Parsing Screenplays for Extracting Social Networks from Movies

Apoorv Agarwal[†], Sriramkumar Balasubramanian[†], Jiehan Zheng[‡], Sarthak Dash[†]

[†]**Dept. of Computer Science**
Columbia University
New York, NY, USA

[‡]**Peddie School**
Hightstown, NJ, USA

apoorv@cs.columbia.edu

jzheng-14@peddie.org

Abstract

In this paper, we present a formalization of the task of parsing movie screenplays. While researchers have previously motivated the need for parsing movie screenplays, to the best of our knowledge, there is no work that has presented an evaluation for the task. Moreover, all the approaches in the literature thus far have been regular expression based. In this paper, we present an NLP and ML based approach to the task, and show that this approach outperforms the regular expression based approach by a large and statistically significant margin. One of the main challenges we faced early on was the absence of training and test data. We propose a methodology for using well structured screenplays to create training data for anticipated *anomalies* in the structure of screenplays.

1 Introduction

Social network extraction from unstructured text has recently gained much attention (Agarwal and Rambow, 2010; Elson et al., 2010; Agarwal et al., 2013a; Agarwal et al., 2013b; He et al., 2013). Using Natural Language Processing (NLP) and Machine Learning (ML) techniques, researchers are now able to gain access to networks that are not associated with any meta-data (such as email links and self-declared friendship links). Movies, which can be seen as visual approximations of unstructured literary works, contain rich social networks formed by interactions between characters. There has been some effort in the past to extract social networks from movies (Weng et al., 2006; Weng et al., 2007; Weng et al., 2009; Gil et al., 2011). However, these approaches are primarily regular expression based with no evaluation of how well they work.

In this paper we introduce a formalization of the task of parsing screenplays and present an NLP and ML based approach to the task. By parsing a screenplay, we mean assigning each line of the screenplay one of the following five tags: "S" for scene boundary, "N" for scene description, "C" for character name, "D" for dialogue, and "M" for meta-data. We expect screenplays to conform to a strict grammar but they often do not (Gil et al., 2011). This disconnect gives rise to the need for developing a methodology that is able to handle anomalies in the structure of screenplays. Though the methodology proposed in this paper is in the context of movie screenplays, we believe, it is general and applicable to parse other kinds of noisy documents.

One of the earliest challenges we faced was the absence of training and test data. Screenplays, on average, have 7,000 lines of text, which limits the amount of annotated data we can obtain from humans. We propose a methodology for using well structured screenplays to create training data for anticipated *anomalies* in the structure of screenplays. For different types of anomalies, we train separate classifiers, and combine them using ensemble learning. We show that our ensemble outperforms a regular-expression baseline by a large and statistically significant margin on an unseen test set (0.69 versus 0.96 macro-F1 measure for the five classes). Apart from performing an intrinsic evaluation, we also present an extrinsic evaluation. We show that the social network extracted from the screenplay tagged by our ensemble is *closer* to the network extracted from a screenplay tagged

Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLfL) @ EACL 2014, pages 50–58,
Gothenburg, Sweden, April 27, 2014.

by a human, as compared to the network extracted from a screenplay tagged by the baseline.

The rest of the paper is structured as follows: in section 2, we present common terminology used to describe screenplays. We survey existing literature in section 3. Section 4 presents details of our data collection methodology, along with the data distribution. Section 5 gives details of our regular-expression based system, which we use as a baseline for evaluation purposes. In section 6, we present our machine learning approach. In section 7, we give details of the features we use for machine learning. In section 8, we present our experiments and results. We conclude and give future directions of research in section 9.

2 Terminology

Turetsky and Dimitrova (2004) describe the structure of a movie screenplay as follows: a screenplay describes a story, characters, action, setting and dialogue of a film. Additionally, they report that the structure of a screenplay follows a (semi) regular format. Figure 1 shows a snippet of a screenplay from the film – *The Silence of the Lambs*. A scene (tag "S") starts with what is called the *slug line* (or scene boundary). The slug line indicates whether the scene is to take place inside or outside (INT, EXT), the name of the location ("FBI ACADEMY GROUNDS, QUANTICO, VIRGINIA"), and can potentially specify the time of day (e.g. DAY or NIGHT). Following the scene boundary is a scene description. A scene description is followed by a character name (tag "C"), which is followed by dialogues (tag "D"). Character names are capitalized, with an optional (V.O.) for "Voice Over" or (O.S.) for "Off-screen." Dialogues, like scene descriptions, are not associated with any explicit indicators (such as INT, V.O.), but are indented at a unique level (i.e. nothing else in the screenplay is indented at this level). Screenplays may also have other elements, such as "CUT TO:", which are directions for the camera, and text describing the intended mood of the speaker, which is found within parentheses in the dialogue. For lack of a name for these elements, we call them "Meta-data" (tag "M").

3 Literature Survey

One of the earliest works motivating the need for screenplay parsing is that of Turetsky and Dimitrova (2004). Turetsky and Dimitrova (2004) proposed a system to automatically align written screenplays with their videos. One of the crucial steps, they noted, is to parse a screenplay into its different elements: scene boundaries, scene descriptions, character names, and dialogues. They proposed a grammar to parse screenplays and show results for aligning *one* screenplay with its video. Weng et al. (2009) motivated the need for screenplay parsing from a social network analysis perspective. They proposed a set of operations on social networks extracted from movies and television shows in order to find what they called *hidden semantic* information. They proposed techniques for identifying lead roles in *bilateral* movies (movies with two main characters), for performing community analysis, and for automating the task of story segmentation. Gil et al. (2011) extracted character interaction networks from plays and movies. They were interested in automatically classifying plays and movies into different genres by making use of social network analysis metrics. They acknowledged that the scripts found on the internet are not in consistent formats, and proposed a regular expression based system to identify scene boundaries and character names.

While there is motivation in the literature to parse screenplays, none of the aforementioned work addresses the task formally. In this paper, we formalize the task and propose a machine learning based approach that is significantly more effective and tolerant of anomalous structure than the baseline. We evaluate our models on their ability to identify scene boundaries and character names, but also on their ability to identify other important elements of a screenplay, such as scene descriptions and dialogues.

4 Data

We crawled the Internet Movie Script Database (IMSDB) website[1] to collect movie screenplays. We crawled a total of 674 movies. Movies that are well structured have the property that scene boundaries and scene descriptions, character names, and dialogues are all at different but fixed levels of indentation.[2] For example, in the movie in Figure 1, all scene boundaries and scene

[1] http://www.imsdb.com

[2] By level of indentation we mean the number of spaces from the start of the line to the first non-space character.

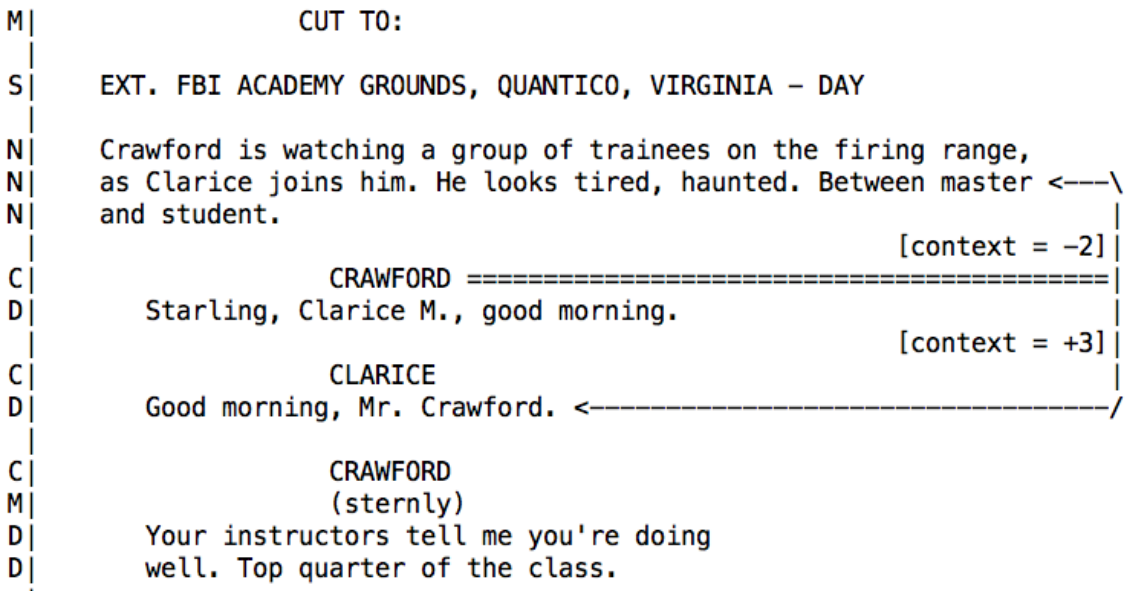

Figure 1: Example screenplay: first column shows the tags we assign to each line in the screenplay. M stands for "Meta-data", S stands for "Scene boundary", N stands for "Scene description", C stands for "Character name", and D stands for "Dialogue." We also show the lines that are at context -2 and +3 for the line "CRAWFORD."

descriptions are at the same level of indentation, equal to five spaces. All character names are at a different but fixed level of indentation, equal to 20 spaces. Dialogues are at an indentation level of eight spaces. These indentation levels may vary from one screenplay to the other, but are consistent within a well formatted screenplay. Moreover, the indentation level of character names is strictly greater than the indentation level of dialogues, which is strictly greater than the indentation level of scene boundaries and scene descriptions. For each crawled screenplay, we found the frequency of unique indentation levels in that screenplay. If the top three unique frequencies constituted 90% of the total lines of a screenplay, we flagged that the movie was well-structured, and assigned tags based on indentation levels. Since scene boundaries and scene descriptions are at the same level of indentation, we disambiguate between them by utilizing the fact that scene boundaries in well-formatted screenplays start with tags such as INT. and EXT. We programmatically checked the *sanity* of these automatically tagged screenplays by using the following procedure: 1) check if scene descriptions are between scene boundaries and character names, 2) check if dialogues are between character names, and 3) check if all character names are within two scene boundaries. Using this methodology, we were able to tag 222 movies that pass the sanity check.

Data	# S	# N	# C	# D	# M
TRAIN	2,445	21,619	11,464	23,814	3,339
DEV1	714	7,495	4,431	9,378	467
DEV2	413	5,431	2,126	4,755	762
TEST	164	845	1,582	3,221	308

Table 1: Data distribution

Table 1 gives the distribution of our training, development and test sets. We use a random subset of the aforementioned set of 222 movies for training purposes, and another random subset for development. We chose 14 movies for the training set and 9 for the development set. Since human annotation for the task is expensive, instead of getting all 23 movies checked for correctness, we asked an annotator to only look at the development set (9 movies). The annotator reported that one out of 9 movies was not correctly tagged. We removed this movie from the development set. From the remaining 8 movies, we chose 5 as the first development set and the remaining 3 as the second development set. For the test set, we asked our annotator to annotate a randomly chosen screenplay (*Silver Linings Playbook*) from scratch. We chose this screenplay from the set of movies that

we were unable to tag automatically, i.e. *not* from the set of 222 movies.

5 Baseline System

Gil et al. (2011) mention the use of regular expressions for tagging screenplays. However, they do not specify the regular expressions or their exact methodology. We use common knowledge about the structure of the screenplay (underlined text in section 2) to build a baseline system, that uses regular expressions and takes into account the grammar of screenplays.

Since scene descriptions, characters and dialogues are relative to the scene boundary, we do a first pass on the screenplay to tag scene boundaries. We created a dictionary of words that are expected to indicate scene boundaries. We use this dictionary for tagging lines in the screenplay with the tag "S". We tag all the lines that contain tags indicating a character (V.O., O.S.) with "C". We built a dictionary of meta-data tags that contains patterns such as "CUT TO:, DISSOLVE TO." We tag all the remaining untagged lines containing these patterns with the tag "M." This exhausts the list of regular expression matches that indicate a certain tag.

In the next pass, we incorporate prior knowledge that scene boundaries and character names are capitalized. For this, we tag all the untagged lines that are capitalized, and that have more than three words as scene boundaries (tag "S"). We tag all the untagged lines that are capitalized, and that have less than four words as character (tag "C"). The choice of the number four is not arbitrary; we examined the set of 222 screenplays that was tagged using indentation information and found that less than two percent of the character names were of length greater than three.

Finally, we incorporate prior knowledge about relative positions of dialogues and scene descriptions to tag the remaining untagged lines with one of two tags: "D" or "N". We tag all the untagged lines between a scene boundary and the first character occurrence as "N". We tag all the lines between consecutive character occurrences, the last character occurrence and the scene boundary as "D".

We use this baseline system, which incorporates all of the prior knowledge about the structure of screenplays, to tag movies in our first development set DEV1 (section 8). We report a macro-F1 measure for the five tags as 0.96. This confirms that our baseline is well suited to parse screenplays that are well structured.

6 Machine Learning Approach

Note that our baseline system is not dependent on the level of indentation (it achieves a high macro-F1 measure without using indentation information). Therefore, we have already dealt with one common problem with screenplays found on the web: bad indentation. However, there are other problems, some of which we noticed in the limited data we manually examined, and others that we anticipate: (1) missing scene boundary specific patterns (such as INT./EXT.) from the scene boundary lines, (2) uncapitalized scene boundaries and (3) uncapitalized character names. These are problems that a regular expression based system is not well equipped to deal with. In this section, we discuss a strategy for dealing with screenplays, which might have anomalies in their structure, without requiring additional annotations.

We *synthesize* training and development data to *learn* to handle the aforementioned three types of anomalies. We create eight copies of our TRAIN set: one with no anomalies, represented as TRAIN_000, [3] one in which character names are uncapitalized, represented as TRAIN_001, one in which both scene boundaries and character names are uncapitalized, represented as TRAIN_011, and so on. Similarly, we create eight copies of our DEV1 set: {DEV1_000, DEV1_001, ..., DEV1_111}. Now we have eight training and eight development sets. We train eight models, and choose the parameters for each model by tuning on the respective development set. However, at test time, we require one model. Moreover, our model should be able to handle all types of anomalies (all of which could be present in a random order). We experiment with three ensemble learning techniques and choose the one that performs the best on the second development set, DEV2. We add all three types of anomalies, randomly, to our DEV2 set.

For training individual models, we use Support Vector Machines (SVMs), and represent data as feature vectors, discussed in the next section.

[3] Each bit refers to the one type of anomaly described in the previous paragraph. If the least significant bit is 1, this means, the type of anomaly is uncapitalized characters names.

7 Features

We have six sets of features: bag-of-words features (BOW), bag-of-punctuation-marks features (BOP), bag-of-terminology features (BOT), bag-of-frames features (BOF), bag-of-parts-of-speech features (POS), and hand-crafted features (HAND).

We convert each line of a screenplay (input example) into a feature vector of length 5,497: 3,946 for BOW, 22 for BOP, 2*58 for BOT, 2*45 for POS, 2*651 for BOF, and 21 for HAND.

BOW, BOP, and BOT are binary features; we record the presence or absence of elements of each bag in the input example. The number of terminology features is multiplied by two because we have one binary vector for "contains term", and another binary vector for "is term." We have two sets of features for POS and BOF. One set is binary and similar to other binary features that record the presence or absence of parts-of-speech and frames in the input example. The other set is numeric. We record the normalized counts of each part-of-speech and frame respectively. The impetus to design this second set of features for parts-of-speech and frames is the following: we expect some classes to have a characteristic distribution of parts-of-speech and frames. For example, scene boundaries contain the location and time of scene. Therefore, we expect them to have a majority of *nouns*, and frames that are related to location and time. For the scene boundary in Figure 1 (*EXT. FBI ACADEMY ... - DAY*), we find the following distribution of parts of speech and frames: 100% nouns, 50% frame LOCALE (with frame evoking element *grounds*), and 50% frame CALENDRIC_UNIT (with frame evoking element *DAY*). Similarly, we expect the character names to have 100% nouns, and no frames.

We use Stanford part-of-speech tagger (Toutanova et al., 2003) for obtaining the part-of-speech tags and Semafor (Chen et al., 2010) for obtaining the FrameNet (Baker et al., 1998) frames present in each line of the screenplay.

We devise 21 hand-crafted features. Sixteen of these features are binary (0/1). We list these features here (the feature names are self-explanatory): has-non-alphabetical-chars, has-digits-majority, has-alpha-majority, is-quoted, capitalization (has-all-caps, is-all-caps), scene boundary (has-INT, has-EXT), date (has-date, is-date), number (has-number, is-number), and parentheses (is-parenthesized, starts-with-parenthesis, ends-with-parenthesis, contains-parenthesis). We bin the preceding number of blank lines into four bins: 0 for no preceding blank lines, 1 for one preceding blank line, 2 for two preceding blank lines, and so on. We also bin the percentage of capitalized words into four bins: 0 for the percentage of capitalized words lying between 0-25%, 1 for 25-50%, and so on. We use three numeric features: number of non-space characters (normalized by the maximum number of non-space characters in any line in a screenplay), number of words (normalized by the maximum number of words in any line in a screenplay), and number of characters (normalized by the maximum number of characters in any line in a screenplay).

For each line, say $line_i$, we incorporate context up to x lines. Figure 1 shows the lines at context -2 and +3 for the line containing the text *CRAWFORD*. To do so, we append the feature vector for $line_i$ by the feature vectors of $line_{i-1}, line_{i-2}, \ldots line_{i-x}$ and $line_{i+1}, line_{i+2}, \ldots line_{i+x}$. x is one of the parameters we tune at the time of training. We refer to this parameter as CONTEXT.

8 Experiments and Results

In this section, we present experiments and results for the task of tagging the lines of a screenplay with one of five tags: {S, N, C, D, M}. Table 1 shows the data distribution. For parameter tuning, we use DEV1 (section 8.1). We train separate models on different types of known and anticipated anomalies (as discussed in section 6). In section 8.2, we present strategies for combining these models. We select the right combination of models and features by tuning on DEV2. Finally, we show results on the test set, TEST. For all our experiments, we use the default parameters of SVM as implemented by the SMO algorithm of Weka (Hall et al., 2009). We use a linear kernel.[4]

8.1 Tuning learning parameters

We tune two parameters: the amount of training data and the amount of CONTEXT (section 7) required for learning. We do this for each of the eight models (TRAIN_000/DEV1_000, ..., TRAIN_111/DEV1_111). We merge training

[4]We tried the polynomial kernel up to a degree of four and the RBF kernel. They performed worse than the linear kernel.

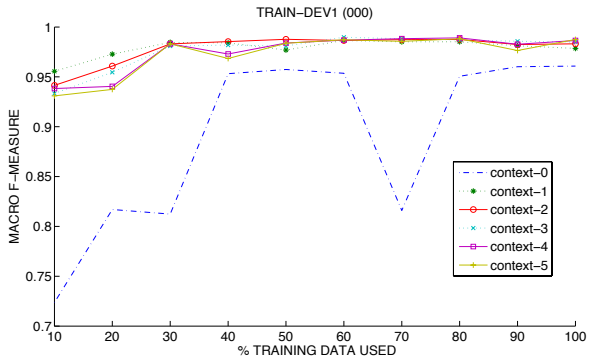

Figure 2: Learning curve for training on TRAIN_000 and testing on DEV1_000. X-axis is the % of training data, in steps of 10%. Y-axis is the macro-F1 measure for the five classes.

data from all 14 movies into one (TRAIN). We then randomize the data and split it into 10 pieces (maintaining the relative proportions of the five classes). We plot a learning curve by adding 10% of training data at each step.

Figure 2 shows the learning curve for training a model on TRAIN_000 and testing on DEV1_000.[5] The learning curve shows that the performance of our classifier without any context is significantly worse than the classifiers trained on context. Moreover, the learning saturates early, and stabilizes at about 50% of the training data. From the learning curves, we pick CONTEXT equal to 1, and the amount of training data equal to 50% of the entire training set.

Table 2 shows a comparison of our rule based baseline with the models trained using machine learning. For the 000 setting, when there is no anomaly in the screenplay, our rule based baseline performs well, achieving a macro-F1 measure of 0.96. However, our machine learning model outperforms the baseline by a statistically significant margin, achieving a macro-F1 measure of 0.99. We calculate statistical significance using McNemar's significance test, with significance defined as $p < 0.05$.[6] Results in Table 2 also show that while a deterministic regular-expression based system is not well equipped to handle anomalies, there is enough value in our feature set, that our machine learning based models learn to adapt to any combination of the three types of anomalies, achieving a high F1-measure of 0.98 on average.

[5]Learning curves for all our other models were similar.

[6]We use the same test for reporting other statistically significance results in the paper.

8.2 Finding the right ensemble and feature selection

We have trained eight separate models, which need to be combined into one model that we will make predictions at the test time. We explore the following ways of combining these models:

1. MAJ: Given a test example, we get a vote from each of our eight models, and take a majority vote. At times of a clash, we pick one randomly.

2. MAX: We pick the class predicted by the model that has the highest confidence in its prediction. Since the confidence values are real numbers, we do not see any clashes.

3. MAJ-MAX: We use MAJ but at times of a clash, we pick the class predicted by the classifier that has the highest confidence (among the classifiers that clash).

Table 3 shows macro-F1 measures for the three movies in our DEV2 set. Note, we added the three types of anomalies (section 6) randomly to the DEV2 set for tuning the type of ensemble. We compare the performance of the three ensemble techniques with the individual classifiers (trained on TRAIN_000, ... TRAIN_111).

The results show that all our ensembles (except MAX for the movie *The Last Temptation of Christ*) perform better than the individual models. Moreover, the MAJ-MAX ensemble outperforms the other two by a statistically significant margin. We thus choose MAJ-MAX as our final classifier.

Table 4 shows results for removing one of all feature sets, one at a time. These results are for our final model, MAJ-MAX. The row "All" shows the results when we use all our features for training. The consecutive rows show the result when we remove the mentioned feature set. For example, the row "- BOW" shows the result for our classifier that was trained without the bag of words feature set.

Table 4 shows that the performance drops the most for bag of words (BOW) and for our handcrafted features (HAND). The next highest drop is for the bag of frames feature set (BOF). Error analysis revealed that the major drop in performance because of the removal of the BOF features was *not* due the drop in the performance of scene boundaries, counter to our initial intuition. The drop was because the recall of dia-

	000	001	010	011	100	101	110	111
Rule based	0.96	0.49	0.70	0.23	0.93	0.46	0.70	0.24
ML model	**0.99**	**0.99**	**0.98**	**0.99**	**0.97**	**0.98**	**0.98**	**0.98**

Table 2: Comparison of performance (macro-F1 measure) of our rule based baseline with our machine learning based models on development sets DEV1_000, DEV1_001, ..., DEV1_111. All models are trained on 50% of the training set, with the feature space including CONTEXT equal to 1.

Movie	000	001	010	011	100	101	110	111	MAJ	MAX	MAJ-MAX
LTC	0.87	0.83	0.79	0.94	0.91	0.86	0.79	0.96	0.97	0.95	**0.98**
X-files	0.87	0.84	0.79	0.93	0.86	0.84	0.79	0.92	0.94	0.94	**0.96**
Titanic	0.87	0.87	0.81	0.94	0.86	0.83	0.82	0.93	0.94	0.95	**0.97**
Average	0.87	0.85	0.80	0.94	0.88	0.84	0.80	0.94	0.95	0.95	**0.97**

Table 3: Macro-F1 measure for the five classes for testing on DEV2 set. 000 refers to the model trained on data TRAIN_000, 001 refers to the model trained on data TRAIN_001, and so on. MAJ, MAX, and MAJ-MAX are the three ensembles. The first column is the movie name. LTC refers to the movie "The Last Temptation of Christ."

Feature set	LTC	X-files	Titanic
All	0.98	0.96	0.97
- BOW	**0.94**	**0.92**	**0.94**
- BOP	0.98	**0.97**	0.97
- BOT	**0.97**	**0.95**	**0.96**
- BOF	**0.96**	**0.93**	**0.96**
- POS	0.98	0.96	**0.95**
- HAND	**0.94**	**0.93**	**0.93**

Table 4: Performance of MAJ-MAX classifier with feature removal. Statistically significant differences are in bold.

	Baseline			MAJ-MAX		
Tag	P	R	F1	P	R	F1
S	0.27	1.00	0.43	0.99	1.00	**0.99**
N	0.21	0.06	0.09	0.88	0.95	**0.91**
C	0.89	1.00	0.94	1	0.92	**0.96**
D	0.99	0.94	0.96	0.98	0.998	**0.99**
M	0.68	0.94	0.79	0.94	0.997	**0.97**
Avg	0.61	0.79	0.69	0.96	0.97	**0.96**

Table 5: Performance comparison of our rule based baseline with our best machine learning model on the five classes.

	$\mathcal{N}_B$	$\mathcal{N}_{\text{MAJ-MAX}}$	$\mathcal{N}_G$
# Nodes	202	37	41
# Links	1252	331	377
Density	0.036	0.276	0.255

Table 6: A comparison of network statistics for the three networks extracted from the movie *Silver Linings Playbook*.

logues decreases significantly. The BOF features were helping in disambiguating between the metadata, which usually have no frames associated with them, and dialogues. Removing bag of punctuation (BOP) results in a significant increase in the performance for the movie *X-files*, with a small increase for other two movies. We remove this feature from our final classifier. Removing parts of speech (POS) results in a significant drop in the overall performance for the movie *Titanic*. Error analysis revealed that the drop in performance here was in fact due the drop in performance of scene boundaries. Scene boundaries almost always have 100% nouns and the POS features help in capturing this characteristic distribution indicative of scene boundaries. Removing bag of terminology (BOT) results in a significant drop in the overall performance of all movies. Our results also show that though the drop in performance for some feature sets is larger than the others, it is the conjunction of all features that is responsible for a high F1-measure.

8.3 Performance on the test set

Table 5 shows a comparison of the performance of our rule based baseline with our best machine learning based model on our test set, TEST. The results show that our machine learning based models outperform the baseline with a large and sig-

Model	Degree	Weighted Degree	Closeness	Betweenness	PageRank	Eigen
$\mathcal{N}_B$	0.919	0.986	0.913	0.964	0.953	0.806
$\mathcal{N}_{\text{MAJ-MAX}}$	**0.997**	**0.997**	**0.997**	**0.997**	**0.998**	**0.992**

Table 7: A comparison of Pearson's correlation coefficients of various centrality measures for $\mathcal{N}_B$ and $\mathcal{N}_{\text{MAJ-MAX}}$ with $\mathcal{N}_G$.

nificant margin on all five classes (0.96 versus 0.69 macro-F1 measure respectively). Note, as expected, the recall of the baseline is generally high, while the precision is low. Moreover, for this test set, the baseline performs relatively well on tagging character names and dialogues. However, we believe that the performance of the baseline is unpredictable. It may get *lucky* on screenplays that are well-structured (in one way or the other), but it is hard to comment on the robustness of its performance. On the contrary, our ensemble is robust, hedging its bets on eight models, which are trained to handle different types and combinations of anomalies.

In tables 6 and 7, we present an extrinsic evaluation on the test set. We extract a network from our test movie screenplay (*Silver Linings Playbook*) by using the tags of the screenplay as follows (Weng et al., 2009): we connect all characters having a dialogue with each other in a scene with links. Nodes in this network are characters, and links between two characters signal their participation in the same scene. We form three such networks: 1) based on the gold tags ($\mathcal{N}_G$), 2) based on the tags predicted by MAJ-MAX ($\mathcal{N}_{\text{MAJ-MAX}}$), and 3) based on the tags predicted by our baseline ($\mathcal{N}_B$). Table 6 compares the number of nodes, number of links, and graph density of the three networks. It is clear from the table that the network extracted by using the tags predicted by MAJ-MAX is *closer* to the gold network.

Centrality measures are one of the most fundamental social network analysis metrics used by social scientists (Wasserman and Faust, 1994). Table 7 presents a comparison of Pearson's correlation coefficient for various centrality measures for $\{\mathcal{N}_B, \mathcal{N}_G\}$, and $\{\mathcal{N}_{\text{MAJ-MAX}}, \mathcal{N}_G\}$ for the top ten characters in the movie. The table shows that across all these measures, the statistics obtained using the network $\mathcal{N}_{\text{MAJ-MAX}}$ are significantly more correlated to the gold network ($\mathcal{N}_G$), as compared the the baseline network ($\mathcal{N}_B$).

9 Conclusion and Future Work

In this paper, we presented a formalization of the task of parsing movie screenplays. We presented an NLP and ML based approach to the task, and showed that this approach outperforms the regular expression based approach by a large and significant margin. One of the main challenges we faced early on was the absence of training and test data. We proposed a methodology for learning to handle anomalies in the structure of screenplays without requiring additional annotations. We believe that the machine learning approach proposed in this paper is general, and may be used for parsing noisy documents outside of the context of movie screenplays.

In the future, we will apply our approach to parse other semi-structured sources of social networks such as television show series and theatrical plays.

Acknowledgments

We would like to thank anonymous reviewers for their useful comments. We would also like to thank Caronae Howell for her insightful comments. Agarwal was funded by IBM Ph.D. fellowship 2013-2014. This paper is based upon work supported in part by the DARPA DEFT Program. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Apoorv Agarwal and Owen Rambow. 2010. Automatic detection and classification of social events. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1024–1034, Cambridge, MA, October. Association for Computational Linguistics.

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013a. Automatic extraction of social networks from literary text: A case study on alice in wonderland. *In the Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)*.

Apoorv Agarwal, Anup Kotalwar, Jiehan Zheng, and Owen Rambow. 2013b. Sinnet: Social interaction network extractor from text. In *Sixth International Joint Conference on Natural Language Processing*, page 33.

C. Baker, C. Fillmore, and J. Lowe. 1998. The berkeley framenet project. *Proceedings of the 17th international conference on Computational linguistics*, 1.

Desai Chen, Nathan Schneider, Dipanjan Das, and Noah A. Smith. 2010. Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 264–267, Uppsala, Sweden, July. Association for Computational Linguistics.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting social networks from literary fiction. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147.

Sebastian Gil, Laney Kuenzel, and Suen Caroline. 2011. Extraction and analysis of character interaction networks from plays and movies. Technical report, Stanford University.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *SIGKDD Explorations*, 11.

Hua He, Denilson Barbosa, and Grzegorz Kondrak. 2013. Identification of speakers in novels. *The 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. *In Proceedings of HLT-NAACL*.

Robert Turetsky and Nevenka Dimitrova. 2004. Screenplay alignment for closed-system speaker identification and analysis of feature films. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, volume 3, pages 1659–1662. IEEE.

Stanley Wasserman and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. New York: Cambridge University Press.

Chung-Yi Weng, Wei-Ta Chu, and Ja-Ling Wu. 2006. Movie analysis based on roles' social network. *In Proceedings of IEEE Int. Conference Multimedia and Expo.*, pages 1403–1406.

Chung-Yi Weng, Wei-Ta Chu, and Ja-Ling Wu. 2007. Rolenet: treat a movie as a small society. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 51–60. ACM.

Chung-Yi Weng, Wei-Ta Chu, and Ja-Ling Wu. 2009. Rolenet: Movie analysis from the perspective of social networks. *Multimedia, IEEE Transactions on*, 11(2):256–271.