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 presented here, give the reader new freedoms in controlling how they consume their media.

Having a large cast of characters, in particular point of view (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 POV; that is to say the reader has little or no 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 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 (Booth, 2010). 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 characters. 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 a 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 they 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 parts of speech, and named entity tags. From this features are extracted for each named entity. Theses feature vectors 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. A broadly similar idea, for primary location classification in news articles, was presented by (Imani et al., 2017).

2.1 Baseline systems

To the best of our knowledge no systems have been developed for this task before. As such we have developed two deterministic baseline character classifiers. These are both potentially useful to the end-user in our deployed system (Section 5), and used to gauge the performance of the more complicated systems in the evaluations presented in Section 4.

It should be noted that the baseline systems, while not using machine learning for the character classification steps, 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.

2.1.1 First Mentioned

An obvious way to determine the main character of the section is to select the first named entity. We use this to define the "First Mentioned" baseline In this system: the **Feature Extraction** step is simply retrieving position of the first use of the named entity; and the **Character Scoring** step is assigning them a score such that earlier is higher. 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 stormy 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.

2.1.2 Most Mentioned

A more robust method to determine the main character, is to use the occurrence counts. We call this the "Most Mentioned" baseline The **Feature Extraction** step is to extract occurrence counts. The **Character Scoring** assign a score based on the proportion of all occurrences of any named entity we for a given character. This works well for many books. The more important a character the more often they would occur. However, it is fooled, for example, by book chapters that are about the point of view 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 appears, 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

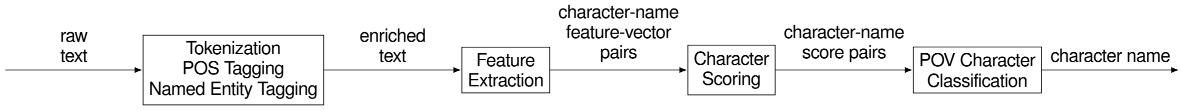

Figure 1: The general structure of the character classification systems. This in turn is the classification step of part of the large stem in Figure 2.

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. The task is to predict if a given named entity is the point of view character. For each possible character (i.e. each named-entity that occurs), a feature vector is extracted (see Section 2.2.1). This feature vector is the input to a binary classifier, which determines the probability that it represents the main character. The **Character Scoring** step is thus the running of the binary classifier: the score is the output probability normalised over all the named entities

2.2.1 Feature Extraction for ML

For the models we investigated two feature sets. The different feature sets correspond to different **Feature Extraction** steps in Figure 1. A hand-engineered feature set, that we call the "Classical" feature set; and a more modern "Word Embedding" feature set". Both feature sets give information about how the each named entity token was used in the text.

The "Classical" feature set uses features that are well established in NLP related tasks. The features can be described as *positional features*,like in the First Mentioned baseline; *occurrence count features*, like in the *Most Mentioned* baseline and *adjacent POS counts*, to give usage context. The *positional features* are the index (in token counts) of the first and last occurrence of the named entity. The *occurrence count features* are simply the number of occurrences of the named entity, supplemented with its rank on that count compared to the others (this is the only rank feature provided). The *adjacent POS counts* are the counts of each part of speech tag on the word prior to the named entity, and on the word after. 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 more verbs. This give 100 base features. To allow for text length invariance we also provide each as a percentage its maximum possible value (e.g. for a given POS tag occurring before a named entity, the percent of times this tag occurred). This gives a total of 200 features.

The "Word Embedding" feature set uses FastText word vectors (Bojanowski et al., 2016). We concatenate the word embedding for the word immediately prior to, and immediately after each occurrence of a 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 (White et al., 2015; Mikolov et al., 2013). This has a total of 600 features.

2.2.2 Classifier

The binary classify that predicts if a named entity is the main character is the key part of the **Character Scoring** step for the machine learning systems. 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. The score for a character is the classifiers predicted probability of it's feature vector being for the main character. This idea of using the output of a binary classifier as a score has also been used, for example, by (Corston-Oliver et al., 2001) where the estimated probability of translation being a written by a human to rate machine translations.

Our approach of using a binary classifier to rate each possible class, may seem similar to the one-vs-rest approach for multi-class 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.

For the Classical Feature-set, we used logistic

Dataset	Chapters	POV Characters
ASOIAF	256	15
SOC	91	9
WOT	432	52
combined	779	76

Table 1: The number of chapters and point of view characters for each dataset

regression, with the features being preprocessed with 0-1 scaling. During preliminary testing we found that many classifiers had similarly high degree of success, and so chose the simplest. With the higher dimensional feature-sets we used a radial bias support vector machine, with standardisation during preprocessing, as has been commonly used with word embeddings on other tasks.

3 Experimental Setup

3.1 Datasets

We make use 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 9 volumes of Robert Jordan's "Wheel of Time" series (hereafter referred to as WOT). In Section 4 we consider the use of each as a training and testing dataset. In the online demonstration (Section 5), we deploy models trained on the combined total of all the datasets.

The requirements of a 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. WOT's ground truth comes from an index created by readers[1]. This ground truth may be less reliable – we corrected one error in it before training, there may be others.

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 identified (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. During evaluation, we sum the scores of the characters alternative aliases/nick-names used in the books. For example merging `Ned` into `Eddard`. This roughly corresponds to the case that a normal user can enter multiple aliases into our application when selecting sections to keep. We do not use these aliases during training, though that is an option that could be investigated in a future work.

3.3 Implementation

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

The text is preprocessed using NLTK (Bird and Loper, 2004). The text is first tokenised, part of speech (POS) tagged , and finally named entity chunked (binary), using NLTK's default methods. Specifically, using the Punkt sentence tokenizer (Kiss and Strunk, 2006), 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. Fantasy novels often use "exotic" names for characters, we found that this often resulted in character named entities being classified as organisations or places. Note that this is particularly disadvantageous to the First Mentioned 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 for the machine learning and evaluations (Pedregosa et al., 2011).

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. We evaluate the machine learning methods using each dataset in turn as the testset and the training set.

The First Mentioned baseline is very weak. The Most Mentioned baseline is much stronger. All the machine learning methods out-perform both baselines;

[1] http://wot.wikia.com/wiki/List_of_Point_of_View_Characters

Test Set	Method	Train Set	Acc
ASOIAF	First Mentioned	—	0.250
ASOIAF	Most Mentioned	—	0.914
ASOIAF	ML Classical Features	SOC	0.953
ASOIAF	ML Classical Features	WOT	**0.984**
ASOIAF	ML Word Emb. Features	SOC	0.863
ASOIAF	ML Word Emb. Features	WOT	0.977
SOC	First Mentioned	—	0.429
SOC	Most Mentioned	—	0.791
SOC	ML Classical Features	ASOIAF	0.923
SOC	ML Classical Features	WOT	0.923
SOC	ML Word Emb. Features	ASOIAF	**0.945**
SOC	ML Word Emb. Features	WOT	0.934
WOT	First Mentioned	—	0.044
WOT	Most Mentioned	—	0.660
WOT	ML Classical Features	ASOIAF	**0.745**
WOT	ML Classical Features	SOC	0.701
WOT	ML Word Emb. Features	ASOIAF	0.699
WOT	ML Word Emb. Features	SOC	0.551

Table 2: The results of the character classifier systems.

with the exception of the SOC trained word embeddings for WOT. We attribute the poor performance of the SOC trained word embeddings, on both the WOT and ASOIAF test sets, to the small amount of training data. With only 91 chapters to generate its training cases from, and 600 dimensions of input to fit it can very easily over-fit – thus causing these bad results.

Using more training data for the ML methods generally gives better results. The datasets are, as was presented in Table 1, of very different sizes. In all cases, except Word Embedding features testing on SOC, the larger training dataset performed at least as well as the smaller. In the exceptional case, of Word Embedding features testing on SOC, we will note that the difference between the results for the ASOIAF and WOT test set, is getting 1 additional test case correct, and so may not be significant. In the demonstration we deploy models training on the combined dataset of all the books.

Between the methods the results on the ASOIAF and SOC are very strong. The results for WOT are much weaker, though still accurate enough to be useful when combined with manual checking. We believe the reason for this is that the WOT is practically a more difficult text. It comes closer to the line between 3rd person omniscient, than the more clear 3rd person limited point of view of the other texts. It also has a much larger set of named entities mentioned in each chapter. As a more difficult task features which may be clearly indicative of main characters in other texts are less indicative on WOT.

It should be noted that the word embedding feature set only contains context information, and not frequency information, but still performs well. Not only that, but it only uses a single word window to each side of the named entity. This suggests that the point of view character can largely be inferred only from the usage of the named entity This is in-line with a humans ability to take a arbitrary single paragraph of text and identify the point of view character.

Overall the performance between word embeddings and classical features is quick close. Particularly when the small SOC training set is excluded. It may be that with more training data the word embeddings could consistently out-perform the classical features. With the limited training data we currently have it seems that the classical feature set is marginally more useful. Pragmatically, the word embeddings do have an issue in that they require storing a large lookup table in memory (Over 10GB for the FastText embeddings); though there are methods to handle this. In our demonstration we allow the user to select from either of the machine learning systems or the baseline systems.

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[2]. This allows the user to apply any of the model discussed to their own novels.

The user uploads an ebook, and selects one of the character detection systems we have discussed above. The user is then presented with a page displaying a list of sections, with the predicted main character(/s) paired with an excerpt from the beginning of the section. 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 spent 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.

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

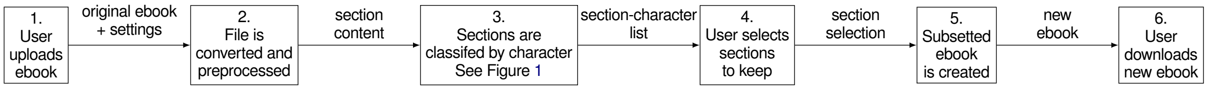

Figure 2: The full process of the using NovelPerspective. Note that step 5 uses the original ebook to subset.

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 issue 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.

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 analytic, 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.

Consumer fiction processing is an interesting area to develop useful tools that can change how we consume written media.

References

Bird, S. and Loper, E. (2004). Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Booth, W. C. (2010). *The rhetoric of fiction*. University of Chicago Press.

Corston-Oliver, S., Gamon, M., and Brockett, C. (2001). A machine learning approach to the automatic evaluation of machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 148–155. Association for Computational Linguistics.

Imani, M. B., Chandra, S., Ma, S., Khan, L., and Thuraisingham, B. (2017). Focus location extraction from political news reports with bias correction. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1956–1964.

Kiss, T. and Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

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.

White, L., Togneri, R., Liu, W., and Bennamoun, M. (2015). How well sentence embeddings capture meaning. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, pages 9:1–9:8. ACM.