NovelPerspective

Anonymous ACL submission

Abstract

We present a proof of concept for a tool to allow consumers to subset ebooks, based on the main character of the section. Many novels have multiple main characters, and vary with each chapter (or sub-chapter) which character the story is focused on. A well known example is George R. R. Martin's "Game of Thrones" novel, and others from that series. The NovelPerspective tool detects which character the section is about, and allows the user to generate a new ebook with only those section. The detection of main character can be done by many means. We present two simple baselines, and several machine learning based methods.

1 Introduction

Many books have multiple main characters, often each character is written from the perspective of a different main character. Different sections are written from the perspective of different characters. Generally, these books are written in limited third-person point of view (POV); that is to say the reader has little or or more knowledge of the situation described than the main character does. Having a large cast of character, in particular POV characters, is a hallmark of the epic fantasy genre.

We propose a method here to detect the main/POV character for each section of the book. Detecting the main character is not a difficult task, as the strong baseline result shows. However to our knowledge there does not exist any current software to do this. We attribute this lack to it being impractical to physically implement until recent times. The surge in popularity of ebooks has opened a new niche for consumer discourse processing. Tools such as the one present here, give the reader new freedoms in controlling how they consume their media.

We focus here on novels written in the limited third-person point of view. In these stories, the main character is the point of view (POV) characters. Some examples include: Across its 15 books, Robert Jordan's "Wheel of Time" series which has 146 POV characters[1]). Only about one fifth of the total word count was from the POV of the "main character". George R.R. Martin's "A Song of Ice and Fire", have over 30 POV characters in the books published so far [2]. Other well-known books meeting this requirement include: Robert Jordan's "Wheel of Time" series, all the novels from Brandon Sanderson's "Cosmere" universe, Brent Week's "Nightangel" and "Lightbringer" series, Steven Erikson's "Malazan Book of the Fallen" series, and thousands of others. This is also of interest for works written in omniscient third person point of view, such as J. R. R. Tolkien's "Lord of the Rings", which also may feature a focus on different main characters however the correct split is much less clear in these cases.

The utility of dividing a book in this way varies with the book in question. Some books will cease to make sense when the core storyline crosses over different character. Other novels, particularly the large epic fantasy stories we are primarily considering, have many parallel story lines focused on the different characters that only rarely intersect. While we are unable to find formal study on this, many readers speak of "skipping the chapters about the boring characters", or "Only reading the real main character's sections". Particularly on a re-read, or after already having consumed the media in some other form such as watching a movie adaptation, or reading a summary. We note that sub-setting the novel once does not prevent the reader going back and reading the intervening chapters if it ceases to make sense, or from sub-setting again to get the chapters for another character who's path the

[1] http://wot.wikia.com/w3iki/Statistical_analysis

[2] http://awoiaf.westeros.org/index.php/POV_character

one they are reading intersects. We (the first author) can personally attest for some books reading the chapters one character at a time is indeed possible, and indeed pleasant: having read George R.R. Martin's "A Song of Ice and Fire" series in exactly this fashion.

2 Character Detection Systems

2.1 Baseline systems: First and Most Common

The obvious way to determine the main character of the section is to select the first named entity. This works well for many examples: "It was a dark and stormy night. Bill heard a knock at the door."; however it fails for many others "'Is that Tom knocking on my door' thought Bill, one storm night.". Sometimes a section may go several paragraphs describing events before it even mentions the character who is perceiving them. This is a varying element of style.

A more robust method is to use the most commonly named entity. This works well, as once can assume the most commonly named entity is the main character. However, it is fooled, for example, by book chapters that are about the main character's relationship with a secondary character. In such cases the secondary character may be mentioned more often.

A better system would combine both the information about when a named entity appeared, with a how often it occurs, and other information about how that named entity token is being used. It is not obvious as to how these should be combined to determine which named entity section is about. We thus attempt to solve it using machine learning, to combine these features to make a classifier.

2.2 Machine learning systems

One can see the determination of the main character as a multi-class classification problem. From the set of all named entities in the section, classify that section as to which one is the main character. Unlike typical multi-class classification problems the set of possible classes varies per section being classified. Further, even the total set of possible named characters, and thus classes, varies from book to book.

An information extraction approach is required which can handle these varying classes. As such, any machine learning model used can not incorporate knowledge of the classes themselves into it's learned system.

We reconsider the problem as a series of binary predictions. For each possible character (i.e. each named-entity that occurs), a feature vector is extracted. This feature vector is the input to a binary classifier, which determines the probability that it represents the main character. We consider than binary probability as the score for the corresponding character. We chose the highest scoring character.

It should be noted that the base-line systems, while not using machine learning for the final character classification, they do make extensive use of machine learning based systems during the preprocessing stages (in the same way the machine learning systems to also for preprocessing). The POS-tagger, and the Named Entity recogniser are based on machine learning.

2.3 Classifier

XGBoost tree ensemble's are used for the machine learning methods (Chen and Guestrin, 2016). We use the default hyper-parameters: 100 trees with a max depth of 3, using the logistic loss function.

During training, from each chapter in the training dataset, we generated a training example for every named entity that occurred. All bar one of these was a negative example.

2.4 Feature Extraction

For the models we investigated several feature sets. XGBoost is based on decision trees with limited depth. It thus performs implicit pruning, so we are not concerned with redundant or useless features. As such we use large numbers of features, many of which are not actually used in the trained model, as discussed in Section 4.2.

We define the "Classical Features-Set" using features that are well established in NLP related tasks. We start with the features from the Baseline systems. The position in the text that the named Entity first occurs, and for symmetry also the last position. The the number of occurrences, as well as the rank of that score compared to the other named entity in this text. This occurrence rank is the only feature which gives direct reference to the other possibly labels. It would be possible to create more rank based-features for the other features. This give 100 base features. To allow for text length invariance we also provide each as a percentage its maximum possible value, For a total of 200 features.

We define a "Word Embedding Feature-Set" using FastText word vectors

3 Experimental Setup

3.1 Datasets

We make uses of two series of books selected from our own personal collections. The first four books of George R. R. Martin's "A Song of Ice and Fire" series

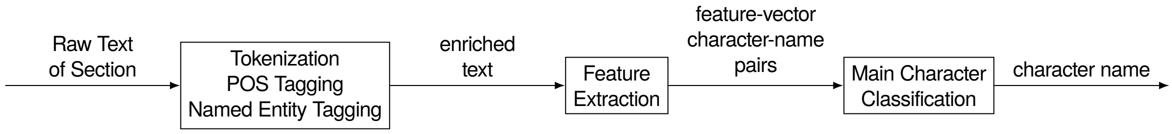

Figure 1:

(hereafter referred to as ASOIAF); and the two books of Leigh Bardugo's "Six of Crows" duology (hereafter referred to as SOC). ASOIAF has a total of 256 chapters and 15 point of view characters. SOC has a a total of 92 chapters and 7 point of view characters.

The requirements of the books to use in the training and evaluation of the NovelPerspective system is that they provide ground truth for the section's main characters. These books do so in the chapter names – each matching to a character name.

We do not have any datasets with labelled sub-chapter sections, though the tool does support such works.

3.2 Evaluation Details

In the evaluation, the books chapters are pre-separated into body text and chapter name (character name). The detection systems are given the body text and asked to predict the character names. To mimic the human users ability to select multiple aliases of a character, before final classification the scores of character's nicknames are merged. For example merging `Ned` into `Eddard`.

3.3 Evaluation Metrics

We report overall accuracy, and weighted-average precision, recall and F1-score. The weighted-average scores are calculated by calculating the macro scores for each true class (as in macro averaging (Sokolova and Lapalme, 2009)), then weighting by the size of the class. Note that this means the overall accuracy is the same as the recall..

In the case of the cross-validation results, we report the mean results over 10 random folds, using the same 10 folds for each model being evaluated.

3.4 Implementation

The full implementation is available at `https://github.com/oxinabox/NovelPerspective/`

The text is preprocessed using NLTK (Bird et al., 2009) to added features. The text is first tokenised, part of speech (POS) tagged , and finally named entity chunked (binary), using NLTK's default methods. That is the Punkt sentence tokenizer (?), regex based improved TreeBank word tokenizer, Greedy Averaged Perceptron POS tagger, and the Max Entropy Named Entity Chunker. The use of a binary, rather than a multi-class named entity chunker is significant. Because fantasy novels often use "exotic" names for characters, we found that it often fooled the multi-class named entity recogniser, into thinking characters were organisations or places rather than people. Note however this is particularly disadvantageous to the First Mentioned Named Entity Baseline, as any kind of named entity will steal the place. Never-the-less, it is required to ensure that all character names are a possibility to be selected.

Scikit-Learn is used to calculate the evaluation metrics and to orchestrate the cross-validation tests (Pedregosa et al., 2011).

4 Results and Discussion

4.1 Main results

4.2 Feature Weights

From a model trained on the full combined dataset, we can extract the feature weights.

5 Demonstration

An online demonstration is available at `http://white.ucc.asn.au/tools/np`. This is a web-app, made using the CherryPy framework[3].

The users uploads an ebook, and selects one of the character detection systems we have discussed above. The users is then presented with a page displaying a list of sections, with the predicted main character in the left, and an excerpt from the beginning of the section on the right. To avoid the user having to wait while the whole book is processed this list is dynamically loaded as it is computed. We find that the majority of the time is spend on running the preprocessing to annotate the data before the classification.

The user can select sections to keep. This is done via checkboxes to the right of each author The user can input a regular expression for a character name to have the corresponding check-boxes marked. This uses the character name as predicted by the model. As none of the models is perfect, some mistakes are likely to be

[3] `http://cherrypy.org/`

Testset	Method	P	R	F1	Acc
ASIAF	First Mentioned	0.251	0.242	0.247	0.242
ASIAF	ML Classical Features trained on SOC	0.914	0.914	0.914	0.914
ASIAF	ML Hybrid Features trained on SOC	0.914	0.914	0.914	0.914
ASIAF	ML Word Emb. Features trained on SOC	0.895	0.895	0.895	0.895
ASIAF	Most Commonly Mentioned	0.926	0.926	0.926	0.926
SOC	First Mentioned	0.341	0.326	0.333	0.326
SOC	ML Classical Features trained on ASIAF	0.913	0.913	0.913	0.913
SOC	ML Hybrid Features trained on ASIAF	0.913	0.913	0.913	0.913
SOC	ML Word Emb. Features trained on ASIAF	0.696	0.696	0.696	0.696
SOC	Most Commonly Mentioned	0.848	0.848	0.848	0.848

Table 1:

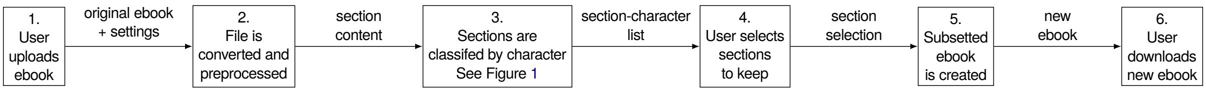

Figure 2: The full process of the using NovelPerspective. Note that step 5 uses the original ebook to subset.

mode. The user can then manually correct the selection using the check-boxes before downloading the book.

6 Conclusion

References

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python*. " O'Reilly Media, Inc.".

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427 – 437.