NovelPerspective: Identifying Point of View Characters

Lyndon White, Roberto Togneri, Wei Liu, and **Mohammed Bennamoun**
lyndon.white@research.uwa.edu.au, roberto.togneri@uwa.edu.au,
wei.liu@uwa.edu.au, and mohammed.bennamoun@uwa.edu.au
The University of Western Australia.
35 Stirling Highway, Crawley, Western Australia

Abstract

We present NovelPerspective: a tool to allow consumers to subset their digital literature, based on point of view (POV) character. Many novels have multiple main characters each with their own storyline running in parallel. A well-known example is George R. R. Martin's novel: "A Game of Thrones", and others from that series. Our tool detects the main character that each section is from the POV of, 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 heuristic-based baselines, and two 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 storylines being unfolded by the author. As readers, we have often desired to read just one storyline at a time, particularly when reading the book a second-time. In this paper, 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 (POV). The critical part of this system is the detection of the POV character. This is not an insurmountable task, building upon the well established field of named entity recognition. However to our knowledge there is no software to do this. Such a tool would have been useless, in decades past when booked were distributed only on paper. But today, the surge in popularity of ebooks has opened a new niche for consumer narrative processing. Methods are being created to extract social relationships between characters (Elson et al., 2010; Wohlgenannt et al., 2016); to align scenes in movies with those from books (Zhu et al., 2015); and to otherwise augment the literature consumption experience. 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 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, and 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 POV. 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 the perspective of another character (Booth, 1961). 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 in epic fantasy genre, have parallel storylines which 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 a matter of the consumer's personal 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 from 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 whose path intersects with the storyline they are currently reading. We can personally attest for some books reading the chapters one character at a time is indeed possible, and pleasant: the first author of this paper 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 attributing each section to its POV character. 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 system. We investigated several options which the main text of this paper will discuss.

2 Character Classification Systems

The full NovelPerspective pipeline is shown in Figure 1. The core character classification step (step 3), is detailed in Figure 2. In this step the raw text is first enriched with parts of speech,

and named entity tags. We do not perform coreference resolution, working only with direct entity mentions. From this, features are extracted for each named entity. These feature vectors are used to score the entities for the most-likely POV character. The highest scoring character is returned by the system. The different systems presented modify the **Feature Extraction** and **Character Scoring** steps. A broadly similar idea, for detecting the focus location of 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 pre-processing stages.

2.1.1 "First Mentioned" Entity

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 the position of the first use of each name; and the **Character Scoring** step assigns each a score such that earlier is higher. This works for many examples: *"One dark and stormy night, Bill heard a knock at the door."*; however it fails for many others: *" 'Is that Tom?' called out Bill, after hearing a knock."*. 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" Entity

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 count how often the name is used. The **Character Scoring** step assigns each a score based what proportional of all names used were for this entity. This works well for many books. The more important a character
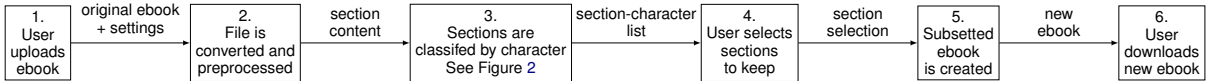
Figure 1: The full NovelPerspective pipeline. Note that step 5 uses the original ebook to subset.
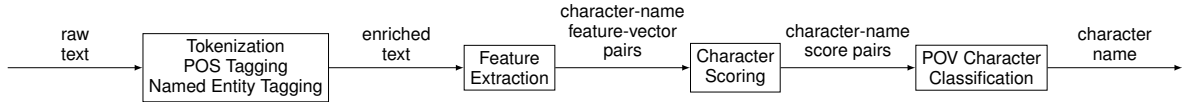

Figure 2: The general structure of the character classification systems. This repeated for each section of the book during step 3 of the full pipeline shown in Figure 1.

is, the more often their name occurs. However, it is fooled, for example, by book chapters that are about the POV character's relationship with a secondary character. In such cases the secondary character may be mentioned more often.

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, i.e. classes, varies from book to book. An information extraction approach is required which can handle these varying classes. As such, a machine learning model for this task can not incorporate direct knowledge of the classes (i.e. character names).

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

We investigated two feature sets as inputs for our machine learning-based solution. They correspond to different **Feature Extraction** steps in Figure 2. 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 the 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. The *adjacent POS counts* are the occurrence counts of each of the 46 POS tags on the word prior to the named entity, and on the word after. We theorised that this POS information would be informative, as it seemed reasonable that the POV character would be described as doing more things, so co-occurring with more verbs. This gives 100 base features. To allow for text length invariance we also provide each of the base features expressed as a portion of its maximum possible value (e.g. for a given POS tag occurring before a named entity, the potion of times this tag occurred). This gives a total of 200 features.

The "Word Embedding" feature set uses FastText word vectors (Bojanowski et al., 2017). We use the pretrained 300 dimensional embeddings trained on English Wikipedia [1]. We concatenate the 300 dimensional 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 concatenated vector over all occurrences of the entity. Such averages of word embeddings have been shown to be a useful

---

[1] https://fasttext.cc/docs/en/pretrained-vectors.html

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 classifier, 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 but 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 classifier's predicted probability of its feature vector being for the main character.

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 fundamental problem is information extraction, and the classifier is a tool for the scoring which is the correct information to report.

With the classical feature set we use logistic regression, with the features being preprocessed with 0-1 scaling. During preliminary testing we found that many classifiers had similar high degree of success, and so chose the simplest. With the word embedding feature set 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.

To use a book for the training and evaluation of our system, we require a ground truth for each section's POV character. ASOIAF and SOC provide

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.

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.[2] 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` in ASOIAF. 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 source code is available on GitHub. [3] Scikit-Learn (Pedregosa et al., 2011) is used for the machine learning and evaluations, and NLTK (Bird and Loper, 2004) is used for textual preprocessing. The text is tokenised, and tagged with POS and named entities using NLTK's default methods. Specifically, these are the Punkt sentence tokenizer, the regex-based improved Tree-Bank word tokenizer, greedy averaged perceptron POS tagger, and the max-entropy binary named entity chunker. The use of a binary, rather than

---

[2] http://wot.wikia.com/wiki/List_of_Point_of_View_Characters
[3] https://github.com/oxinabox/NovelPerspective/

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 Classical Features	WOT+SOC	0.977
ASOIAF	ML Word Emb. Features	SOC	0.863
ASOIAF	ML Word Emb. Features	WOT	0.977
ASOIAF	ML Word Emb. Features	WOT+SOC	0.973
SOC	First Mentioned	—	0.429
SOC	Most Mentioned	—	0.791
SOC	ML Classical Features	WOT	0.923
SOC	ML Classical Features	ASOIAF	0.923
SOC	ML Classical Features	WOT+ASOIAF	0.934
SOC	ML Word Emb. Features	WOT	0.934
SOC	ML Word Emb. Features	ASOIAF	**0.945**
SOC	ML Word Emb. Features	WOT+ASOIAF	**0.945**
WOT	First Mentioned	—	0.044
WOT	Most Mentioned	—	0.660
WOT	ML Classical Features	SOC	0.701
WOT	ML Classical Features	ASOIAF	**0.745**
WOT	ML Classical Features	ASOIAF+SOC	0.736
WOT	ML Word Emb. Features	SOC	0.551
WOT	ML Word Emb. Features	ASOIAF	0.699
WOT	ML Word Emb. Features	ASOIAF+SOC	0.681

Table 2: The results of the character classifier systems. The best results are **bolded**.

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 misclassified 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. Nevertheless, it is required to ensure that all character names are a possibility to be selected.

4 Results and Discussion

Our evaluation results are shown in Table 2 for all methods. 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 as a test set, and using each of the other two and their combination as the training set.

The First Mentioned baseline is very weak. The Most Mentioned baseline is much stronger. In almost all cases machine learning methods outperform both baselines. The results of the machine learning method on the ASOIAF and SOC are very strong. The results for WOT are weaker, though they are still accurate enough to be useful when combined with manual checking.

It is surprising that using the combination of

Test Set	Method	Train Set	Acc
ASOIAF	ML Classical Features	ASOIAF	0.980
ASOIAF	ML Word Emb. Features	ASOIAF	0.988
SOC	ML Classical Features	SOC	0.945
SOC	ML Word Emb. Features	SOC	0.956
WOT	ML Classical Features	WOT	0.785
WOT	ML Word Emb. Features	WOT	0.794

Table 3: The training set accuracy of the machine learning character classifier systems.

two training sets does not always out-perform each on their own. For many methods training on just one dataset resulted in better results. We believe that the difference between the top result for a method and the result using the combined training sets is too small to be meaningful. It can, perhaps, be attributed to a coincidental small similarity in writing style of one of the training books to the testing book. To maximise the generalisability of the NovelPerspective prototype (see Section 5), we deploy models trained on all three datasets combined.

Almost all the machine learning models resulted in similarly high accuracy. The exception to this is word embedding features based model trained on SOC, which for both ASOIAF and WOT test sets performed much worse. We attribute the poor performance of these models to the small amount of training data. SOC has only 91 chapters to generate its training cases from, and the word embedding feature set has 600 dimensions. It is thus very easily to over-fit which causes these poor results.

Table 3 shows the training set accuracy of each machine learning model. This is a rough upper bound for the possible performance of these models on each test set, as imposed by the classifier and the feature set. The WOT bound is much lower than the other two texts. This likely relates to WOT being written in a style that closer to the line between third-person *omniscient*, than the more clear third-person *limited* POV of the other texts. We believe longer range features are required to improve the results for WOT. However, as this achieves such high accuracy for the other texts, further features would not improve accuracy significantly, without additional more difficult training data (and may cause over-fitting).

The results do not show a clear advantage to either machine learning feature set. Both the classical features and the word embeddings work well.

Though, it seems that the classical feature are more robust; both with smaller training sets (like SOC), and with more difficult test sets (like WOT).

5 Demonstration System

The demonstration system is deployed online at https://white.ucc.asn.au/tools/np. A video demonstrating its use can be found at https://youtu.be/iu41pUF4wTY. This web-app, made using the CherryPy framework,[4] allows the user to apply any of the model discussed to their own novels.

The web-app functions as shown in Figure 1. The user uploads an ebook, and selects one of the character classification systems that we have discussed above. They are 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.

The user can select sections to retain. They can use a regular expression to match the character names(/s) they are interested in. The sections with matching predicted character names will be selected. As none of the models is perfect, some mistakes are likely. The user can manually correct the selection before downloading the book.

6 Conclusion

We have presented a tool to allow consumers to re-structure their ebooks around the characters they find most interesting. The system must discover the named entities that are present in each section of the book, and then classify each section as to which character's point of view the section is narrated from. For named entity detection we make use of standard tools. However, the classification is non-trivial. In this design we implemented several systems. Simply selecting the most commonly named character proved successful as a baseline approach. To improve upon this, we developed several machine learning based approaches which perform very well. While none of the classifiers are perfect, they achieve high enough accuracy to be useful.

A future version of our application will allow the users to submit corrections, giving us more training data. However, storing this information poses copyright issues that are yet to be resolved.

---

[4]http://cherrypy.org/

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. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Booth, W. C. (1961). *The rhetoric of fiction*. University of Chicago Press.

Elson, D. K., Dames, N., and McKeown, K. R. (2010). Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 138–147, Stroudsburg, PA, USA. 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.

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.

Wohlgenannt, G., Chernyak, E., and Ilvovsky, D. (2016). Extracting social networks from literary text with word embedding tools. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 18–25.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.