Dating Text From Google NGrams

Kelsey Josund
Computer Science
Stanford University
kelsey2@stanford.edu

Akshay Rampuria
Computer Science
Stanford University
rampuria@stanford.edu

Aashna Shroff
Computer Science
Stanford University
aashna@stanford.edu

Abstract

Google makes hundreds of gigabytes of n-gram data available as part of the Google Books project, a massive dataset of words, phrases, and metadata that has been underutilized. We aim to predict the distribution of an unseen 5-gram and display it similarly to the phrase occurrence graph Google's NGram Viewer has and then use 5-gram distribution predictions to determine the age of longer texts. In addition to this, we plan to classify when a 5-gram was likeliest to have occurred, and we use this to date books

1 Introduction and Prior Work

The task of identifying the age of text has been approached using a variety of corpuses using several different techniques. In previous offerings of CS224N at Stanford, before the addition of the deep learning emphasis, students attempted to date books from Project Gutenberg using Support Vector Machines [1] and basic Naive Bayes and Maximum Entropy Classifiers [2]. Both achieved reasonable results but suffered from a small, skewed dataset that had to be verified by hand as Project Gutenberg does not make year of authorship available. Of these, a unigram-based SVM had best performance, achieving an F1 score of 0.68 when deciding the binary problem of whether a text was written before or after the year 1600. The Naive Bayes classifier could distinguish between five centuries with an F1 score of only 0.17.

Outside of this class, Dalli and Wilks [3] applied time series analysis to word frequency within the English Gigaword Corpus, noting that word use has a periodic component that repeats itself in a predictable manner, and a non-periodic component that is left after the periodic component has been filtered out. They attempted to classify specific dates within a small window of nine years and had predictions, on average, within 14 weeks of the correct date. Finally, Ciobanu et al. [4] employed a pairwise approach between documents to attempt to create an ordering and then extrapolated from some known text ages to predict text age. Although the Google Ngrams dataset makes year of publication available with each n-gram in its corpus and the Google Books project is famous for its visualization of n-gram use over time, there are no known studies attempting to predict this distribution.

Taking inspiration from the distribution-based evaluation metric in Socher et al. [5], we used a deep learning approach to model the distribution of the Google 5-gram fiction corpus. Because our dataset is comprised of 5-grams and GLoVe provides vectors for individual words, one of our

approaches generates "5-gram vectors" in a RNN-based manner similar to Palangi et al. [6], which they showed to be an effective manner of combining word vectors to represent a phrase in the context of information retrieval.

2 Current Project

2.1 Problem Description

Our project uses Google n-grams to attempt to solve two problems in historical linguistics, which is the study of the evolution of language over time. More concretely, our problems are to:

1. Predict the age of text documents
2. Predict how the frequency of an n-gram has varied over time

Both these problems have a wide range of applications in both technical and humanities fields. The first is a useful tool in dating texts and predicting when texts with controversial authorship times, like Shakespeare's plays and the anonymous Federalist papers [1, 7], may have been written. The second is useful in seeing how trends in writing styles and word choices emerged and re-emerged over time in literary works. Predicting the trends of certain words is also useful for scientific researchers to determine facts about the history of science and acquiring references often unknown to researchers [8].

2.1.1 Dataset

Google makes n-grams of length 1, 2, 3, 4, and 5 available for all books scanned and processed as part of the Google Books project. We selected the English Fiction 5-grams 2012 subset of this data for several reasons:

- 5-grams are likely to contain additional information about the relationship between words that smaller n-grams would not include.
- There has been criticism of the full n-grams dataset for over-representing religious texts in early centuries (roughly pre-1900) and scientific papers in more recent years (roughly post-1950) which would skew the dataset's representative power.
- The 2012 update to the corpus fixed many previous errors due to computer vision failings that led to misspellings or incorrect words making it into the dataset.

We then removed all 5-grams prior to 1781 or after 2000, as there were too few old texts and too many recent texts to properly balance the dataset, and converted listed years of publication to twenty year buckets, rounded up (for example, anything listed between 1781 and 1800 was labeled 1800). The buckets approach attempts to correct for uncertainty in year of writing and publication, as books often are written over the course of several years, and also reduces the number of final classes.

Most 5-grams occurred in multiple buckets, so at this point, the dataset consisted of 5-grams along with the counts of their occurrences in each bucket from 1781-2000. However, as Figure 1 shows, the dataset is heavily biased towards later buckets, therefore simple counts do not capture the trend of a n-gram over years. To resolve this, we normalized our dataset in two directions to create distributions of occurrences for each 5-gram, similar to those plotted by Google on the Ngram Viewer website. The steps are described below:

1. **Probability of a 5-gram occurring in a particular bucket:** Consider a 5-gram x that appears across multiple buckets. Let the number of times it appears in bucket i be n_i. Let the total number of occurrences of all 5-grams that occurred in bucket i be t_i. Then, the probability of x occurring in i is computed by $p_x^i = \frac{n_i}{t_i}$. This is computed for each of the 11 buckets for each 5-gram, so that each 5-gram now has the following vector associated with it: $x' = \begin{bmatrix} \frac{n_1}{t_1} & \frac{n_2}{t_2} & \dots \frac{n_{11}}{t_{11}} \end{bmatrix}$.
2. **Normalizing each 5-gram:** The vector computed in the above step is normalized using an L2 norm so that its values sum to 1. This is done in order to represent 5-gram with a distribution that can be later predicted using a softmax function: $x = \begin{bmatrix} \frac{p_x^1}{\|x'\|} & \frac{p_x^2}{\|x'\|} & \dots \frac{p_x^{11}}{\|x'\|} \end{bmatrix}$

For all experiments, we split the data into train (83%), dev (8.5%) and test (8.5%).

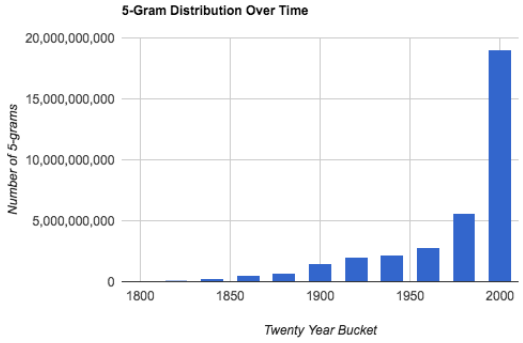

Figure 1: The imbalance of the raw Google Ngrams dataset, with a column for each of our eleven buckets.

2.1.2 Tasks

We begin by first classifying 5-grams into buckets and later classify entire books into buckets by splitting them into 5-grams and selecting the most frequently predicted bucket. The input x to our models is a 5-gram represented by concatenated 50-dimensional GloVe word vectors along with an additional indicator of capitalization. Then, our tasks our to predict the following:

- **Classifying 5-grams into buckets:** y, the twenty-year bucket class that the 5-gram has the highest probability of occurring in.
- **Predicting distribution of 5-grams:** y, a distribution of the probabilities of the 5-gram occurring over the eleven two decade buckets.

2.1.3 Evaluation Metrics

- **Age of Books:** We classify each text into a two-decade bucket within the time range 1781 through 2000. Since the buckets from the later years are more densely populated, we evaluate using F1-score for each bucket class:

  $$F_1 = \frac{2pr}{p+r}$$

  where p and r stand for precision and recall respectively. We then average the F_1 score across each class to obtain micro-averaged and macro-averaged F_1 scores. Macro-averaging gives equal weight to all classes, whereas micro-averaging gives equal weight to each 5-gram classification. Due to the small number of examples in earlier buckets, we report micro-F_1 scores.
- **Popularity of 5-gram over time:** Since we want to predict a distribution of occurrences of each 5-gram over time that most resembles the actual distribution, we evaluate using KL-Divergence, which is a measure that compares probability distributions:

  $$KL(g||p) = \sum_i \log(\frac{g_i}{p_i})$$

  where g is the gold distribution and p is the predicted one. We report the average KL-Divergence over all n-grams in the evaluation dataset, where a smaller value indicates lower error.

2.2 Baselines

We attempted to solve the two tasks using different deep learning approaches, described in the following sections. In order to compare our models to traditional methods, we used the following methods:

- **Random:** Since there are 11 classes, this gives .09 F1.
- **Most Frequent:** Selecting the bucket which has the most occurrences of 5-grams (this is the 1991-2000 bucket).
- **Logistic Regression:** Basic regression between classes using a logistic function.
- **Naive Bayes:** Predicting buckets and distributions based on a strong independence assumption and maximum likelihood.

2.3 Deep Learning Architecture

We employed different deep learning approaches to our problem. The models, described below, were used for both the bucket classification and distribution tasks. Each model predicted a probability distribution for a 5-gram. This was simply used as the output for the distribution task, whereas for the bucket classification task, the output was the bucket with the highest probability from the distribution.

2.3.1 Simple Neural Networks

We use 1-hidden layer, 2-hidden layer and 4-hidden layer feedforward neural networks which are trained on 5-gram vectors comprised of concatenated GloVe word vectors. Each layer in the neural network comprised of a sigmoid activation function followed by a softmax output.

Tables 1 and show the results for these models. Even without any additional hyperparameter tuning, data processing, or more complex models, these simple neural networks outperformed the other baselines. Of these, the 2-hidden layer performed the best. The mathematical formulation of the 2-hidden neural network model is below:

$$
\begin{aligned}
e &= xL \\
h_1 &= \sigma(eW_1 + b_1) \\
h_2 &= \sigma(h_1W_2 + b_2) \\
\hat{y} &= \text{softmax}(h_2U + b3)
\end{aligned}
$$

where $L \in \mathbb{R}^{V \times D}$ are the word embeddings, $W_1 \in \mathbb{R}^{D \times H}, W_2 \in \mathbb{R}^{H \times H}, b_1, b_2 \in \mathbb{R}^H$, are the parameters for the hidden layer, and $U \in \mathbb{R}^{H \times C}$ and $b_3 \in \mathbb{R}^C$ are parameters for the softmax. Note that V is the size of the vocabulary, D is the size of the word embedding, H is the size of the hidden-layer and C is the number of classes, which in our case, is 11.

Then, for the task of bucket classification, we use cross-entropy loss with L2-regularization:

$$
\begin{aligned}
J(\theta) &= CE(y, \hat{y}) + \frac{\lambda}{2} \|\theta\|^2 \\
CE(y, \hat{y}) &= \sum_{i=1}^{C} y_i \log \hat{y}_i
\end{aligned}
$$

For the task for predicting distributions, KL Divergence (as described in the Evaluation Metrics section) was itself used as the loss function.

2.3.2 5-gram Vector Embeddings using LSTM-RNN

In the simple neural network architectures, each word was represented by its pre-trained GloVe vector representation, and then concatenated to obtain a 5-gram vector representation. In the following section, we attempt to directly learn a semantic vector for a 5-gram using a LSTM-RNN architecture, similar to Palangi et al. [6].

The goal of using an LSTM-RNN model is to sequentially take each word of a sentence, extract its information, and embed it into an embedding vector. The LSTM-RNN can accumulate increasingly richer information as it goes through each word in a 5-gram, and when the fifth (i.e., the last) word is reached, the hidden layer of the network provides a semantic vector representation of the entire 5-gram. A visual depiction of the model is shown in Figure 2.

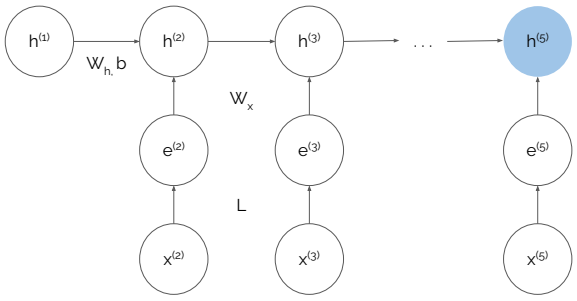

Figure 2: The basic architecture of the RNN for 5-gram embedding. The hidden activation vector corresponding to the last word is the sentence embedding vector (blue).

The output of the LSTM-RNN is $x' \in \mathbb{R}^H$ where H is the size of the hidden-layer. This is then fed-into the 2-hidden layer neural network described in the section above to obtain predictions for buckets and distributions.

3 Results

3.1 5-grams Bucket Classification

Table 1 summarizes the performance of all our baselines and implemented models for classifying 5-grams into buckets. Although we expected the LSTM-RNN phrase vector encoding to outperform naive concatenation of the five word vectors, the simple two-hidden-layer architecture without the RNN step performed slightly better. The simple 2-hidden layer model obtains a F1-score of 0.26, with the LSTM-RNN model with 2-hidden layer following closely behind with a F1 score of 0.25. The 2-hidden neural network model was trained with a learning rate of 0.001 and hidden layers of size 300 and 400, with 50% dropout and no regularization. It is possible that the semantic n-gram encoding using LSTM-RNN still had the potential to be superior, and increasing the hidden layer size and using more data would have improved performance.

Table 1: F1-Scores for 5-gram bucket classification

Model	F1-score
Random	0.09
Most Frequent	0.03
Naive Bayes	0.11
Logistic Regression	0.13
1-hidden-layer sigmoid/softmax	0.21
2-hidden-layer sigmoid/softmax	0.26
4-hidden-layer sigmoid/softmax	0.22
LSTM-RNN and 2-hidden-layer	0.25

Having seen some success with the simple 2-hidden layer model, we train on larger datasets, which had the same normalization scheme but approximately thirty million lines, to evaluate how it changes the results. As shown in Table 2, the performance becomes substantially better, with the large dataset achieving a F1-score of 0.35, indicating that increasing the training data and training time substantially help improve predictive power of the model. The corresponding confusion matrix is shown in Figure 3. We also show how train loss and squared error vary over time in Figure 4.

Table 2: Dataset comparison for 5-gram bucket classification using simple 2-hidden layer model

Dataset	No. of lines (train/dev/test)	F1-score
Small	3 mil/300k/300k	0.25
Medium	10 mil/1 mil/1 mil	0.3
Large	30 mil/2 mil/2 mil	0.35

	1800	1820	1840	1860	1880	1900	1920	1940	1960	1980	2000
1800	1869	1127	2313	5096	2656	464	275	2225	471	69	2398
1820	400	3029	2555	6053	3700	691	408	2890	732	148	3318
1840	519	1215	6270	10138	5641	1273	857	5234	1253	171	6323
1860	302	944	3269	17766	9282	1871	1747	9508	2058	202	11008
1880	118	745	1940	11051	16029	2151	2479	11789	2788	315	14876
1900	79	562	1453	8686	9808	7679	3372	12408	3182	443	15813
1920	46	332	993	6728	8054	2607	10057	15169	4701	480	22429
1940	85	313	847	7319	7607	2001	4071	29394	7148	569	29478
1960	36	231	576	4607	5282	1408	2562	14734	14732	1252	40144
1980	16	143	397	2369	2529	813	1155	7002	5722	5125	30259
2000	25	75	261	1741	2713	472	1430	6581	5431	1137	145373

Figure 3: Confusion matrix of our final model's predictions (across the top) versus gold label (down the side) twenty year buckets, evaluated on 5-grams.

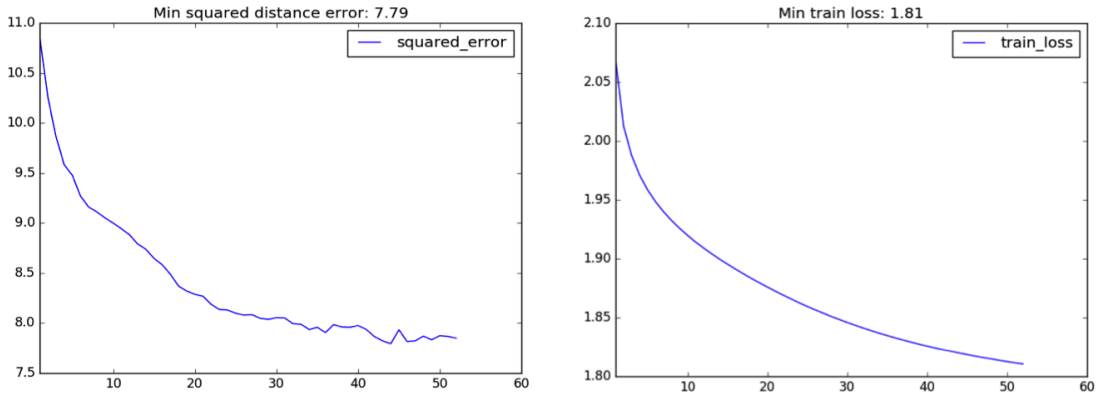

Figure 4: Train loss and squared loss vs. epochs for simple 2-hidden layer model used to classify 5-grams into buckets

3.1.1 Distribution Prediction

Table 3 summarizes the KL-Divergence results for the baseline algorithms and implemented models. Like bucket classification, the GloVe word vector concatenation outperformed the learned 5-gram vector representation for 2-hidden layers. The lowest KL Divergence resulted from a learning rate of 0.01, hidden layers of size 500, 40% keep probability for dropout, and 0.0001 as the delta for L2 regularization. We ran the simple 2-hidden layer neural network on larger datasets, results of which are reported in Table 4. As expected, we achieved a considerably lower KL-Divergence of 0.51 with the large dataset.

We evaluated our model's performance on distributions both quantitatively, with overall KL Divergence on an unseen set of test n-grams, and qualitatively, by inspecting generated plots of actual versus predicted distributions. On the plots that follow, the distribution calculated from the Google Ngram data is in blue and our model's prediction is in red.

With all of the neural network approaches we tried, it was fairly easy to get a KL Divergence below 0.7 with minimal hyperparameter tuning on a variety of dataset sizes. Our final

Table 3: Average KL-Divergence for distribution prediction

Model	KL Divergence
Random	3.45
Naive Bayes	2.96
Logistic Regression	0.74
1-hidden-layer sigmoid/softmax	0.63
2-hidden-layer sigmoid/softmax	0.6
4-hidden-layer sigmoid/softmax	0.69
LSTM-RNN and 2-hidden-layer	0.67

Table 4: Dataset comparison for 5-gram distribution prediction using simple 2-hidden layer model

Dataset	No. of lines (train/dev/test)	KL Divergence
Small	3 mil/300k/300k	0.56
Medium	10 mil/1 mil/1 mil	0.54
Large	30 mil/2 mil/2 mil	0.51

architecture as described above achieved a KL Divergence of 0.5 after training on nearly 30 million example n-grams for four epochs.

Some 5-grams, even when some of the words in the phrase did not have a corresponding GloVE vector, had predictions that matched the gold distribution very well, as demonstrated by Figure 5. This example demonstrates the power of individual words within a 5-gram: in this case, the word ”squeers” has the most predictive power. It, alone, had a very similar distribution of the 5-gram in this example, and is the last name of a character in a Charles Dickens novel written in the 1830s.

The majority of plots looked more like those in Figure 6, however. The model predicted a distribution with the general correct shape and trend but which did not match the actual distribution in the details. In most cases, our model predicted smoother distributions than the gold data suggests.

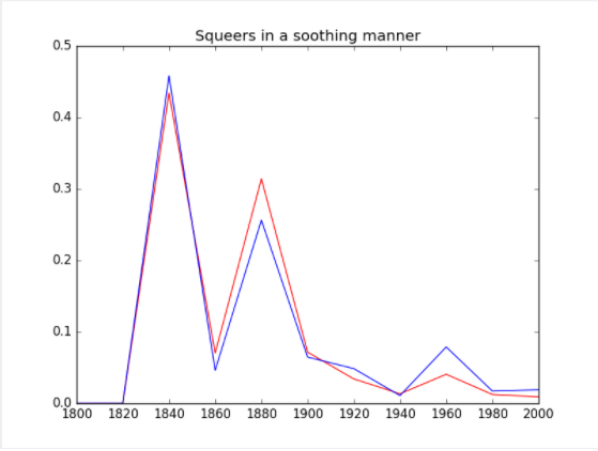

Figure 5: Example of an accurate distribution prediction.

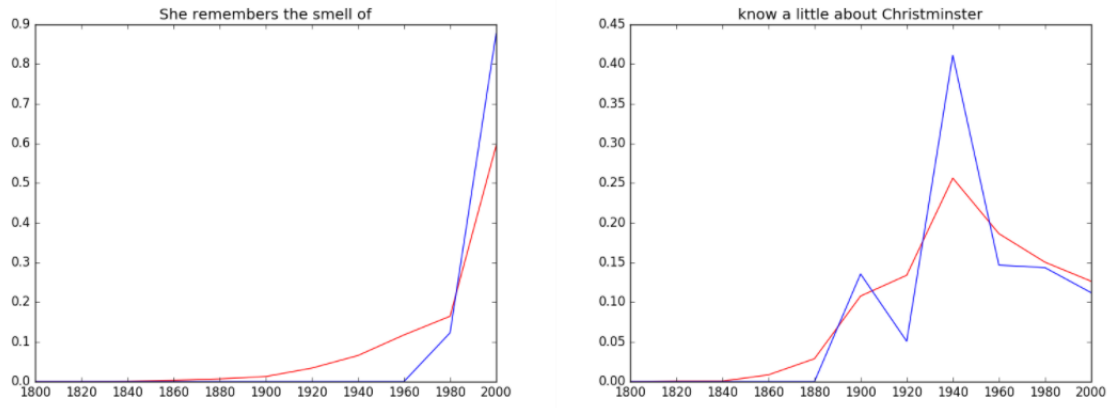

Figure 6: Examples of more typical distribution predictions, which demonstrate accurate trends but not details.

4 Book Classification

4.1 Motivation and Description of Problem

One of the biggest shortcomings of predicting a single bucket to classify a 5-gram is that the 5-gram does not actually fit into just one bucket. While we predict the bucket in which it is most likely to appear, a 5-gram can appear very frequently over several decades. For example, the 5-gram "Where have you been ?" may occur in each of our buckets ranging from 1800-2000. Assigning just one bucket to a phrase as ubiquitous as this is inadequate. In order to make the applicability of our model more meaningful, we evaluate our model to date entire documents. The model is trained on the Google 5-gram corpus as before, however it is tested on entire documents by splitting them into 5-grams. In this case, the classification problem is well formed, as each book can only belong to the bucket it is written. We download 120 books from Project Gutenberg that are evenly distributed across 1781-2000 and attempt to predict the twenty-year bucket it belongs to. Additionally, we attempt solve a binary classification problem to predict which century, the 19th or the 20th, the book was written in. We were inspired to perform this problem from Cope [1], where an F1 score of 0.68 was realized in predicting whether a document was written before or after the 16th century. We were able to significantly improve on this score by using deep learning techniques, and by working on the much harder problem of classifying before or after the 19th century.

4.2 Method and Subsampling

We use a 2-hidden layer neural network, the best model for n-gram bucket classification as a starting point for this task. Since the Google ngram training data is heavily biased to include more recent books (see Figure 1), we implement subsampling of each minibatch at variable percentages for training and evaluation. Our subsampling implementation balanced each minibatch equally or capped the percentage of a bucket as a fraction of the entire minibatch. This, consequently, reduced our dataset on some runs. The subsampling cap was set to be very low, and performance did not improve linearly with the increase in balance across bucket data. The best subsampling percentage allowed enough representation to learn, but not so much as to overwhelm the model.

The 2-hidden layer neural network with subsampling outputs a distribution of probabilities for each bucket for every 5-gram from the document. For each bucket, we add up the probabilities across all 5-grams to obtain a single probability distribution across buckets for the entire document. For the century classification problem, a similar procedure was followed, with the additional step of adding the probabilities of each bucket of to a century to obtain a single distribution over two centuries. For both problems, the final result was the bucket with the maximum probability.

4.3 Results

In order to fully appreciate our results, we must try and understand how difficult it is to get this prediction right. Most 5-grams in a book are common phrases that are not really indicative of the year of publication. Also, since most of our data is from more recent times, little biases are built up in otherwise universally occurring phrases, and our model is skewed to predict a book belonging to more recent year. After lots of experimentation, we used subsampling of our minibatches to solve this problem. Please note that Table 5 records a comparison of how well we did when sampling the same result time and again. Note that our best performance on the bucket classification is an 0.39 F1 score, which is much better than a random prediction, and very powerful given that we had 11 buckets.

Table 5: Performance comparison for book classification across 20-year buckets using various sampling techniques

Dataset	F1 Score
No minibatch sampling	0.24
15% minibatch sampling	**0.39**
10% minibatch sampling	0.19
5% minibatch sampling	0.19

With regards to century classification, first, we must note how it is computed: add all the predicted probabilities of buckets between 1800 and 1900, add all the predicted probabilities 1900-2000. Compare the two sums, and report the century with the higher score as the predicted century, Keeping this in mind, let us examine Figure 7 closely and look at at the actual dates of books that have been classified one way or another. Obtaining an F1 score of 0.74, when compared to random and naive baselines, is a very good performance.

Looking closely at the graph, we see that the only two books that have wrongly been assignment to the 19th century are books published in 1901 and 1902. Moreover, virtually every book that occurs more than 20 years before or after the demarcating date, is in the appropriate bucket. This does give us some intuition about how the algorithm works. The algorithm learns to pay particular attention to certain kinds of words by back-propagating into them during training. For example, in Table 6, we have covered the purported reason why each of the 5-grams have a very high confidence of being in a particular century. Sometimes, words occur because of culture, sometime it is because of linguistic patterns, and some times it is due to both.

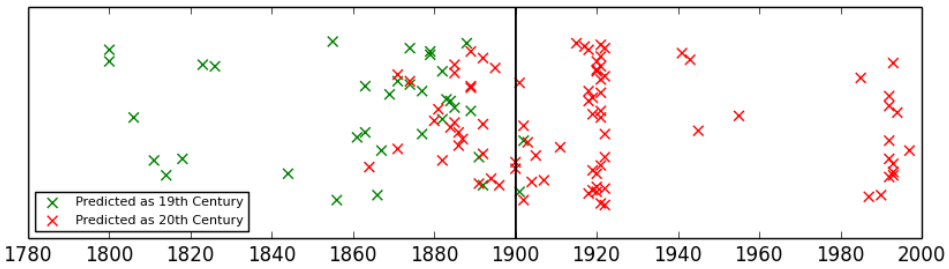

Figure 7: Example of an accurate distribution prediction.

5 Future Work

As we have shown, applying deep learning to the text dating problem with Google Ngrams is an under explored area with great potential.

Performance of our model was significantly better when trained on 30,000,000 examples

Table 6: Examples of 5-grams with high predictive power for 19th and 20th centuries.

5-gram	Century	Description
('case', 'of', 'the', 'Cistercian', 'buildings')	19th	A new order of the Cisterians religous sect called the Trappists was formed in 1892, which justifies their increased usage over the 19th century.
('allegiance', 'to', 'Urbain', 'de', 'Bellegarde')	19th	Urbain de Bellegarde was a character in a Henry James book entitled 'The American'. The term 'allegiance' is of importance here, since it was almost solely used in the 19th century, and its usage has been declining since.
('teenager', 'enthralled', 'by', 'computers', ',')	20th	Popularity of computers, especially amongst teenagers, is uniquely a 20th century phenomenon.
('he', 'was', 'experiencing', 'computer', 'problems')	20th	Since computers are a 20th century phenomenon, it only makes sense for this to be categorized as 20th century

than when trained on 1,000,000 examples. It seems likely, therefore, that the addition of both longer and shorter n-grams or simply additional 5-grams could improve performance; given more computational time and storage, providing the model with more data from which to learn would have been our next step.

Our selection of twenty-year buckets for our distribution calculation was somewhat arbitrary. Previous studies have not thoroughly justified their choice of classification range as well, and researchers have attempted similar projects using buckets as small as a week or as large as a five centuries. Given the structure of Google Ngram data, with its recency bias but overall enormous dataset size, dating older n-grams (and texts) to larger windows or more recent n-grams (and texts) to smaller windows could be an effective means of taking advantage of Google's rich dataset. An alternate approach, as suggested in [4], would be to model the problem as a regression problem to make more flexible temporal predictions for books, as opposed to a multi-class classification problem with different time-intervals.

6 References

[1] Cope, A. (2013) Authorship and date classification using syntactic tree features. *CS 224N 2013, Stanford University.*

[2] Tausz, A. (2011) Predicting the Date of Authorship of Historical Texts. *CS224N 2011, Stanford University.*

[3] Dalli A. & Wilks Y. (2006) Automatic dating of documents and temporal text classification. *In Proceedings of the Workshop on Annotating and Reasoning about Time and Events* (ARTE '06). Association for Computational Linguistics, Stroudsburg, PA, USA, 17-22.

[4] Ciobanu A. N., Dinu L. P., Zampieri M. & Niculae V. Temporal Text Ranking and Automatic Dating of Texts. In *European Chapter of the Association for Computational Linguistics* (EACL '14).

[5] Socher R., Pennington, J., Huang E. H., Ng A. Y. & Manning C. D. (2011) Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (EMNLP '11). Association for Computational Linguistics, Stroudsburg, PA, USA, 151-161.

[6] Palangi H., Deng L., Shen Y., Gao J., He X., Chen J., Song X. & Ward R. (2016) Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval. *IEEE/ACM Trans.* Audio, Speech and Lang. Proc. 24, 4, 694-707.

[7] Holmes D. I. & Forsyth R. S. (1995) The Federalist evisited: New Directions in Authorship Attribu- tion. *Lit Linguist Computing 2*.

[8] Marazzato, R., & Sparavigna, A.C. (2015). Using Google Ngram Viewer for Scientific Referencing and History of Science. *CoRR, abs/1512.01364*.