**Tim Zhang (110746199)**

**CSE519 Project Proposal: Dating Documents**

**Overview**

The task of dating documents is a k-classification problem of natural language processing. Here the objects in our feature space are collections of text and the k-classification objective is defined over 200 possible publication years (1817-2017). To achieve reasonable predictive performance, I will need to carefully curate my datasets and preprocess the parsed data in an intelligent manner.

There are a variety of techniques for processing text data into useable formulations, and I will leverage the considerable work in NLP to my advantage for this project. While there are many corpora for learning, few have easily accessible meta-data which includes the year of the document. Unfortunately, this will require me to work a little harder to get access to the precious data.

The remainder of my project proposal is structured as follows. First I will provide a distillation of my literature survey in methods for document dating. Next, a rudimentary implementation framework will be presented. Then I will discuss the candidate datasets and their relative pros and cons. The report will close with a description of how I will measure and evaluate my progress for the next project milestones.

**Literature Survey**

**Jong et al. [1]** present a survey of methods for the temporal classification problem. Their approach uses a unigram model which assigns the unigram probabilities conditioned on the year label. They define a set of constraints that a good reference corpus (training set) should satisfy: sufficient size, balanced distribution over the time span, same domain as the test set, and temporal span over the same period as the test set. These constraints seem reasonable, and I discuss how my datasets meet or violate them later in the report.

Next they discuss granularity of classification based on time periods (say decade level vs year level). I believe that this is a reasonable method, and I also came up with the idea independently. They define two general techniques for classification: document level comparison and partition level comparison.

In document level comparison, all training documents are compared against the test document. A ranking is given and the top *k* are chosen for further computation. Using these top candidates, a weighted sum is computed for each time span partition. The maximum sum is chosen as the class of the test document. It seems to me that what they are proposing is a variant of k-nearest neighbors, although they never go as far as to say it explicitly.

In partition level comparison, a language model is built from all the documents in the training corpus which is conditioned on the time class. Then the target classification is simply the maximizer of the probability conditional on the class. This is a general class of algorithms, and I believe Naïve Bayes will suffice for my problem.

**Kanhabua and Norvag [2]** present a more detailed description of a dating algorithm based on [1]. Here they use different distance metrics and score computations. They also discuss incorporating Google Zeitgeist search statistics to add another parameter to their scoring model. I plan on incorporating Google Ngrams as a parameter for my models, so it was interesting to see how they justified their approach.

**Implementation**

I plan on using a Naïve Bayes classifier and fitting the distributions of words with respect to 10 year bins. I believe that decade resolution will provide for a better baseline than yearly resolution since there is likely an uneven representation of documents per year. After using solving the k=20 problem at decade resolution, I will then feed the results to a different Naïve Bayes classifier trained at the inter-decade level. This finer detailed model will possibly call Google Ngrams to help disambiguate between the year classification.

At first I will consider Naïve Bayes using unigram bag of words features as my baseline model. For extensions on the baseline, I will try using higher order Markov processes, such as bigrams and trigrams. I will also consider k-nearest neighbor approaches and some of the other methods discussed earlier in the literature survey. It may also turn out that mixing and matching different learners for the different resolutions of classification will lead to more accurate results.

As with all practical implementations, there are some tricks that I will need to perform to get reasonable results. I will use NLTK for my processing library and scikit-learn for modeling. As far as preprocessing goes, I will apply the Porter stemmer to regularize the string representations. I will also apply stop word removal over my vocabulary to reduce noise from commonly occurring words skewing the distributions. Smoothing will be applied as well to the probability estimations via Laplace smoothing (or a more sophisticated smoothing technique).

**Datasets**

There are many datasets which capture some publication date data. While this was initially promising, there are some serious problems which need to be addressed. One of the main problems is that of coverage. Not all years are equally represented in the acquired datasets. For example, the Brown Corpus only covers newspaper articles from 1961.

Luckily there is some guarantee of full coverage given by the yearly presidential addresses, however there will be a natural imbalance with the naïve approach to dataset integration. I also have implemented selective scraping of the New York Times for abstracts and snippets of articles from 1851 to the present. I will discuss some ideas to circumvent the time coverage problem after presenting the corpora which I have already acquired and their relative strengths and weaknesses.

**Gutenberg Corpus:** The Gutenberg Project is an archive of textual works which have lost their copyright protections. The largely volunteer based project has over 54,000 English text documents stored in various formats. They provide a scraper interface to download parts of their archives using the following command:

**wget -w 2 -m -H "http://www.gutenberg.org/robot/harvest?filetypes[]=txt&langs[]=en"**

Using this interface, I acquired 2386 of their documents, which represents approximately 5% of their collection.

**Pros:** Each book will provide a large amount of data for my models. There is also a large volume of data, and if I need additional data I can always download more from their servers. I also suspect that there will be relatively good date coverage in the dataset relative to the other corpora.

**Cons:** The documents have some assorted meta-data which are largely irrelevant. I will need to parse and throw away this data before allowing my algorithms to consume it. Additionally, the meta-data does not include a publication date field. I will need to implement some regex to parse the year from the books. Finally, given that the source material are books, there may be the problem of “historical” pieces. What I mean by this is that it is possible that a classic work like *Antigone* was re-published in 1850. This would highly pollute the language model for the year in question.

**Brown Corpus:** The Brown Corpus, collected at Brown University, was the first million+ sized text corpus ever compiled. The text documents represent 500 different sources from 15 different categories (news, learned, hobbies, etc) all published in 1961.

**Pros:** From what I understand the entire corpus is already stored in NLTK servers, so I will not need to implement yet another data parser. No date processing is required since the text documents are all from 1961. Good coverage of topics will help avoid overfitting a year category to a specific topic (in the case of sparse data for the year).

**Cons:** The data has limited coverage, only spanning the year of 1961.

**Reuters-21578 Corpus:** Similar in scope to the Brown Corpus, the Reuters-21578 Corpus is a collection of text spanning 90 different categories over 22578 news articles published in 1987.

**Pros:** Regular markup language which includes publication date meta-data (although all articles come from the same year) leads to ease of parsing. High coverage of different topics.

**Cons:** Must parse the text files to remove meta-data. Each file consists of 1000 documents, so I will need to parse the file into documents as well. Only consists of data from 1987.

**Cather Archive:** Willa Cather was an author who published from 1892-1949. The University of Nebraska – Lincoln has archived her works (short stories, books, interviews, etc) and I plan to scrape this data for my models.

**Pros:** Good coverage of dates and a lot of text to learn from.

**Cons:** I must scrape the data by hand and remove HTML markup via BeautifulSoup.

**Inaugural Address Corpus:** This dataset contains every presidential inaugural address.

**Pros:** Large coverage of dates, easy to parse the year, no meta-data to clean.

**Cons:** Relatively short documents, limited scope of topics.

**State of the Union Corpus:** Transcription of each presidential state of the union address.

**Pros:** Full coverage of dates, no meta-data to clean. NLTK had the 1900’s to the early 2000’s archived in txt format.

**Cons:** I must scrape a website to get the remaining data. Limited scope of topics.

**New York Times:** The New York Times offers a web API to query the archives by fields such as publication year (1851-2017). While this is a generous feature, the main website is still blocked by a paywall, so I will be forced to only scrape the abstracts from the JSON data.

**Pros:** Large year and topic coverage. Exact year queries will allow me to fill in any gaps which are present in the data.

**Cons:** Limited number of queries allowed per day. Must a use custom API to acquire the data. Only can scrape the abstract of the article.

**\*\*\***

As shown above, I have gathered many streams of language data from a variety of corpora. Although some of the datasets have limited date coverage, I believe that I can somewhat alleviate this problem with selective querying from the New York Times. Ultimately, it may be the case that I cannot fully depend on the New York Times data due to the throttled querying rate. If necessary I may need to balance the corpora by purposely not using some of the data (although I will try to avoid this).

I also plan on incorporating Google Ngrams as a selective probability query engine. The main drawback of using Ngrams is that it is painfully slow to query via Python over all possible dates (not to mention all possible words). I will need to take care to only use Ngrams when there is need for disambiguating decisions.

**Evaluation**

My main evaluation metric for the next milestone will be dataset parsing completion. Since I have many different datasets, all which need to be parsed and processed in different ways, this will be a tedious task. After most of the data has been extracted, I will then start performing exploratory data analysis over quantities such as word counts, distribution of words with respect to publication year, and coverage of publication years in the dataset. Using this information, I will be able to selectively query the New York Times abstracts to even out the language model. Currently I am scraping the New York Times for data ranging from 1990 onwards as I believe there will not be as much recent data in the corpora I have collected.

If I can complete both the tasks of dataset extraction and implementation of New York Times API parser, I will then focus on implementing my learning models. To implement my models correctly I will first need to apply the preprocessing steps outlined above in the implementation section. Finally, I will be able to experiment with my model architecture and see where improvements can be made.

**Resources**

**Papers:**

* **[1] Temporal Language Models for the Disclosure of Historical Text** - <https://ris.utwente.nl/ws/portalfiles/portal/5421323>
* **[2] Using Temporal Language Models for Document Dating** - <https://pdfs.semanticscholar.org/f346/a93874a760deadaf5401ad6f5f6d5aff4a08.pdf>

**Code:**

* **NLTK** - <http://www.nltk.org>
* **Google Ngrams Scraper** - <https://github.com/econpy/google-ngrams>
* **New York Times API -** <https://developer.nytimes.com/article_search_v2.json#/README>

**Datasets:**

* **Gutenberg Corpus** - <https://www.gutenberg.org/>
* **Brown Corpus** - <https://www1.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/corpora/list/private/brown/brown.html>
* **Reuters-21578 Corpus** - <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>
* **Cather Archive** - <http://cather.unl.edu/>
* **Inaugural Address Corpus** - <https://archive.org/details/Inaugural-Address-Corpus-1789-2009>
* **State of the Union Corpus** - <http://stateoftheunion.onetwothree.net/texts/index.html>