Extracting Book Names from Unstructured Data

Jai K. Smith
Dartmouth College / Hanover, NH
jai.k.smith.22@dartmouth.edu

Abstract

This project examines optimal strategies by which to extract structured book rosters from unstructured course data. It will detail how to use natural language processing to create a classifier that distinguishes between segments of text containing a book name, and segments of text *not* containing a book name, with very high accuracy. This approach can be used to improve data parsing techniques pertaining to unstructured course data, offering better resources to developers targeting college students and college workflows. The resulting classifier and data-set have been fully open-sourced on my GitHub for further use.

1 Introduction

Colleges maintain an online course catalog with class descriptions, prerequisites, and required readings. Although very useful, this data is rarely offered in a structured format, and is instead scattered through course descriptions and miscellaneous fields. Required readings are a particularly useful portion of this data to parse into a structured collection, as it offers indicators of course content, and a popularity metric for books within a given institution.

Indicators of course content, in particular, are helpful for a variety of reasons. Current course search within Dartmouth is performed via direct keyword match with the course title/description, or by course code. This means that to find a course via the search tool in tools like the timetable or third party review sites, you must already know the title of the course or a specific keyword that appears in its description. This only covers a small portion of the use cases for a course search tool. Far more useful is the ability to search a course registry by content. To find classes that discuss 'data mining', for instance, students should be able to search 'data mining' and see a list of all courses that discuss data mining. Access to relevant readings to courses (made available via this project) will allow us to gather information on the content of courses beyond their description, allowing us to include classes that read about 'data mining' in such a search query.

Existing work in this field includes a semi-supervised approach to attribute extraction from book names (Putra et al., 2020), which details strategies by which specific attributes can be detected and extracted from complete names. This approach offers mediocre accuracy overall, and fails to distinguish non-book entities from book entities, so it was not applicable to the problem we are attempting to solve. Another approach involves parsing unstructured course syllabi into structured course data (Yu et al., 2007). This technique was more applicable to our problem statement, but is inaccurate and too broadly scoped for our purposes. We only want to extract complete book names, not tag every segment of text in a syllabus.

2 Methodology

Dartmouth (and most other colleges) makes available to students a collection of data on all offered courses. The official registrar timetable offers a `textbooks` field for each course, which includes unstructured text supposedly specifying all 'book requirements'. We will use this as our source data to classify.

Ordinarily, loading this data into our project would require extensive web scraping from the timetable website. Fortunately, I've done this web scraping work in the past for Dartmouth's student-run course review site, and we can directly tap into this repository for up-to-date timetable and catalog data:

`https://github.com/D-Planner/data/tree/master/current`

Very basic normalization and tokenization techniques primarily involving regular expressions allow us to transform the unstructured text blobs in each `textbook` field into an array of text segments by splitting at line breaks and removing empty whitespace. This yields segments that are primarily either book requirements or *not* book requirements, the rest are slices of book requirements (this occurs when a name is split over several lines). Using a simple rule-based approach we can combine most of these slices into full book names, by treating every segment starting with the phrase 'by' or 'isbn' as an addendum to the previous segment.

The collection of text segments we have remaining are either book names or not book names, leaving us with the task of filtering out all segments that *are* book names. To do this, we'll build a text classifier. Starting a model from scratch at this point does not make sense, as there are a fair amount of linguistic cues that tell us whether these segments are book names or not. Instead, we want to start with a basic understanding of English, upon which we can build our classifier. We'll use a small `BERT` model as a starting point, but it is by no means the only eligible contender.

We chose not to use traditional named-entity recognition (NER) libraries because, based on our research and experimentation, they tend to detect entities *too narrowly*. This results in classifying subsets of book names as book names, but not the full segment. We did find some research online regarding the combination of NER with NP chunking (Osenova and Kolkovska), but this seemed overly complex given the availability of models like BERT that operate directly on full segments.

To train our `BERT` model, we require a substantial amount of tagged data. Ideally, this data would be perfectly reflective of the actual segments within the timetable, and we would manually tag a training set. However, since we probably want to train with more segments than *exist* in the timetable, and manually tagging every single one for training would defeat the purpose of our project, we'll compromise and *generate* our training data instead. This allows us to train on a far larger data-set than would otherwise be possible, and saves us substantial time tagging.

In order to generate data reflective of our timetable samples, we must generate book names that mimick the structure of our timetable data, and negative samples that include text that is similar to the 'filler' we find in the timetable. To positive samples, we used the following data-set of book attributes:

`https://www.kaggle.com/jealousleopard/goodreadsbooks`

Using a random sampling of fields from each book in the data-set (weighted by the approximate frequency with which said fields appeared in our textbook data, based on a quick manual scan) we constructed over ten thousand text samples similar in structure to our timetable data.

To generate negative samples, we selected random sentences from course descriptions in the Dartmouth course catalog. This content is written by the same professors that fill in the textbook field, and refer to the same courses, so they *should* be similar to the filler text found within the timetable.

This technique was not specified in any of the papers I read, but it made sense to me given the somewhat regular structure of samples I leafed through in my data-set. Template-filling is no newcomer to natural language processing, this is a mere extension of that. After several iterations tweaking the data we generated, we added a collection of roughly seven hundred manually tagged samples from the timetable to improve model accuracy. The same effect could have been achieved by tweaking the generator to be more reflective of our actual data, but this allowed us to account for some of our trickiest edge cases without spending too long tweaking parameters.

Once we collected an appropriate set of data (approximately twenty-three thousand total samples), we trained a BERT model using tensorflow in Google Colab. Nothing in this project requires tensorflow or Google Colab, but they simplified the development process significantly. You can find a copy of our work at the following link, which can be easily loaded in Google Colab and experimented with:

`https://github.com/jaismith/book-name-extraction`

3 Results

Using the aforementioned methodology, we were able to train a classifier that distinguishes between book name and non-book name text segments with very high accuracy. After training for five epochs, we were able to achieve a 99.4593% accuracy and 3.0168% loss on our *testing set*.

Metric	Value
Precision	92.9%
Accuracy	93.1%
F-1 Score	93.0%

Table 1: Model performance on real timetable data.

These results were a good indicator that *something* was working, but not that our classifier was necessarily accurate on real data from the timetable. If our generated data wasn't reflective of the timetable data, it would not be reflected here, as we're still testing primarily on generated data.

As an initial test of performance on real data, we copied over a few samples that looked tricky to *us* and ran the model on them. This showed very good performance on samples that included author names, dates, and ISBN numbers, and much lower performance on standalone book titles (which oftentimes don't include proper nouns, and rarely numbers). This made sense, as they were the most ambiguous samples to us as well, not immediately clear which class they belonged to.

After tweaking the model slightly to perform better on book titles (by changing the field weights in our initial random sampling), we continued on to test with a set of real data. To obtain this data we manually tagged another 700 samples (using a basic tagging utility we wrote) and ran the model against them, comparing our classification to the model's classification.

Our model consistently classified segments with titles and author names correctly, and yielded a very low false positive rate. However, we found that it consistently classified links (URLs) as books, when they should not be, and missed some books that lacked punctuation and multiple authors. We believe these issues can be resolved with more parameter tuning, and the data quality could be improved substantially by improving our rule-based merge and slicing tools (they are too simple for the data we are ingesting).

4 Discussion

We are very happy with the results of our project, and they significantly outperformed our initial expectations. Our methodology uses several overly simplistic processes for data cleaning and preparation which we believe leave substantial room for improvement.

Although we developed this in the English language, using English course data, the same approach can be applied in most other languages (because most data is generated, access to large quantities of training data is not required). Unfortunately, our methodology *does* rely on access to substantial pre-trained learning models, which limits the languages in which this can be applied. This constitutes a very mild ethical issue, but is nothing new to this project.

5 Conclusion

By generating samples from an existing data-set of book attributes and snippets of course descriptions, we were able to train a `BERT` classifier with an F-1 score of 93% in classifying text segments as book names or not.

These preliminary results are good enough to start building a data-set of book requirements for all courses at Dartmouth, and we have no reason to believe this approach isn't applicable at other institutions. Once we've collected book requirements in plain text, we will run them through the Google or Amazon book APIs to normalize them and retrieve pre-existing content metrics (and in some cases full texts). Using those content metrics and full-text blobs we will build a search index for fulfilling content-based course queries.

We strongly believe that such a utility will have an overwhelmingly positive impact on course discoverability and the process of course election. We will be rolling out an early version of this search tool during the coming months in Dartmouth's LayupList site, and the classifier is already available for public use in other projects.

References

Petya Osenova and Sia Kolkovska. Combining the named-entity recognition task and np chunking strategy for robust pre-processing. *BulTreeBank Project*.

Hadi Syah Putra, Faisal Satrio Priatmadji, and Rahmad Mahendra. 2020. Semi-supervised named-entity recognition for product attribute extraction in book domain. In *Digital Libraries at Times of Massive Societal Transition*, pages 43–51, Cham. Springer International Publishing.

Xiaoyan Yu, Manas Tungare, Weiguo Fan, Manuel Pérez-Quiñones, Edward A. Fox, William Cameron, and Lillian Cassel. 2007. Using automatic metadata extraction to build a structured syllabus repository. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pages 337–346, Berlin, Heidelberg. Springer Berlin Heidelberg.