Book Name Extraction from Text Files

Sowrabha Horatti Gopal sgopal3@wisc.edu
Pradeep Kashyap Ramaswamy pradeep.kashyap@wisc.edu
Sharath Prabhudeva Hiremath shiremath@wisc.edu

1 Enity Type

Book Names

The source for our text files is **Goodreads.com**. We scraped reviews of books from **Goodreads.com** and are trying to extract Book Names from them. Though this might seem straightforward, book names involve a lot more complexities than extracting any other names like a person's name or a place's name. For example, "A Game of Thrones" doesn't contain any proper nouns. We can see it's not that simple.

A few examples of occurrences of Book names in our text files (excerpts from review comments)-

- In ***Harry Potter and the Deathly Hallows*** , the seventh and final installment of the ridiculously popular Harry Potter series, J.K. Rowling going to keep going until I succeed or die.
- ***The Girl Who Kicked the Hornet's Nest*** picks up exactly where The Girl Who Played With Fire left us, with Lisbeth Salander riddled with bullets, Zalachenko the ex-Russian spy with half his face hacked off, the blonde giant Niedermann on the run, and Mikael Blomkvist, journalist and co-owner of Millennium magazine, trying to explain to the police the true story of Lisbeth's past.
- The author of ***The Boy in the Striped Pajamas***, on the other hand, clearly thinks that children are idiots.
- There is not a single lawyer in ***A Painted House***; the best we get is a single policeman, because this story is set in rural Arkansas in 1952.

2 Size of Datasets

We have a total data set of **1400** documents. We have split our dataset into two parts as per the project requirements, namely -

1. **Training Dataset (I)** - This is the dataset on which we perform all our testing to determine the best classifier. This is used as a development dataset to design features and form post processing rules. This set contains **900** documents. In total, we are training our classifiers with **1800** positive mentions and **4500** negative mentions. This set is further divided into set **P** with **600** documents for training and set **Q** with **300** documents for testing and debugging of false positives and false negative.
2. **Testing Dataset (J)** - This is the dataset on which we run the classifier and post processing rules and use to determine the accuracy of our extractor. This set contains **500** documents. In total, there are around **1000** positive mentions and **2500** negative mentions.

3 Features

Selection of features of book name is complicated than it looks. Many of the book names are similar to person names or characters. For example, the book **John Adams** is exactly a person name as well. Another example would be **Harry Potter and the Deathly Hallows** in which the name of the character **Harry Potter** is part of the book name. Thus after much deliberation over the features, we have come up with the below features set. These are the final set of features that we have decided upon after all the refinement and debugging. Some of these features look for the context in which the mentions are present and others look into the specifics within the mentions (We'll be using **phrase** and **mention** interchangeably).

1. **First word is 'A' or 'The'** : *Boolean*: Whether the first word in phrase extracted is 'The' or 'A' and it is followed by word with first letter capitalized. This is a common pattern in book names. For example - *The Book Thief*, *A Thousand Splendid Suns*

2. **Capitalized First Letter** : *Boolean* : First letter in all the words in the extracted phrase, which are not stop words, should be capitalized. Book names have this unique feature that the first letter in all of their words will be capitalized or or else they will be *stop words*. This features enables detecting book names better. This is not true for all books; For example, the name **Shantaram** is a person's name as well as a book's name. This feature returns $True$ for such names and is a good negative example as well. Thus, because of such names, with one or two words with all capitalized, some false positives exist.

3. **Period in the Phrase** : *Boolean* : If there exists a period within the selected phrase, then it might correspond to the initials of a person name; For example, J.K. Rowling, George R. R. Martin, E.L. James etc. Hence it is a potential negative instance.

4. **Colon in the Phrase** : *Boolean*: If phrase has colon then it mostly corresponds to a book name. For example, "The Dark Side**:** The Inside Story of How the War on Terror Turned Into a War on American Ideals", "Hegemony or Survival**:** America's Quest for Global Dominance".

5. **Phrases between Braces** [1] : *Boolean*: Whether phrase exists within "(" and ")" or not. If the phrase is inside brackets, then it is mostly negative; In most of the cases, text within brackets tries to provide some additional information regarding character or book
   [1] *This is the coolest sounding feature name. Don't you think?*

6. **Phrases between Periods** : *Boolean*: Whether the phrase exists between fullstops or not. If the phrase is between fullstops, then it is mostly a long sentence. Book names alone cannot appear as a sentence; They have to be mentioned within some context.

7. **Phrases between Commas** : *Boolean*: Whether the phrase exists between commas (, phrase,) or not. If the phrase is between commas, then it is mostly negative. These kind of phrases generally try to give a description of something being explained. They are not usually book names, instead they explain books.

8. **Word Length Constraint** :*Boolean*: Whether word length is word length$\geq$ 2. Differentiating book names from author names and character names can be difficult. Large number of books have $\geq$ 2 words. For example, The Surgeon, America America, The Twelve, Skinny Dip etc. Whereas, author or character names will usually be of one or two word. E.g, Tina, Henry, Harry, Benjamin etc.

9. **Word Length** *Integer*: Word length of the phrase; It is a good indicator of book names. Generally book names range from 1 word to 7 words.

10. **Word Length** :*Boolean*: Whether word length is word length$\geq$ 2. Differentiating book names from author names and character names can be difficult. Large number of books have $\geq$ 2 words. For example, The Surgeon, America America, The Twelve, Skinny Dip etc. Whereas, author or character names will usually be of one or two word. E.g, Tina, Henry, Harry, Benjamin etc.

4 Classifier Selection

To select the best classifier for our entity extraction, first we performed cross validation the dataset I using the following classifiers -

- Decision Tree
- Random Forest
- k-Nearest Neighbours

- Support Vector Machine
- Logistic Regression
- Linear Regression

The results obtained from this first trivial run are shown in the Table 1 in Section Metrics. As the results show, ***Decision Tree*** gave us the best precision and best recall (Random forest came very close), so we selected it to further tune and improve the performance.
After debugging and refining our feature set based on the classifier that we selected, we performed cross validation again using all the classifiers listed. The results obtained from this refined execution are shown in the Table 2 in Section 6.

As evident from the results, ***Random Forest*** is the classifier with the highest precision, a good recall and the best F1 score. So we settled upon this classifier for our entity extraction.

5 Post Processing

Once we settled on the classifier ***Random Forest***, we tried to achieve further improvement in performance by using post processing rules. We looked for patterns in the results that were being wrongly classified.

- We started looking at false positive examples. We observed that some of the author names were being considered as book names falsely. To filter these results out we developed a rule, which removed all the results which had only **proper nouns** in them.
- We also observed that some incomprehensible phrases were being considered for classification. There were phrases with **unicode characters**, so we added a rule to filter them out as well.

6 Metrics

We performed cross validation on the set I to select a good classifier for development. We obtained the following results -

Classifier	**Precision**	**Recall**	**F1**
Logistic Regression	0.8595	0.4775	0.6129
Random Forest	0.7465	0.6825	0.7152
Support Vector Machine	0.7575	0.6717	0.7086
k-Nearest Neighbours	0.7461	0.6533	0.6881
Linear Regression	0.8722	0.3935	0.5423
Decision Tree	0.7580	0.6842	0.7164

Table 1: Scores of different Classifiers on Training Dataset I **before** debugging

Though Logistic Regression and Linear Regression have high precision values, they have very **poor recall** values. SVM and Decision Tree gave good results for both precision and recall but, Decision Tree had better F1 score. Thus, we picked **Decision Tree** classifier for debugging and development. Then we split our training dataset I further into two datasets **P** and **Q** to debug and improve the performance of the classifier. Our training dataset I had 900 records, so we split it into two parts such that the dataset P gets 600 records, which will be further used to train the classifier and the dataset Q gets 300 records, which will be used to test and debug.

After developing a refined final feature set, we executed our tests on all the classifiers again. To our surprise, **Random Forest** started performing better (*maybe it was always striving to do well in life*). As seen in the table 2, Random Forest has the best precision, a good recall and—hands down—the best F1 score. So we picked Random Forest as our **final classifier**.

Classifier	Precision	Recall	F1
Logistic Regression	0.8312	0.7892	0.8094
Random Forest	0.9101	0.7433	0.8181
Support Vector Machine	0.8912	0.7233	0.7983
k-Nearest Neighbours	0.8255	0.7225	0.7678
Linear Regression	0.8407	0.6216	0.7147
Decision Tree	0.8991	0.7483	0.8166

Table 2: Scores of different Classifiers on Training Dataset I **after** Debugging

Then looking at the pattern of the false positive/negatives we developed post processing rules as mentioned in Section 5. After applying our post processing rules, we obtained the following results -

Classifier	Random Forest
Precision	**0.9644**
Recall	**0.7734**
F1 Score	**0.8584**

Table 3: **Final Scores for Extraction on Testing Dataset J**

7 Conclusion

After the grueling process of tagging the documents, developing the features, picking a classifier, developing post processing rules and developing the super powers of patience and spidey sense, we learnt the gory details of the process of Information Extraction from texts. We learnt the importance of cross validation and the behaviour of different classifiers. It was a challenging and subtle task to develop the feature set and post processing rules. Finally we were able to develop an amazing extractor with a precision of **96.44%** and a recall of **77.34%**. ***Now Witness the Firepower of this fully Armed and Operational Extractor!***