NovelPerspective

Anonymous ACL submission

Abstract

We present a proof of concept tool to allow consumers to subset ebooks, based on the main character of the section. Many novels have multiple main characters each with their own storyline running in parallel. 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 sections. This gives consumers new options in how they consume their media, allowing them to pursue the storylines sequentially, or skip chapters about characters they find boring. We present two simple baselines, and several machine learning based methods for the detection of the main character.

1 Introduction

Often each section of a novel is written from the perspective of a different main character. The characters each take turns in the spot-light, with their own parallel story-lines being unfolded by the author. As readers we've often desired to read just one storyline at a time, particularly when reading the book a second-time. We present a tool, NovelPerspective, to give the consumer this choice.

Our tool allows the consumer to select which characters of the book they are interested in, and to generate a new ebook file containing just the sections from that character's point of view. The critical part of this system is the detection of the point of view character. This is not an insurmountable task, building upon the well established field of named entity recognition. However to our knowledge there does not currently exist any 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 narrative processing. Tools such as the one present here, give the reader new freedoms in controlling how they consume their media.

Having a large cast of character, in particular POV characters, is a hallmark of the epic fantasy genre. Well known examples include: George R.R. Martin's "A Song of Ice and Fire", Robert Jordan's "Wheel of Time", Brandon Sanderson's "Cosmere" universe, Steven Erikson's "Malazan Book of the Fallen", amongst thousands of others. 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.

We focus here on novels written in the limited third-person point of view. In these stories, the main character is, for our purposes, the point of view (POV) character. Limited third-person POV is written in the third-person, that is to say the character is referred to by name, but with the observations limited to being from the perspective of that character. This is in-contrast to the omniscient third-person POV, where events are described by an external narrator. Limited third-person POV is extremely popular in modern fiction. It preserves the advantages of first person, in allowing the reader to observe inside the head of the character, while also allowing the flexibility to switch to narrate from another character (**?**). This allows for multiple concurrent storylines around different characters. Our tool helps users un-entwine such storylines, giving the option to process them sequentially.

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, anecdotally many readers speak of:

- "Skipping the chapters about the boring characters"
- "Only reading the *real* main character's sections"
- "Reading ahead past the side-stories to get on with the main plot"

Particularly if the have read the story before, and thus do not risk confusion. Such opinions are, of-course, but a matter of the consumer's taste. The NovelPerspective tool gives the reader the option to customise the book in this way, according to their personal preference.

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

The primary difficulty in segmenting ebooks this way is in classifying the sections as to which character they are from the perspective of. That is to say detecting who is the point of view character. Very few books indicate this clearly, and the reader is expected to infer it during reading. This is easy for most humans, but automating it is a challenge. To solve this, the core of our tool is its character classification systems. We investigated several options which the main text of this paper will discuss.

2 Character Classification Systems

The common structure of all our character classification systems is shown in Figure 1. First the raw text is enriched with part of speech and named entity targets, from which features are extracted for each named entity. Theses are used to score the named entities for the most-likely to be the POV character, and the highest scoring is returned by the system. The different systems presented modify the the Feature extraction and Character scoring steps.

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. In this system feature extraction and scoring is just to give a score of one to the first named entity found, and zero to the others. 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.

Here the feature extraction and scoring step scores each named entity proportionate to how often it occurred. 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

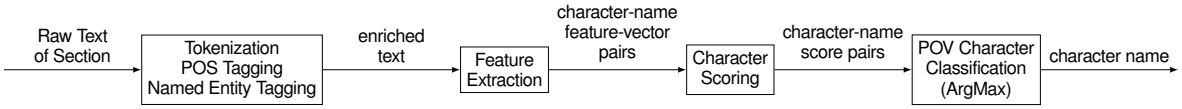

Figure 1: The general structure of the character classification systems. This in turn is the classification step of part of the large stem in Figure 2.

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.

2.2.1 Classifier

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

During training, from each text in the training dataset, we generated a training example for every named entity that occurred. All bar one of these was a negative example. We then trained it as per normal for a binary classifier, using the logistic loss function. This predicted probability of that feature vector being for POV character was used as the score for that character.

This may seem similar to an one-vs-rest approach to using binary classifiers for multiclass classification. However, there is an important difference. Our system only uses a single binary classifier; not one classifier per class, as the classes in our case vary with every item to be classified. The problem is truly one of information extraction, and the classifier is a tool for the scoring.

2.2.2 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 4.3.

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. The other features are based on frequency of each POS tag of the immediately adjacent words. We theorised that this POS information would be informative, as the it seemed reasonable that the POV character would be described as doing more things, so co-occurring with verbs seemed useful. 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 (**?**). We concatenate the word embedding for the a 5 word window to either size of the named entity; and take the element-wise mean of this concentrated vector over all occurrences of the named entity. Such averages of word embeddings have been shown to be a rich and useful feature in many tasks (**??**).

3 Experimental Setup

3.1 Datasets

We make uses of three 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 (hereafter referred to as ASOIAF); the two books of Leigh Bardugo's "Six of Crows" duology (hereafter referred to as SOC); and the first three books of Brandon Sanderson's "Stormlight Archive" (hereafter referred to as SA). 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 POV character.

ASOIAF and SOC provide ground truth for the main character in the chapter names. Every chapter only uses the POV of that named character. SA indicates using an chapter image corresponding to the main character. Problematically, each chapter of SA has sections which feature different POV characters, though the indicated character is normally the most prevalent. Ideally, the sub-chapter sections individually labelled, however they are not. For this reason the ground truth of SA is much weaker. As such we do not use it for the testing of the main results, but consider it as supplementary training data in **??**.

Dataset	Chapters	POV Characters
ASOIAF	256	15
SOC	91	9
SA	275	6

Table 1: Dataset key features.

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

The total counts of chapters and characters in the datasets, after preprocessing, is shown in Table 1. Preprocessing consisted of discarding chapters for which the POV character was not identifies (e.g. prologues); and of removing the character names from the chapter titles as required.

3.2 Evaluation Details

In the evaluation, the 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 Implementation

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

The text is preprocessed using NLTK (**?**) 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 that 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 (**?**).

Test Set	Method	Train Set	Acc
ASOIAF	First Mentioned	—	0.250
ASOIAF	ML Classical Features	SOC	0.957
ASOIAF	ML Hybrid Features	SOC	0.938
ASOIAF	ML Word Emb. Features	SOC	0.117
ASOIAF	Most Commonly Mentioned	—	0.902
SOC	First Mentioned	—	0.429
SOC	ML Classical Features	ASOIAF	0.923
SOC	ML Hybrid Features	ASOIAF	0.923
SOC	ML Word Emb. Features	ASOIAF	0.527
SOC	Most Commonly Mentioned	—	0.780

Table 2: The results of the character classifier systems.

4 Results and Discussion

4.1 Main results

The results of all the methods on both datasets are shown in Table 2. This includes the two baseline methods, and the machine learning methods with the different feature sets. The machine learning methods are each train on the dataset which they are not tested on. As expected the First Mentioned baseline is very weak.

The Most Commonly Mentioned baseline is much stronger. It marginally outperforms the ML system using Classical Features on the ASIAF dataset. However, it performs much more poorly on the SOC dataset, 84% vs 91% for the Classical Features ML system. The difference in the Most Commonly Mentioned baseline can be attributed to the difference in writing styles between the novels, how often other characters are being described from the point of view of the (so called) main character.

The Word Embedding model performs generally worse than the classical features model. This is actually a surprising good result. The word embedding model contains no features about how frequent the named entity occurs. Only the average of the word embeddings for its context. Yet it is still able to gene… perform the task. This suggests that word vector…

The hybrid results, including both word embedding features, and classical features perform identically to the Classical Features system. This can be attributed to during training, the word embedding features being found to be less useful, and thus not used in any of the trees.

The weak ground truth of SA can be seen is the relatively poor performance of the ML syste… trained on SA, compared to the other ML systems.

Dataset	Method	Acc
ASOIAF	First Mentioned	0.250
ASOIAF	ML Classical Features	0.945
ASOIAF	ML Hybrid Features	0.965
ASOIAF	ML Word Emb. Features	0.945
ASOIAF	Most Commonly Mentioned	0.914
Combined	First Mentioned	0.238
Combined	ML Classical Features	0.894
Combined	ML Hybrid Features	0.900
Combined	ML Word Emb. Features	0.891
Combined	Most Commonly Mentioned	0.868
SOC	First Mentioned	0.431
SOC	ML Classical Features	0.913
SOC	ML Hybrid Features	0.946
SOC	ML Word Emb. Features	0.944
SOC	Most Commonly Mentioned	0.781

Table 3: 10 fold cross-validation results. The ML systems are trained and tested on distinct slices of the same dataset.

4.2 Cross Validation Results

To determine the effect of author/book style on the performance of the systems, we also evaluated them using 10 fold cross validation. By training and testing on different chapters from within the same book we control for variations in the writing style. The results are shown in Table 3. It can be seen that they are indeed significantly better for the classical features system than the results in Table 2 where the systems trained on different books. With the SOC result being a perfect score This suggests that indeed the features that indicate a point of view character in one book, do not perfectly transfer to another; but that they are reasonably consistent within a single book.

4.3 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[1]. This allows the user to apply any of the model discussed to their own novels. With the exclusion of the word embedding based models, as while interesting, these were less performant and much more demanding on computational resources.

[1] http://cherrypy.org/

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(/s) in the left, and an excerpt from the beginning of the section on the right. The user can adjust to show the top-k most-likely characters on this screen, to allow for additional recall. 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 using a series of check-boxes. The user can input a regular expression for the name of the characters they wish to keep. The sections with matching predictions character-names will be selected. As none of the models is perfect, some mistakes are likely. The user can manually correct the selection using the check-boxes before downloading the book.

6 Conclusion

We have presented a tool to allow consumers to restructure their ebooks around the characters they find most interesting. The system must discover the named entities present in each section of the book, and then classify the section as to which character is the Point of View character. For named entity detection we make use of standard tools. The classification is non-trivial. In its design we implemented several systems. Simply selecting the most commonly named character proved very successful as a baseline approach. It is outperformed by a machine learning based classifier using classical features: primarily occurrence counts of the named entity, and of the different parts of speech that occur adjacent to it. While none of the classifiers are perfect, they achieve high enough accuracy to be useful.

The results are presented to the user via a web-interface. The user can use the results to select the chapters from the point of view of the character/s they are most interested in; and correct any errors the system has made. The user can then download the selected subset of their book.

A primary issues is the limited amount of labelled training data. A future version of our application will allow the users to submit corrections. This would be achieved by allowing them to select the true main character from amongst the named entities, then sorting those named entities along side their feature vector, for use in future training.

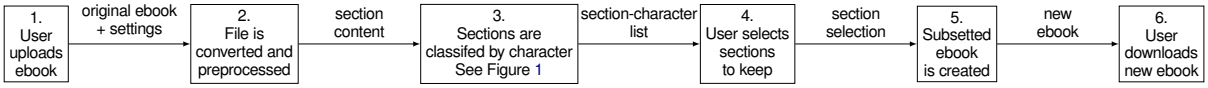

Figure 2: The full process of the using NovelPerspective. Note that step 5 uses the original ebook to subset.

Further tools along these lines have potential as writing aids for authors. To allow them to assess how much "screentime" is being given to each character of their work in progress novels. With additional discourse processing, it would be possible to display other useful analytics, such as how often characters occur in the same scenes.

Another application of related work would be the automatic indexing of texts, by adding features such as the main character name, the perspective, etc. This could be applied to resources such as Wikisource, or Project Gutenberg; which have texts, but not annotations such as these.