Data Collection and Analysis of Print and Fan Fiction Classification

Channing Donaldson[1][a] and James Pope[2][b]
[1]*University of Arizona, Department of Linguistics, Tucson AZ 85721, U.S.A.*
[2]*University of Bristol, Faculty of Engineering, Bristol BS8 1QU, U.K.*

Keywords: Text Classification, Natural Language Processing, Fan Fiction, Comparative Literature.

Abstract: Fan fiction has provided opportunities for genre enthusiasts to produce their own story lines from existing print fiction. It has also introduced concerns including intellectual property issues for traditional print publishers. An interesting and difficult problem is determining whether a given segment of text is fan fiction or print fiction. Classifying unstructured text remains a critical step for many intelligent systems. In this paper we detail how a significant volume of print and fan fiction was obtained. The data is processed using a proposed pipeline and then analysed using various supervised machine learning classifiers. Given 5 to 10 sentences, our results show an accuracy of 80-90% can be achieved using traditional approaches. To our knowledge this is the first study that explores this type of fiction classification problem.

1 INTRODUCTION

The explosive growth of fan fiction in the last two decades has resulted in significant intellectual property issues. Traditional print fiction writers as well as fan fiction writers require protection. Given a segment of text, literary theorists need to be able to distinguish between hobbyist writers (e.g. fan fiction) and professional writers. This is a special case of a text classification problem that we more simply denote as *fiction classification*. An additional number of research questions are raised. How does the number of sentences in the text segment influence the classification performance? Can digital fan fiction ”look like” printed fiction?

To address these questions we first collected a sizeable amount of print and fan fiction text. We then pre-process the data and perform feature extraction using the traditional term frequency, inverse document frequency (TF-IDF) bag of words technique and a word embedding approach (*word2vec* (Mikolov et al., 2013; Goldberg and Levy, 2014)). Several canonical classifiers are then used to assess the accuracy of the various approaches. We show that given 5 to 10 sentences per instance, approaches can differentiate print from fan fiction with an accuracy between 80-90%. We further provide evidence that TF-IDF followed by the Naive Bayes (NB) classifier provides the best accuracy to computation performance. The contributions of the paper are as follows.

1. Text classification approach for the fiction classification problem
2. Comparative analysis of various approaches for fiction classification
3. Annotated corpus for the research community

The data can be obtained by contacting the authors. The rest of the paper is organised as follows. First the introduction along with the related work is presented. The data collection is then detailed. The data analysis explains the experimental setup and models. Finally, the results are evaluated followed by the conclusion.

2 RELATED WORK

Text analysis is a vast and active area of computing. Within the past ten years researchers have explored such similar problems as large text corpora comparative analysis of document sets using phrase semantic commonality and pairwise distinction (Ren et al., 2017), topic modelling across multiple document sets (Hua et al., 2020), feature selection for literature review (Pintas et al., 2021), automated attribution analysis of quoted speech from 19th century fiction using logistic regression, decision tree, and JRip (Elson and

[a] https://orcid.org/0000-0002-2579-1844
[b] https://orcid.org/0000-0003-2656-363X

Donaldson, C. and Pope, J.
Data Collection and Analysis of Print and Fan Fiction Classification.
DOI: 10.5220/0010774100003122
In *Proceedings of the 11th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2022)*, pages 511-517
ISBN: 978-989-758-549-4; ISSN: 2184-4313

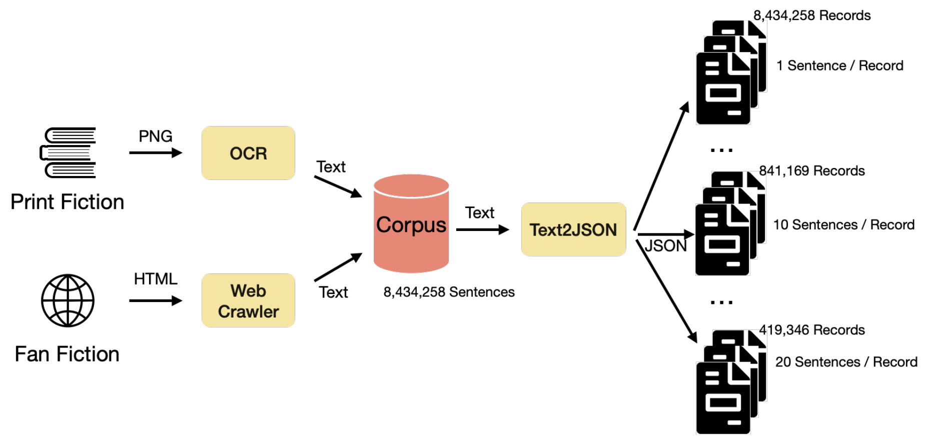


Figure 1: Data Collection Overview.

McKeown, 2010) and analyses of interpersonal relationships and gender roles in 19th century Sweden fiction (Kokkinakis et al., 2014), to name a few.

Text mining has gained significant interest in the last two decades. In it's early stages, Kalt and Croft (Kalt, 1996) suggested treating a document as a bag of words (BOW). Salton, et al. (Salton et al., 1994), made a notable extension to the BOW model by adding weights based on term and document frequency (TF-IDF). Subsequently, Mikolov, et al. (Mikolov et al., 2013), presented *word2vec* that describes how words and their surrounding words can be transformed into a vector (i.e. word embedding) useful for subsequent classification. Recently, research has focused upon context and how the previous words and subsequent words influence meaning. Melamud, et al. (Melamud et al., 2016), present *context2vec* that passes the word embeddings to a Bidirectional LSTM (Bi-LSTM) neural network. Bahdanau, et al. (Bahdanau et al., 2016) show how adding an *attention* layer can improve sequence based models (e.g. LSTM). More recently, Devlin, et al. (Devlin et al., 2019), present the Bidirectional Encoder Representations from Transformers (BERT) that produces contextualised word embeddings.

Our research leverages these existing works for classifying text as either print or fan fiction. To the best of our knowledge, our work is the first to compare fan fiction and print fiction using recent text classification approaches.

3 DATA COLLECTION

Two types of data were collected for this experiment. The first set is from published fiction novels and the second is from literature fan fiction. Figure 1 is a visualisation of our data collection pipeline, in which the two sets of data are converted to UTF8 TXT format (hereby refered to as TXT) and combined into a single corpus with two distinct categories; Print and Fan Fiction. From this binary corpus, we create our JSON files used for data analysis (Pezoa et al., 2016). Table 1 shows a summary of the corpus size.

Table 1: Corpus Summary.

	# Files	GB	# Sentences
Print Fiction	67488	33	7,759,881
Fan Fiction	9724	1.7	674,377

3.1 Print Fiction

The print corpus data set was collected from free book donation areas. The requirement for selection included the modernity of the book of no more than 100 years unless the novel still has an active fan-base and being a fiction novel. Due to restrictions upon scanning capability, ultimately paperback novels make up the majority of the corpus since the weight of the paper effected the ease of which it could be scanned. Thinner paper, like that which is used in hardback books, could not be effectively and quickly scanned with the available tools. Once the novels were collected, an industrial sized cutter was used to remove the spines of the novels, making them easier to be scanned as a whole. These scanned pages were saved in PDF format and organised in files according to the author and book title. Book covers, publishing details, and marketing pages were saved separately from the novel's text and the code is written to ignore PDFs under a certain size. This consideration was done to limit possible noise from these pages affecting the scope of this analysis. In totality the size of the print corpus is roughly 33GB.

Tesseract was used to OCR the PDF document

(Kay, 2007). Apache PDFBox was used to extract the PNGs from the PDF and the remaining text was saved separate in TXT format (Foundation, 2009). During this stage we found that some PDFs had flipped orientation which resulted in nonsensical text extractions. We resolved the orientation problem by annotating approximately 234 PNGs (i.e. individual pages) randomly selected from the print corpus and rotating them into four total orientations to produce 936 PNGs. Our orientation correction tool was developed in previous research (Pope. et al., 2020).

3.2 Fan Fiction

The digital fan fiction corpus data set was collected from the online platform archiveofourown.org. Fan fiction was targeted by collections associated with literature, where each collection is connected to an author and a specific book, and collections had more than 1,000 associated digital fictions within the specified collection. Collections that were associated with only an author and not a book, or collections associated with mixed media such as literature and TV/-movie, were ignored. The goal was to be selective in the comparative text, in which we wanted to associate the literature fan-base directly to printed literature. Code was written to target the specified Collection pages and the first 20 most recent submissions were downloaded only the body of the text in HTML format if it met the requirement of being written in English. Even after collecting at most 40 instances per collection, the size of the Fan fiction was significantly smaller coming in at just under 2GB.

3.3 Text 2 JSON

Once both Print and Fan fiction were in TXT format we converted into a consistent JSON format for processing. We parsed 20 separate JSON file where each file number corresponded to the number of sentences in each instance such that file 1 had 1 sentence per instance, up to file 20 which had 20 sentences per instance totalling in 20 seperate JSON files for analysis. Listing 1.1 are examples of 2 sentence JSON file from both the Print and the Fan Fiction data sets.

Listing 1: JSON Record Example (2 sentences per instance).

```
{
        {"category": "print",
        "text": "Some things were simply better left
              unexamined. The next day, Harry hired a horse
              .",
        "filename": "2020_08_16_15_22_24.txt",
        "paragraph": 95}
}
{
        {"category": "fan",
        "text": "I can't help it. I know it'll ruin me one
              day, alas, perhaps it's one of those parts of
               me I can't change.",
        "filename": "removed for privacy",
        "paragraph": 115}
}
```

4 DATA ANALYSIS

This sections first details how the data is converted from the corpora into formats suitable for several machine learning approaches. Each approach and its results are then examined and finally compared. Our natural language processing code utilised the same framework as created by Pietro (Pietro, 2021). The experiments also used the natural language processing toolkit (NLTK) (Steven Bird, 2021), and the Gensim topic modelling library (Řehůřek and Sojka, 2010).

4.1 Data Preparation

Using the pipeline shown in Figure 2, experiments were conducted using a varying number of sentences per instance for each approach. the data preparation block prepares the text for an experiment in which instances are selected and processed prior to feature representation. For each experiment, the instances are first stratified into print and fan fiction sets. Due to the class imbalance, we under sample the majority fan fiction instances so that 50% of the print fiction is selected. For example, given 1000 fan fiction instances and 100 print fiction instance, we randomly select 50 instance from the fan fiction and then 50 from the print fiction, leaving the classes balanced. We normalise the text by removing Rainbow stop words (Kalt, 1996), proper nouns, and punctuation (Steven Bird, 2021) then preforming lemmatisation upon the text.

4.2 Word Count Approach

Beginning with one of the most traditional approaches to text analysis, the Bag of Words (BOW) method, requires feature engineering to decide which word(s) are more important by weighing them across the entirety of the corpus. BOW does this through vectorisation of text segments. Our code takes the top 10,000 most occurring words and reduces that number with feature selecting using the chi-2 algorithm. We take the resulting smaller vector and perform TF-IDF statistics for weighing, providing numerical representations from the remaining vocabulary. We perform classification upon these representations using both multi-nominal NB and SVM on a 70/30 training

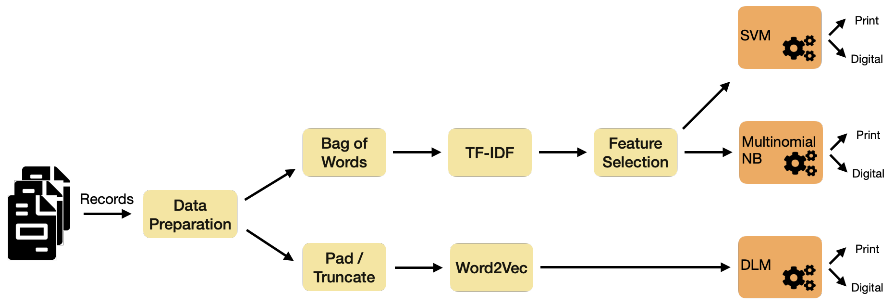
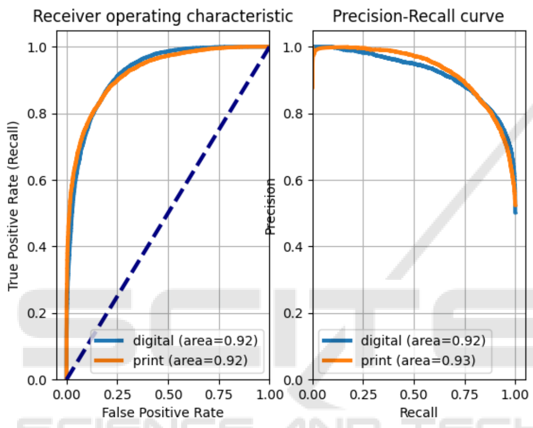
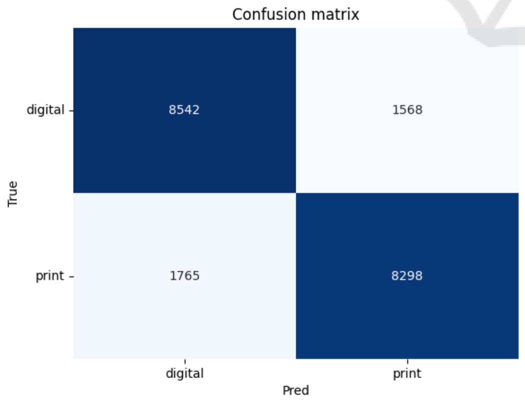


Figure 2: Data Analysis Overview.

test split. Figure 3 shows the receiver operating character, precision-recall curve, and confusion matrix for 10 sentences per instances. These results suggest that the two types of fiction can equally look alike.



(a) ROC at 10 sentences.



(b) Confusion Matrix at 10 sentences.

Figure 3: 10 Sentences Per Instance.

4.3 Deep Learning Model (DLM) Approach

Representation learning approaches attempt to learn word embeddings suitable to be passed to a classifier, directly from the raw text avoiding feature extraction steps. The presented model, derived from Pietro (Pietro, 2021), is similar to *context2vec* (Melamud et al., 2016) with an attention layer (Bahdanau et al., 2016) added between the word embeddings and LSTM layers. The transformed text's vector representation is passed to a softmax layer for classification. Table 2 summarises the architecture. Collectively we denote the approach *Word2Vec+DLM*.

The text length needs to be fixed before presenting to the input layer, so it is either padded or truncated prior to the input layer. The text length is based on the average number of words per sentence times the number of sentences per instance (the corpus average sentence word length is approximately 5). The deep learning model architecture first takes the text and embeds into a 300 dimensional vector. The embedding layer uses the *word2vec* pre-trained weights from Gensim's *word2vec-google-news-300* data library. Training *word2vec* on the corpus would perform better but for convenience and computational reasons the pre-trained weights are used. The word embeddings are then passed to the attention layer (Bahdanau et al., 2016) that decides which parts of the source text to focus on. The output is then passed to two Bi-LSTM layers. Their output is then passed to a ReLU layer which outputs to the softmax layer to produce the probabilities for the classes. The *sparse_categorical_crossentropy* loss function and the *adam* optimiser are used for training.

4.4 Results Comparison

Since precision and recall curves in Figure 3 are effectively the same, we choose to use accuracy instead of the f score for comparison. The experiments are repeated seven times and the mean accu-

Table 2: Deep Learning Model (DLM) Architecture.

Layer	Parameters
Input	textLength=5 x sentences per instance
Embedding	output=(textLength, 300)
Attention	output=(textLength, 300)
LSTM	dropout=0.2, output=(textLength, 2 x textLength)
LSTM	dropout=0.2, output=(2 x textLength)
Dense	activation=ReLU, output=(64)
Dense	activation=Softmax, output=(2)

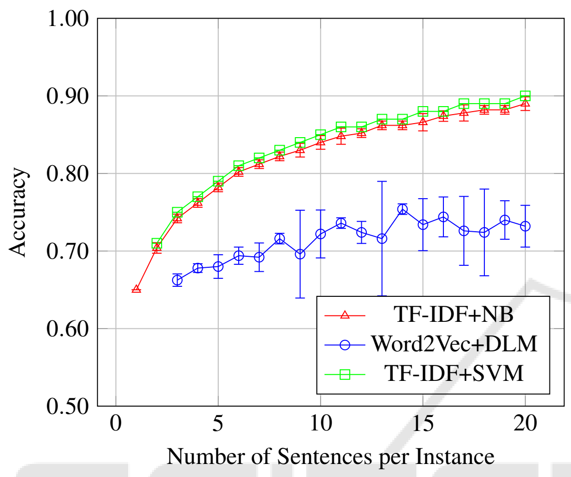


Figure 4: Accuracy for varying number of sentences per instance.

racy is determined along with 95% confidence intervals (using the t-distribution). Figure 4 shows the results of the experiments. Starting from the left of the graph and moving right, the number of instances (sentences, in this case) are decreasing. We found that while TF-IDF+SVM out-performed TF-IDF+NB and *word2vec*, it also greatly under-performed in computational time. The Word2Vec+DLM runtime is better than TF-IDF+SVM. However, TF-IDF+NB performs has the best accuracy and computational performance. Figure 4 shows the accuracy results for the different approaches.

With less than 5 sentences per instance the approaches perform about the same around 70% accuracy. Clearly the TF-IDF+SVM and NB approaches are quickly improving with more sentences per instance. From 1 to 10 sentences per instance the accuracy increased by 20% but from 10 to 20 it only increased by 5%. The best accuracy achieved was ~90% by TF-IDF+SVM. For TF-IDF there is diminishing return with the inflection point around 5 sentences per instance. This provides evidence for how many sentences are sufficient to make a classification and answers one of the research questions.

The Word2Vec+DLM approach performs slightly better from 1 to 10 sentences per instance after which it does not perform any better and possibly slightly worse after 15 sentences per instance. The results clearly show that the Word2Vec+DLM representation learning approach performs worse than the TF-IDF bag of words approach for the fiction classification problem. The results suggest between 5-10% improvement in accuracy can be achieved by performing some feature engineering. A possible improvement may be to train on the corpus instead of using pre-trained weights for *word2vec*.

For comparison, we also determined the processing time that each approach took. This time includes both the training time and the prediction time on the test set. The data preparation time from Figure 2 is not included. All experiments were run on the same CPU (not GPU acceleration). Figure 5 shows the results where the y-axis is in seconds processing time and the x-axis is the number of sentences per instance. Since the number of sentences in the corpus is fixed, the x-axis can also be interpreted as the number of instances, decreasing from left to right. The Word2Vec+DLM approach takes much more processing time than TF-IDF+NB. For example, when the number of sentences per instance is 5, Word2Vec+DLM takes 1435 seconds versus only 30 seconds for TF-IDF+NB. This ratio of roughly 45 times faster holds for the other number of sentences per instance. Most notably, however, is the time required for TF-IDF+SVM as the number of instances increases. Only after 18 sentences per instances does the time beat Word2Vec+DLM. Obviously, TF-IDF+SVM does not scale well with the number of instances. For 5 sentences per instance, TF-IDF+SVM takes ~6 times more than Word2Vec+DLM and ~280 times more than TF-IDF+NB. Given near identical accuracy results between TF-IDF+SVM and TF-IDF+NB, clearly TF-IDF+NB provides a better accuracy to computational time trade-off.

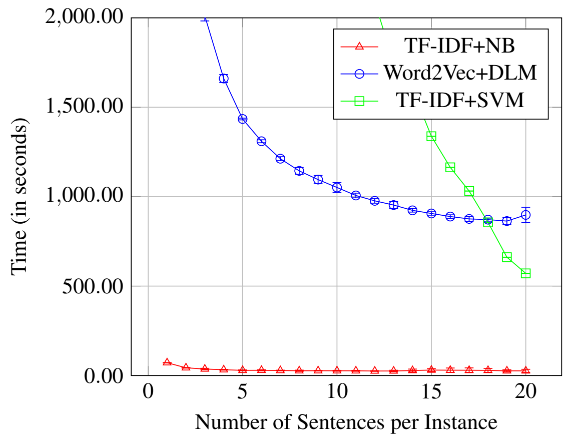


Figure 5: Computational Time for varying number of sentences per instance.

5 CONCLUSION

We conclude that the needed amount of information for fiction classification is between 5 to 10 sentences which returns sufficient accuracy. Furthermore we conclude that while SVM has a higher accuracy rate in classification, its computational runtime easily increases as the number of instance increases, and that Word2Vec has a steady and unimpressive accuracy rate across all sentence counts. For the approaches analysed, we finally conclude that NB with TF-IDF is the better approach for fiction classification.

ACKNOWLEDGEMENTS

The authors would like to thank Gus Hahn-Powell, Valerie Johnson, and Cynthia Mwenja for their valuable feedback and support.

REFERENCES

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Elson, D. K. and McKeown, K. R. (2010). Automatic attribution of quoted speech in literary narrative. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1013–1019. AAAI Press.

Foundation, A. S. (2009). *PDFBox*. Last accessed 26 Sep 2021.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Hua, T., Lu, C.-T., Choo, J., and Reddy, C. K. (2020). Probabilistic topic modeling for comparative analysis of document collections. *ACM Trans. Knowl. Discov. Data*, 14(2).

Kalt, T. (1996). A new probabilistic model of text classification and retrieval.

Kay, A. (2007). Tesseract: An open-source optical character recognition engine. *Linux J.*, 2007(159):2.

Kokkinakis, D., Malm, M., Bergenmar, J., and Ighe, A. (2014). Semantics in storytelling in swedish fiction. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, page 137–142, New York, NY, USA. Association for Computing Machinery.

Melamud, O., Goldberger, J., and Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., and Vrgoč, D. (2016). Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273. International World Wide Web Conferences Steering Committee.

Pietro, M. (2021). *Natural Language Processing Toolkit*. Last accessed 4 Oct 2021.

Pintas, J., Fernandes, L., and Garcia, A. (2021). Feature selection methods for text classification: a systematic literature review. *Artificial Intelligence Review*.

Pope., J., Powers., D., Connell., J. A. J., Jasemi., M., Taylor., D., and Fafoutis., X. (2020). Supervised machine learning and feature selection for a document analysis application. In *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods - ICPRAM,*, pages 415–424. INSTICC, SciTePress.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Ren, X., Lv, Y., Wang, K., and Han, J. (2017). Comparative document analysis for large text corpora. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, page

325–334, New York, NY, USA. Association for Computing Machinery.

Salton, G., Allan, J., and Buckley, C. (1994). Automatic structuring and retrieval of large text files. *Commun. ACM*, 37(2):97–108.

Steven Bird, E. L. (2021). *Natural Language Processing Toolkit*. Version 3.6.2.