Predicting Proppian Narrative Functions from Stories in Natural Language

Josep Valls-Vargas,[1] Jichen Zhu,[2] and Santiago Ontañón[1]
[1]Computer Science, [2]Digital Media
Drexel University
Philadelphia, PA 19104
josep.vallsvargas@drexel.edu, jichen.zhu@drexel.edu, santi@cs.drexel.edu

Abstract

Computational narrative systems usually require knowledge about the story world and narrative theory to be encoded in some form of structured knowledge representation formalism, a notoriously time-consuming task requiring expertise in both storytelling and knowledge engineering. In this paper we present an approach that combines supervised machine learning with narrative domain knowledge toward automatically extracting such knowledge from natural language stories, focusing specifically on predicting Proppian narrative functions. Our experiments on a dataset of Russian fairy tales show that our system outperforms an informed baseline and that combining top-down narrative theory and bottom-up statistical models inferred from an annotated dataset increases prediction accuracy with respect to using them in isolation.

Introduction

Computational narrative systems such as story generation (Gervás et al. 2005) and drama management (Fairclough 2007) usually require knowledge about the story world and narrative theory to be encoded in some form of structured knowledge (Meehan 1977; Riedl 2004; Zhu and Ontanón 2010). Despite efforts towards computer assisted tools (Poulakos et al. 2015; Kapadia et al. 2015), this knowledge is mostly hand-authored in a time-consuming task requiring expertise in storytelling and knowledge engineering.

This paper focuses on automatically extracting narrative information from stories written in natural language, and specifically, extracting Proppian *narrative functions* (Propp 1973). We propose an approach that combines supervised machine learning, domain knowledge and probabilistic inference to predict Propp's narrative functions for a given segment of text from a story written in natural language. Our experiments show that our approach outperforms an informed baseline and that combining top-down narrative theory and bottom-up statistical models inferred from an annotated dataset increases prediction accuracy with respect to using them in isolation.

In the rest of the paper, we first introduce Proppian functions. Then we describe the problem of narrative function prediction and present our approach. Next we report our experimental evaluation on a dataset of 15 Russian fairy tales drawn from Propp's work. The paper closes with a discussion of related work, conclusions and future work.

Proppian Narrative Functions

In the context of narratology, a narrative function is a fundamental building block of storytelling: *"an act defined in terms of its significance for the course of the action in which it appears; an act considered in terms of the role it plays at the action level"* (Prince 2003). Specifically, in this paper, we draw from Vladimir Propp's theory of narrative functions described in his Morphology of the Folktale (Propp 1973).

Propp (1973) described a series of narrative functions which he claimed represent canonical and invariant acts that constitute the underlying structure of any Russian fairy tale. Furthermore, Propp's theses state that the sequence of functions in a fairy tale is constant and there are explicit interdependencies between the functions that appear in a particular fairy tale. Propp's work ultimately reduces a fairy tale to a sequence of variations of his functions (which he called subfunctions) represented using a formal language.

Even though Propp gave definitions of his functions and subfunctions, identifying these in a given text is a non-trivial task even for trained annotators (Finlayson 2012). Referring to a generative use of his work, Propp wrote (Propp 1973, pp. 111–112):

> *In order to create a fairy tale artificially, one may take any A, then one of the possible B's then a C↑, followed by absolutely any D, then an E, the one of the possible F's, then any G, and so on. In doing this, any elements may be dropped, or repeated three times, or repeated in various forms. [...] The storyteller is constrained [...] in the overall sequence of functions, the series of which develops according to the above indicated scheme. [...] The storyteller is free [...] in the choice of those functions which he omits, or, conversely, which he uses and in the choice of the means (form) through which a function is realized. [...]*

where A, B, C, ↑, D, F, and G refer to some of the narrative functions defined in his work. From an annotator or recognizer's perspective, the freedom of choice of the means of realizing a narrative function makes the identification process a nontrivial task. Furthermore, functions might not be explicitly realized in the text, but can be left implicit by contextual cues, i.e., a human reader might infer that events happen in the story corresponding to a function, even if they are never explicitly mentioned.

Predicting Proppian Narrative Functions

Our long-term goal is to automatically extract narrative information from stories written in natural language in order to alleviate the "authorial bottleneck" problem in computational narrative systems. Exploiting local information, commonsense and domain knowledge, in our previous work (Valls-Vargas, Ontañón, and Zhu 2015), we presented a system (*Voz*) that given a story written in natural language, is capable of extracting characters with an accuracy of over 90% and identifying their Proppian narrative roles with an accuracy of about 65%, in addition to also extracting actions (i.e., verbs) and identifying their participants. In this paper we extend *Voz* to also identify Proppian narrative functions, a significant step towards our long-term goal.

Problem Statement

Given a story S written in natural language and a finite set of narrative functions $\mathbb{F}$, the problem we address in this paper is to predict the sequence of narrative functions $[f_1, ..., f_n]$ ($\forall_{i=1...n} f_i \in \mathbb{F}$) that describes the story and the sequence of contiguous, non-overlapping, potentially empty segments of text $[s_i, ..., s_n]$ where the narrative functions are realized.

Moreover, in this paper we make one simplification assumption, and start with a story S that has been manually divided into text segments where narrative functions are realized. Then, our goal is to identify the narrative function present in each of the text segments. Therefore, given a story S divided into a sequence of unannotated text segments $[s_1, ..., s_n]$ in natural language, associate each segment s_i with a function $f_i \in \mathbb{F}$.

In order to address this problem, we present an approach that integrates supervised learning, narrative domain knowledge and probabilistic inference, described below. In this work, we use a set $\mathbb{F}$ of 34 narrative functions derived from Propp's original collection of narrative functions. Table 1 enumerates the 34 narrative functions in $\mathbb{F}$ and the number of instances in the dataset used in our experiments.

Technical Approach

Our approach requires the existence of a dataset to train the machine learning components of the system. Specifically, the dataset contains a set of stories $S_1, ..., S_n$ manually annotated with ground truth. Each story S_i in the dataset is manually annotated as follows:

- S_i is divided into a sequence of contiguous, non-overlapping, potentially empty text segments $[s^i_1, ..., s^i_{m_i}]$ where narrative functions are realized (i.e., if a part of a story does not realize any function, then that part might not be present in any of the text segments).
- Each text segment s^i_j is automatically converted to a feature vector x^i_j, and manually associated to a narrative function $f^i_j \in \mathbb{F}$ (a description of the annotations and features in the feature vector is provided in the experimental evaluation section).

This results in two training datasets:

- A text segment to narrative function mapping dataset:

$$D_f = \{\langle x_1, f_1\rangle, ..., \langle x_N, f_N\rangle\}$$

- A function sequences dataset:

$$D_s = \{[f^1_1, ..., f^1_{m_1}], ..., [f^n_1, ..., f^n_{m_n}]\}$$

Table 1: Enumeration of the narrative functions used in our system ($|\mathbb{F}| = 34$), the symbols Propp used to refer to them, and the total number of instances in our dataset. The third column is the total number of instances annotated, the fourth column, the number of those that are actually realized in the text (i.e., explicit) and the fifth column the number of instances once the direct speech (e.g., dialog) is filtered out.

Function	*Symbol*	*# Instances*		
Initial Situation	α	13	13	12
Absentation	β	4	2	2
Interdiction	γ	3	3	1
Violation	δ	2	2	2
Reconnaissance	ϵ	0	0	0
Delivery	ζ	1	1	1
Trickery	η	2	2	2
Complicity	θ	1	1	1
Deceit (merged with θ)	λ	0	0	0
Villainy	A	13	11	11
Lack	a	3	1	1
Mediation, Connective Incident	B	7	6	3
Beginning Counteraction	C	11	6	2
Departure	$\uparrow$	14	14	14
First Function of Donor	D/d	8	8	3
Protagonist's Reaction	E	8	8	7
Acquisition of Magical Agent	F/f	12	11	6
Transference, Guidance	G	5	5	3
Struggle	H	9	7	6
Branding	J	0	0	0
Victory	I	12	11	10
Liquidation	K	13	11	9
Return	$\downarrow$	14	11	9
Pursuit	Pr	8	7	5
Rescue (from Pursuit)	Rs	8	8	8
Unrecognized Arrival	o	2	2	1
Unfounded Claims	L	0	0	0
Difficult Task	M	0	0	0
Solution	N	0	0	0
Recognition	Q	2	2	1
Exposure	Ex	1	1	1
Transfiguration	T	3	2	1
Punishment	U	1	1	1
Wedding	W/w	10	10	7
Total		190	167	130

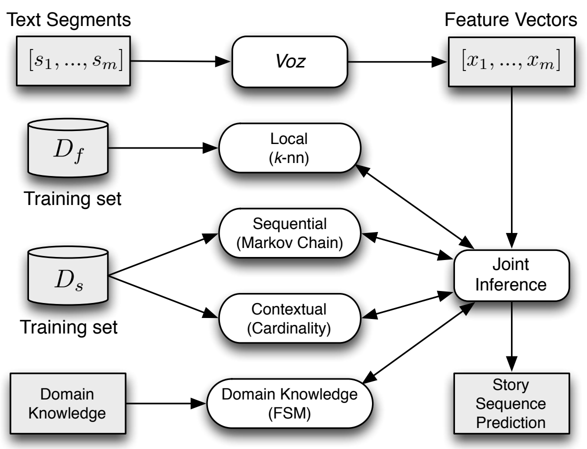

Figure 1: Overall system diagram. Both databases indicate automatically learned training sets from our annotated dataset. *Voz* is our previous narrative information extraction system which we use to automatically compute feature vectors from text segments in a given story. The output is a sequence of narrative function predictions such as: $\langle \alpha, \beta, \delta, A, B, C, \uparrow, H, I, K, \downarrow, W \rangle$

This second dataset ignores the feature vectors and contains narrative function sequences representing each of the stories in the original dataset.

D_f and D_s are used to train our system. At run time, in order to predict the functions associated with a given sequence of text segments $[s_1, ..., s_m]$, we employ our system *Voz*[1] to automatically extract the necessary narrative information (characters, narrative roles, and actions) from the natural language text and translate each text segment s_i into a feature vector x_i which is then used for prediction.

Our approach is based on a search process over the space of possible function sequences, returning the sequence with highest probability as predicted by a set of predictors and aggregated by a module we call *joint inference*. In the following sections we describe our proposed probabilistic predictors and the search process in the *joint inference* module. Figure 1 illustrates the overall workflow of the system.

Predictors

Local Predictor (k-nn). Our first predictor is a supervised machine learning predictor that uses the D_f training dataset mapping feature vectors to narrative functions. Given a feature vector x representing a text segment, it uses the euclidean distance in the feature vector space to retrieve the k nearest neighbors ($k = 5$ in our experiments) from the D_f training dataset. Then, using the subset of retrieved pairs $R_x = \{\langle x_1, f_1 \rangle, ..., \langle x_5, f_5 \rangle\} \subseteq D_f$, it computes a probability distribution of the likelihoods of each narrative function class $f \in \mathbb{F}$ to be the function appearing in the text segment represented by x. A Laplacian smoothing (with a pseudocount $a = 0.1$ in our experiments) is applied to the probability distribution. So, specifically, the probability that the local predictor assigns to a function f is:

$$P_{k\text{-nn}}(f|x) = \frac{|\{\langle x_i, f_i \rangle \in R_x | f_i = f\}| + a}{|R_x| + a|\mathbb{F}|}$$

Using these probabilities, the *joint inference* module can compute the likelihood of a sequence of functions $[f_1, ..., f_m]$ given the sequence of feature vectors $[x_1, ..., x_m]$ as:

$$P_{k\text{-nn}}([f_1, ..., f_m]|[x_1, ..., x_m]) = \prod_{i=1...m} P_{k\text{-nn}}(f_i|x_i)$$

Sequential Predictor (Markov Chain). Our second predictor uses sequential information and the D_s training dataset containing function sequences. First, the D_s training dataset is used to automatically learn a Markov Chain model of narrative function transition probabilities $P(f_i|f_{i-1})$. An extra sentinel f_0 with a special narrative function class $\perp$ is prepended to the function sequences in D_s to mark the beginning of the sequences. This predictor assesses the likelihood of a sequence of functions $[f_1, ..., f_m]$ as:

$$P_{MC}([f_1, ..., f_m]) = \prod_{i=1...m} P(f_i|f_{i-1})$$

A Laplacian smoothing (with a pseudocount $a = 0.1$ in our experiments) is used to estimate the conditional probability distribution $P(f_i|f_{i-1})$ from the training set D_s.

Contextual Predictor (Cardinality). Our third predictor uses the D_s training dataset containing function sequences to estimate the likelihood of a sequence of functions by considering the frequency with which each function appears in the sequence. First, the D_s training dataset is used to count the number of sequences in the dataset in which a given function class f appears exactly n times. From the counts $C_{f,n}$, it estimates, using Laplacian smoothing (with a pseudocount $a = 0.1$ in our experiments), the probability $P_C(f, n)$ of a given function to appear a certain number of times in a story. Given a function sequence $F = [f_1, ..., f_n]$, its likelihood is then assessed as:

$$P_C(F) = \prod_{f \in \mathbb{F}} P_C(f, count(f, F))$$

where $count(f, F)$ is the number of times f appears in F.

Domain Knowledge Predictor (FSA). This last predictor uses domain knowledge and encodes our interpretation of the rules in Propp's narrative theory, such as those outlined in the excerpt earlier in this paper (Propp 1973, pp. 111–112). A subset of the rules in Propp's theory was manually encoded in a probabilistic finite state machine, namely:

- The precedence relationships defined by Propp's ordering may not be violated. Table 1 enumerates the functions in the expected order.
- Each function may only appear once (our dataset only includes stories with a single *move*, as described below).

[1]For source code and datasets, visit: https://sites.google.com/site/josepvalls/home/voz

- The first function must be one of the introductory functions ($\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \lambda$), villainy ($A$) or lack ($a$).
- Return ($\downarrow$) should not appear without prior departure ($\uparrow$).
- Rescue (Rs) should not appear without prior pursuit (Pr).

At run time, given a sequence of functions $[f_1, ..., f_n]$, this predictor works as a finite state recognizer. Starting at a sentinel start state, the predictor tries to consume each function the sequence. If the consumed prediction matches a *valid* function transition, it updates the internal state of the finite state machine, otherwise each state has a special *unrecognized* function transition that goes back to itself for functions that do not match any valid transition. Each transition is labeled with a probability, and the probability of a sequence is computed as the product of the probabilities of all the transitions that were fired when consuming the sequence. The probability associated with a transition t coming out of a state i is defined as:

$$P(t,i) = \begin{cases} \frac{1+a}{n_i + a|\mathbb{F}|} & \text{if } t \text{ is a } \textit{valid} \text{ transition at state } i \\ \frac{0+a}{n_i + a|\mathbb{F}|} & \text{for } \textit{unrecognized} \text{ function transitions} \end{cases}$$

where n_i is the number of *valid* function transitions coming out of state i. These probabilities correspond to a Laplacian smoothing (with $a = 0.1$) assuming we observe each of the *valid* transitions identified in Propp's theory once, and zero times all the *unrecognized* ones.

Search

In order to explore the space of sequences of narrative functions for an input sequence $[x_1, ..., x_m]$, systematic search is unfeasible given the size of the search space ($|\mathbb{F}|^m$). In our experiments, we report our results using beam search. The beam search algorithm searches the space of possible sequences, searching for the sequence with the highest probability given the predictions provided by the four predictors. The likelihoods returned by each predictor are multiplied together to obtain the joint likelihood for a given sequence.

Beam search evaluates the nodes at depth i using the four predictors (except the contextual predictor that can only be used for complete sequences, i.e., only for the leaf nodes of the search tree), and discards all but the top N candidates (where N is the size of the beam, set to $10,000$ in our experiments) before expanding the next level of depth $i+1$. The assignment with highest likelihood at the end is returned as the sequence of narrative functions for the input sequence of feature vectors $[x_1, ..., x_m]$ representing a story S.

Notice that each of the four predictors was designed to capture a different aspect of the prediction task: the local predictor predicts functions based on the content of the text segments, the sequential and contextual predictors make predictions based on function sequences alone (without taking into account the input), and the final domain knowledge predictor exploits narrative domain knowledge.

Experimental Evaluation

In this section we first describe our dataset and the features describing text segments used by our local predictor. Then we describe our experimental setup and results.

Dataset

For our experimental evaluation we use a dataset of 15 Russian fairy tales collected and annotated by Mark Finlayson (2009). The dataset contains 23,291 tokens (words and punctuation) and includes annotations for referring expressions, coreference, verbs, semantic roles, narrative roles and narrative functions. There is a total of 190 annotated narrative functions. In this work we ignore narrative functions that are implicit or don't have an explicit text realization. The dataset was manually filtered to include text segments where narrative functions are realized. The experiments reported in this paper use 167 text segments corresponding to the explicit functions covering a total of 3,915 tokens. The sequence of segments is not altered in any other manner and it's processed in the order it is mentioned in the text of the story. All the stories in our dataset were identified by Propp as single *move* (a higher-level narrative structure defined by Propp that contains narrative functions).

We report results with 3 different scenarios:

- *Automatic*: In this scenario, the input to the system are the sequences of text segments in natural language without any further annotation. Our system uses *Voz* to automatically process the text and build the feature vectors. *Voz* is not currently able to process direct speech (e.g., dialog) and thus narrative functions that appear solely in it are removed from the dataset before the experiments. The dataset used to run experiments in this scenario contains 130 text segments which span over 2,342 tokens.
- *Annotated*: This scenario, instead of using *Voz* to generate the feature vectors, they are computed directly from the dataset annotations contained within each text segment and includes the 167 text segments. This scenario provides an upper bound that ignores any error introduced by *Voz* due to inaccuracies in the natural language processing and information extraction.
- *Filtered Annotated*: This scenario replicates the *Annotated* scenario, except that removes text segments contained within direct speech (e.g., dialog) in order to provide results comparable to the *Automatic* scenario (i.e., using the annotations from the filtered 130 text segments).

Table 1 lists the functions and their instance counts in our dataset. The third column reports the total number of instances annotated (including implicit narrative functions), the fourth column reports the number of those instances that are actually realized in the text (i.e., included in the *Annotated* scenario) and the fifth column the number of instances once the direct speech (e.g., dialog) is filtered out (included in the *Automatic* and *Filtered Annotated* scenarios).

Feature Vectors

The main contributor to our joint inference process is the supervised machine learning local predictor that uses feature vectors to represent text segments. A feature vector x_i is composed of 3 types of features:

- *Positional feature*: one feature with value in $[0.0, 1.0]$ representing the position relative to the story where a given segment appears, 0.0 being the beginning of the story and 1.0 the end of the story.

- *Proppian role features*: there are 7 features of this type with value in $[0.0, 1.0]$. The first 6 correspond to Propp's narrative roles that can be automatically identified by *Voz* (*Hero*, *Villain*, *Sought-for-person*, *False Hero*, *Donor+Magical Helper*, and *Other* that includes the dispatcher, family member and other minor roles). Additionally an extra feature is used to account for mentions to *Non-characters* (e.g., locations, time expressions, non-animated objects, etc.). Given a segment of text, *Voz* extracts all the referring expressions (i.e. entities), and maps them to their narrative roles. The 7 features represent the distribution of extracted roles.
- *Acts and actions*: there are currently 10 binary features of this type, representing acts and actions and are primarily identified by verbs that appear in the segment of text. The features are computed as follows: For each verb in the text, *Voz* finds the most likely semantic frame in Framenet (Baker, Fillmore, and Lowe 1998). This generates 186 binary features indicating whether different frames appear in the text segment or not. Then, automatic feature selection (using correlation coefficient between features and narrative function labels) is used with the training set to select the top 10 of these features and discard the rest.

Experimental Setup

For our experimental evaluation we followed the leave-one-story-out protocol. For each story S_i in the dataset, we construct the training sets D_f and D_s with the remaining 14 stories in order to train the system. Then we apply the joint inference methodology described to compute a sequence of predictions $[f_1, ..., f_m]$ for the story S_i at hand. We compare the sequence of predictions element-wise with the annotations in the dataset and we report the average accuracy weighted by the length m (in functions) of each story.

The number of neighbors k in k-nn, the neighbor distance metric, the value a of the Laplacian smoothing pseudocount, the number of features to include during selection, the size of the beam and the verb grouping were set experimentally.

We performed an exhaustive analysis of the different combinations of the predictors described earlier. For simplicity we report a representative subset of the experimental results. Moreover, we tested using the FSA domain knowledge predictor in two different ways: 1) using it during the search process as all other predictors, and 2) only using it once we reach the leaves of the search process. We call these two scenarios "FSA Search" and "FSA Leaves" respectively. This is done since the FSA predictor used slightly more computational resources than the others, and using it during the search increases execution time significantly.

Experimental Results

In order to validate our proposed approach and evaluate the usefulness of the individual predictors and their combination we repeated the application of our methodology to several combinations of predictors in the three different scenarios described in the dataset section. Table 2 reports the accuracies of the different combinations.

Table 2: Classification accuracy obtained in different scenarios. The first five columns describe which predictors were used. The remaining three columns report the classification accuracies for narrative function predictions in three different scenarios. The last row shows the accuracy obtained by predicting the most common function in the dataset.

k-nn	Markov Chain	Cardinality	FSA Search	FSA Leaves	Automatic	Annotated	Filtered Ann.
✓					0.202	0.21	0.217
	✓				0.194	0.12	0.194
		✓			0.155	0.114	0.155
			✓		0.054	0.036	0.054
				✓	0.047	0.036	0.047
✓	✓				0.194	0.257	0.225
✓		✓			0.194	0.275	0.209
✓				✓	0.178	0.234	0.186
✓	✓	✓			0.24	0.216	**0.287**
✓	✓			✓	0.171	0.275	0.24
✓		✓		✓	0.209	**0.281**	0.248
✓		✓	✓		0.225	**0.281**	0.202
✓	✓	✓		✓	**0.271**	0.246	0.233
Informed Baseline					0.109	0.084	0.109

To provide a reference to our results, the bottom row of Table 2 shows the accuracy obtained with an informed baseline that always predicts the most common function in our dataset (e.g., $\uparrow$ or $\downarrow$ in the *Annotated* scenario), which is 0.084 in the *Annotated* scenario and 0.109 in the other two scenarios. Accuracy is low since this is a prediction over a large set of labels ($|\mathbb{F}| = 34$, random baseline is 0.029).

The first five rows of Table 2 show the performance obtained by each of the four predictors in isolation (and also when we used the FSA during the search or only at the leaves). As we can see, when used in isolation, the k-nn local predictor achieves the best performance (unsurprising, since it is the only one that considers the actual text features to make predictions). The Markov Chain predictor achieves the second best accuracy, since it is accurately able to predict the first functions of a story (stories regularly start with the α function and are often followed by A). Even with low performance, notice the local predictor by itself doubles the performance of the informed baseline in all three scenarios.

The rest of the rows in Table 2 show the performance of our approach with an increasing number of predictors (all the way to using all four predictors, in the bottom rows). As we can see, the highest performance in all scenarios is achieved when combining at least three predictors. We do not include rows when the k-nn predictor was not used, since none of those produced good results. This indicates that integrating information from multiple sources (including Propp domain knowledge) significantly improves performance. The highest performance achieved by our system is between 0.271 and 0.287 depending on the scenario, which is close to three times that of the informed baseline.

Concerning differences across the three scenarios, if we average the performance of the bottom 8 rows in the table (corresponding to configurations with 2, 3 and 4 modules), the performance in the *Automatic* it is 0.210, in the *Annotated* scenario is 0.258, and in the *Filtered Annotated* it is 0.229. This shows that, as expected, making predictions from annotated text is more accurate. However, looking at the configurations with the highest performance (0.281 for *Filtered annotated* and 0.271 for *Automatic*), we can see that the difference is not too large, and that the performance of *Voz* is not the main bottleneck in this prediction task.

One interesting observation is that the domain knowledge predictor seems to actually hinder the performance of k-nn when only those two predictors are used together. We attribute this phenomena to the fact that many properties of the theory are violated in actual stories (specially in the *Automatic* and *Filtered Annotated* datasets where many functions are removed since they appear in dialog). However, when used in combination with the other predictors trained from D_f and D_s, this seems to be alleviated.

Looking at the *Automatic* scenario, we can see how our automatically extracted features encode some relevant information but the results are unstable and some combinations seem to actually hinder the performance. The best combination includes all four predictors and achieves an accuracy of 0.271. A close inspection of the results pointed out that overlooking relevant information in the dialogue (we found an instance where 2 consecutive lines of dialogue included 3 narrative functions) and the limited size of the dataset (130 segments) are the main causes for the reduced performance with respect to the *Annotated* scenario (167 segments).

In conclusion, these results show that our approach significantly outperforms an informed baseline, and that incorporating different types of knowledge (text information, sequence information, domain knowledge) also significantly improves performance. Although performance is far from optimal, results seem promising, and we believe a larger dataset could significantly help improve results.

Related Work

Propp's theory has served as a reference for defining grammars for small story generators (Seifert et al. 2005) and full fledged interactive narrative systems such as Fairclough's *Opiate* (Fairclough 2007). Opiate contains a Case-Based Reasoning (CBR) *story director* which builds new plots by reusing plot fragments represented in terms of Proppian functions. This system maps Propp's function to scripts and fills in characters from a taxonomy derived from Propp's narrative roles. Gervás et al. (2005) presented a story generation system also based on CBR that uses an ontology of moves, functions, characters and roles derived from Propp's theory in order to guide the plot generation process.

Beside uses in story generation, computational narrative systems using automatic and semi-automatic narrative information extraction have been used to study literature and narrative theories. Finlayson (2012) uses a semi-automatic annotation procedure and a machine learning algorithm to extract plot patterns from a set of Russian fairy tales. In his work he reports results that aim to approximate Propp's model via automatic analysis of the annotations. Elson (2010) used an automatic system to extract and study character interactions in classic literary works.

In our previous work (Valls-Vargas, Zhu, and Ontañón 2014; Valls-Vargas, Ontañón, and Zhu 2015), we proposed the use of natural language processing to automatically identify narrative information such as characters and their narrative roles, from text and alleviate the well-known "authorial bottleneck" problem. The AESOP system (Goyal, Riloff, and Daumé 2010) explored how to extract characters and their affect states from textual narratives in order to produce plot units (Lehnert 1981) for Aesop fables. Calix et al. (Calix et al. 2013) proposed an approach for detecting characters in stories based on commonsense, spoken, and textual features. More recently, Suciu et al. (Suciu and Groza 2014) proposed an approach that uses an ontology to identify characters in stories. Bamman et al. (Bamman, O'Connor, and Smith 2014) extract characters, their attributes and the actions in which they participate and propose an unsupervised approach to identify latent character archetypes or personas by clustering their stereotypical actions and attributes.

Automatically extracting high level narrative elements from text in natural language has received less attention. Systems such as the *Story Workbench* (Finlayson 2012) and *Scheherazade* (Elson 2012) employ a semi-automatic approach to text annotation. Both systems enable the annotation of high level narrative information such narrative functions. More recently, Li et al. (2013; 2015) proposed a combination of NLP and crowdsourcing to acquire narrative information. A key difference between these systems and our work is that we seek to completely automate the process.

Conclusions and Future Work

In this paper we presented a probabilistic joint inference approach for the task of identifying narrative functions for segments of stories. When predicting sequences our method achieves an accuracy of 0.281 from an annotated dataset and 0.286 when predicting directly from natural language text segments using *Voz*. These accuracies are significantly higher than a random and an informed baseline (0.109). Our experimental results confirm our initial hypotheses that combining top-down narrative theory and bottom-up statistical models inferred from an annotated dataset increases prediction accuracy with respect to using them in isolation.

In our future work we want to improve the robustness of *Voz* in order to handle full stories including dialog, which carries useful information which we are currently ignoring. The feature set used by our local predictor can also be significantly improved (for example, currently no feature captures which character roles are performing which actions, which is key to distinguish some of the functions). We would also like to improve our domain knowledge predictor in order to be able to handle stories with multiple *moves*, and to consider the information extracted from the text segments rather than relying just on function sequences alone. The current approach also relies on a known segmentation of the stories before predicting the narrative functions. We would like to experiment with expectation-maximization methods to automatically segment the stories.

References

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The Berkeley FrameNet Project. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 86–90.

Bamman, D.; O'Connor, B.; and Smith, N. A. 2014. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 352–361.

Calix, R. A.; Javadpour, L.; Khazaeli, M.; and Knapp, G. M. 2013. Automatic Detection of Nominal Entities in Speech for Enriched Content Search. *The Twenty-Sixth International FLAIRS Conference* 190–195.

Elson, D. K.; Dames, N.; and McKeown, K. R. 2010. Extracting social networks from literary fiction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 138–147. Association for Computational Linguistics.

Elson, D. K. 2012. *Modeling Narrative Discourse*. Ph.D. Dissertation, Columbia University.

Fairclough, C. R. 2007. *Story Games and the OPIATE System*. Ph.D. Dissertation, University of Dublin, Trinity College.

Finlayson, M. A. 2009. Deriving narrative morphologies via analogical story merging. In *Proceedings, 2nd International Conference on Analogy*, 127–136.

Finlayson, M. A. 2012. *Learning narrative structure from annotated folktales*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Gervás, P.; Díaz-Agudo, B.; Peinado, F.; and Hervás, R. 2005. Story Plot Generation based on CBR. *Knowledge-Based Systems* 235–242.

Goyal, A.; Riloff, E.; and Daumé, III, H. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 77–86.

Kapadia, M.; Falk, J.; Zünd, F.; Marti, M.; Sumner, R. W.; and Gross, M. 2015. Computer-assisted authoring of interactive narratives. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, 85–92. ACM.

Lehnert, W. G. 1981. Plot units and narrative summarization. *Cognitive Science* 5(4):293–331.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the 27th AAAI Conferece on Artificial Intelligence*, 598–604.

Li, B. 2015. *Learning Knowledge to Support Domain-Independent Narrative Intelligence*. Ph.D. Dissertation, Georgia Institute of Technology.

Meehan, J. R. 1977. Tale-spin, an interactive program that writes stories. In *IJCAI*, volume 77, 91–98.

Poulakos, S.; Kapadia, M.; Schüpfer, A.; Zünd, F.; Sumner, R. W.; and Gross, M. 2015. Towards an accessible interface for story world building. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference, Workshop on Intelligent Narrative Technologies*.

Prince, G. 2003. *A dictionary of narratology*. University of Nebraska Press.

Propp, V. 1973. *Morphology of the Folktale*. University of Texas Press.

Riedl, M. O. 2004. *Narrative generation: balancing plot and character*. North Carolina State University.

Seifert, L.; Lim, C.; Tan, L.; and Wee, N. 2005. Digital propp: Proppian fairy tale generator.

Suciu, D., and Groza, A. 2014. Interleaving ontology-based reasoning and natural language processing for character identification in folktales. In *IEEE Tenth International Conference on Intelligent Computer Communication and Processing*, 67–74.

Valls-Vargas, J.; Ontañón, S.; and Zhu, J. 2015. Narrative hermeneutic circle: Improving character role identification from natural language text via feedback loops. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2517–2523.

Valls-Vargas, J.; Zhu, J.; and Ontañón, S. 2014. Toward automatic role identification in unannotated folk tales. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 188–194.

Zhu, J., and Ontanón, S. 2010. Story representation in analogy-based story generation in riu. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence in Games*, 435–442. IEEE.