Mark Alan Finlayson

Inferring Propp's Functions from Semantically Annotated Text

Vladimir Propp's Morphology of the Folktale *is a seminal work in folkloristics and a compelling subject of computational study. I demonstrate a technique for learning Propp's functions from semantically annotated text. Fifteen folktales from Propp's corpus were annotated for semantic roles, co-reference, temporal structure, event sentiment, and dramatis personae. I derived a set of merge rules from descriptions given by Propp. These rules, when coupled with a modified version of the model merging learning framework, reproduce Propp's functions well. Three important function groups—namely A/a (villainy/lack), H/I (struggle and victory), and W (reward)—are identified with high accuracies. This is the first demonstration of a computational system learning a real theory of narrative structure.*

Keywords
AFS ETHNOGRAPHIC THESAURUS: Artificial intelligence, computational linguistics, folklore, functionalism, semantics

Introduction

Vladimir Propp's *Morphology of the Folktale*, published in 1928 and first translated into English in 1958 (Propp [1928] 1968), is a seminal work in folkloristics, having ushered in an era of structuralism, provided a template for later studies of the narrative structure of folklore, and inspired generations of folklorists. One of the most precise formulations of narrative structure to date, Propp's morphology presents a compelling subject of machine learning. It would be of wide-ranging interest if a morphology could be automatically and reliably extracted from a given set of folktales. For folklorists and literary theorists, such a tool would be invaluable for comparison, indexing, and classification. For cultural anthropologists, it would provide a new technique for studying culture and its variations across time and space. For cultural psychologists, it would point the way to new experiments for investigating culture and its impact on thought. For cognitive scientists, it would serve as a model of understanding abstractions from texts, and the nature of narrative understanding. For computational linguists, it would be a step toward understanding the higher-level meaning of natural

MARK ALAN FINLAYSON is Assistant Professor of Computer Science, Florida International University

Journal of American Folklore 129(511):55–77

language. And for researchers in artificial intelligence and machine learning, it would represent an advance in our ability to extract deep structure from complex datasets. Each of these fields would naturally also find advances in the others of interest.

Unfortunately, the extraction of morphologies has until now remained a manual task, the purview of scholars such as A. J. Greimas, Claude Lévi-Strauss, Alan Dundes, and, of course, Vladimir Propp. Constructing a morphology for a particular set of folktales takes many years of reading and analysis. It is unclear how much the morphology, once complete, owes to the folklorist's personal biases or familiarity with other extant morphologies, rather than being a true reflection of the character of the tales under investigation. Furthermore, blind reproduction or validation of a morphological analysis is a prohibitively difficult endeavor, requiring a scholar with the necessary skills to retrace the years-long paths of reading, analysis, and synthesis required to generate a morphology by hand.

I demonstrate a technique that gives computational purchase on the problem of identifying a morphology from a given set of stories. The algorithm is a modification of a machine learning technique called model merging (Stolcke and Omohundro 1994), and uses a set of rules derived from Propp's descriptions of his own process for finding similarities between tales. In this technique, the algorithm runs over semantically annotated texts as data, folktales whose surface semantics have been encoded in a computer-readable representation. For this particular demonstration, the data are a selection of single-move Russian fairy tales analyzed by Propp, and translated into English. Importantly, the encoding of the surface semantics of the texts is human-assisted; the actual learning of the identities of Propp's function is done by computer.

The paper is organized as follows: First, I explain the machine learning problem at hand, pointing out those parts of Propp's theory that I will target for learning. Second, I describe the structure of the learning technique used, and how it differs from regular model merging. Third, I describe the data used in the experiment, including the texts, the semantic annotation schemes, and the gold standard data (Propp's analyses) against which the performance of the algorithm was measured. Fourth, I lay out the set of merge rules, derived from Propp's descriptions, that work within the model-merging framework to reproduce a substantial portion of Propp's functions. Finally, I describe the performance of the algorithm in extracting the identities of Propp's functions.

Learning Target

Propp's morphology comprises a set of character categories plus three levels of plot structure: gross structure (*moves*), intermediate structure (*functions*), and fine structure (what I will here call *subtypes*: Propp himself had no specific term for them). The character categories are called *dramatis personae*, of which Propp identified seven: Hero, Villain, Princess, Dispatcher, Donor, Helper, and False Hero. A move, made up of functions, is a rudimentary tale, and a normal tale is made up of one or more moves, possibly intertwined in complex ways. A function is a plot element that is "an act of a character, defined from the point of view of its significance for the course of the action" (Propp [1928] 1968:21). Each function falls into a major type determined by its position in the move, purpose for the plot, and the dramatis personae involved. Propp identifies 31 different functions. Each function defines what was happening, but

does not necessarily specify how it happened—that is, functions can be instantiated in many different ways, which I call the *subtype* of the function.

Propp's plot structure defines a grammar, in the formal, computational sense of the word. In this work, I endeavor to learn some part of that grammar from the text itself. As we know from work in grammatical inference (Higuera 2010), the power of a grammar influences how difficult that grammar is to learn. So how powerful is Propp's grammar?

Propp defines the top-level structure of a tale to be an optional preparatory sequence, followed by some number of moves, possibly intermingled. The grammatical complexity of this level is at least context-free, an observation consistent with Lakoff's analysis (1972), and a fairly powerful grammar. The intermediate level, where functions occur in a restricted order, is a regular grammar, which is less powerful than a context-free grammar, and thus easier to learn. The subtype level can have long-distance effects within the story arc, in that the choice of a particular subtype early on in the tale (e.g., instantiating A, the Villainy, as a kidnapping) forces the choice of a particular subtype later on (e.g., instantiating K, the Resolution, as a rescue of the kidnapped person). This subtype effect adds additional complexity but can be incorporated into the function-level regular grammar (or move-level context-free grammar) in the form of a feature grammar (Goodman 2000) or generalized phrase structure grammar (Gazdar, Pullum, and Sag 1985). Thus, leaving aside the dramatis personae, the overall grammatical complexity of Propp's theory is at least that of a generalized phrase structure grammar (GPSG), which is complex indeed.

We currently do not have the computational technology to learn the alphabet and transitions of Propp's GPSG simultaneously with the categories of dramatis personae. Even if the dramatis personae are given, learning the GPSG is currently too hard. I focus here, therefore, on learning only the identities of Propp's functions, noting that the function classes can perhaps be considered Propp's most important contribution. Almost everything else is defined in reference to functions: moves are complexes of functions, and subtypes are a modulation on functions. The dramatis personae are defined, in part, by the functions in which those characters participate. The majority of folkloristic and computational work that has built upon Propp has focused on the function level (e.g., Dundes 1964; Colby 1973; Díaz-Agudo, Gervás, and Peinado 2004; Halpin, Moore, and Robertson 2004).

I will leave the following for future work learning: the dramatis personae categories, the function subtype categories, the move-level grammar, and the transition structure of the function-level regular grammar. My concern here will only be with learning the function categories, which is the equivalent of learning only the alphabet of the regular grammar of the function level. Learning regular grammars (with a known alphabet) is a problem on which there has been some traction, and I leveraged that work to construct a new algorithm for learning the alphabet of the regular grammar.

Learning Technique

Model merging is an automated technique for learning regular grammars from positive examples (Omohundro 1992; Stolcke and Omohundro 1994) and is the conceptual foundation of my approach. My technique takes model merging and adds two key

augmentations. First, whereas model merging assumes the grammatical alphabet is known, one of the major challenges when learning Propp's morphology is to learn identities of the functions themselves. To achieve this, I start with an extremely large possible alphabet and incorporate a filtering stage at the end for identifying the actual alphabet from the final model. Second, whereas model merging considers model states to be relatively atomic, with only a probability distribution over the symbols emitted by model states, my technique considers a rich internal structure of each model state (derived from the semantic annotations on the text) when making its merge decisions.

Model merging may be used to derive a regular grammar from a set of positive examples. Consider the set of two characters sequences {ab, abab}. What is the pattern that most concisely describes these two sequences? One guess is the regular grammar (ab|abab), or rather, either the first string or the second. Our intuition, however, is that this guess is unsatisfactory because it does not generalize beyond the examples provided. Anyone can see that a more plausible guess is the substring ab repeated one or more times, or, written as a regular expression, (ab)+. Model merging is a framework that allows us to find a good approximation to this pattern; all we need is a method for searching the space of possible grammars.

Model merging follows the grammar inference paradigm that starts with a model constructed to accept the finite language composed of exactly the positive examples observed (Young-Lai 2009). One achieves generalization by applying a merge operation over states in the model, where two states are removed from the model and replaced with a single state that inherits the removed states' transitions and emissions. This merge operation induces a large space of models to search.

Figure 1 illustrates my technique by showing the extraction of a simple morphology from two extremely short stories, written to illustrate the technique. The first story is about an old man and maidservant: they meet on the road, he chases her, she runs away, and it ends with her thinking he is an ugly man. The second story is about a dragon and a princess: the dragon stalks the princess, which scares her, so she flees into hiding, and it ends with her deciding the dragon is an evil creature. At some level of abstraction, these two stories are similar. The chasing and stalking events

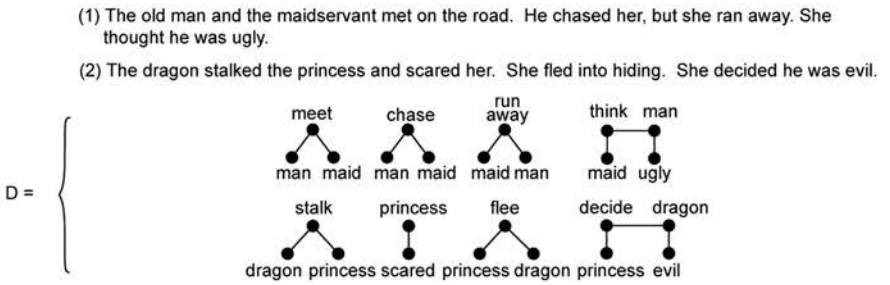

Figure 1. Example merge of two simple stories. Model M_3 describes not only the two input stories, but an additional two stories that alternatively include or exclude both nodes 1 and 6. Thus the model has generalized beyond the two input examples. The filtering step produces model M_4, which represents the final morphology.

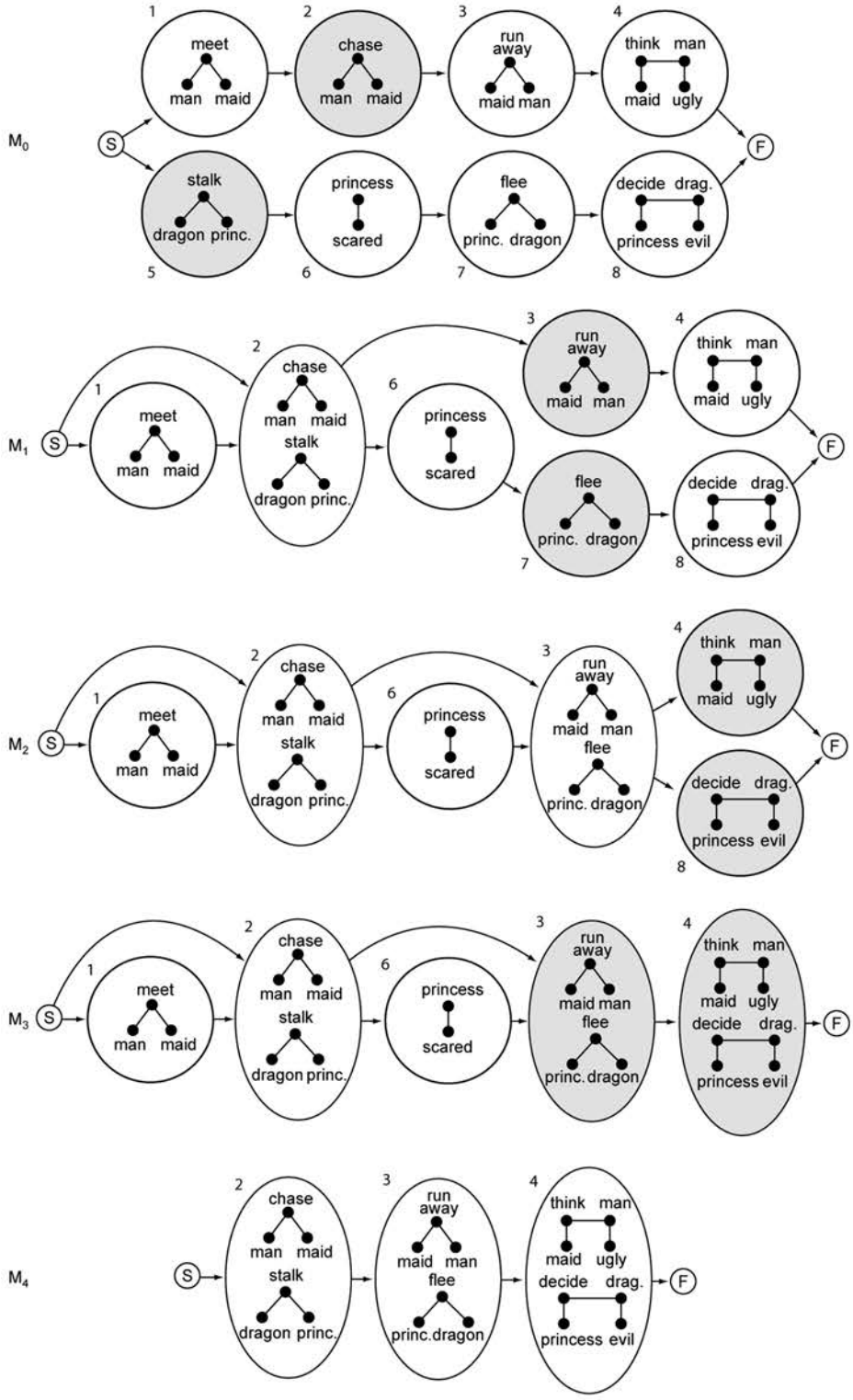

Figure 1 (*continued*).

are similar in that they involve one participant following after another; the running away and fleeing events are similar because they involve movement of one participant away from the other; and the thinking and deciding events are both mental events that involve evaluation. With a representation of the semantics of these events, one can use semantic distance measures and an analogical mapping algorithm to find the semantic and structural similarities. In the set of merges shown in figure 1, first the chasing and stalking events are merged, then the running away and fleeing, and then the thinking and deciding events. The result is a story morphology that could be seen as generating stories with an optional playing event at the beginning, a pursuit event, followed by an optional scared event, followed by the fleeing and evaluation events. Once the final model is filtered, three states remain, which might be titled Pursuit, Flee, and Judgment.

The initial model is derived from the time line of events in the story world itself. Each initial state in the model is a single event from each story, and they are ordered as they occur in the story time line. Each individual story time line is then incorporated into the initial morphology as a single linear branch. An example initial model, labeled M_0, can be seen at the top of figure 1, where each of the two simple example stories with their four constituent events is transformed into a sequence of four states.

There are many ways of driving the search for the appropriate set of merges. One popular way, which I have explored elsewhere, is a search driven by probabilities induced by Bayes's rule (Finlayson 2011). In contrast, the work described here uses a search driven by a set of semantic and structural merge rules derived from Propp's monograph. These rules will be outlined in the section titled "Merge Rules," after I explain the data on which the experiment was run. What is clear, however, is that some set of rules, heuristics, or biases is needed to find a good model: an exhaustive search is not possible in most cases.[1]

Filtering Stage

As can be seen in figure 1, the penultimate model in the chain (M_3) is not quite a morphology: it contains states that do not correspond to abstracted similarities between the two stories (i.e., states 2 and 3). This is because the initial model began with an alphabet of all possible symbols. A filtering step is used to go from the merged model to one that represents an actual morphology. The filtering process constructs another model from the final merged model from which all states that do not meet certain criteria are removed. The states that survive this culling become the alphabet of Propp's grammar, or the functions. The details of this filtering process are given in the "Merge Rules" section below.

Data

Propp selected a specific set of tales to analyze from which he derived his morphology: the first 100 tales of the Russian fairy tale collection by Alexander Afanasyev (1957).[2] Propp provides, in his Appendix III, function markings for only about half

the tales he analyzed: in the English translation of Propp's work, there are 45 tales in the function table, with a small number of additional analyses distributed throughout the text. In this work, I do not attempt to learn the move-level grammar, nor the subtype grammar, nor the dramatis personae categories. This restriction of scope leads to a particular approach to the preparation of the data. First, variations at the move level are filtered out by only including tales that Propp identified as containing a single move. Second, the learning data explicitly include identification of the dramatis personae categories for characters.

Because I restricted myself to single-move tales, the set of available tales was reduced from Propp's 45 analyzed tales; across several translations of Propp's morphology, I found a total of 21 single-move tales for which function analyses were provided. My ability to apply detailed semantic annotations to this set was further reduced by my research budget. In the end, I was left with 15 single-move tales, comprising 18,862 words, which I was able to have fully annotated.

Additionally, while Propp did his work in the original language of the tales (Russian or sometimes Belorussian or Ukrainian) for practical reasons, I did my analysis using English translations. Folklorists also sometimes study tales in translation, and the consensus is that, for first-order structural and semantic analyses, the important information of the tale is retained in a good translation. As J. L. Fischer notes: "If one translated a tale into another language, the tale structure and the essential features of the tale images would remain the same" (1963:249).

Semantic Annotation

My use of the word "annotation" here is the same as in corpus linguistics, in that it covers "any descriptive or analytic notations applied to raw language data" (Bird and Liberman 2001). Automatically producing high-quality annotations of the many different aspects of semantics required for this work is beyond the reach of the current state of the art of natural language processing (NLP). Consequently, to achieve high-quality, low-noise semantic annotations, I needed to hire people to either correct automatically produced annotations (so-called semi-automatic annotation) or to provide a fully manual annotation from scratch. The benefit of having people do semi-automatic or manual annotation is that, while it is slow and expensive, we can obtain annotations that we cannot yet create automatically with high quality. Thus, while the learning of Propp's function is done by machine, the raw data of the study, "the formalized semantics of the texts," were produced, essentially, by people.

All automatic, semi-automatic, or manual annotation performed by the annotators for this work was done using the Story Workbench annotation tool (Finlayson 2008, 2011). The Story Workbench is a general text-annotation tool that allows annotation of multiple layers of semantics, provides a user-friendly graphic user interface, and supports the annotation of arbitrary text. The annotated layers are listed in table 1. Propp's morphology is founded on characters and event structure, namely who is doing what to whom, and when: I call this the "surface semantics" of the text. Each of the listed layers was key to extracting the surface semantics from each text.

Table 1. Annotations used in this work. Agreement is expressed as either an F_1-measure or a chance-adjusted Rand index. The F_1-measure ranges from 0 (no agreement) to 1 (perfect agreement). The Rand Index ranges from –1 (perfect disagreement) to 1 (perfect agreement)

Layer	Semantics captured	Annotation style	Agreement
Referring expressions (REs)	Objects in the world	Manual	0.91
Co-reference	Which REs co-refer	Manual	0.85*
Time expressions	Times and dates	Manual	0.66
Events	Occurrences and states	Semi-automatic	0.69
Time links	Temporal order	Manual	0.59
Semantic roles	Arguments to verbs	Semi-Automatic	0.60†
Wordnet senses	Dictionary definitions	Manual	0.78
Event valence	Event's impact on the Hero	Semi-automatic	0.78
Dramatis personae	Propp's character types	Manual	0.70
Functions	Propp's functions	Manual	0.71‡

*Chance-adjusted Rand Index

†Core arguments only

‡Region overlap

Referring Expressions and Co-Reference

The raw information for calculating the characters in the story is given by the "referring expression" and co-reference annotations (Hervás and Finlayson 2010). The referring expression representation marks collections of words that refer to something, where the set of words may be continuous or discontinuous. This representation was annotated manually. Example (1) shows three examples of referring expressions, which are underlined.

(1) Ivan had a sword. It was sharp.

In this sentence, the referents are people and things, concrete things in the story world, but they need not always be so. Referring expressions can refer to abstract objects (such as ideas), events, times, actions, emotions, and many other things.

Example (1) also illustrates the important and obvious point, that a single referent can be mentioned several times in a text. In this case, there is a single referent (the sword) with two referring expressions (the phrases "the sword" and "it"). These last two referring expressions are co-referential because they refer to the same referent. To annotate referents using referring expressions, collections of referring expressions that co-refer are brought together in a co-reference bundle. Therefore, a co-reference bundle is a list of referring expressions referring to the same thing. This representation was annotated manually.

A second aspect of co-reference that was annotated was set-member relationships. Example (2) below shows a simple form, where the referring expression "Jack and Jill" refers to the set including both Jack and Jill. The word "they" is co-referential with the set Jack and Jill, and both Jack and Jill are members of that set. This information was important for determining which individual characters were actually participating in which events.

(2) Jack and Jill went up the hill. They fetched a pail of water.

Time Expressions, Events, and Time Links

To construct the time line of the story, I used the TimeML annotation scheme (Pustejovsky et al. 2003). TimeML comprises three representations: time expressions, events, and time links. The first two mark the objects that populate the time line, and the last defines the order of those objects on the time line. Examples in the section are drawn from those in the TimeML annotation guide (Saurí et al. 2006).

Time expressions mark the location, type, and value of temporal expressions. Each expression is a sequence of tokens, potentially discontinuous, that indicate a time or date, how long something lasted, or how often something occurs. Temporal expressions may be calendar dates, times of day, or durations such as periods of hours, days, or even centuries. Temporal expressions can be precise or ambiguous.

(3) The dragon arrived at noon. (Time)
(4) The dragon arrived on the last day of spring. (Date)
(5) He lived in the underworld for almost a year. (Duration)

Interestingly, time expressions are extremely sparse in the fairy tales analyzed in this study, with only 142 instances over the entire corpus of 18,862 words, an average of only 7.5 time expressions per 1,000 words. Indeed, most of the tales had fewer than 10 time expressions, and two had only one. This is perhaps because folktales generally occur on unspecified dates, or altogether outside of history. Regardless of the reason, time expressions proved to have little importance for the time lines as a whole.

Events are the second sort of object that populates the time line. Events are defined as happenings or states. They can be punctual, as in (6) below, or they can last for a period of time, as in (7). For the most part, circumstances in which something obtains or holds true, such as "shortage" in (8), are considered events.

(6) Ivan quickly struck the dragon's head from his body.
(7) The heroes traveled to faraway lands.
(8) There was a shortage of food across the land.

Events and times are connected together via time links, which represent temporal order. Time links fall into three major categories. Temporal links indicate an ordering between two times, two events, or a time and an event, as in Examples (9) and (10).

(9) Ivan's brothers only arrived *after* the fight. (Temporal—After)
(10) He lived in the underworld *for* almost a year. (Temporal—During)

Aspectual links indicate a relationship between an event and one of its subparts, as in (11) below. Subordinating links indicate relationships involving events that take arguments, as in (12) below. Good examples of events that start subordination links are those events that impose some truth-condition on their arguments, or imply that their arguments are about future or possible worlds.

(11) Ivan started to search for his wife. (Aspectual-Begin)
(12) Ivan forgot to bring the magic word. (Subordinating-Factive)

Word Senses

Word sense disambiguation (WSD) is a well-known NLP task in which each open-class token or multi-word expression (i.e., each noun, verb, adjective, or adverb) is assigned a single sense from a sense inventory, which gives us a proxy for the actual meaning of each word (Agirre and Edmonds 2007). For this work, the annotators sense-disambiguated every word using the electronic dictionary Wordnet 3.0 (Fellbaum 1998). Because most WSD algorithms are not much better than the default most-frequent-sense baseline, the annotators did this annotation completely manually. While they were assigning word senses, they also corrected multi-word expression boundaries, part-of-speech tags, and word stems. Although Wordnet's coverage is excellent, it occasionally lacks an appropriate word sense. In those cases, the annotators found a reasonable synonym and substituted that sense. In the rare case that they could not find an appropriate substitute, annotators were allowed to mark "no appropriate sense available."

Semantic Roles

Annotators also captured the argument structure of all the verbs in the texts, a task known as semantic role labeling. Specifically, we used the PropBank scheme (Palmer, Kingsbury, and Gildea 2005). This annotation was done semi-automatically by a basic statistical semantic role-labeler modeled on the analyzers described in the literature (Pradhan et al. 2005; Gildea and Jurafsky 2002). This labeler was run over the texts to create argument boundaries and semantic role labels for each verb. Each verb was also assigned a PropBank frame, which is a list of allowed roles and their descriptions. The identity of this frame was the only piece of information not automatically annotated. Annotators were required to add the frame, any missing arguments, and semantic role labels, and to correct the extant argument boundaries and labels. As was the case for word senses, sometimes an appropriate frame was not available in the PropBank frame set. This happened perhaps once or twice per text, and, in these cases, the annotators found the closest matching frame and assigned that instead.

Event Valence

Each TimeML event was also marked for its valence, a marking intended to capture how positive or negative an event is for the Hero. The scale is akin to Wendy Lehnert's positive or negative mental states (1981). My scale ran from –3 to +3, including 0 (neutral) as a potential valence, rather than being restricted to just positive or negative, as in Lehnert's representation. The import of each valence on the scale is laid out in table 2. This representation was annotated manually.

Dramatis Personae

Propp identified seven types of characters found in his folktales, and these character types are critically important in his theory. As noted earlier, I am leaving the learning

Table 2. Valence scale, which describes each level of affect and gives some examples

Valence	Description	Example
+3	Immediately good for the hero or his allies	The hero marries the princess; The hero is given gold
+2	May lead directly to a +3 event	Someone hides the hero from pursuit
+1	Someone promises an event that would be +2 or +3	An old man promises to help someday when the hero most needs it
0	Neither good nor bad	
–1	Someone threatens an event that would be –2 or –3	A witch threatens death to the hero
–2	May lead directly to a –3 event	The hero and the dragon fight
–3	Immediately bad for the hero or his allies	The princess is kidnapped; The hero is banished

of the dramatis personae for future work. Consequently, the dramatis personae were annotated, and this information was used to help derive the morphology structure. This representation consisted of the seven labels: Hero, Villain, Princess, Dispatcher, Donor, Helper, and False Hero. Any number of these could be attached to a particular referent in the text. Not all characters fulfilled a dramatis personae role, as Propp noted, and, in such cases, no tag was attached to that referent. In other cases, as Propp also noted, a single character fulfilled more than one role. This representation was annotated manually.

Functions

The final annotation captured Propp's functions. This annotation served as the standard against which the results of the learning algorithm were measured. Annotating Propp's functions was a delicate task. While Propp described his morphology in great detail, it still was not specified in such a way as to allow unambiguous annotation in text. Propp's monograph is enlightening, but it is not an effective annotation guide. There are at least four main problems with Propp's scheme as described: unclear placement; implicit functions; inconsistent marking of trebling (function groups that were repeated two, three, or four times in succession); and, in a small number of cases, apparent disagreement between Propp's own categorization scheme and what is found in the tale.

Regarding unclear placement, consider, for example, the following excerpt of Afanasyev's tale No. 148.

> The tsar went in person to beg Nikita the Tanner to free his land from the wicked dragon and rescue the princess. At that moment Nikita was currying hides and held twelve hides in his hands; when he saw that the tsar in person had come to see him, he began to tremble with fear, his hands shook, and he tore the twelve hides. But no matter how much the tsar and tsarina entreated him, he refused to go forth against the dragon. So they gathered together five thousand little children and sent them to implore him, hoping that their tears would move him to pity. The little children came to Nikita and begged him with tears to go fight the dragon. Nikita himself began to shed tears when he saw theirs. He took twelve thousand pounds of hemp, tarred it

> with pitch, and wound it around himself so that the dragon could not devour him, then went forth to give him battle. (Afanasyev 1957, 1975:310–1)

Propp indicates the presence of functions B and C in this tale. Propp calls B the "mediation, the connective incident," with this extended definition: "Misfortune or lack is made known; the hero is approached with a request or command; he is allowed to go or he is dispatched" ([1928] 1968:36). He calls C the "beginning counteraction," with this extended definition: "The Seeker agrees to or decides upon counteraction" ([1928] 1968:38). Roughly, these two functions are the presentation of the task to the hero (B), and the acceptance of that task (C).

Finding these two functions in this passage is not trivial. Where exactly is B? Is it the whole section? Is it from the word "entreated" to the word "begged"? Should function boundaries correspond to sentence or paragraph boundaries? Is their imploring of the children to be considered part of B? Annotators marked two groups of tokens when identifying functions. First, they marked a region that captures the majority of the sense and extent of a function. This was usually a sentence, but extended to a paragraph or more in some cases. Second, they marked a defining word for the function, which usually took the form of a single verb. In cases where that single verb or its synonyms were repeated in close proximity to the first marking, and referred to the same action, these repeats were marked as well. In the case above, annotators marked the region "the tsar and tsarina entreated . . . and begged him with tears to go fight the dragon" as B, and picked the verbs "entreated" and "begged" as the defining verbs.

Where exactly is C? This is the decision to go forth against the dragon. It seems to happen somewhere between Nikita's shedding of tears and his preparation for battle by obtaining hemp, but it is not expressed anywhere directly in words; that is, the function is implicit. When the annotators could find no explicit mention of a particular function that Propp indicated as having occurred in a tale, they chose the logically most closely related event and marked it either as an Antecedent or a Subsequent, as appropriate. For C in the section above, the region was the sentence "Nikita himself began to shed tears when he saw theirs," and "shed" was marked as the defining verb. This implicit function was marked as an Antecedent.

When trebling was inconsistently marked, or when indicated functions did not seem to match the tale itself, the annotators did their best to determine the correct marking. Fortunately, most typographical errors in Propp's table were restricted to disagreement in function subtypes, which does not directly affect these results.

Agreement

Measuring inter-annotator agreement can provide an assessment of the quality of the annotations. In cases where an established layer was being annotated, I prepared an annotation guide from the available material for the annotation team. An annotation team consisted of two annotators and an adjudicator. The adjudicator was either an annotator already experienced in that representation, or myself (if no other adjudicator was available). After the annotation of the same few thousand

words (two or three texts) by the two annotators, the whole annotation team met and merged the annotations together into a single document, which was then corrected by discussion guided by the adjudicator. This process was repeated until all the texts were annotated.

The most uniform measure of agreement across the different layers is the F_1-measure, familiar to statisticians, which, calculated in the standard way, provides the harmonic mean of precision and recall (van Rijsbergen 1979; see also Nikolić and Bakarić in this issue). I used the F_1-measure instead of the more common Kappa statistic, which measures the chance-adjusted agreement (Carletta 1996) because of the difficulty of calculating the chance-level of agreement for most of the layers. The F_1-measure is a natural outgrowth of the merge process, has a clear interpretation with regard to the data, and allows a direct comparison between different layers. Table 1 summarizes the agreements for the different layers annotated either manually or semi-automatically. Overall, the agreement values are good.

Initial Model Construction

With the human-annotated data in hand, we can proceed to the automated portion of the study. Constructing the initial model for the merge algorithm required the following steps: first, a time line of events for each story was automatically extracted from the annotations. Second, each event was automatically associated with a set of Agent and Patient characters. Figure 2 represents schematically the information included in the initial model.

The TimeML annotations allowed the extraction of the time line of events for each tale. The fairy tales in the corpus are quite simple in their temporal structure; all of them, except for one, are describable by a linear time line. To construct the linear time line for each tale, I first removed all subordinated events from consideration. Events connected only by subordinating links indicated events that did not actually occur on the time line. Second, using the straightforward definition of the temporal links (Before, After, Simultaneous, etc.), I wrote a simple algorithm that arranged events in the order of their starting point (Finlayson 2011).

It should be noted that quite a few of the events on the time lines were generic, and these were not distinguishable on the basis of surface semantics alone from other non-Functional events. These events were eventually filtered from consideration; this is discussed more in the section on "Merge Rules." Table 3 shows the 15 tales that were annotated, the counts of events, the number of events on the full time line (excluding subordinated events), and the filtered time line that was used in the final experiment.

Once I constructed the event time line, I assigned automatically, if possible, an Agent and a Patient to each event. I extracted this information from the semantic role, referring expression, and co-reference annotations. Every verb in the corpus was marked with a semantic role, which gives the arguments to that verb represented as spans of text. Nearly every event in the corpus was associated with at least one semantic role via its verb expressions. In fact, of the 3,438 events on the tale time lines, only two events had no semantic role. I manually specified the Agents and Patients of these two

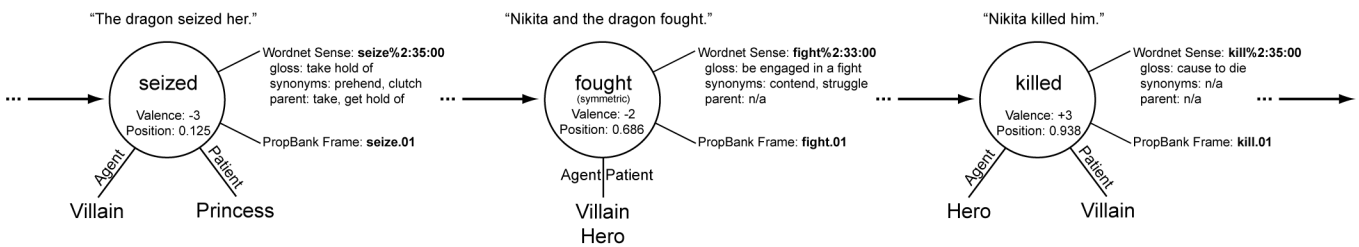

Figure 2. Schematic of the information extracted from the annotations. Each story is represented by an ordered list of events (the time line), which is extracted from the TimeML annotations. Each event is assigned a set of Agent and Patient roles, if possible, gathered from the dramatis personae labels attached to the participating referring expressions, where groups of characters are replaced by individuals. Each event is also associated with one or more Propbank frames, one or more Wordnet senses, and an Event Valence.

events in post-processing. When an event had more than one semantic role, meaning that the event was mentioned several times using a verb, I merged the subject and object fillers for each associated semantic role, favoring the first-mentioned semantic role in case of a conflict.

I used each semantic role's associated PropBank frame to find the subject and object. According to PropBank conventions, the argument to the verb marked ARG0 is usually the subject, and the argument marked ARG1 is usually the object. Nevertheless, many PropBank frames do not have this ARG0-ARG1 subject-object structure because of the idiosyncrasies of the frame definitions. Furthermore, some PropBank frames can be considered symmetric, where the Agent and Patient roles are not semantically distinct (e.g., the verb "marry" when used intransitively, as in "Anna and Bob were married"). Because this information is not encapsulated anywhere in PropBank, I manually classified the symmetric nature and Agent and Patient roles for all the PropBank frames found in the corpus.

Once the correct subject and object span was determined, the largest referring expression inside each span was automatically chosen as the most appropriate role filler. Once the referring expressions filling the subject and object roles of an event were determined, it remained to substitute automatically that referring expression with one or more elementary referents. Sometimes this required replacing a composite referent with member referents.

Table 3. Texts analyzed. All texts were single-move folktales for which Propp provided function analyses. Listed are the number of words in the English translation; the number of TimeML events annotated for each story; the number of non-subordinated events that appear in the full time line of the story; and the number of events that appear in the "filtered" time line used in the learning experiment

Tale No.	Russian title	English title	# Words	# Events	Full time line	Filtered time line
148	*Никита кожемяка*	*Nikita the Tanner*	646	104	75	16
113	*Гуси-лебеди*	*The Magic Swan Geese*	696	132	94	43
145	*Семь симеонов*	*The Seven Simeons*	725	121	87	42
163	*Бухтан Бухтанович*	*Bukhtan Bukhtanovich*	888	150	107	62
162	*Хрустальная гора*	*The Crystal Mountain*	989	150	104	43
151	*Шабарша рабочий*	*Shabarsha the Laborer*	1202	236	122	55
152	*Иванко Медведко*	*Ivanko the Bear's Son*	1210	223	143	65
149	*Змей и цыган*	*The Serpent and the Gypsy*	1210	250	138	80
135	*Иван Попялов*	*Ivan Popyalov*	1228	220	170	46
131	*Фролка-сидень*	*Frolka Stay-at-Home*	1388	248	169	56
108	*Ивашко и ведьма*	*Ivashko and The Witch*	1448	276	157	61
154	*Беглый солдат и черт*	*The Runaway Soldier and the Devil*	1698	317	190	76
114	*Князь Данила-Говорила*	*Prince Danila Govorila*	1774	341	223	92
127	*Купеческая дочь и служанка*	*The Merchant's Daughter and the Housemaid*	1794	331	234	89
140	*Зорька, вечорка, и полуночка*	*Dawn, Evening, and Midnight*	1934	339	250	78
		Average	1258	229	151	60
		Sum	18862	3438	2253	904

Merge Rules

To design merge rules that reproduce Propp's functions within the model-merging framework, I considered three of the same features to which Propp himself was sensitive in his analysis. Propp describes in his monograph three features through which he found similarity between events: event semantics, the dramatis personae involved, and the position of the event in the arc of the move. I leveraged these three aspects of similarity in a two-stage merge process. The first stage merged together, roughly, events that were semantically similar. The second stage merged only states that contained more than one event, and, of these states, it merged those that were nearby with the same emotional valence for the Hero.

Both stages only merged together states that involved consistent sets of dramatis personae. The dramatis personae involved in two events were considered consistent when they were either identical or proper subsets of each other. More specifically, the dramatis personae labels for each participant in both the Agent and Patient positions were added to an Agent or Patient set of labels. If the Hero tag was in a set, the Helper tag was also added, and vice versa. Two events were considered to have consistent dramatis personae if one event's Agent and Patient sets of dramatis personae labels equaled (or was a proper subset of, or vice versa) the other event's Agent and Patient sets, respectively. If one of the events was marked as a symmetric event, where Agent and Patient positions are interchangeable, the dramatis personae sets for each event were combined into one set for the purposes of matching.

Stage One: Semantics

The merge rule for stage one was as follows. Two states were automatically merged if (1) all events in the resultant state were non-generic (see below), (2) all pairs of events in the resultant state were synonymous or hyper-synonymous with regard to their Wordnet senses, and (3) each unique PropBank frame attached to all events in the resultant state was represented at least twice. I define these conditions in more detail below.

Generic Events: I identified a class of verb type, which I called "generic" verbs. They were automatically excluded from merging because it was impossible to distinguish an informative, functional use of these words from a generic filler sense. The verb "say" and its synonyms are a good example: these made up nearly three-quarters of all events, and every one of Propp's functions included at least one "say" event. That is, characters could accomplish all of Propp's functions through speech acts. Characters could threaten each other (A, Villainy, or Pr, Pursuit), meet for the first time or offer assistance (D, First Encounter with the Donor), react to other's actions (E, Hero's Reaction to the Donor), offer one's services (C, Decision to Counteract), send a Hero on a quest (B, Dispatch), and so forth. More precisely, generic events were those whose verbs were marked with Wordnet senses falling into the lexicographer files of verbs of communication, perception, or motion. These include verbs such as "say," "see," or "go."

Synonymity: Two events were considered synonymous if their attached Wordnet senses, or the sense's hypernyms, shared synonyms. This defined a loose semantic similarity that allowed events to be clustered together on the basis of meaning.

Doubled PropBank Frames: PropBank frames were attached to events through the semantic role annotation, as described previously. For two states to be merged, each PropBank frame found on an event in that state needed to be found on at least one other event in that state. This more specific semantic similarity served as a balance to the more generous similarity provided by Wordnet synonymity.

Stage Two: Valence and Position

In the second stage of merging, two states were automatically merged if (1) the two states already contained more than one event each, (2) the valence across the events of the states was compatible, and (3) the two states were the closest pairs of events in the story arcs.

Matching Valence: Two states were only automatically merged in this stage if the valences across the events in a state were compatible. As shown in table 2, event valence was measured on a 7-point scale running from +3 to –3. Two valences were compatible if their values were equal, with the exception that a neutral valence (value of 0) was allowed to match any other valence.

Closest Pairs: This stage also automatically merged states in a particular order, according to how far apart the state's constituent events were, relatively, on their time lines. The position of each state was calculated as follows. The position of an event was defined as a fraction between 0 and 1, inclusive, corresponding to its relative position in its original linear time line. The position of a merged node was the average position of its constituent events. Then pairwise merges were ranked according to the difference in position between the states they were merging, where the smallest differences were pushed to the front of the search queue.

Results

From the Propp function annotations described previously, I constructed the gold standard against which the final model was measured. The final, gold standard set of function markings was actually much reduced from the list of functions in Propp's monograph for three reasons: Propp's omissions, functions not present or too sparse in the corpus data, and implicit functions.

Of the 31 functions, Propp did not indicate the presence of the first seven functions (these were the preparatory functions, marked with Greek letters). These functions therefore had to be excluded from the analysis. Of the remaining 24 functions, functions J, L, M, and N were not found in the 15 tales in my corpus, leaving 20 functions. Of these, an additional four—o, Q, Ex, and U—had two or fewer instances, and were excluded because they were too sparse to learn.

There were 276 function markings, of which 186 were explicit and 90 implicit. Because I did no commonsense inference, these implicit functions, or over 30 percent

of the data, had no actual event instantiation in the text. This problem was largely circumvented by noting that the majority of the implicit functions was one of functions involved in the two pairs of E-F (Reaction & Receipt) and H-I (Struggle & Victory). In these cases, when one of a pair was implicit, the other would be explicit. For example, in the case of a Hero fighting with a Villain, only the actual fight was mentioned and the victory was left implicit, or the victory was mentioned and the fight left implicit. Thus for the purposes of measurement, I merged these two sets of functions together. This resulted in the merging of 45 implicit function markings into explicit function instances, leaving 234 explicit function markings out of 276; the remaining 45 implicit markings were excluded from the target. These data are summarized in table 4, with the rightmost column indicating the number of functions present after filtering out generic events (see next section).

I used three different measures to analyze the performance of the learning procedure. The first was the application of the chance-adjusted Rand Index, a measure of the overall quality of the clustering of events into Propp's functions (Rota 1964). The second was the application of individual F_1 measures for each of Propp's functions. The third was a cross-validation analysis of how well the implementation works with smaller amounts of data.

Event Clustering

I used the chance-adjusted Rand Index (Hubert and Arabie 1985) to examine the quality of clustering of events into Propp's function categories. I created three measures that may be ranked, colloquially, from "strict" to "lenient." They were (1) a Strict score, where the clusters in the final model were compared against all Propp's clusters of explicit function markings as listed in the Explicit Raw column in table 4; (2) an Interactive-Only score, where the clusters in the final model were compared against Propp's explicit clusters with Non-Interactive events removed; and (3) an Interactive

Table 4. Function present in the corpus before and after time line filtering

Symbol	Description	# Explicit raw	# Explicit filtered
A/a	Villainy/Lack	18	15
B	Mediation/Dispatch	7	7
C	Beginning counteraction	7	5
up	Departure	13	7
D	Encounter with the donor	16	16
EF	Reaction & Receipt	30	29
G	Transference	4	2†
HI	Struggle & Victory	71	66
K	Tension liquidated	12	9
down	Return	10	2†
Pr	Pursuit	18	14
Rs	Rescue	13	10
T	Transfiguration	3	2†
W/w	Reward	12	8
	Total	234	186

†Too few instances in data to expect extraction, not included in total

Table 5. Three different chance-adjusted Rand index measures of cluster quality. The scores range from most strict through most lenient

Method	Score
Strict	0.511
Interactive only	0.581
Interactive non-generic only	0.714

Non-Generics Only score, where the clusters in the final model were compared against Propp's explicit clusters with both Non-Interactive and Generic Events removed. These three results are listed in table 5. For the most lenient measure (Interactive Non-Generics Only), the algorithm performs fairly well, achieving an overall 0.714 chance-adjusted Rand Index against Propp's original functions. Here, I say "fairly well" because it is actually unclear how good this performance is, as there is no prior work: this work is the first attempt ever to learn Propp's functions from folktales by computer, so there is no previous technique against which to compare.

Function Categories

The second metric was the F_1-measure for individual function categories. Of the 14 function categories in the final data, eight were recovered. These results are shown for the Interactive Non-Generics O measure in table 6. Importantly, the algorithm extracted the most central functions of the morphology: the initial villainy (A), the donor-encounter triplet (DEF), the struggle with and victory over the villain (HI), the liquidation of the villainy (K), the Pursuit-Rescue doublet (Pr-Rs), and the final reward (W). These are all key functions, not only in the tales analyzed, but across Propp's morphology.

The most striking success was the extraction of HI, the Struggle-Victory combination function. A full 51 instances were classified correctly, and, when measured against the filtered time line, this resulted in an overall F_1-measure of 0.823. This success

Table 6. F_1-measures of identification of functions

Symbol	Description	Semantics	#False pos.	# True pos.	# False neg.	F_1
A/a	Villainy/Lack	devour, drag, hurt, seize	3	12	3	0.8
D	Donor encounter	drag, hit	0	6	10	0.585
E-F	Reaction & Receipt	cover, eat, make	3	9	20	0.839
H-I	Struggle & Victory	attack, break, cut, defeat, drag, fight, hit, hurt, push, race, seal, seize, throw, whistle	7	51	15	0.823
K	Tension liquidated	fill	0	3	4	0.6
Pr	Pursuit	chase, chew	0	5	9	0.526
Rs	Rescue	strike, throw	1	6	4	0.706
W	Reward	gift, marry	1	6	2	0.8

can probably be attributed to substantial uniformity of semantics for this particular function, in that all the verbs were verbs of competition and fighting.

The next notable success is the identification of A (Villainy/Lack) and W (Reward), functions with an F_1-measure of 0.8. These are two key functions because they book-end the action. Similar to HI, the semantic uniformity of these functions was important to their successful extraction. In Russian tales, the most common villainy is a kidnapping of a princess or other weak party. The reward is most commonly either marriage to the rescued princess or a gift of money.

Cross-Validation

The third metric of success was a cross-validation study where the algorithm was run over different subsets of the data, and exhibited a smooth degradation with smaller amounts of data. Remarkably, the technique still achieves an average chance-adjusted Rand Index of 0.457 when examining just two stories. Figure 3 shows this performance, measured by the three chance-adjusted Rand Index measures as in table 5, at the optimal parameter values over different subsets of the corpus. Each data point in the graph is an average of all *n*-sized subsets of stories from the folktale corpus. As can be seen, the algorithm's performance drops off smoothly, until, when only two stories are being considered at a time, it retains a surprisingly good value of 0.457 for the No-Generics measure, 0.360 for the Interactive Only measure, and 0.325 for the Strict measure. This measurement shows that the implementation is actually quite robust to variation in the data.

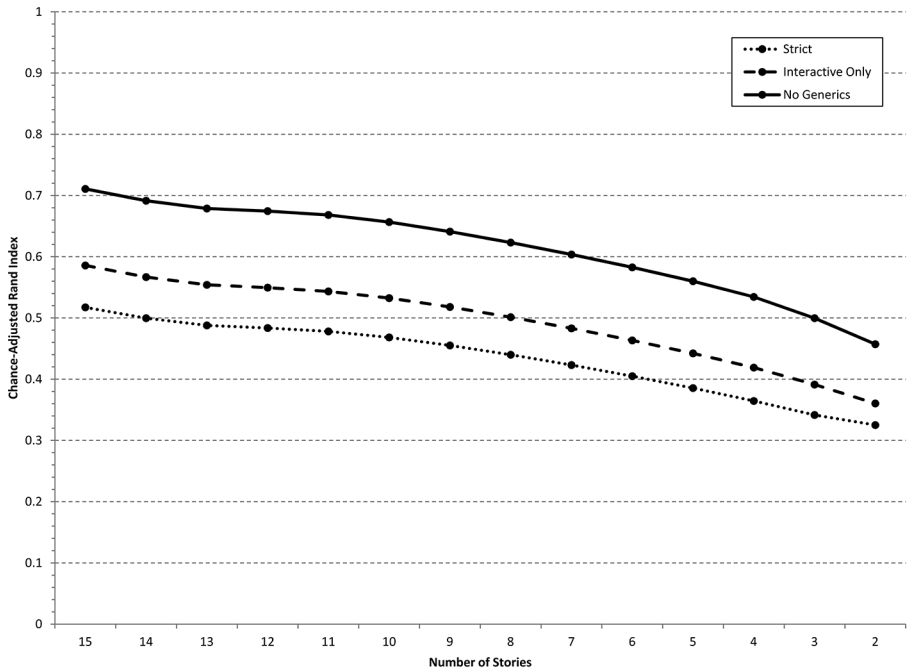

Figure 3. Performance of the ProppASM implementation on all subsets of the corpus.

Related Work

While this is the first work that has shown learning of an actual theory of narrative structure by computational means, there has been some interesting recent work on learning more general narrative patterns. First, Nathanael Chambers and Dan Jurafsky have leveraged distributional learning over very large corpora to identify common sequences of events (2008, 2009). The technique relies on a point-wise mutual information score between verbs that share arguments to build up common pairs of events and their orders. These pairs are then knitted together to form narrative chains. Narrative chains have several interesting points of commonality and difference with this work. Chambers and Jurafsky and I are similarly trying to identify chains of events commonly found across sets of texts. Furthermore, their work is another data point supporting the argument that knowing the roles of the characters (e.g., who is the Protagonist) is critically important for identifying common narrative structure. On the other hand, the technique relies on an incredible weight of texts (they train on over 1 million texts) to find similarities. This approach is in contrast to my own algorithm, which my cross-validation study shows works passably well on a mere two tales. Also in contrast to my approach, the narrative chain model used by Chambers and Jurafsky sits quite close to the meaning of the texts: verbs are considered identical when they share root forms. With my technique, I go beyond this surface representation to abstract and generalize from the data—for example, using semantic knowledge to unify items such as "kidnap" and "seize," and then further unify these with a verb such as "torment" to achieve a "harm" or "villainy."

Other work, including that by Michaela Regneri, Alexander Koller, and Manfred Pinkal (2010), seeks to learn event scripts from lists of actions. The technique is a variation of the multiple-sequence alignment technique from bio-informatics. In their work, they were able to extract reasonable script-like structure from the data. Differences from my work include the type of data (subject-generated lists of key actions in performing a task versus natural stories) and, as with Chambers and Jurafsky, the inability to learn cycles. There is also no attention paid to filtering out unimportant events, as their starting data contained only events relevant to a particular script.

Conclusion

This work represents an advance for both the artificial intelligence field and the field of folkloristics. For artificial intelligence, it demonstrates a technique for learning a level of semantics rarely attempted and never before learned in such a verified manner. For folkloristics, it demonstrates that computational techniques can provide significant help in examining the deeper structure of folklore, and do not have to operate only at the surface level of lexical or keyword analyses.

There are many avenues to explore in future work. First, we should pursue extending these techniques to automatically learning the other levels of Propp's theory: moves, subtypes, and dramatis personae. Second, regarding functions, it is natural to apply this work to other morphological analyses, such as those by Colby (1973) and Dundes (1964). Third, the basic technique itself affords much improvement: greater integration of commonsense knowledge about cause, generics, and other semantics;

attempts to learn implicit functions; and closing the loop by verifying the validity of morphological analyses via psychological or cultural experiments. With these efforts, artificial intelligence and folkloristics can expect much exciting future interdisciplinary interaction that will enrich and advance both fields.

Acknowledgments

The work described here and the preparation of this article were supported by the Defense Advanced Research Projects Agency under contract number D12AP00210, as well as the Office of Naval Research under award number N00014-09-1-0597.

Notes

1. For non-trivial starting stories, the search space for model merging becomes unmanageably large: it is equal to Bell's number, B_n, where n is the number of initial states in the model (Rota 1964). Bell's number quickly becomes extremely large as n increases. For example, while $B_2 = 2$ and $B_3 = 5$, $B_{10} = 115{,}975$ and $B_{55} \approx 3.59e + 31$.

2. Note that Propp used an older version of Afanasyev's collection. We provide this more modern citation for convenience.

References Cited

Afanasyev, Aleksander N. 1957. *Narodnye Russkie Skazki*. 3 vols. Moscow: Gos. Izd-vo Khudozh. Lit-ry.

———. 1975. *Russian Fairy Tales*, trans. Norbert Guterman. New York: Pantheon Books.

Agirre, Eneko, and Philip Edmonds, eds. 2007. *Word Sense Disambiguation*. Dordrecht: Springer.

Bird, Steven, and Mark Liberman. 2001. A Formal Framework for Linguistic Annotation. *Speech Communication* 33(1–2):23–60.

Carletta, Jean. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics* 22(2):249–54.

Chambers, Nathanael, and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pp. 789–97, Columbus, OH.

———. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, pp. 602–10, Suntec.

Colby, Benjamin. 1973. A Partial Grammar of Eskimo Folktales. *American Anthropologist* 75(3):645–62.

Díaz-Agudo, Belén, Pablo Gervás, and Federico Peinado. 2004. A Case Based Reasoning Approach to Story Plot Generation. In *Proceedings of the European Conference on Case Based Reasoning (ECCBR)*, pp. 142–56, Madrid.

Dundes, Alan. 1964. The Morphology of North American Indian Folktales. FF Communications No. 195. Helsinki: Suomalainen Tiedeakatemia.

Fellbaum, Christiane, ed. 1998. *Wordnet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Finlayson, Mark Alan. 2008. Collecting Semantics in the Wild: The Story Workbench. In *Proceedings of the AAAI Fall Symposium on Naturally Inspired Artificial Intelligence*, pp. 46–53, Washington, DC.

———. 2009. Deriving Narrative Morphologies via Analogical Story Merging. In *New Frontiers in Analogy Research (Proceedings of the 2nd International Conference on Analogy)*, ed. Boris Kokinov, Keith Holyoak, and Dedre Gentner, pp. 127–36. Sofia: New Bulgarian University Press.

———. 2010a. Learning Narrative Morphologies from Annotated Folktales. In *Proceedings of the 1st Automated Motif Discovery in Cultural Heritage and Scientific Communication Texts (AMICUS) Workshop*, pp. 99–102, Vienna.

———. 2010b. The Story Workbench: An Extensible Semi-Automatic Text Annotation Tool. In *Proceedings of the 4th Workshop on Intelligent Narrative Technologies (INT-4)*, pp. 21–4, Stanford, CA.

———. 2011. Learning Narrative Structure from Annotated Folktales. PhD diss., Massachusetts Institute of Technology.

Fischer, J. L. 1963. The Sociopsychological Analysis of Folktales. *Current Anthropology* 4(3):235–95.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Oxford: Basil Blackwell.

Gildea, Daniel, and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics* 28(3):245–88.

Goodman, Joshua. 2000. Probabilistic Feature Grammars. In *Advances in Probabilistic and Other Parsing Technologies*, ed. Harry Bunt and Anton Nijholt, pp. 63–84. Dordrecht: Springer.

Greimas, Algirdas Julien. 1966. *Sémantique structurale: Recherche de méthode*. Paris: Larousse.

Halpin, Harry, Johanna Moore, and Judy Robertson. 2004. Automatic Analysis of Plot for Story Rewriting. In *Proceedings of the Conference on Experimental Methods in Natural Language Processing (EMNLP)*, pp. 127–33, Barcelona.

Hervás, Raquel, and Mark Alan Finlayson. 2010. The Prevalence of Descriptive Referring Expressions in News and Narrative. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 49–54, Uppsala.

Higuera, Colin de la. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge: University of Cambridge Press.

Hubert, Lawrence, and Phipps Arabie. 1985. Comparing Partitions. *Journal of Classification* 2(1):193–218.

Lakoff, George. 1972. Structural Complexity in Fairy Tales. In *The Study of Man*, Vol. 1:128–50. Irvine, CA: School of Social Sciences, University of California. https://georgelakoff.files.wordpress.com/2010/12/structural-complexity-in-fairy-tales-lakoff-1972.pdf.

Lehnert, Wendy G. 1981. Plot Units and Narrative Summarization. *Cognitive Science* 5(4):293–331.

Lévi-Strauss, Claude. 1978. *Myth and Meaning*. New York: Routledge.

Omohundro, Stephen M. 1992. Best-First Model Merging for Dynamic Learning and Recognition. In *Advances in Neural Information Processing Systems 5*, ed. John E. Moody, Stephen J. Hanson, and Richard P. Lippmann, pp. 958–965. San Mateo, CA: Morgan Kaufmann.

Palmer, Martha, Paul Kingsbury, and Daniel Gildea. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics* 31(1):71–105.

Pradhan, Sameer, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurky. 2005. Support Vector Learning for Semantic Argument Classification. *Machine Learning* 60(1–3):11–39.

Propp, Vladimir. [1928] 1968. *Morphology of the Folktale* (2nd edition), trans. Laurence Scott. Austin: University of Texas Press.

Pustejovsky, James, Jose Castano, Robert Ingria, Roser Sauri, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proceedings of Fifth International Workshop on Computational Semantics (IWCS-5)*, p. 193, Tilberg.

Regneri, Michaela, Alexander Koller, and Manfred Pinkal. 2010. Learning Script Knowledge with Web Experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 979–88, Uppsala.

Rota, Gian-Carlo. 1964. The Number of Partitions of a Set. *American Mathematical Monthly* 71(5):498–504.

Saurí, Roser, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. TimeML Annotation Guidelines, Version 1.2.1. http://www.timeml.org/site/publications/timeMLdocs/annguide_1.2.1.pdf.

Stolcke, Andreas, and Stephen Omohundro. 1994. Inducing Probabilistic Grammars by Bayesian Model Merging. In *Grammatical Inference and Applications*, ed. Rafael C. Carrasco and Jose Oncina, pp. 106–18. Berlin: Springer.

van Rijsbergen, C. J. 1979. Evaluation. In *Information Retrieval*, by C. J. van Rijsbergen, pp. 112–40. London: Butterworths.

Young-Lai, Matthew. 2009. Grammar Inference. In *Encyclopedia of Database Systems*, ed. Ling Liu and M. Tamer Ozsu, pp. 1256–60. Berlin: Springer.