CS221 Final Report: Extraction Based Text Summarization

Names: [Reginald Long, Michael Xie, Helen Jiang]
SUIDs: [reglong, sxie, helennn]

1 Motivation

Most information in the world is stored in text because of its permanence and ability to be shared. For example, according to Google, there are 130 million books in the entire world. Much of this information is inaccessible because there's simply too much of it. No human can possibly read all of the books or research papers in the world. This is why books have summaries, why research papers have abstracts, and why Wikipedia exists. However, in order to make a summary, a person had to manually compile information and write it, which is a time-intensive task. We believe that building a system that can automatically construct summaries for us would allow us to access information in a more digestible format and save countless human hours spent summarizing documents.

2 Problem Definition

Our problem is defined as follows: Given a document D with sentences $(x_0, ...x_n)$, return Y, where Y is a set containing the K most important sentences from D, where K is a given natural number.

3 Model

We model text summarization as a binary classification problem. Given a sentence, the program should determine whether the sentence is "important" or "unimportant" using document context and the features that we use. Thus, each training example consists of the local features of a sentence as well as features from the document context, such as position of the sentence in the document. We classify examples in batches, with each batch representing a document. Each classification batch also requires a desired sentence count K, which is the number of sentences desired in the returned summary. To extract a summary from a body of text, we apply our classification algorithm to each sentence and return the desired number of sentences deemed as "important" by similarity score to the catchphrases.

4 Train/Test Data

Our dataset is a set of 2021 legal cases found in UCI's ML Repository. These documents have been annotated with "catchphrases", which represent the sentences containing the most valuable information. These catchphrases were annotated by the Astralasian Legal Information Institute (operated by the University of Technology, Sydney and the University of New South Wales). To the best of our knowledge, there is only one annotation for each document. Each document has around 200 sentences; however, the number of sentences ranges from 50 to 2000+. Each document has generally around 5-10 catchphrases. This represents a case of imbalanced classes, which we attempt to account for in our algorithm.
(Dataset: https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports)

4.1 Example Titles

1. Beyazkilinc v Manager Baxter Immigration Reception amp; Processing Centre [2006] FCA 16 (18 January 2006)

2. Communications, Electrical, Electronic, Energy, Information, Postal, Plumbing amp; Allied Services Union of Australia v ACI Operations Pty Ltd [2006] FCA 7 (16 January 2006)

3. MZWQW v Minister for Immigration and Multicultural and Indigenous Affairs [2006] FCA 23 (31 January 2006)

4.2 Example Catchphrases

1. whether penalty should be paid to applicant

2. contravention of part xa of the workplace relations act 1996 (cth)

3. whether appellant had sufficient time to plead case

4.3 Example Sentences (These Correspond to Catchphrases Above)

1. On that day the Court ordered the reinstatement of Mr Colin Williams and deferred consideration of penalty and compensation pending the receipt of written submissions.

2. Pursuant to s 298U(a)(i) of the Workplace Relations Act 1996 (Cth) a penalty of $16,500 be imposed on the respondent for its contravention of s 298K(1)(a) of the Act for the prohibited reason referred to in s 298L(1)(a) of the Act.

3. 10 The appellant filed a notice of appeal dated 22 August 2005 contending that the adjournment application should not have been refused by Riethmuller FM because he was ill on the day of the hearing and he was not given sufficient time to plead his case in detail.

4.4 Labelling Important and Unimportant Sentences

Since catchphrases do not match up exactly with sentences (they cannot always be found word-for-word in the document), we loop through the entire document and calculate the similarity between a sentence in the document and the catchphrase. The similarity is calculated by the dot product of the word count vectors of the catchphrase and sentence. For each catchphrase, we take one sentence with the highest similarity score and label it as one of the ”important” sentences.

4.5 Data Validation

We use 10-fold validation, meaning that 90% of the documents randomly selected as the train set, and 10% are used as the test dataset.

5 Model

5.1 Preprocessing

5.1.1 Convert Document to Lower Case

We don't want to have two separate weightings for the capitalized version of a word and a lower case version of a word ("Worse" and "worse" are not semantically different).

5.1.2 Remove Stop Words

Before the processing of our text, we filter out a list of words such as "the", "a", and "an". We do this because these words are semantically meaningless in a summarization context, and could end up causing noise in the weights we use for our algorithm. The list of words can be found in stopwords.txt.

5.1.3 Lemmatize

Lemmatizing is a process that converts different forms of the same base word to the same word, taking context into account. For example, if we encountered the word "worse", a lemmatizer would be able to convert it to its base form "bad". Its important to lemmatize because many words in fact share the same semantic meaning, (Ex: "connect" and "connected") and we would like our system to be robust to changes in word forms.

5.2 Algorithm

Our algorithmic pipeline works as follows:

```
Load Corpus
Preprocessing
Remove Stop Words from Each Document
Lemmatize the words (Ex: plays to play...documentation to document)
Create train/test example (x, y), where x contains the sentences in the document and y contains the catchphrases and number of catchphrases K
Train
Learn binary classifier (SVM, Naive Bayes, SGDClassifier)
Test
for each document do
  Extract features
  Apply classifier to each sentence, getting a score S
  Return the K highest scoring sentences in the document.
end for
```

5.3 Hyperparameters

5.3.1 SVM

A linear kernel SVM with L2 norm loss and penalty was employed on the 309556 dimensional dataset. Additionally, the SVM used regularization parameter $C = 1$ and a class weighting of $\frac{1}{\alpha_i}$

where α_i is the total number of instances of class i in the training set. Thus, the high number of non-important sentences will cause the non-important class to be weighted much less than the important class. The SVM is trained by optimizing the dual formulation of the optimization objective.

5.3.2 Naive Bayes

Multinomial and Bernoulli formulations of Naive Bayes were employed on the dataset without Laplace Smoothing.

5.3.3 SGDClassifier

We use L2 regularization on a hinge loss classifier. We use 5 iterations and use 'optimal' learning rate, where the learning rate is decided by sci-kit.

5.4 Local Features

5.4.1 Word Counts

This feature is the count of unique words in every sentence. We assume that these topics are expressed in similar words, and thus sentences with high frequencies of words in catchphrases should be important. In text classification, a frequently used normalization technique used with word counts is TF-IDF (term frequency-inverse document frequency). The TF-IDF value increases proportionally to the number of times a word appears in the document, but is inversely offset by the frequency of the word in the corpus, which helps to control for words that are just more common in the language.

5.4.2 Part of Speech Tags

Semantic information would give insight into sentence type and structure. We hypothesize that certain sentence types and structures could be correlated with importance (Ex: Perhaps sentences with more verbs are more important).

5.4.3 Part of Speech Locations

We keep track of the relative location of each POS tag for a given sentence. The hypothesis is that certain sentence structures could be inherently more important than others.

5.5 Global Features

5.5.1 Sentence Location

Most writing in English tends to have important sentences in the beginning, to signal what the rest of the document will talk about, and in the end, to conclude or summarize what the document was about. We keep track of which decile a sentence falls in to take advantage of this structure and to accomodate for varying document sizes.

5.5.2 High Frequency Words

An on-topic sentence (which is likely to be important) usually contains words found frequently in the document as a whole. This differs from word counts, as word counts hypothesize universal importance/lack of importance.

5.5.3 Similarity to Title

The title of a document is a strong indicator of what will be important in the said document. We have a feature that tracks how many words in the title are found in the sentence. Generally, a higher number of words should correlate with a greater importance.

6 Example Run-Through (06_15.xml)

Because each example is generally 200-300 sentences long, we have an abbreviated example of a document below where there is only one catchphrase and three sentences.

6.1 Initial Data

Catchphrase: “whether get up and colour scheme of respondents’ shop is misleading or deceptively similar to applicants’ get up and colour scheme”
Important Sentence: Sentence 3 (K=1)
Title: “Intellectual Property Pty Ltd (ACN 089 962 041) v Mygroups Pty Ltd (ACN 116 583 748) [2006] FCA 15 (23 January 2006)”
Sentences:

1. “The first Clark Rubber store was opened in Melbourne in 1947 and at the present time there are approximately 70 stores operated by franchisees of Clark Rubber throughout Australia.”

2. “There is also displayed on the exterior of Clark Rubber stores the ”Clark Rubber” name and trademark.”

3. “The applicants claim that the manner in which the first respondent’s store has been painted has been to use a colour scheme which is misleading or deceptively similar to the Clark Rubber colour scheme.”

6.2 Pre-Preprocessing (Lemmatize, Lowercase, Remove Stop Words)

Catchphrase: whether get up colour scheme respondents’ shop misleading deceptively similar applicants’ get up colour scheme,
Title: intellectual property pty ltd (acn 089 962 041) v mygroups pty ltd (acn 116 583 748) [2006] fca 15 (23 january 2006)
Sentences:

1. first clark rubber store opened melbourne 1947 present time approximately 70 store operated franchisees clark rubber throughout australia

2. also displayed exterior clark rubber store “clark rubber” name trademark
3. applicant claim manner which first respondent’s store ha been painted ha been use colour scheme which misleading deceptively similar clark rubber colour scheme

6.3 Feature Extraction

We extract the features described above in our model. Denote the feature vectors as $\phi(x_1)$, $\phi(x_2)$, and $\phi(x_3)$ for the above sentences.

6.4 Compute Scores

Intuitively, we wish to find the sentence that has the highest score or probability, where a higher score/probability indicates greater importance.

6.4.1 SVM/SGD

We find $\arg\max_{x_i} w^T\phi(x_i) + b$, for i=1,2,3, where b is our bias term. We select the K=1 highest scoring sentence, in this case, the 3rd one, and select it as our one important sentence. We then return the sentence.

6.4.2 Naive Bayes

Let $Y \in \{0,1\}$ denote whether a sentence is important (1=important). Let $\phi(x_i)_j$ denote the element at the jth index of $\phi(x_i)$. We find $\arg\max_{x_i} P(Y = 1) * \prod_{j=1}^{n} P(\phi(x_i)_j|Y = 1)$. Finally, we select the K=1 highest scoring sentence and return it.

7 Compared Models

7.1 Baseline

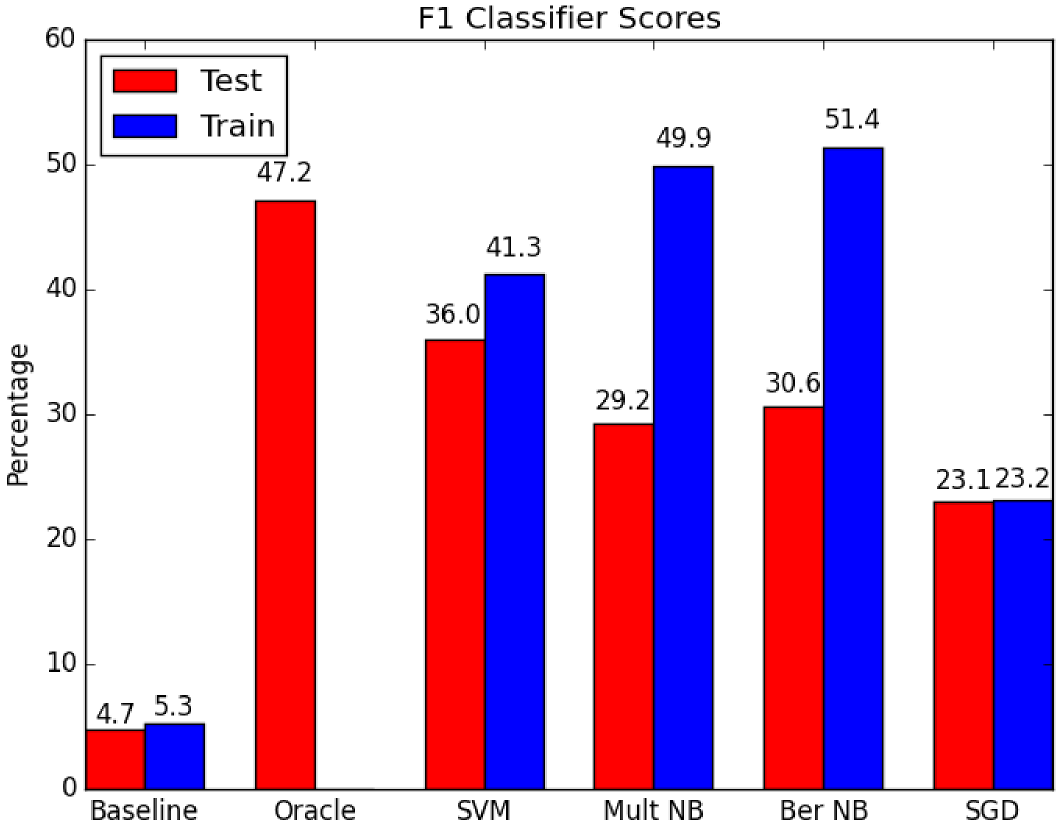
Our Baseline model is a hinge loss linear classifier that takes as input a sentence x and outputs y $\in \{0,1\}$, which correspond to unimportant and important sentence, respectively. It uses only word features. It is trained using SGD, with $\eta = 0.1$ and 3 iterations.

7.2 Oracle

Our Oracle is a set of workers on Amazon Mechanical Turk select the K most important sentences from a document (where the number K is given to the summarizers). Due to cost, we were only able to summarize 10 documents. Each document was summarized by 5 different people on AMT.

8 Results

Note: SGD is SGDClassifier described in Hyperparameters



9 Error Analysis

Naive Bayes models were able to attain high accuracy on training examples, but seemed to overfit and failed to generalize to the test set in both small and large corpus sizes. In contrast, the SVM has a larger training error than the Naive Bayes models, but has higher test accuracy than the Naive Bayes models - this suggests that the SVM generalizes better, but the classifier has higher bias. Both the SVM and Naive Bayes Models seem to fit the data and generalize better than the SGDClassifier. In all cases, the baseline hinge loss linear classifier performed worse than the more sophisticated models.

9.1 Explanation of Reduced Train Performance

The phenomenon seen that training accuracy decreased as the corpus grew is indicative of either high classifier bias or inherently large variance in the data. This implies that the data is not linearly separable in the 309556 dimensions that each feature vector is in. We could tackle this by increasing the dimensionality of the classifer, such as using a Gaussian kernel for the SVM. However, the lack of more powerful computing resources makes training a LIBSVM's Gaussian kernel SVM intractable, since LIBSVM is not as optimized for high dimensionality and a high number of training examples as the LIBLINEAR implementation of the linear kernel SVM.

10 Challenges

10.1 Imbalance of Important Vs. Unimportant Sentences

Since most sentences in a document are, by definition, unimportant, most train data is not important. This is more of a problem when we feed a classifier a sentence at a time, as opposed to examining the entire document and picking the K best sentences. We would be able to get a very good accuracy (True Positives + True Negatives) / (Positive + Negative) by simply saying that every sentence is unimportant. However, we still have this problem even when we have the algorithm to return the K-best sentences (although it's somewhat mitigated by forcing positive predictions). For example, if we were running Stochastic Gradient Descent to train a linear classifier, we would only make at most K updates (The ones we predict are important) in a document of approximately 200 sentences. This makes training less efficient than if the classes were balanced because we require more data.

10.2 Context Dependencies

Intuitively, the most important sentences of a document are strongly correlated with what the document is talking about. For example, a summarization about a criminal legal case would have sentences that are quite different from a document summarization on C syntax. Within each document, the importance of some words are augmented, whereas others are diminished. As a simple example, a sentence containing the word "guilty" in a criminal case would be more important than "guilty" in a guide on C syntax (Ex: "The jury found the defendant guilty." vs. "I'm guilty of abusing C syntax"). As such, a good automatic text summarizer must employ global features that allow us to make use of context.

10.3 Inherent Subjectivity

The summarizers disagreed on 0.6166 of sentences selected (Calculated by the expression: (Number of different sentences picked) / (Total Number of important sentences)). As shown above, humans disagree about what is important in a document. This makes it difficult to measure success in a universal sense. In other words, a successful summarizer is not transitive; doing well in one data set does not necessarily imply good performance in another data set. One possible approach is to include annotations from a wide of annotators, which would allow for broader coverage, but would also introduce more noise. In our approach, we assume the annotations are the "gold standard", which is less than ideal.

11 Future Work

We believe that focusing on how to better model context will lead to better summarization results. Extracting the most important sentences is a useful first step, but often leads to choppy, abrupt sentences. Many of the extracted sentences may be important, but they may also be redundant. One first step is to extract the most important sentences and then remove irrelevant parts of the sentence. From there we can run some similarity algorithm to maximize the dissimilarity between the important sentences, which, we hypothesize, would decrease the redundancy between important sentences.Ultimately, it seems as that the future of automatic text summarization lies in abstraction-based methods as opposed to extraction-based, where the machine builds up a high level summary of a document through synthesizing information, much like an abstract of a research paper.

References

[1] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874.

[2] G. Carenini, J.-C.-K. Cheung. Extractive vs. NLG-based Abstractive Summarization of Evaluative Text: The Effect of Corpus Controversiality

[3] J. Goldstein, V. Mittal, J. Carbonell, M. Kantrowitzt. Multi-Document Summarization By Sentence Extraction. NAACL-ANLP-AutoSum '00 Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization - Volume 4. 2000.