

Normalization - Since our dataset has 8 prompts each with different score ranges, we also normalize the essay scores by dividing them by the maximum possible scores (thus giving us a score in the range [0, 1]. This gives us an even dataset so we can predict our scores using the sigmoid function.

Tokenization & Vocabulary

For this, we simply use the ‘nltk’ tokenizer function to split the sentences into tokens and create a vocabulary for our training data. **We do not do the usual steps in preprocessing such as lemmatization or removal of stop words** because while this would largely retain the meaning of the essay, we also want to account for the flow, structure, and writing. This would be difficult to determine if we had wrong forms of the word (by lemmatizing) or missing words (by removing stopwords). In fact, on removing stop words, our kappa metric dropped to around 0.38.

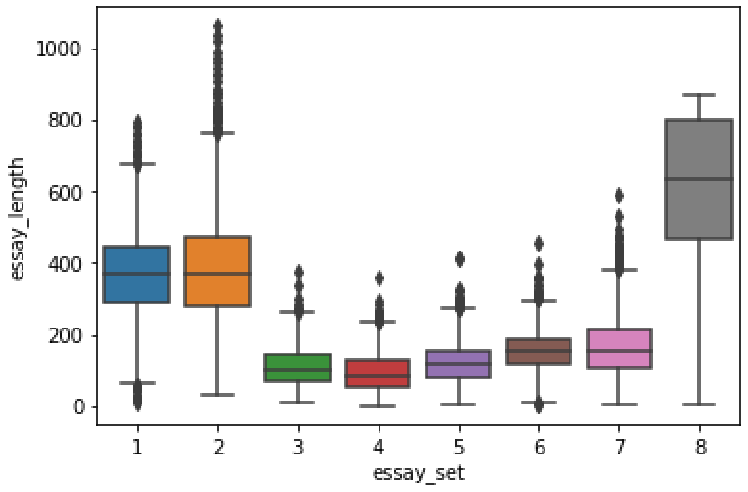
After tokenization, we create a dictionary to map each unique word/ token to a count of the number of times the word occurs in the entire dataset. This gives us the distribution of each word, and in our vocabulary we decide to only keep words that occur more than 10 times. For all such words that occur fewer than 10 times, we indicate them using an “UNK” token which indicates unknown value (this is also part of our vocabulary).

Encoding & Padding

Now, after the vocabulary is ready, we map each token/ word to an index that is a numerical value to indicate the word’s presence. This helps us convert the text to a list of integers for use in our model.

For example, using a vocabulary dictionary of { ‘dog’ : 1; ‘I’ : 2; ‘have’ : 3; ‘a’ : 4; ‘UNK’ : 5}, the text “I have a cat.” would map to [2, 3, 4, 5]

Moreover, we have also had to restrict the essays to a particular word length due to differences i essay length based on the essay set:



We found that considering 550 as the maximum length was sufficient since it covered almost all essays from all sets. **Later, we did see a drop in accuracy for essay set 8, but since there are the least number of samples in set 8, we went with 550.** We assume that these many words are enough for the model to understand the style/structure of the writing to adequately grade it. For essays shorter than this, we pad them with a 0 (to which no word is mapped). The reason for not padding it with the length of the longest essay is to ensure that shorter essays are not padded too much - hence the tradeoff.

Embeddings using Pre-trained Word Vectors (GloVe)

After our preprocessed text is ready, we also make use of pre-trained word vectors to find better relationships between our words in essays. For this problem, we use GloVe embeddings. These are similar to word2vec in the sense that they map text to vectors and capture the distance between words such that similar words/vectors are closer together and different words/vectors are far apart. Word2vec uses neural embeddings accounting for surrounding words in a corpus and maps words to vectors using these surrounding words (or a *local* distribution). But while word2vec accounts for the local distributions, the advantage of GloVe is that it also accounts for *global* distributions of words. So, while it accounts for surrounding words in a text, it also uses a co-occurrence matrix for the text as a whole to keep track of the occurrences of these phrases together. This accounts for the text as a whole, and this is an advantage in a task like this because the essays we have are, on an average, 250-300 words in length, and hence bound to have multiple sentences with different sentence structures. This global representation helps us account for those structures.

Upon trying the problem with word2vec, we noticed good results, but we wanted to check for better possibilities, so we tried for GloVe and found it to be slightly better as well. For this problem, we use **GloVe trained on Wikipedia** (with 100 dimensions). We decided to use the Keras “Embedding()” layer to embed our encoded input into appropriate matrices using GloVe. We create a matrix that maps the index of the specific words to vectors that we have received from the pre-trained GloVe embeddings.

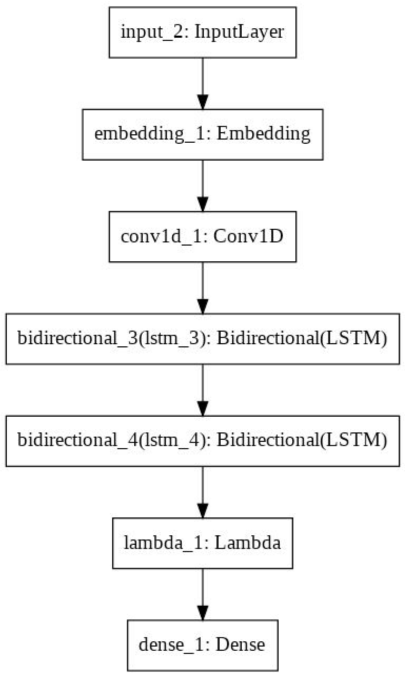
Model

After our preprocessed text is ready and we have prepared our embedding matrix, we used an LSTM model with mean-over-time to complete the task. Though we experimented with deeper networks, we found marginal improvement and even overfitting.

In our model, after the encoded text has been embedded, it is passed through **a CNN layer** to extract local features before passing it through the LSTM layer. It is kind of like looking at local correlations within sentences before using the LSTM to extract long-term dependencies across sentences/paragraphs. One may even remove this if the text is shorter.

Next, we pass it through 2 **bidirectional LSTM layers** to capture temporal and long-term dependencies to hopefully capture the sentence structure and style. The bidirectional allows for the same propagation to occur on the input as well as a reversed copy of the input. This helps account for the states in a forward and backward manner, to account for the effects that both the previous and latter words have instead of just checking in one direction.

Lastly, we pass the output from the LSTM layer to a **mean-over-time layer** which is simply a function to average out multiple vectors to one vector of the same length. **This basically averages the hidden states that we get from the LSTM layer for more efficient passing through the succeeding dense layer**. (This decreased our training time considerably.) After this, a dense layer gives one output. In between, we also use dropout (usually 0.25) to prevent overfitting.

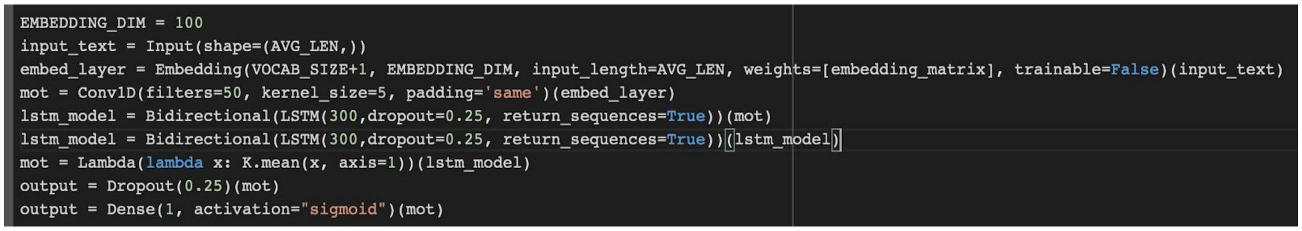

input_2: InputLayer
embedding_1: Embedding
conv1d_1: Conv1D
bidirectional_3(lstm_3): Bidirectional(LSTM)
bidirectional_4(lstm_4): Bidirectional(LSTM)
lambda_1: Lambda
dense_1: Dense

The final dense layer gives the predicted score for the essay. Here, we use the sigmoid activation function to give the final output score. We use this function for the following reasons:

- It is non-negative and can give values in the range [0,1] which works with our normalized scores.
- An alternative for the same would be not to normalize these scores, but then training would be difficult since the essay sets have vastly different ranges (while some go till 10 some go to 60).
- Without normalization, an alternative would be ReLU, but that might not be the best idea because that is unbounded. Because we are training on the entire dataset, essays with higher score ranges might affect predictions for other essays, and we might even predict scores that are much higher than the score range for that set.

Like most, if not all, activation functions, sigmoid also gives real numbers. Unfortunately, this cannot be changed, and hence we **round the answers predicted by the final model** for our kappa score calculation.

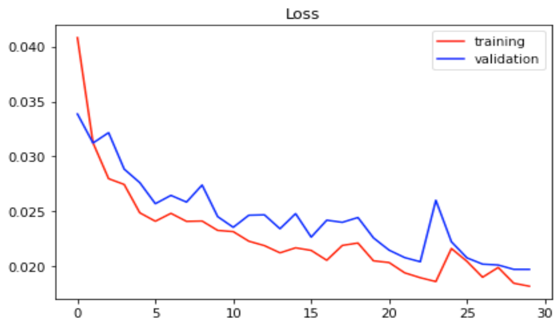
Thus, in this model, our encoded text data is embedded using GloVe embeddings and passed through LSTM layers to give a final essay score value. Given that the output is using sigmoid (real value), and this task can be posed as a regression task (between the human and model grader), we use the **'mse' loss function**. This ensures that larger differences between scores are penalized more.

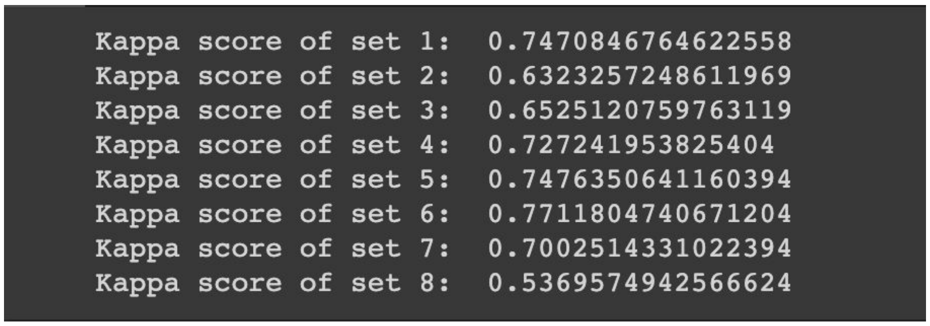


```
EMBEDDING_DIM = 100
input_text = Input(shape=(AVG_LEN,))
embed_layer = Embedding(VOCAB_SIZE+1, EMBEDDING_DIM, input_length=AVG_LEN, weights=[embedding_matrix], trainable=False)(input_text)
mot = Conv1D(filters=50, kernel_size=5, padding='same')(embed_layer)
lstm_model = Bidirectional(LSTM(300,dropout=0.25, return_sequences=True))(mot)
lstm_model = Bidirectional(LSTM(300,dropout=0.25, return_sequences=True))(lstm_model)
mot = Lambda(lambda x: K.mean(x, axis=1))(lstm_model)
output = Dropout(0.25)(mot)
output = Dense(1, activation="sigmoid")(mot)
```

Results:

On training the model on 80% of the data and testing on 20% of it using the adam optimizer for 30 epochs and a batch size of 256, we get the following results:

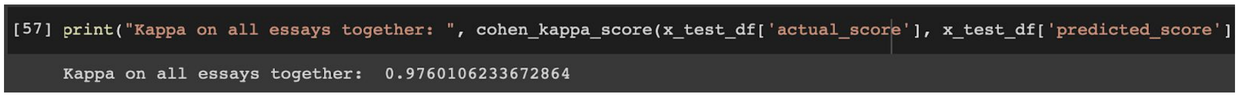




```
Kappa score of set 1:  0.7470846764622558
Kappa score of set 2:  0.6323257248611969
Kappa score of set 3:  0.6525120759763119
Kappa score of set 4:  0.727241953825404
Kappa score of set 5:  0.7476350641160394
Kappa score of set 6:  0.7711804740671204
Kappa score of set 7:  0.7002514331022394
Kappa score of set 8:  0.5369574942566624
```

We notice that while the rest of the essays have good results, set 8 has poor performance. This may be since we are restricting the length of the essay to 550 and a majority of essays in set 8 are of higher length than that. **In fact, upon trying to train it simply on set 8 and using a 800 max length gave us a slight improvement in set 8's kappa, reaching 0.6+.** The increase may not be much because there's not enough data for set 8.

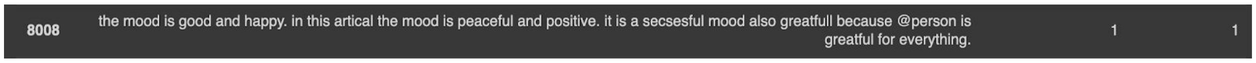
Weighing this by the size of the sets in the test case, this gives us an average kappa of **0.703.** On computing the kappa on all datasets together, we get **~0.97** (though this may not be an accurate metric, as mentioned by many examples in the literature, described in the next section).

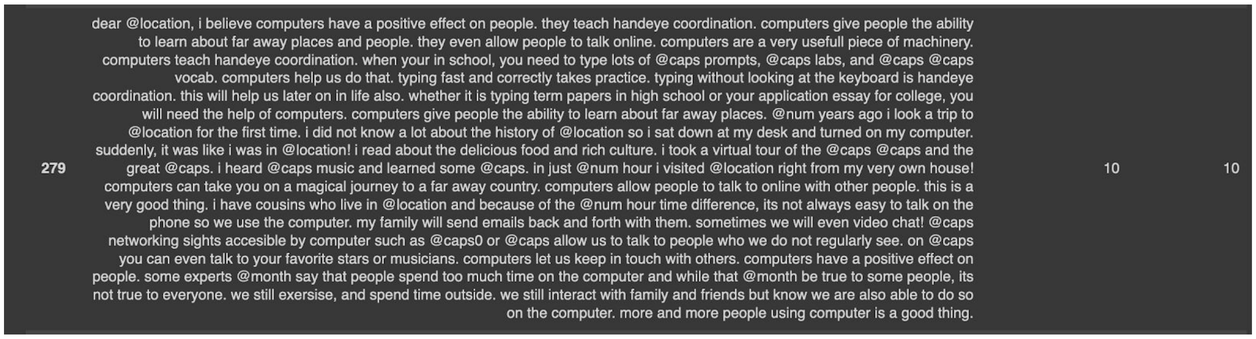


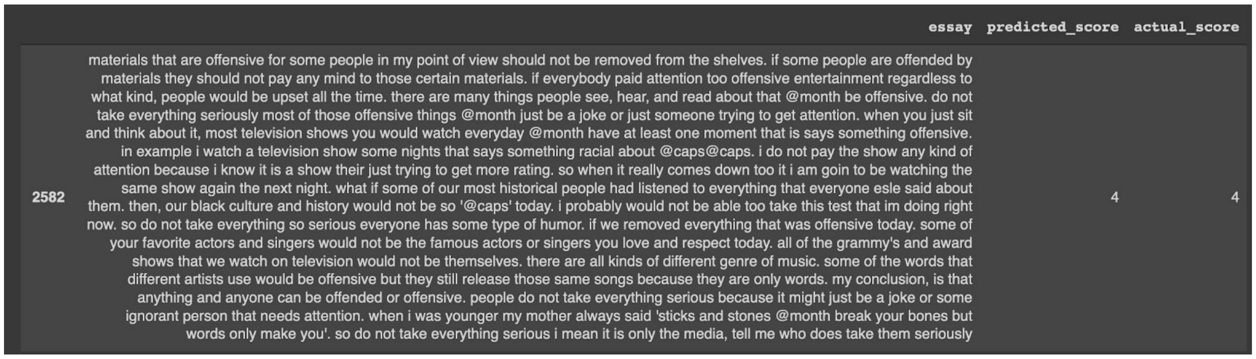
```
[57] print("Kappa on all essays together: ", cohen_kappa_score(x_test_df['actual_score'], x_test_df['predicted_score']

     Kappa on all essays together:  0.9760106233672864
```

These are some examples of our results:

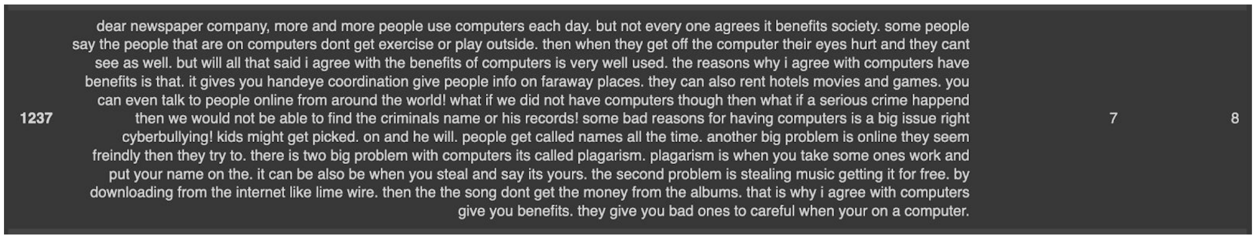




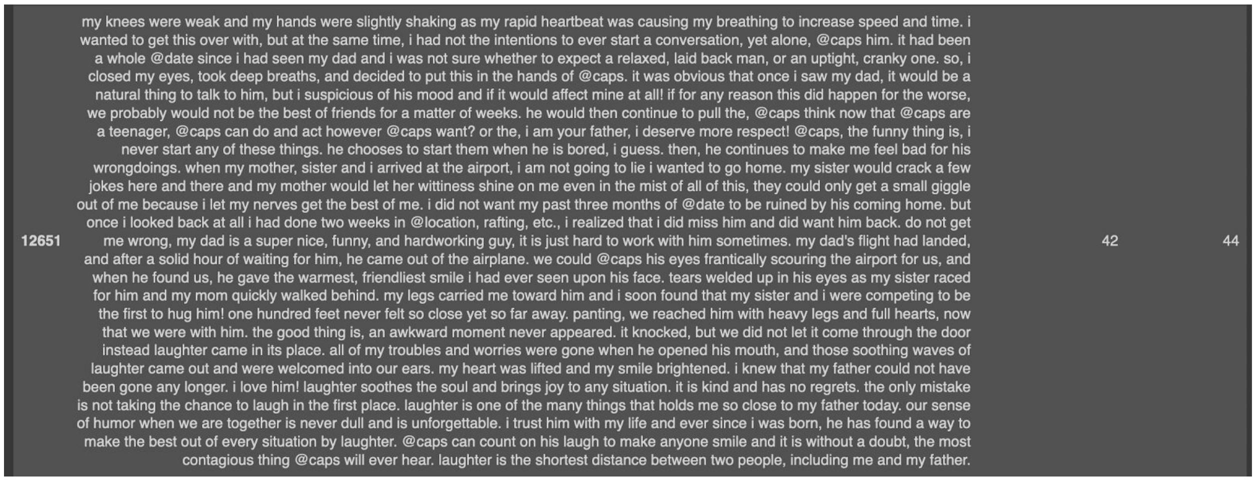


	essay	predicted_score	actual_score
8008	the mood is good and happy. in this artical the mood is peaceful and positive. it is a secsesful mood also greatfull because @person is greatful for everything.	1	1
279	dear @location, i believe computers have a positive effect on people. they teach handeye coordination. computers give people the ability to learn about far away places and people. they even allow people to talk online. computers are a very usefull piece of machinery. computers teach handeye coordination. when your in school, you need to type lots of @caps prompts, @caps labs, and @caps @caps vocab. computers help us do that. typing fast and correctly takes practice. typing without looking at the keyboard is handeye coordination. this will help us later on in life also. whether it is typing term papers in high school or your application essay for college, you will need the help of computers. computers give people the ability to learn about far away places. @num years ago i look a trip to @location for the first time. i did not know a lot about the history of @location so i sat down at my desk and turned on my computer. suddenly, it was like i was in @location! i read about the delicious food and rich culture. i took a virtual tour of the @caps @caps and the great @caps. i heard @caps music and learned some @caps. in just @num hour i visited @location right from my very own house! computers can take you on a magical journey to a far away country. computers allow people to talk to online with other people. this is a very good thing. i have cousins who live in @location and because of the @num hour time difference, its not always easy to talk on the phone so we use the computer. my family will send emails back and forth with them. sometimes we will even video chat! @caps networking sights accesible by computer such as @caps0 or @caps allow us to talk to people who we do not regularly see. on @caps you can even talk to your favorite stars or musicians. computers let us keep in touch with others. computers have a positive effect on people. some experts @month say that people spend too much time on the computer and while that @month be true to some people, its not true to everyone. we still exersise, and spend time outside. we still interact with family and friends but know we are also able to do so on the computer. more and more people using computer is a good thing.	10	10
2582	materials that are offensive for some people in my point of view should not be removed from the shelves. if some people are offended by materials they should not pay any mind to those certain materials. if everybody paid attention too offensive entertainment regardless to what kind, people would be upset all the time. there are many things people see, hear, and read about that @month be offensive. do not take everything seriously most of those offensive things @month just be a joke or just someone trying to get attention. when you just sit and think about it, most television shows you would watch everyday @month have at least one moment that is says something offensive. in example i watch a television show some nights that says something racial about @caps@caps. i do not pay the show any kind of attention because i know it is a show their just trying to get more rating. so when it really comes down too it i am goin to be watching the same show again the next night. what if some of our most historical people had listened to everything that everyone esle said about them. then, our black culture and history would not be so '@caps' today. i probably would not be able too take this test that im doing right now. so do not take everything so serious everyone has some type of humor. if we removed everything that was offensive today. some of your favorite actors and singers would not be the famous actors or singers you love and respect today. all of the grammy's and award shows that we watch on television would not be themselves. there are all kinds of different genre of music. some of the words that different artists use would be offensive but they still release those same songs because they are only words. my conclusion, is that anything and anyone can be offended or offensive. people do not take everything serious because it might just be a joke or some ignorant person that needs attention. when i was younger my mother always said 'sticks and stones @month break your bones but words only make you'. so do not take everything serious i mean it is only the media, tell me who does take them seriously	4	4

Despite the vast ranges we are able to get good results. Even when our predictions are wrong they are fairly close:



1237	dear newspaper company, more and more people use computers each day. but not every one agrees it benefits society. some people say the people that are on computers dont get exercise or play outside. then when they get off the computer their eyes hurt and they cant see as well. but will all that said i agree with the benefits of computers is very well used. the reasons why i agree with computers have benefits is that. it gives you handeye coordination give people info on faraway places. they can also rent hotels movies and games. you can even talk to people online from around the world! what if we did not have computers though then what if a serious crime happend then we would not be able to find the criminals name or his records! some bad reasons for having computers is a big issue right cyberbullying! kids might get picked. on and he will. people get called names all the time. another big problem is online they seem freindly then they try to. there is two big problem with computers its called plagarism. plagarism is when you take some ones work and put your name on the. it can be also be when you steal and say its yours. the second problem is stealing music getting it for free. by downloading from the internet like lime wire. then the the song dont get the money from the albums. that is why i agree with computers give you benefits. they give you bad ones to careful when your on a computer.	7	8



12651	my knees were weak and my hands were slightly shaking as my rapid heartbeat was causing my breathing to increase speed and time. i wanted to get this over with, but at the same time, i had not the intentions to ever start a conversation, yet alone, @caps him. it had been a whole @date since i had seen my dad and i was not sure whether to expect a relaxed, laid back man, or an uptight, cranky one. so, i closed my eyes, took deep breaths, and decided to put this in the hands of @caps. it was obvious that once i saw my dad, it would be a natural thing to talk to him, but i suspicious of his mood and if it would affect mine at all! if for any reason this did happen for the worse, we probably would not be the best of friends for a matter of weeks. he would then continue to pull the, @caps think now that @caps are a teenager, @caps can do and act however @caps want? or the, i am your father, i deserve more respect! @caps, the funny thing is, i never start any of these things. he chooses to start them when he is bored, i guess. then, he continues to make me feel bad for his wrongdoings. when my mother, sister and i arrived at the airport, i am not going to lie i wanted to go home. my sister would crack a few jokes here and there and my mother would let her wittiness shine on me even in the mist of all of this, they could only get a small giggle out of me because i let my nerves get the best of me. i did not want my past three months of @date to be ruined by his coming home. but once i looked back at all i had done two weeks in @location, rafting, etc., i realized that i did miss him and did want him back. do not get me wrong, my dad is a super nice, funny, and hardworking guy, it is just hard to work with him sometimes. my dad's flight had landed, and after a solid hour of waiting for him, he came out of the airplane. we could @caps his eyes frantically scouring the airport for us, and when he found us, he gave the warmest, friendliest smile i had ever seen upon his face. tears welded up in his eyes as my sister raced for him and my mom quickly walked behind. my legs carried me toward him and i soon found that my sister and i were competing to be the first to hug him! one hundred feet never felt so close yet so far away. panting, we reached him with heavy legs and full hearts, now that we were with him. the good thing is, an awkward moment never appeared. it knocked, but we did not let it come through the door instead laughter came in its place. all of my troubles and worries were gone when he opened his mouth, and those soothing waves of laughter came out and were welcomed into our ears. my heart was lifted and my smile brightened. i knew that my father could not have been gone any longer. i love him! laughter soothes the soul and brings joy to any situation. it is kind and has no regrets. the only mistake is not taking the chance to laugh in the first place. laughter is one of the many things that holds me so close to my father today. our sense of humor when we are together is never dull and is unforgettable. i trust him with my life and ever since i was born, he has found a way to make the best out of every situation by laughter. @caps can count on his laugh to make anyone smile and it is without a doubt, the most contagious thing @caps will ever hear. laughter is the shortest distance between two people, including me and my father.	42	44

On multiple runs, we get average kappa scores between 0.69 - 0.71. **Note that this is also susceptible to the manner in which the data is split since the data is not well balanced.**

<u>Limitations and Ways to Improve:</u>

- The current model does not perform well for a dataset 8, but we could improve that by training it separately on dataset 8 with a longer maximum length (though it would be a marginal increase due to the small size of dataset 8).
- This task is also very susceptible to the manner in which we preprocess the essays. Different styles of tokenization (like not including commas etc.) have an impact on the result, so it would be worthwhile to experiment with those.
- At the moment, while validating (during training), we look at the entire dataset. It would be interesting to account for datasets during this, to ensure that kappa scores are high for all datasets.
- We only take the encoded and embedded essay as our input. We could also include other features such as essay length, the dataset number, and so on.
- A more complex model could be helpful, though we did find some overfitting with more layers.

<u>Discussion & Comparison with State-of-the-Art:</u>

This problem has been approached many times over the years, however with the rise of LSTMs/neural models and better embedding models, very high accuracies have already been reached by many people since the seminal papers have been published. It seems as if the general problem on this given dataset has been tackled enough, and now papers are focussing on more specific problems like tone/ content etc. or different datasets/ languages altogether.

On going through some of the state-of-the-art methods in neural methods, we saw that many of them were older approaches and fairly simple in their architecture with high accuracies. For example, Taghipour and Ng (2016) used the mean-over-time method (that inspired us) to get a kappa score of **~0.75**. They did not use any sophisticated word embeddings, only numerical encodings. They even compared their approach to neural attention, but did not get as good a result. Perhaps this could be explained by attention generally being used in encoder-decoder seq2seq models when the input and output are of varied lengths. If we wanted to place more emphasis on the semantic meaning of the essay, perhaps attention would have been a better option. In 2016, Dong and Zhang tried this with a LSMT-CNN architecture with attention (without sophisticated word embeddings) but only reached **~0.76**, which is a marginal improvement. Dasgupta et. al. (2018) get better results by using other features along with the data. They try to 'qualitatively' enhance the dataset by adding psychological features (such as tone, emotion), lexical diversity (by analyzing the word choices), sentence embeddings from GloVe word embeddings, as well as the 'informativeness' of the text. Through this, they were able to reach around **~0.78** kappa**. They were also able to reach 0.97 kappa on all datasets combined, much like ours.**

These methods and scores hinted to us that there is worth in using the simple and effective mean-over-time architecture and experimenting the GloVe embeddings. Hence, we tried using these pretrained embeddings with mean-over-time and reached a 0.70 kappa score which brings us close to the state of the art even without anything complex.

On the other hand, the highest score so far has been Alikaniotis et. al. (2016) use score-specific word embeddings and bidirectional LSTMs to get a kappa *on the entire dataset* of around 0.96 (as reported above, we also reached this score on the entire dataset). These 'score-specific word embeddings' are similar to GloVe in the sense that they account for local and global distributions, but they also account for the *usage* of the words. They account for the occurrence of words across datapoints and weigh under-informative words as less. They also account for noise in the data such as spelling errors. This has been able to get them to the highest kappa score so far. **They do not provide essay-specific scores.**

<u>Conclusion:</u>

On experimenting with GloVe embeddings, **we were surprised at how good the results were even with a simple architecture**. Unlike other tasks such as chatbots and text summarization that need an even better understanding of the sentence structure and dependencies and may need more sophisticated architectures like attention (as we did in our Advanced Machine Learning class), this task did not require that. This perhaps indicates that even without these dependencies, the model is able to decipher the flow of the essay, vocabulary, and grammar fairly well. **This also highlights the importance of strong word embeddings like GloVe that are able to understand the meaning of the words to help understand the meaning and structure of sentences.** Largely, we were able to use sophisticated embedding techniques to get better results than older results that did not use those.

(The code for the report is attached on Google Classroom. We also have a few pre-trained weights and data objects saved, if needed.)

References:

Alikaniotis, Dimitrios, Helen Yannakoudakis, and Marek Rei. "Automatic text scoring using neural networks." *arXiv preprint arXiv:1606.04289* (2016).

Dasgupta, Tirthankar, et al. "Augmenting textual qualitative features in deep convolution recurrent neural network for automatic essay scoring." *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018.

Dong, Fei, Yue Zhang, and Jie Yang. "Attention-based recurrent convolutional neural network for automatic essay scoring." *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. 2017.

Kakkonen, Tuomo, Niko Myller, and Erkki Sutinen. "Applying latent Dirichlet allocation to automatic essay grading." *International Conference on Natural Language Processing (in Finland)*. Springer, Berlin, Heidelberg, 2006.

Rudner, Lawrence M., and Tahung Liang. "Automated essay scoring using Bayes' theorem." *The Journal of Technology, Learning and Assessment* 1.2 (2002).

Taghipour, Kaveh, and Hwee Tou Ng. "A neural approach to automated essay scoring." *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016.