Emotion Classification of Textual Data Linear Classification Against Transformer Models

Yang Kai Yam, Xu Michael, and Tian Yu Hao

School of Computer Science, McGill University

Abstract

In this project, we explored the Naive Bayes model and pre-trained transformer models in the context of emotional classification of real-world text data. A big part of the project this time was learning how to approach pre-processing of the data for our use. To that end, we explored sklearn's text processing libraries and auto-tokenizer from the transformers package. From the Numpy implementation of Naive Bayes, the best accuracy we achieved was 0.83 *and* 0.93 *from the BERT-based model. We found that although transformer models far outperformed the Naive Bayes implementations for emotion classification, the choice of model should be made carefully with available computational resources in mind.*

Introduction

The first challenge this time around was figuring out how we wanted to approach vectorization and tokenization of the input data as the experimental accuracy greatly varies depending on the pre-treatment applied. This is discussed in the Dataset section in conjunction with Section A. Our first implementation of Naive Bayes was done with a multinomial likelihood and Laplace smoothing and gave us a decent baseline model for the future. Tuning the model through different pre-processing strategies, we found the best test accuracy for Naive Bayes at 0.83.

In the second part of the experiment, we fine-tuned pre-trained BERT-based models and compared them to our previous Naive Bayes strategy. Our baseline performance here was a strong 0.927. So strong that, in fact, any further modifications brought to the model in the hopes of improving accuracy was only observed to be only marginal. The best prediction accuracy obtained overall was achieved by the Roberta-base-emotion model although every BERT-based model was very close in accuracy to the baseline.

We conclude that although BERT-based models are better at predicting emotion by a non-neglectable amount, Naive Bayes models are significantly lighter and require much less computational power to run.

Dataset

The dataset comprises of training and test sets split 16000/2000, each with 15186 features. We first vectorized the dataset using a bag-of-words representation. Further improvements were made in section A while tuning the Naive Bayes model, please refer to that section for the complete list of changes implemented. To summarize, we found that removing stopwords and adding lematization to the dataset is generally beneficial while vaying the word encoding strategy improved the accuracy however marginally.

Results

A. Naive Bayes (NB)

The model adeptly calculates both prior and posterior probabilities for emotion classification. The class prior probability is derived from the logarithm of class frequencies in the training data. Simultaneously, feature posterior probabilities are computed using the frequency of each feature within classes, augmented by Laplace smoothing to avoid zero frequencies. Our experiments are designed to assess the impact of each parameter, aiming to optimize the Naive Bayes configuration for enhanced performance.

A1. Choice of Vectorizer and Bag-Of-Word Our study focuses on comparing CountVectorizer and TF-IDF Vectorizer.

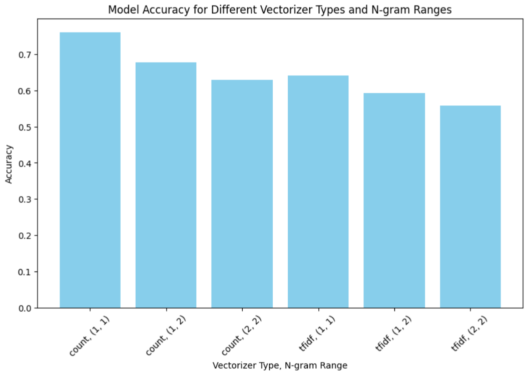


Figure 1: *The accuracy with different vectorizers and BoWs*

Using different ranges of n-gram encoding, Figure 1 shows CountVectorizer consistently outperforms TF-IDF Vectorizer. The highest accuracy observed is 0.76 with a unigram (1, 1) setup, followed by 0.68 using a combination of unigram and bigram (1, 2) in the count vectorizer setup. The former unigram configuration will serve as our baseline performance for the Naive Bayes (NB) model going forward.

A2. Influence of Stop Word Removal Using NLTK library [1], the impact of removing stopwords is investigated, assuming a count (1, 1) configuration. Figure 2 indicates a performance improvement, with accuracy increasing from 0.76 to 0.80 after stopwords are removed.

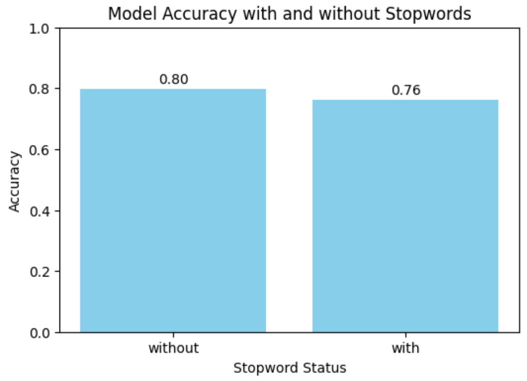


Figure 2: *The accuracy of NB (1,1) with and without stopwords*

A3. Influence of Lemmatization Assuming the same NB (1,1) setup with stopwords removed, we explored the effect of lemmatization on performance. As detailed in Appendix A, lemmatization's impact is marginal, yielding only a slight increase in accuracy of approximately 0.06.

With Lemma	Without Lemma
0.803	0.797

Table 1: *Test accuracies depending on lemmatization configuration*

A4. Finetuning with Smoothing We found experimentally that the optimal smoothing parameter ranges from 0.1 to 1 for our purpose. We determine the best Lidstone parameter in this subsection while building upon our previous best parameter findings (NB (1,2) with stopwords removed and added lemmatization). Validation accuracy is used here to tweak the parameter in question. Figure 3 illustrates the effect of varying the Lidstone parameter. Notably, a value of 0.2 seems to a yield the best result at 0.8295.

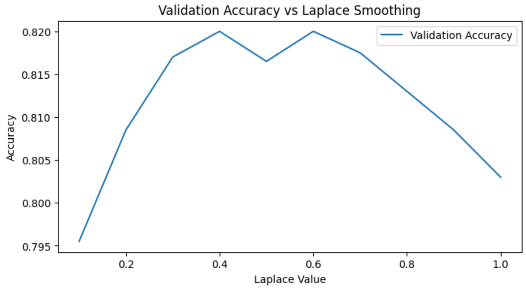


Figure 3: *The accuracy of different smoothing alpha*

Additionally, Appendix A2 presents the performance of different smoothing approaches: CountVectorizer with (1, 1) and TF-IDF Vectorizer with (1, 2), achieving accuracies of 0.821 at alpha 0.4 and 0.785 at alpha 0.1, respectively. We observe that Lidstone smoothing generally outperforms traditional Laplace smoothing.

A5. Using Gaussian Naive Bayes For experiment sake, we also implemented a Gaussian Naive Bayes model to compare with the Multinomial likelihood approach. As shown in Figure 4, the Multinomial Naive Bayes achieved a higher accuracy at 0.83, whereas the Gaussian Naive Bayes only reached 0.64. This suggests that a Gaussian assumption on the features may not be well-suited for our approach to vectorizing the dataset. We will delve into the reasons for this in the Discussion.

A6. Optimal Configuration of NB

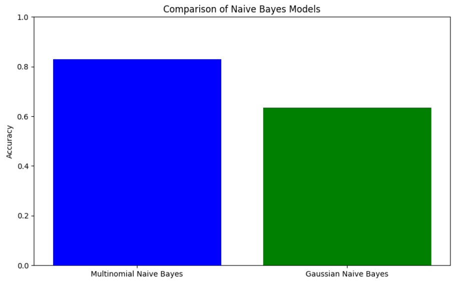


Figure 4: *Comparison to Gaussian Naive Bayes*

Configuration	Details
Smoothing Technique	Lidstone
Smoothing Alpha	0.2
Vectorizer	CountVectorizer
Stop Words	Removed
Lemmatization	Applied
Bag of Words Range	(1, 2)
Use of Gaussian	No
Best Accuracy	**0.83**

Table 2: *Optimal Configurations for Naive Bayes*

B. BERT-based emotion classifier

In the following section, we compare the different pre-trained BERT-based models and experiment with fine-tuning their parameters through various methods. A final comparison is made between the best bert-based model configurations and the results section A. The models in question can be found here.

B1. Baseline Performance and Light Tuning for Distilled-Bert The distilled-bert model we have as a baseline has 6 encoding and decoding blocks with GELU activation feed forward neural networks in between. The output layer has a linear pre-classifier and a final softmax output classifier. A preliminary run for untuned distilled-bert is performed, yielding an accuracy 0.927. This will serve as our baseline going forward.

Following that, we trained the weights for three epochs on our dataset with a learning rate of $\alpha = 5 * 10^{-5}$. The resulting training and validation loss curves are plotted in Figure 5. The resulting test accuracy is 0.9235. For reasons outlined in the discussion, we also decided to try training the using only the last few layers of distilled-bert, whose loss curves are plotted in Figure 6.

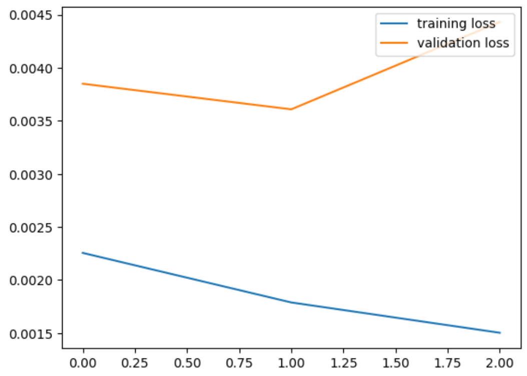


Figure 5: *Training and validation loss curves for distilled-bert (all weights). Accuracy = 0.9235*

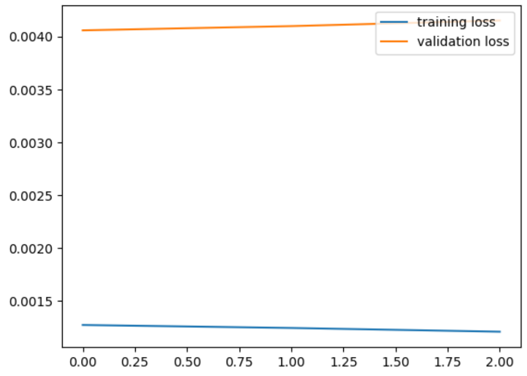


Figure 6: *Training and validation loss curves for distilled-bert (only output layer weights). Accuracy = 0.9275*

B2. BERT Experiments With the goal of further refining the experimental accuracy of the pre-trained BERT models, we tried modifying an extensive set of the tunable parameters within the transformer structure. The summary of results we obtained are outlined in Table 3. One thing we notice is that the accuracies yielded by slight modifications to the distilled-bert model are generally within marginal differences of the baseline result (untuned distilled-bert) outlined in the previous section. In actuality, we posit that the slight improvements or decreases observed in accuracy can be attributed mostly to random noise. That is to say that we were unsuccessful in tuning the model to significantly improve the prediction accuracy. On the other hand, we could also say that the base model was already powerful enough. This will be further elaborated upon in the discussion.

B3. Performance Comparisons In Table 4 are presented the BERT configurations we've discussed so far and the best test accuracies we

Parameter	Accuracy
Optimizer: SGD	0.9265
Optimizer: ADAM	0.9260
Learning Rate: $5 * 10^{-1}$	0.3475
Learning Rate: $5 * 10^{-2}$	0.6290
Learning Rate: $5 * 10^{-3}$	0.9245
Learning Rate: $5 * 10^{-4}$	0.9280
Learning Rate: $5 * 10^{-5}$	0.9260
Learning Rate: $5 * 10^{-6}$	0.9285
Drop Out (Attention Layer): 0.5	0.9240
Drop Out (Attention Layer): 0.7	0.9215
Drop Out (Attention Layer): 0.9	0.8985
Activation: ReLU	0.9220
Activation: Tanh	0.9170
Activation: Sigmoid	0.9250
Activation: Softplus	0.9245
Drop Out (Classification Layer): 0.5	0.9290
Drop Out (Classification Layer): 0.7	0.9270
Drop Out (Classification Layer): 0.9	0.9265

Table 3: *Accuracies of Distilled-BERT Model with Various Parameters*

Best Configurations	Accuracy
NB	0.8300
Distilled Bert (untuned)	0.9270
Distilled Bert (tuned)	0.9295
Non-Distilled Bert	0.9265
Roberta-base-emotion	0.9305

Table 4: *Best configurations and their following accuracies we obtained. Note: Distilled-Bert (tuned) has learning rate* $= 5 * 10^{-5}$*, epochs = 3 as well as training for all weights and a drop out rate of 0.5 instead of 0.2 (base) between the encoder and final classification layer.*

obtained from each of them.

B4. A deeper insight into the Self-Attention Layers To understand the transformer model better let's have a look at the self-attention layers.

Starting with incorrectly detected examples: In Figure 7 and Appendix B.1, we have the bert-base-uncased-emotion model that incorrectly classifies the sentence "I feel like my sweet company is finally coming together" as expressing "love" rather than "joy". On analyzing the attention matrix from the first transformer layer and its initial attention head, we observe that the attention weight between "sweet" and "company" is high, approaching one, while the weight between "sweet" and "together" is low, nearing zero. This pattern suggests that the model disproportionately focuses on the relationship between "sweet" and "company", overlooking the combined semantic contribution of "sweet" with "together". In common understanding, the conjunction of "sweet" and "together" strongly connotes the emotion of "love", indicating a possible area of improvement in the model"s attention mechanism. To more accurately differentiate between "joy" and "love", the model may benefit from a more balanced attention distribution across all three key words.

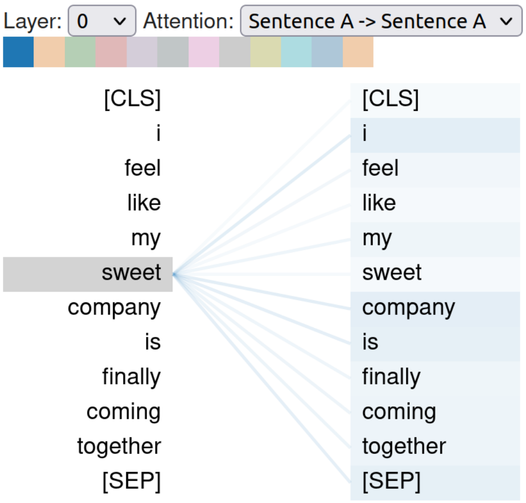


Figure 7: *Incorrectly predicted example's self-attention layer*

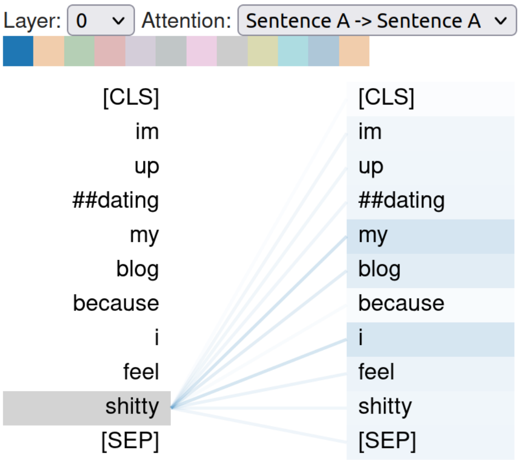


Figure 8: *Correctly predicted example's self-attention layer*

The model can accurately classify sentences like "I'm updating my blog because I feel shitty" and "I felt anger at the end of a telephone call" may largely due to the clear association of key words with specific emotions. From Figure 8 and Appendix B.2, analysis of the first transformer layer and the first attention head proves our assumption. In the both sentences, the weight between "shitty" and "I", "anger" and "I" are close to one, indicating a strong focus on these word pair. And these word pairs should be quite common with emotion labels in the training corpus. Since other word pairs, such as "blog" and "my" or "telephone" and "call", do not have clear emotional connotations, the model does not need to further disambiguate the emotions.

Discussion and Conclusion

Multinomial is better than Gaussian In the previous analysis, we found Naive Bayes may be outperforming Gaussian Naive Bayes. We believe Naive Bayes is better suited for handling the type of data produced by the Count Vectorizer. Count Vectorizer creates features that are counts of words, which are discrete numbers. Naive Bayes can directly use these counts and is specifically designed for such categorical data [2]. Gaussian Naive Bayes, on the other hand, assumes that the data follows a normal (bell curve) distribution, which doesn't fit well with the word counts. Hence, it doesn't perform as well with this kind of feature set.

Also, Naive Bayes is known for being effective even with large datasets. It's a relatively simple model and often works well with high-dimensional data like text.

On understanding BERT based models From our observations in section B4, we come to understand why BERT-based models perform better on emotion classification than Naive Bayes implementations. And that can be boiled down into three distinct reasons: 1. BERT's transformer architecture allows it to understand the context in which words appear. In other words, it has context built-in. It considers the entire sequence of words for its predictions, unlike Naive Bayes which assumes independence between the words (its features) for the conditional likelihood computations. 2. BERT has pre-training on exceptionally large datasets while Naive Bayes is limited by its simplicity. 3. Naive Bayes' accuracy suffers from imbalances in the data, while BERT does not have this issue.

On training BERT based models Our rationale in tuning the weights sparsely in section B1 has to do with the belief that training the last few layers (last transformer layer, pre-classifier, and classifier layers) will yield us good enough results on the accuracy given our limited training time and available resources. And as we see from the results discussed about Table 3, within the context of our emotion classification task, training in itself was unnecessary as we could not obtain anything significantly better than the base performance. This leads us to believe that the untuned distilled-bert model pre-trained on an external corpus is 1. already significantly better than the Naive Bayes implementation without any tuning and 2. powerful enough to the point where further learning is not worth the additional computing resources needed (which is already incredibly high).

Reference

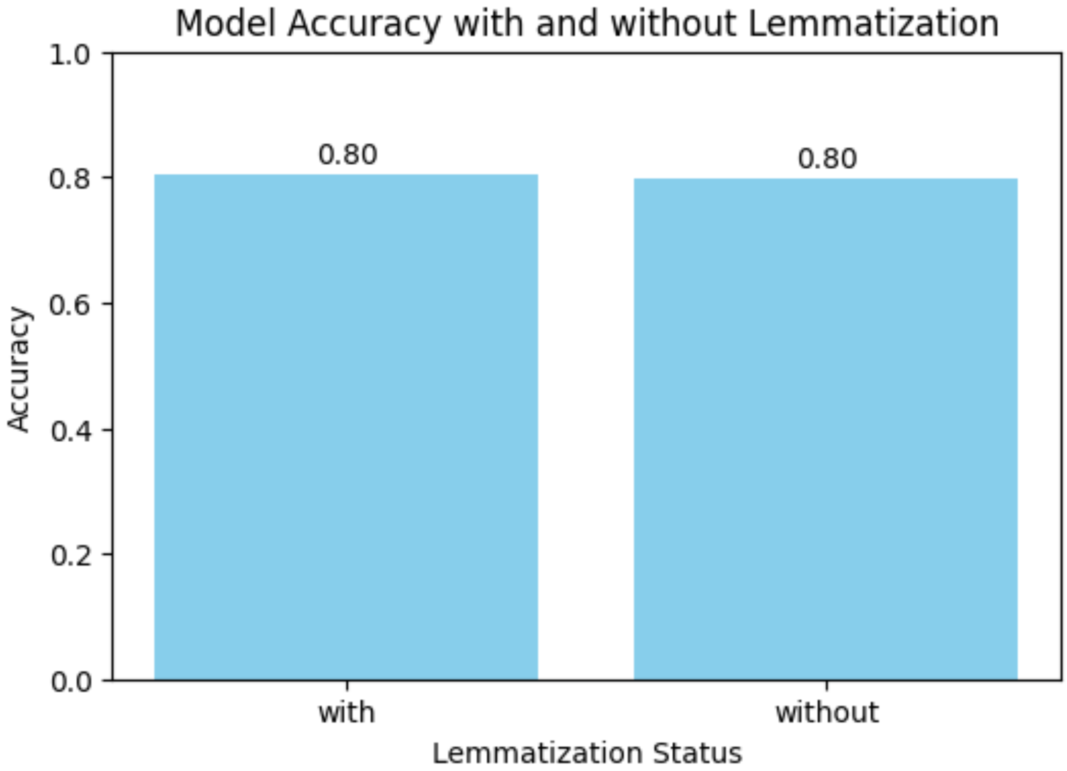
1. NLTK, The NLP Library, https://www.nltk.org/

2. Feldman, R., & Sanger, J. (2006). Text Classification and Naïve Bayes. In The Text Mining Handbook (pp. xx-xx). Cambridge University Press.

Appendix

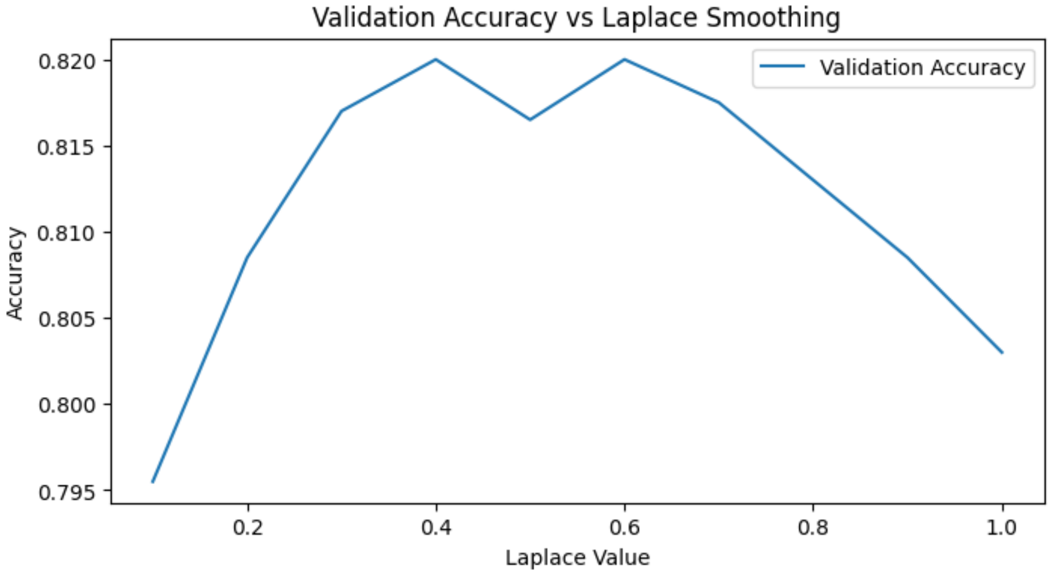
A. Naive Bayes

1. **Analysis on how lemmatization affect the accuracy**
   Additonal analytical graph is shown here to find out the influence of lemmatization on the training data, using Naive Bayes

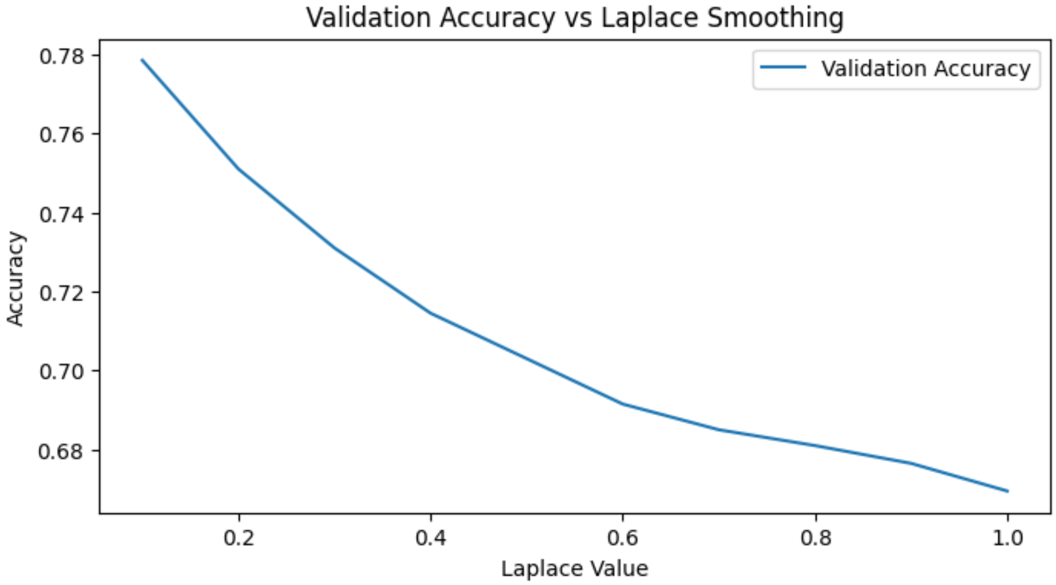


(Figure A1.1 The Figure of the model accuracy with different lemmatization settings)

2. **Analysis on the Libstone and Laplace Smoothing**
   The following graphs shows the effect of different smoothing alpha with other parameter configuration in Naive Bayes.



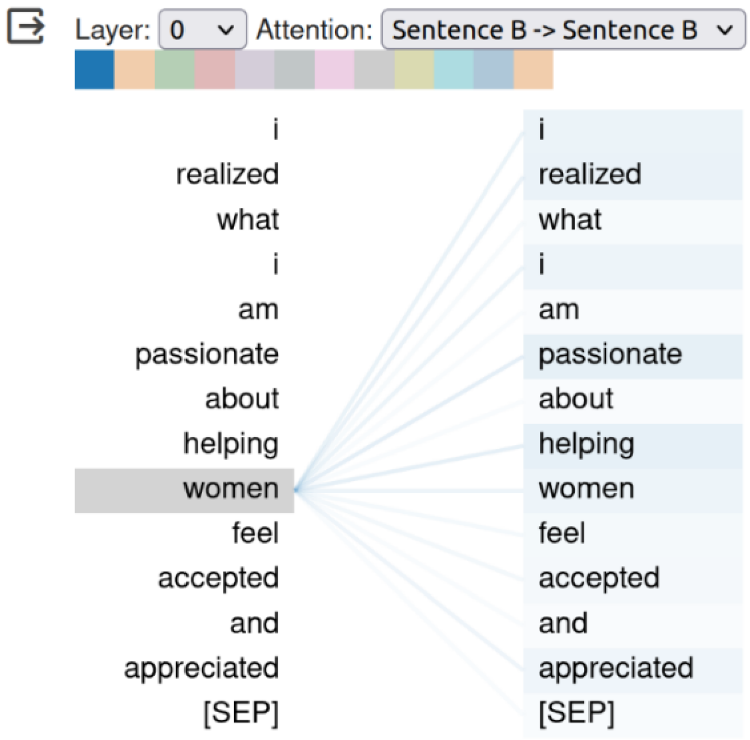
(Figure A2.1 The validation accuracy with CountVectorizer (1, 1))
The highest test accuracy is 0.821 with alpha = 0.4



(Figure A2.2 The validation accuracy with TD-IDF Vectorizer (1, 2))
The highest test accuracy is 0.785 with alpha = 0.1

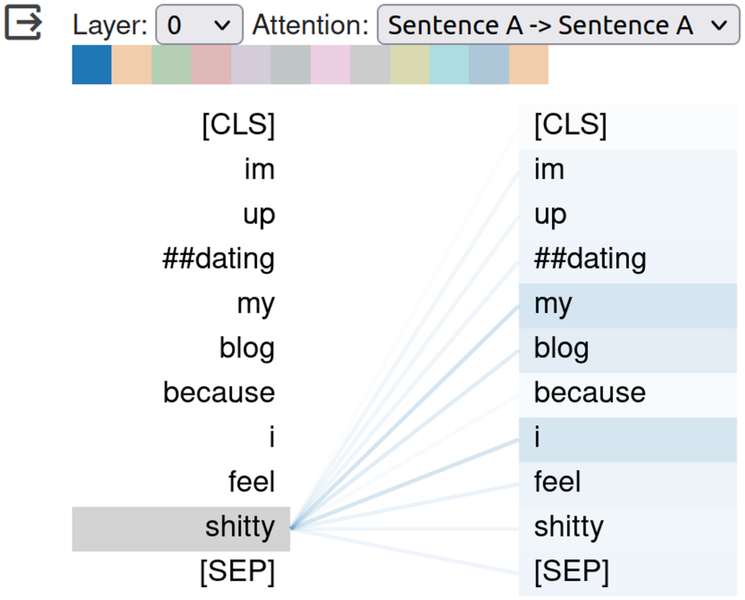
B. BERT Transformer

1. Visualization on attention matrix for incorrect classification



Layer: 0 Attention: Sentence B -> Sentence B

2. Visualization on attention matrix for correct classification



Layer: 0 Attention: Sentence A -> Sentence A