Playscript Classification and Automatic Wikipedia Play Articles Generation

Siddhartha Banerjee
The Pennsylvania State University
University Park, PA-16802, USA
Email: sub253@ist.psu.edu

Cornelia Caragea
University of North Texas
Denton, TX-76203, USA
Email: ccaragea@unt.edu

Prasenjit Mitra
The Pennsylvania State University
University Park, PA-16802, USA
Email: pmitra@ist.psu.edu

***Abstract*—In this work, we aim to create Wikipedia pages on plays automatically by extracting relevant information from various web sources. Our approach involves building an efficient classifier that can classify web documents as playscripts. From the set of correctly classified instances of playscripts, we extract relevant play-related information from the documents and use it to obtain additional information from various sources on the web. This information is aggregated and human-readable Wikipedia pages are created using a bot. The results of our experiments show that classifiers trained by combining our designed features along with "bag-of-words" (bow) features outperform classifiers trained using only bow features. Our approach further shows that good quality human-readable pages can be created using our bot. Such automatic page generation process can eventually ensure a more complete Wikipedia.**

I. Introduction

Since the inception of digital libraries around 1990's, the way they have matured is phenomenal [1], [2]. There are already many digital libraries and digital resources available on the Web in the domain of music (e.g., iTunes and Pandora), movies (e.g., International Movie Database (IMDb)) or scholarly papers (e.g., Google Scholar, CiteSeer^X, and ArnetMiner). However, to the best of our knowledge, there has not been any effort made to automatically create a single-information source in the domain of drama and plays. According to Wikipedia[1], a play is "a form of literature written by a playwright, usually consisting of scripted dialogue between characters, intended for theatrical performance rather than just reading." Currently the users have to search and collect information from multiple sites. Often, they give up and therefore obtain incomplete information.

We seek to build a focused crawler that would crawl play-related websites from the web and an automatic article generator that would generate Wikipedia articles from the information crawled from the web. The Wikipedia Theatre project [2] mentions the goal of making Wikipedia "the finest, most comprehensive resource of theatre topics available online". As an effort to create an information resource on plays, we aim to contribute to the Theatre project by automatically authoring such articles. More importantly, this can also have a broader impact of automatic generation of articles on various topics that have limited coverage on Wikipedia.

Automatic authoring of Wikipedia articles is hard. First, our system has to determine which plays are "notable" and deserve a Wikipedia article. Addressing this issue is beyond the scope of this work and will be addressed in the future. Second, automatic natural language generation is a hard problem [3], [4]. Even if we could generate natural language, generating it in the form and level-of-detail needed for an encyclopedia article is exceedingly hard.

Our system initially crawls URLs that contain playscripts [3] using a focused crawler [6]. We design features that help machine learning algorithms to classify web documents into two classes - playscripts and non-scripts, with high accuracies. The script of a play provides a lot of information on the play such as the title, author, characters, etc. This information is extracted and further used to gather additional information (plot summary, audience reception, performances, etc.) on the play from the web, which is later summarized to generate human-readable Wikipedia entries of high quality. Search queries are formulated that help retrieve links on the play from the web using Google. Text from the web pages is extracted, summarized and populated into the Wikipedia articles in separate sections.

We focus on two problems. First, we address the problem of classifying crawled web documents that contain playscripts. Second, we address the problem of extracting relevant information of plays from various web resources and automatically generating Wikipedia articles. The articles generated should also confirm to Wikipedia standards and be coherent for readability. The research questions that we specifically address are: *Can we design techniques to effectively and efficiently classify web documents as containing playscripts? Moreover, can we create high quality Wikipedia articles adhering to Wikipedia quality requirements?*

In summary, *our contributions* are as follows:

(i) To the best of our knowledge, we are the first to address the problem of playscript classification from web documents. The dataset (1988 documents) [4] built as part of this study will be made publicly available to the research community and would be useful for various tasks.

(ii) We show that the addition of nine structural features, such as presence of delimiters, names of characters, etc. further improves the classification accuracy as obtained using Bag-of-Words features. An ensemble approach that combines two classifiers, one trained on structural features and the other trained on "bag of words" achieves the highest performance (0.96 F-Measure) on our dataset.

(iii) Finally, we found that only about 10% of the plays in our dataset have corresponding Wikipedia pages (i.e., only

[1] http://en.wikipedia.org/wiki/Play_(theatre)

[2] http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Theatre

[3] Wordnet [5] defines playscript as "a written version of a play or other dramatic composition, used in preparing for a performance."

[4] https://dl.dropboxusercontent.com/u/105934454/Playscript_dataset.zip

popular plays such as those written by famous playwrights such as William Shakespeare (e.g, *King Lear*), Oscar Wilde (e.g, *The importance of being earnest*) and Bernard Shaw (e.g, *Caesar and Cleopatra*)). Using our dataset, we create new Wikipedia pages automatically for not very well-known plays.

The rest of the paper is organized as follows. In the next section, we discuss related work. In Section 3, we present the feature design approach to classifying web documents followed by automatic Wikipedia page generation. In Section 4, we explain our dataset construction, followed by experiments and results in Section 5. The last section concludes the paper.

II. Related Work

Classification techniques have been applied extensively in movie and music classification tasks [7], [8], [9], [10], [11], [12]. In the literary domain, though classification tasks have been used in several poetry classification works [13], [14] yet there has been very limited application of machine learning approaches in the domain of drama or play-related text. Some notable work in this domain include genre identification of drama using visualization of dramatic structure [15] as well as identifying gestures from play scripts [16].

In contrast to previous work on drama and plays, in this paper we address the classification of web documents as drama and play scripts versus non-scripts.

We also address the task of automatic Wikipedia page generation on plays. Wikipedia has become an important information source [17]. Our preliminary experiments showed that only few plays have corresponding Wikipedia articles, hence we aim to automatically create Wikipedia pages. Our Wikipedia task is closely related to the work by Sauper and Barzilay [18], yet we are restricted by the fact that plays have very less coverage on the internet when compared to other general topics such as *diseases*. Previous work on Wikipedia have used template learning models for creation of domain independent articles [19]. In our case, we use the traditional concept-to-text generation, where a content planner provides a detailed template for information organization [3].

To the best of our knowledge, automatic Wikipedia page generation on topics having scarce web resources has not been addressed in the past.

III. Proposed Approach

Our proposed approach consists of mainly two stages. Firstly, we classify the set of documents into two classes - playscripts and other documents. Then, using the playscripts, we extract information on the play name and author, use this information to extract relevant information from various web sources and eventually aggregate that information as Wikipedia play pages.

A. Feature Design

Drama and play classification can be seen as a text classification problem, where the instances are web documents and the task is to classify them into drama and playscripts or others. The "bag of words" approach, typically used for text classification, constructs a vocabulary of size d, which contains all words in a collection of documents. A document is represented as a vector $\mathbf{x}$ with as many entries as the words in the vocabulary, where an entry k in $\mathbf{x}$ can record the frequency (in the document) of the k^{th} word in the vocabulary, denoted by x_k. However, we argue that drama and playscripts have specific structural properties that substantially differ from those of non-script documents. These structural properties can be exploited to improve the performance of playscript identification. For example, script pages, by definition, contain dialogues, which would be uncommon for a general web document. We hypothesize that structural properties on script pages can provide additional evidence for the drama and playscript classification.

Proposed Featureset	
Feature Name	**Feature Description**
KeywordFreq	Frequency of keyword matches
NumLinesCapitals	Number of lines with capital letters only
RatioLinesCapitals	Ratio of count of lines in capitals to count of all lines
DelimiterFreq	Frequency of delimiters
RatioDelimiterLines	Ratio of number of delimiters to the number of lines
NumBrackets	Number of brackets
NumPlayActorCapitals	Number of play actors using lines with capital letters
NumPlayActorDelimiters	Number of play actors using lines with delimiters
RatioNonBlankLines	Ratio of the number of non-blank lines to total number of lines

TABLE I: Feature names and descriptions of our proposed feature set.

We design nine structural features for script classifiers, as described below. The features are summarized in Table I.

1) Frequency of keyword matches: This is a word-matching based feature. We create a dictionary of keywords $\mathcal{K}$, commonly found in drama and play script documents. These keywords are: $\mathcal{K}$ ={*drama, play, playscript, script, scene, cast, character, screenplay, act, stage, curtain, comedy, humor, tragedy*}. For this feature, we used the number of times words in $\mathcal{K}$ occur in a web document. If two or more keywords in $\mathcal{K}$ occur on a page, we sum up their counts. Although these keywords may exist in other drama-related web documents, e.g., critics' reviews, their presence on a page can be a potentially good indicator for script pages.

2) Number of lines with capital letters only: Generally, character names and information about the time or place where a scene is taking place are mentioned using capital letters and occur on a separate line. To capture this characteristic of scripts, we use the number of lines with capital letters only, as a feature.

3) Ratio of count of lines in capitals to count of all lines: Lines containing dialogues by characters in plays start with a capital letter followed by lower-case letters. For any drama or play script, there would be a particular structure with lines in capitals only (character names or place, etc.) followed by dialogue lines. This alternating behavior of lines in capital letters and dialogue lines is captured using the ratio of the count of lines having all capital letters to the count of all the lines in a document.

4) Frequency of delimiters: Delimiters such as ':' and '-' are used very frequently in playscripts to separate character names from dialogues. To capture this feature, we use the number of occurrences of delimiters in a document as a feature.

5) Ratio of number of delimiters to the number of lines: Lines in a playscript contain delimiters mostly when they separate character names and dialogues. All other lines generally do not contain delimiters. This feature is captured as the ratio of the number of delimiters to the total number of lines in the document.

6) Number of brackets: Actions or expressions (*AURORA (CONT'D)* or *AURORA (laughs again)*) by characters in

playscripts are represented in brackets. *AURORA* is the name of the character and the brackets provide directives to the actor as to what action is to be performed. We extract the number of brackets as a feature which gives an indication of actions and expressions in the play.

7) Number of play actors using lines with capital letters: Characters in playscripts are either separated from dialogues using a newline character or a delimiter such as ":" or "-". For the first type, we identify the lines having only capital letters. The distinct number of words in capitals which occur at least four times, which indirectly refers to the number of characters in the play is captured as a feature. The underlying assumption is that each character has at least four dialogues in the script.

8) Number of play actors using lines with delimiters: This feature is complementary to the previous feature. Playscripts often have character names and dialogues separated by delimiters. In such a case, we extract the word to the left of the delimiters, and calculate the distinct number of words which occur at least four times.

9) Ratio of the number of non-blank lines to the total number of lines: There are blank lines between the character-dialogue pairs in a playscript. The ratio of the number of non-blank to total lines in a document captures this feature.

B. Automatic Wikipedia Page Generation

Our Wikipedia page generation approach extracts and selects information that can be entered into the new Wikipedia entries. Initially, information on the play title, author and characters is extracted from the playscripts. This information is used to further extract additional information on the plays from various web sources. The steps in the automatic generation includes:

(1) Extraction of play title and author: From the set of web pages which have been correctly classified as playscripts, we extract the title as given by the <title> tag on html pages. To extract the author, we check the first three lines of the play page. The intuition behind this is the fact that in most playscript pages, the title forms the first line followed by the author mentioned within the next two lines. We search for patterns such as "by" or "written by" and extract the string following them. This heuristic worked well in all cases.

(2) Extraction of characters in the play: This step extracts all the character names in the play. Generally, in playscripts, character names are separated from dialogues using delimiters. For example, "CHITRA: I bow to thee, Lord Vasanta." "CHITRA" is the character name separated from the dialogue using a ':'. We extract all such occurrences of dialogue - character pairs from the sentences on the page. For characters, we consider only the string to the left of the delimiter. We considered words which have at least a frequency of four in the entire playscript, under the assumption that each character at least has four dialogues in the entire play.

(3) Play synopsis generation: Synopsis forms an important part of any play. Most of the existing play pages on Wikipedia have a section on synopsis or plot. This stage during the extraction involves four major sub-steps: (i) Identifying relevant URLs from search engines, (ii) Extracting text from the URLs, (iii) Relevant passage detection and (iv) Sentence filtering. We explain each of the steps in detail below:

(i) Identifying relevant URLs: To determine relevant URLs, defining appropriate search queries is essential. Recent work by Sauper and Barzilay [18] has shown that queries using a "conjunction of the document title and topic" performs the best while retrieving topic specific URLs. In our case, the topic is the *plot summary* and the title consists of the *play name and author*. Hence, to retrieve links on the synopsis of the play *Chitra* by *Rabindranath Tagore*, we use the query - "Chitra" "Rabindranath Tagore" plot summary. The top 10 results from Google using the above query are further processed and flagged relevant only if the <body> tag in the corresponding URLs have the mention of the play name and the author's name.

(ii) Extracting text from URLs: Most web pages have cluttered text, with a mix of relevant and non-relevant information that include ads, external links,etc. We decided to use the text extraction module in Repustate API [5] to filter out the relevant text from such web pages.

(iii) Relevant passage detection: We use a well known text segmentation method called TextTiling [20] developed by Hearst, where a piece of text is segmented into multiple topical sections. For each topical section, we check the presence of author name and play title to identify it as play related or unrelated, and is flagged accordingly.

(iv) Sentence filtering: This step ensures that only quality sentences worthy of an encyclopaedic article are extracted. We use certain rules to avoid sentences that might not be usable on Wikipedia. Our rules include exclusion of sentences which end in question marks (e.g. "Enthralling Love story isn't it?"), sentences having hyperlink text (e.g.,"It is a one act play available in the net - http://terebess.hu/english/tagore10.html"), sentences in the first person narrative (e.g., "I read the play online today"), etc. We decided not to use multi-document automatic summarization techniques [21] because they did not preserve the coherence of paragraphs during summarization.

(4) Notable references on the play from scholarly and media resources: One of the key research questions we undertake in this work is that concerning *Wikipedia Notability*. In order to ensure that any new article meets the defined notability criteria, we would have to ensure that the topic has had "significant coverage" from "secondary sources". Hence, we decided to extract mentions of the play from books and news articles not authored by the play author. We extracted excerpts from Google Books [6] and the Google News Archive [7]. We also used extractive summarization to summarize the news excerpts and extracted a two sentence summary of the news articles. For summarization, we used SMMRY [8], a keyword selection based text summarizer.

(5) Linking words/concepts internally to other articles: One of the key features in Wikipedia pages is the existence of links to other Wikipedia articles. Primarily it is aimed at increasing the readers' understanding of the topic at hand, hence we decided to automatically identify words/concepts which have to be linked. We considered a number of Wikipedia pages for a better understanding of linking, and in most of the cases (more than 90%) the links appeared to be named-entities. This is expected as Wikipedia urges users not to link common words understood by most readers in context. We used the Stanford NER tagger [22] for tagging entities in various articles.

[5] https://www.repustate.com
[6] http://books.google.com
[7] http://news.google.com/archivesearch
[8] http://smmry.com/about

(1) www.one-act-plays.com, (2) drama.eserver.org, (3) www.eoneill.com, (4) freedrama.net, (5) drama.eserver.org/plays, (6) www.10-minute-plays.com (7) shakespeare.mit.edu

TABLE II: Sample list of Seed URLs input to Heritrix.

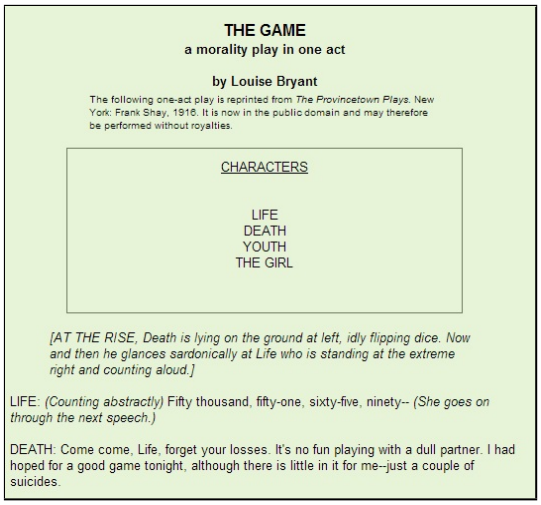

Fig. 1: An example playscript document

IV. Dataset Construction

Table II shows examples of seed URLs that we used for crawling. We used about 30 seed URLs in our crawl. Figure 1 shows a screenshot of a playscript document of a play from our collection - *The Game* by *Louise Bryant*. Every such document typically consists of: Title, Author, Cast or characters, time/act/place/scene (not mandatory) and dialogues.

We manually annotated the documents as playscripts (i.e., the positive class) and non-script documents (i.e., the negative class). The documents had a mix of HTML, PDF and DOC files. Finally, our dataset consisted of 1988 documents - 989 *positive* and 999 *negative*.

V. Experiments and Results

Our experiments are designed to explore the following questions: (i) How do the classifiers trained using the structural features perform as compared to “bag of words" based classifiers? (ii) How does the performance of classifiers trained on the combination of structural features and “bag of words" compare with the performance of classifiers trained on structural features and “bag of words" independently, as well as with the performance of an ensemble of two classifiers, one trained on structural features and the other trained on “bag of words"? (iii) Can we automatically use extracted information on plays from playscripts and other web sources to populate Wikipedia pages?

To answer these questions, we conducted several experiments detailed in the next subsections. In the experiments, we use three different classifiers, namely Support Vector Machine (SVM), Naive Bayes (NB) and Random Forest (RF), and show classifiers' performance using Precision (P), Recall (R) and F-Measure (F-M) averaged in 10-fold cross-validation experiments on our dataset. We used Weka implementation of classifiers with the default set of parameters for all our classification tasks.

A. Comparison of Structural Features with “Bag of Words" Features (BOW)

To assess the quality of our designed structural features, we compared the performance of classifiers trained using these features with that of classifiers trained on the BOW features (our baseline). We used the term occurrences of the words for the BOW features. There were a total of 37968 BOW features.

	Structural Features			BOW (Baseline)		
Classifier	P	R	F-M	P	R	F-M
SVM	0.84	0.84	0.84	0.95	0.95	0.95
NB	0.82	0.78	0.77	0.91	0.91	0.91
RF	0.93	0.93	0.93	0.91	0.91	0.91

TABLE III: Performance of classifiers - comparison between structural features and BOW features.

Table III shows the comparison of structural features and BOW features, using various classifiers. As shown in the table, the SVM and NB classifiers trained using the BOW features outperform, in terms of F-Measure, the classifiers trained using the nine structural features. Random Forest (RF) works better with the nine structural features. Furthermore, we can also see that Support Vector Machine (SVM) trained with the BOW features has the best performance. One thing to be noted here is the fact that Naive Bayes (NB) is used in case of the structural features whereas Naive Bayes Multinomial (NBM) was used with the BOW features.

Error Analysis: We also did error analysis of our document classification task, so that we can improve our feature design in future to correctly classify the documents. The errors can be divided into two types : (i) Playscripts classified as non-playscripts and (ii) Non-playscripts classified as playscripts. Some URLs, such as *freedrama.net* provides *fill in the blanks* exercise documents for students, in which they are required to fill the playscript. Though these documents have been tagged as non-scripts owing to their nature and purpose, these documents do have similar structure to playscripts and hence the classifier (RF) is not able to predict these correctly. On the other hand, some positive class documents were misclassified. Most of such misclassifications occur because either the script is short or has very less number of dialogues, and they do not prominently reveal the specific features we designed.

B. Feature Selection on Structural Features and BOW Features

In order to see if a smaller subset of the structural features would perform better than all the nine features, in other words, if there are redundant or irrelevant features in our set, we carried out the following experiment. We used information gain (IG) to rank our structural features.

Table IV shows the ranked features from the most to the least informative. As can be seen, the first and the second most informative features in the dataset are *NumPlayActorCapitals* and *NumPlayActorDelimiters*, respectively. Both these features are related to extraction of the total number of characters in the play and hence appear to be the most relevant features for identification of playscripts.

Rank	Feature	IG Score
1	NumPlayActorCapitals	0.47
2	NumPlayActorDelimiters	0.34
3	KeywordFreq	0.33
4	NumLinesCapitals	0.24
5	DelimiterFreq	0.19
6	RatioNonBlankLines	0.18
7	RatioDelimiterLines	0.18
8	NumBrackets	0.09
9	RatioLinesCapitals	0.08

TABLE IV: Structural features ranked from most informative to the least, using Information Gain.

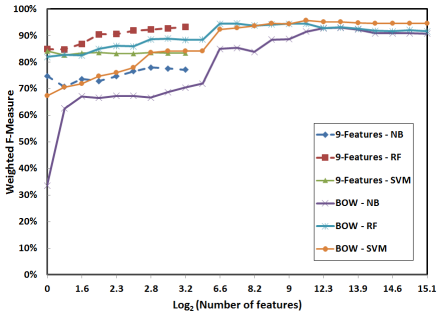

Fig. 2: F-Measure as a function of the number of features on a logarithmic scale. Dotted lines: structural features, Bold lines: BOW features.

Similarly, we performed feature selection on BOW. Figure 2 shows the F1-Measure achieved by the three classifiers, as a function of the number of features (ranked by their IG score) on a logarithmic scale (results are shown for both structural and BOW features). As can be seen from the figure, specifically for RF, as we increase the number of structural features, the weighted F-Measure generally keeps on increasing. On the other hand, for BOW, the performance generally increases as we increase the number of features for all the classifiers. The highest performance is achieved at 1000 features using SVM. We performed a paired t-test between the F1-Measures of structural features and BOW to check the statistical significance of our results and found that the results are statistically significant at significance level of *0.05*.

C. Ensemble of Classifiers vs Single Classifier on Structural and BOW Features

To obtain a composite global model, we also run ensemble of classifiers by combining a classifier trained on structural features and a second one trained on BOW. We run three ensembles: (i) RF+NB, (ii) RF+RF and (iii) RF+SVM. Table V

Ensemble	P	R	F-M
$RF + NB$	0.96	0.96	0.96
$RF + RF$	0.95	0.95	0.95
$RF + SVM$	0.95	0.95	0.95

TABLE V: Performance of ensembles on our dataset

Classifier	P	R	F-M
RF	0.92	0.92	0.92
NB	0.93	0.93	0.93
SVM	0.95	0.95	0.95

TABLE VI: Performance of classifiers on combining structural and BOW features.

shows the performance of ensembles on our dataset. As can be seen, RF+NB outperforms other ensembles, using 10-fold CV. Moreover, comparison of the results with table III shows that there is a 1% increase in F1-Measure using this ensemble over the single SVM classifier trained with BOW features.

We also ran experiments combining all structural features with BOW to assess how the combination of the features would perform when training a classifier. 10 fold CV experiments on the dataset (with features combined) were performed using RF, NB and SVM classifiers. NB in this case is actually NBM as we decided to use the multinomial classifier model owing to the fact that the majority of the combined features are BOW features. Table VI shows the performance of the classifiers using combination of features. As can be seen from the table, SVM works best with the combined set of features. Although RF and NBM show 1-2% improvements using the combined features as compared to only BOW features, F-Measure of SVM does not show any improvement (0.95).

D. Wikipedia Automatic page generation

This experiment is designed to determine whether we can extract relevant information from correctly classified plays and present the extracted information in an organized template.

1) Quality assessment of generated Wiki pages: Our approach as described earlier collected several references from the web. Specifically, the references were used to check relevant synopsis paragraphs on the web, while the news and book links were used to ensure notability and understanding of the public reception on such plays. We will consider the example of a play article we created and was finally accepted after Wikipedia reviewing. The initial page we created using our automatic generation method can be found here [9]. As can be clearly seen, though this play "Chitra" by "Rabindranath Tagore" did not have an existing Wikipedia article, yet it had moderately good coverage on the web, including books and news articles. While it has been studied well in academia, it also had notable public reception as evident from the media articles.

This submitted page was finally accepted after Wikipedia reviewing. There were significant changes made by the reviewers and the final page reflects some of the improvements we can incorporate in our bot. The final accepted page [10] shows that while most of the references have been retained, the scholarly references have been merged in the synopsis section. Also the media reviews underwent several changes in terms of reducing excerpts from news articles. While some of the quotes were retained, yet they were condensed or paraphrased based on the understanding from the actual source. The other play "Fourteen" by *Alice Gerstensberg*, was moved into Wikipedia

[9] http://en.wikipedia.org/w/index.php?title=Chitra_(play)&oldid=566936809
[10] http://en.wikipedia.org/wiki/Chitra_(play)

mainspace with minimal changes. All the references, quotes and paragraphs were retained [11].

Statistics	Count
Number of play pages created	15
Number of play pages accepted by Wikipedia	2
Number of play pages awaiting review	4
Number of play pages rejected due to Notability issue	7
Number of play pages rejected due to quality	2
Average number of references generated	6
Number of plays with synopsis	7
Average number of media articles	2
Average number of scholarly articles	1
Average number of external links	5

TABLE VII: Statistics of generated Wikipedia articles.

Table VII shows some quantitative statistics on some of the plays for which we created automatic pages. We did manual evaluation of the content of the various generated pages. The comments on the rejected pages gave us an insight on what improvements we can propose to enhance quality of the articles. As can be seen from the table, most reviewers cited problems with notability issues as we could not gather many resources from media or scholarly articles. Only in two cases, the article was rejected on grounds of quality stating that it appeared more like an essay rather than an encyclopaedic article. Moreover, information extracted from various sources needs to be summarized in an abstractive [23] fashion to avoid copyright infringement issues. We aim to replicate the success achieved in the two accepted articles by improving information extraction from the web and ensuring better quality of article authoring by employing natural language generation techniques.

VI. Conclusions and Future Work

In this work, we proposed a supervised machine learning model for playscript classification. Our major contribution is the design of structural features for playscript classification, which along with bag-of-words' features can classify playscripts with high accuracies. We also showed that an ensemble of classifiers achieve the highest accuracy on our dataset. We designed a *bot* to automatically create Wikipedia pages by extracting relevant information from plays on the Web. These articles would be a good starting point for readers interested in plays. In future, we plan to improve synopsis generation of such articles. We also plan to use abstractive summarization and natural language generation techniques to improve article generation quality.

References

[1] E. A. Fox, M. A. Gonçalves, and N. A. Kipp, "Digital libraries," in *Handbook on Information Technologies for Education and Training*. Springer, 2002, pp. 623–641.

[2] D. B. Marcum, "Research questions for the digital era library," 2003.

[3] E. Reiter and R. Dale, *Building natural language generation systems*. MIT Press, 2000, vol. 152.

[4] A. Gatt and E. Reiter, "Simplenlg: A realisation engine for practical applications," in *Proceedings of the 12th European Workshop on Natural Language Generation*. Association for Computational Linguistics, 2009, pp. 90–93.

[5] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[6] S. Chakrabarti, M. Van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," *Computer Networks*, vol. 31, no. 11, pp. 1623–1640, 1999.

[7] Z. Rasheed, Y. Sheikh, and M. Shah, "On the use of computable features for film classification," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 1, pp. 52–64, 2005.

[8] Z. Rasheed and M. Shah, "Movie genre classification by exploiting audio-visual features of previews," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2. IEEE, 2002, pp. 1086–1089.

[9] M. F. McKinney and J. Breebaart, "Features for audio and music classification," in *Proc. ISMIR*, vol. 3, 2003, pp. 151–158.

[10] M. I. Mandel and D. P. Ellis, "Song-level features and support vector machines for music classification," in *ISMIR 2005: 6th International Conference on Music Information Retrieval: Proceedings: Variation 2: Queen Mary, University of London & Goldsmiths College, University of London, 11-15 September, 2005*. Queen Mary, University of London, 2005, pp. 594–599.

[11] P. Chaovalit and L. Zhou, "Movie review mining: A comparison between supervised and unsupervised classification approaches," in *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*. IEEE, 2005, pp. 112c–112c.

[12] A. Kennedy and D. Inkpen, "Sentiment classification of movie reviews using contextual valence shifters," *Computational Intelligence*, vol. 22, no. 2, pp. 110–125, 2006.

[13] N. Jamal, M. Mohd, and S. A. Noah, "Poetry classification using support vector machines," *Journal of Computer Science*, vol. 8, no. 9, p. 1441, 2012.

[14] R. M. Roxas and G. Tapang, "Prose and poetry classification and boundary detection using word adjacency network analysis," *International Journal of Modern Physics C*, vol. 21, no. 04, pp. 503–512, 2010.

[15] S. Ramsay, "In praise of pattern," *Faculty Publications–Department of English*, p. 57, 2005.

[16] E. Hanser, P. Mc Kevitt, T. Lunney, and J. Condell, "Scenemaker: Intelligent multimodal visualisation of natural language scripts," in *Artificial Intelligence and Cognitive Science*. Springer, 2010, pp. 144–153.

[17] T. Lucassen and J. M. Schraagen, "Trust in wikipedia: how users trust information from an unknown source," in *Proceedings of the 4th workshop on Information credibility*. ACM, 2010, pp. 19–26.

[18] C. Sauper and R. Barzilay, "Automatically generating wikipedia articles: A structure-aware approach," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, 2009, pp. 208–216.

[19] C. Yao, X. Jia, S. Shou, S. Feng, F. Zhou, and H. Liu, "Autopedia: automatic domain-independent wikipedia article generation," in *Proceedings of the 20th international conference companion on World wide web*. ACM, 2011, pp. 161–162.

[20] M. A. Hearst, "Texttiling: Segmenting text into multi-paragraph subtopic passages," *Computational linguistics*, vol. 23, no. 1, pp. 33–64, 1997.

[21] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz, "Multi-document summarization by sentence extraction," in *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization-Volume 4*. Association for Computational Linguistics, 2000, pp. 40–48.

[22] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 363–370.

[23] D. R. Radev, E. Hovy, and K. McKeown, "Introduction to the special issue on summarization," *Computational linguistics*, vol. 28, no. 4, pp. 399–408, 2002.

[11] https://en.wikipedia.org/wiki/Fourteen_(play)