Building N-Gram Language Models from Scratch

Raymond Lyon
lyon0210@umn.edu

1 Introduction

This project demonstrated and tested the differences between generative and discriminative language models. The models were trained on text from four different authors. The samples that the models were trained and evaluated on were utf-8 format. For evaluation, each of the models was trained on 90 percent of the overall text corpus, and the remaining 10 percent was set aside for development testing. The models were compared using their test accuracy scores, training times, and failure cases. Conclusions were made about the differences in the mod

2 Generative Model

The generative model for this project was implemented using NLTK's LM package[Pro23]. The model used in the final version of the code for this project was a bigram model. Other numbers of grams were tested and it was determined that the bigram worked the best for this specific dataset. Each author had their generative model trained on only their work, otherwise, the models would not be able to generate new tokens similar to the author.

2.1 Data

As mentioned before, the data for training and evaluating these generative models was formatted utf-8. Each line of the data was striped of newline characters and leading or trailing white space to ensure a concise format. Once stripped, the line was tokenized using the NLTK word tokenized function so that any unnecessary characters that did not add context were removed.

All of the models shared the same vocabulary so that new tokens would not automatically be characterized as something that the model has never seen. To account for unknown tokens, the <unk> token was added to the shared vocabulary. When the model found a word it did not know, it would replace the word with this unknown token to continue processing.

Conference'17, July 2017, Washington, DC, USA
© 2024 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Author	Bigram	Trigram	4-Gram
Austen	69.94	67.37	66.45
Dickens	68.89	68.5	66.57
Tolstoy	71.88	69.77	67.10
Wilde	43.78	41.59	41.34

Table 1. Depicts model accuracy vs the different n-gram models with a discount set at 0.75. Numbers represent the accuracy percentage. Names of authors refer to the author's model.

Author	Bigram	Trigram	4-Gram
Austen	**70.85**	68.84	66.18
Dickens	**69.57**	69.28	66.57
Tolstoy	**72.32**	71.01	68.47
Wilde	**44.39**	42.07	41.22

Table 2. Depicts model accuracy vs the different n-gram models with a discount set at 0.90. Numbers represent the accuracy percentage. Names of authors refer to the author's model.

2.2 Model Settings

The final model was a Bigram model that utilized Kneser-Ney smoothing and interpolation. This was chosen throughout testing because when the model found tokens or sequences it had never seen before, it would estimate the probability of the sequence using a distribution and thus could estimate the value of a token.

Further tuning was performed to find the correct 'discount', or weight on higher order n-grams, to find the spot which returned the best accuracy. The final selection was a discount of 0.9 as it seemed to be the highest accuracy for the development set and within the range of real-world application use.

2.2.1 Testing Different Parameters.

Choosing the right number of grams was important because it changed the author's accuracy. Combinations of n-grams and discounts were tested to find the spot where accuracy was highest and perplexity was the lowest.

3 Results

Once the appropriate techniques were used to increase accuracy. The model was trained and evaluated on the development set. The final accuracies for the different authors are bolded above in Table 2.

Number	Prompt
1	In the small village there was a knight in silver
2	The king ran to the queen with
3	He found himself somewhere he had never
4	Once upon a time, in a
5	The people of the city screamed as

Table 3. Prompts that were created to test the models on generation.

Prompt	Austen	Dickens	Tolstoy	Wilde
1	1042.17	437.21	393.48	476.73
2	556.76	541.83	90.02	232.20
3	512.59	224.26	83.93	134.35
4	101.08	135.66	533.62	526.78
5	173.69	258.18	100.70	175.99

Table 4. Perplexity scores of prompt plus the newly generated tokens for each model.

The results showed some drastic differences in the author's accuracies. Wilde was significantly lower than all the rest of the authors. One reason for this is that Wilde had less training data than the rest of the models, around 10k samples vs 13k to 20k. The models show that the authors with the most training data often had the highest accuracy when evaluated on the development set, as Tolstoy had around 20k lines of data and had the highest accuracy.

3.1 Prompt Testing

To test the generative capabilities of these newly trained models, 5 prompts were created and the models were asked to generate the next n tokens. Once each model had generated the next n tokens to the prompt, the text was fed back into the model to calculate the perplexity. The prompts were fairly short which could have caused problems for the models since there was little context. When the models generated text, the text often reads like gibberish. It was hard to understand what the model was trying to add to the sentence but no two models generated the same sentence which was interesting. The generated responses can be found at the top of the document here. Original prompts can be found in table 3, and perplexities are reported in table 4 below.

4 Discriminative Model

The second type of model created in this project was a discriminative model. This was a single model that was trained on the text from all four of the authors.

Similar to the generative model, the data was split in a 90-10 ratio of training data to development data. The data was shuffled so the training data saw samples from all classes and the test data was balanced. Each row of data was tokenized using the auto tokenizer in the Hugging Face library to ensure the model could process the data.

Once the data had been tokenized and separated into training and test data, the model was trained. This model utilized the distilbert-base-uncased model[San+19] from Hugging Face as the pre-trained model and further trained it on the data provided. The training parameters used were a learning rate of 2e-5 with 5 epochs and a weight decay of 0.01. With these parameters, training took around 4 minutes using Google Colab's A100 GPU cluster. The training loss drastically decreased throughout the 5 epochs, starting at 0.88 and ending at 0.16, at which the model was considered fully trained.

Once trained, the model was applied to the test set that was partitioned at the creation time of the model. For each item in the test data, the model was applied to predict the label. The label was then checked against the ground truth where accuracies and misclassifications were recorded. The final results are listed in the table below.

Author	Accuracy
Austen	80.44
Dickens	76.27
Tolstoy	86.50
Wilde	74.14
Average	79.33

Table 5. Discriminative model accuracy for the different authors. Calculations are made using the development set of data. Accuracies are in percentages.

5 Comparison

When comparing the two models, some key differences stand out. The first of these is the accuracy between authors. The generative approach saw drastic differences in the accuracies between Wilde and the other 3 authors, with Wilde being significantly lower. The discriminative approach did not have this large difference. While there was still a slight difference, it was around 5-10 percent lower rather than nearly 20 percent with the generative model. A reason for this could be the architecture of the models. The generative model learns the context and mapping of words from the previous n tokens, for this experiment n was 2. The discriminative model learns the mapping and context from the entire sentence which allows to to better understand the difference in the author's writing style.

Another major difference in the model types was the training and testing time. The generative model could run on Google Colab's CPU machine and complete training and evaluation within 4 minutes. The discriminative model had to be processed using the A100 GPU cluster, a significantly more powerful system and took nearly 5 minutes to complete

Text	Predicted	Truth
way – teaching them , playing with them , and loving them .	Wilde	Austen
Military Library , Whitehall .	Wilde	Austen
eyes , as she said , looking fixedly at me	Tolstoy	Dickens
this , come and see my wife .	Dickens	Tolstoy
fingers on his handkerchief .	Wilde	Tolstoy

Table 6. Generative model failure cases.

Text	Predicted	Truth
of silvery light from the unseen sun.	Tolstoy	Dickens
what there is in you."	Wilde	Tolstoy
given me," he continued, in a voice of great agitation, "was of	Dickens	Austen
but in the present case you know, the connection must be impossible.	Tolstoy	Austen
must be happy, too!"	Tolstoy	Wilde

Table 7. Discriminative model failure cases.

training and evaluation. The reason for this also stems back to the model's architecture. The generative model is only looking at the last n words for context, this means that there should be less computation and the model can work on less powerful machines. The discriminative model uses context from the whole sentence and has to back-propagate model weights to train the model. This requires more computation power and more time but results in a stronger model.

5.1 Failure Cases

Both generative and discriminative models had multiple failure cases, or misclassifications, within the test set. The generative model had nearly twice as many misclassifications as the discriminative model. This could be because the generative model only uses the last 2 words as context and when the sentences lack distinct words that the author uses, it is difficult to predict which author. A list of all the misclassified samples can be found on a document here.

Failure cases between models share one strong similarity. The test text is usually a partial, short, sentence and lacks information about what is going on. The sentence uses general words which do not help distinguish the specific authors. A possible way to combat these failure cases would be to add more context to the test case, add longer sentences, or expand the training data so the model can further learn the relationships between authors and their works.

Success cases were usually seen in long sentences with lots of context and pairs of words that were seen throughout training because the model had learned these relationships.

6 Conclusion

Throughout this project, generative and discriminative language models were tested and compared to determine strengths and weaknesses. Generative models have the luxury of faster training on lower n-gram sizes with the cost of lost context and lower accuracies. Generative models also have the benefit of being able to generate text, whereas discriminative models can only classify text seen.

To improve both of these models, a good step would be to increase the data size. It was shown that models with larger training corpora had higher accuracies when tested on the development sets. After an increase in data, the models of this project could be duplicated to find the best accuracies and parameters once again.

References

[San+19] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *ArXiv* abs/1910.01108 (2019).

[Pro23] NLTK Project. *NLTK LM package*. 2023.