AutoPropp: Toward the Automatic Markup, Classification, and Annotation of Russian Magic Tales

Scott Malec
University of Texas Health Science Center at Houston

AutoPropp:
Toward the Automatic Markup, Classification, and Annotation of Russian Magic Tales

Scott Malec
Software Engineer
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
00 + 1* + (412) 330 7082
malec@andrew.cmu.edu

ABSTRACT

In this paper, I describe the current state of my program of research that uses the R environment and programming language for machine learning and statistics (Feinerer et al, 2008) to automatically analyze and/or parse texts in PFTML (Proppian Fairy Tale Markup Language) (Malec, 2005; Lendvai et al, 2010; Lendvai et al, 2010), an XML grammar that can be used for the semantic markup of folk narratives. R can be applied by researchers to manipulate and analyze motifs in tandem with "functions", the sub- and supra-sentential structures as described by Vladimir Propp in his Morphology of the Folktale (1928) in his analysis of Russian (magic) tales (Afanas'ev, 1945). These methods may be applied to corpora beyond the domain of folklore, but a limited formulaic corpus provides an interesting use case for these methods. The core idea is that this research program would use a corpus that was annotated in PFTML format as a training set to automate the process of annotating folkloric texts. I describe the conceptual framework, the required software packages, the basic steps of preprocessing text (annotation, stemming, removing whitespace and stopwords, importing data into the R environment) and discuss my progress on this project and the questions that I have encountered along the way.

1. INTRODUCTION

The AutoPropp research program attempts to compute the story structure of Russian magic tales given a training corpus marked up in Proppian Fairy Tale Markup Language and a hidden Markov model using some variant of an Expectation-Maximization algorithm, e.g. Viterbi, Baum-Welch in the R programming language. A hypothetically computed Story structure is conceived as the way in which boundaries of sub- and supra-sentential segments may be inferred by both their semantic content and a probabilistic model of the sequence of those segments, given training data for topic content of particular segments and a hidden Markov model that describes the likely sequence of those segments, respectively. This is a supervised learning technique that applies specifically to Propp's original corpus, that of Afanas'ev, in the original Russian. This is a collection of texts with particular attributes that Propp described in his *Morphology of the Folktale* (1928).

Let me define a few terms first. A *function*, as Propp construes it, is a textual unit that generally moves the story forward, irrespective of the dramatis personae that one encounters in it or that perform it. A *story grammar* is an abstract hypothetical schema that characterizes how the semantics and sequence of the elements of a particular genre of stories are constructed. Story grammars can be subject to mathematical rigor; as such, AutoPropp attempts to model these mathematical properties with hidden Markov models and distinct topic distribution for each function.

While there is no consensus vis-a-vis a grand unified theory of narrative, much less a working lexicon for elements of a general story grammar, yet many story grammars have been proposed that apply to specific corpora, e.g., Colby's grammar of Eskimo tales (Colby 1973). Others researchers across a diverse array of disciplines have performed research in this area, including cognitive science (Rumelhart 1975, Lehnert 1981), story generation (Bringsjord 2000 and others), literary theory, anthropology and elsewhere. I have picked Propp because I am familiar with both his work and his corpus. Another researcher may do just as well choosing another corpus and its respective generalized characterization. Propp is as good as any for starters.

2. AUTOPROPP: CONCEPTS AND ALGORITHM HIGHLIGHTS IN PSEUDOCODE

To implement AutoPropp, several sets of data are required: a subset of Afanas'ev's *Russkie nardodye skazki* (1957) with each tale in a separate text file in its own directory and both a subset of Afanas'ev marked up in Proppian Fairy Tale Markup Language (PFTML) and an HMM model (to give AutoPropp a kind of

First International AMICUS Workshop, October 21, Vienna, Austria

roadmap or skeleton key through the terrain of the text) as training data to enable AutoPropp to build a model of the text.

These would be the initial inputs:

PFTML: Afanas'ev's *Russkie nardodye skazki* marked up in PFTML

HMM_Matrix: tab or comma delimited matrix of Proppian functions to create hidden markov model (I have done this using Appendix III in Propp's *Morphology*)

[Note: this would look like A, B, C, Depart, D, E, F, K, Return, W, etc. with each element representing a function and each line representing a “Move”* within a tale in the corpus.]

Raw_Input: Afanas'ev's *Russkie nardodye skazki* in plain text in separate text files. It may be useful to perform some preprocessing on this by way of annotating it given sentence structure as defined by of sentence punctuation (or NLP annotation if that should be available, as these can give useful information to how the windowing function should proceed.

Allow me to sketch out tentatively what AutoPropp might look like in English:

1. Run trainer (“feed” AutoPropp PFTML and HMM Matrix).

a. Preprocess text within markup and create document text matrices for Proppian function across PFTML and generate semantic characterizations of each Proppian function that is encountered using Latent Semantic Analysis or Latent Dirichlet Analysis.

b. Create hidden markov model from HMM_Matrix to provide expectations for function sequences for AutoPropp's story parser.

2. Open Raw_Input, preprocess it, and pick text windows from that output starting with the first (or next) Proppian function in the hidden markov model generated from HMM_Matrix.

a. Compare candidate passage with result of semantic analysis from that function from

PFTML and calculate semantic similarity based on some variant of Bayesian co-occurrence or Levenshtein edit distance.

b. Adjust window parameters left and right in output and calculate output of similarity metric.

c. Compare variations from parser output by using Expectation-Maximization (EM) algorithm of log likelihoods based on hidden markov model and similarity metric:

*BestParse = Max(p*p*p*etc) + Max(s*s*etc.)*

Save the maximized parse (and possibly a given percentage of the top contenders) of text windows that have been traversed through the text.

d. Markup story according to where function segment boundaries have been inferred,

update hidden markov model with new data,

etc.

Now Let us go through this in pseudocode to gain an additional perspective and to be yet more succinct:

AutoPropp Algorithm

```
Preprocess Raw Input and run HMM trainer.
Given PFTML corpus, create topic model of each function using LSA or LDA.
for each candidate passage s where s ∈ Raw Input, do
    - Pick an s (given punctuation and
      other cues) and determine topics
    - Pick a hypothetical function f given
      HMM model of text
    - Determine similarity metric to topic
      given next function in HMM sequence
end for
Calculate for x permutations of s and functions
Pick MAX or top y% from MAX
Annotate texts from Raw Input in accordingly
```

In Figure 1, the AutoPropp research program has been rendered in terms of a digram. The character of the various inputs and outputs should be clear by now.

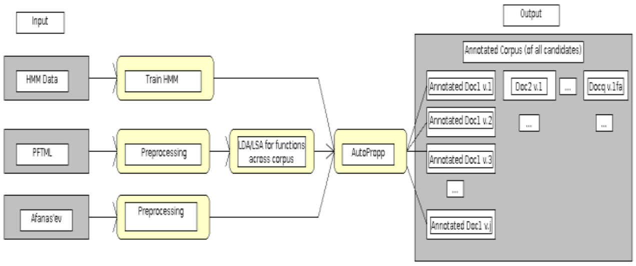

Figure 1: AutoPropp

In the next sub-sections, I will discuss the techniques and sub-components involved in AutoPropp in more depth. For clarity of exposition, the sub-sections below correspond to the sub-sections in section III of this paper, where I outline the packages in R with which I have experimented for implementation purposes.

2.1 Data and Data Preprocessing

2.1.1 XML/PFTML Markup

Approximately 20 tales have been marked up in PFTML. This was painstaking work that I undertook over the course of many excruciating weekends in the late 1990s and early 2000's.

2.1.2 Preprocessing the Data

With unstructured data such as text, one is obliged to preprocess it, i.e., to remove punctuation and stopwords, to remove inflections (stem) it, and the like according to the goals of one's research, the discernment of the individual research, and the power of one's tools to arrive at psychologically compelling results. Many tools exist to make this a fairly painless process. Of course, language is a notoriously complex and shifty substance when it comes to computation. Take, for example, the problem of polysemy – there is no way to distinguish between /bank/ (as in bank of a river) and /bank/ (as in a place where one stores one’s money or takes out a mortgage). Natural language processing can be employed to resolve cases of semantic ambiguity, but for a general bag of words of model, the techniques mentioned above *satisfice* the demands of most projects. The objective, in any case, is to reduce overall variability posed by inflected languages such as English and Russian to a single semantic instantiation of each term.

In the case of AutoPropp, it may ultimately be useful to retain certain types of punctuation, such as semi-colons, commas, and periods, i.e., punctuation that mark clause and sentence boundaries. Punctuation can aid with the efficiency of the story parsing algorithm by providing linguistic delimiters on where the boundaries of hypothetical functions is most likely to be located. This is because it is improbable if not impossible that a boundary of a function (to reiterate, a sub- or supra-sentential block of semantic content that moves a story along without regard to the dramatis personae) would break apart a noun clause (although the same cannot be said for compound sentences).

Specifically, for the purposes of AutoPropp in particular and text data mining projects in general, preprocessing a collection of texts prepares the way to create a document term matrix, a rectangular matrix which may variably count or weigh instances of each term in each document. This matrix may be reduced or decomposed yet further into a Single Value Decomposition, where the matrix is conceived as a product of three matrces – a matrix of terms by the number of terms (U), a matrix of Sigmas, and a matrix of documents (V). Once one arrives at this, a researcher can perform operations upon the document term matrix such as explore the distribution of topics across the corpus using one of the techniques that have been described below, or others.

2.2 Hidden Markov Models

The central idea of hidden Markov models is to evaluate the log likelihood of what the next unknown element might be given what is known about the state of a current, known element. In the case of AutoPropp, this means that if we have an 'E' function, that there might be a .7 probability that it will be followed by some variant of an F function. In other words, this technique gives one insight into the probabilistic relations about the sequence of functions (going both forward and backward through a trellis of observed data). The Baum-Welch and Viterbi algorithms are used to calculate efficient routes through new observations and are commonly deployed in such areas as part of speech analysis and speech recognition.

There is a case where tales may have one or more moves. In the hidden Markov model that I have developed thus far, I have allowed for this case. In short, hidden Markov model is a way of determining how tales are constructed – and, as such, they can be used to dismantle a tales. There may be cases where solutions analogous to those developed in old fashioned transformational-generative grammar may be applied. For example, the surface structure of a text may be inverted, i.e., the function that AutoPropp expects to see first may be second, due to a turn of phrase.

2.3 The Semantic Characterization of Proppian Functions

A sub-hypothesis in this research is that distinct types of content objects, e.g., distinct Proppian functions, will have distinct topic distributions. AutoPropp will experiment with two popular algorithms that are used for inferring topics from collections of texts: Latent Semantic Analysis (LSA) and Latent Dirichlet Analysis (LDA). As input, LSA takes a document term matrix and generates a set of topics based on the distribution of terms across those documents. LDA does much the same. In fact, probabalistic LSA (pLSA) is the equivalent of LDA. The results of these methods are notoriously difficult to interpret. As such, these techniques ought to be considered to be stand-ins for coming developments in the highly dynamic area of topic modeling. The lexical field for each topic model could conceivably be expanded upon accurate match between a segment and a function. The success of this approach has been demonstrated in the question / answer subfield of information retrieval (Celikyilmaz et al. 2010).

2.4 Bayesian Analysis, Finding Associations and Co-Occurrences

AutoPropp has to have a way to measure the similarity of a segment of text, that is, a way of determining whether a hypothesis that a text segment is a member of a set of a particular Proppian function. This could be done with Bayes, with the Weka (http://www.cs.waikato.ac.nz/ml/weka/) packages findAssoc() function[1], or, most simplistically, by calculating the Levenshtein edit distance given semantic matches between the text segment and the topic model for the function to which it may hypothetically belong.

3. R Packages Relevant to AutoPropp

In this section, I will go through some of the packages that will play some role in AutoPropp.

[1] I have turned Appendix III into an .arff file for Waikato's Rweka system. This data can be downloaded, shared, and experimented with in your instance of Weka at your leisure according to the GNU Public Software License from http://www.MalecLabs.com/Propp

3.1 *Preprocessing Revisited with R*

This list is by no means exhaustive, but it is representative of what one would typically encounter when performing the types of research activities that have been described thus far vis-a-vis the AutoPropp project[2].

3.1.1 XML/PFTML Markup

A collection of Russian fairy tales that have been marked up in PFTML can be used to train AutoPropp's model of topics for each Proppian function. This is done by creating a document term matrix of each function, where instead of a collection of documents per se, we have a collection of texts from each function, i.e., a function term matrix. Topic model methods such as LSA and LDA, as discussed above, can then be used to generate a list of topic models for that collection of text segments. With XML, the key is to use the methods available in that package to perform the equivalent of XPATH function and penetrate into PFTMLs data model to deliver the content of a leaf, i.e., the content of a text segment.

```
library(XML); # load the XML library

x ← xmlTreeParse(“95-Morozko.xml”)$doc$children[[1]];
y ← x[“Corpus/Folktale/Move/#”];
```

From there, one may drill depper and venture into more specific areas of content (Proppian functions) within the structure of the PFTML document type.

3.1.2 tm

In the R language, all of the action takes place in the *tm* (text mining) package. Through the methods of the tm package, punctuation and stopwords are stripped and the results are stemmed using *Snowball*, another package, which is required by *tm*. *Snowball*, in turn, uses the famous Porter stemmer; stemmers are available for many language of the Eurasian landmass, including the Russian of the Afanas'ev corpus[3].

The tasks that have been mentioned above (removing punctuation, stopwords, etc.) are performed with the following syntax:

```
library(“tm”); # load the tm library
setwd(“your/working/directory/for/Afanas'ev/”);
# set working directory
afancorpus ← Corpus(DirSource(), readerControl =
list(language = “ru”));
afancorpus ← tm_map(afancorpus, removePunctuation); #
remove ; , . ! ?
afancorpus ← tm_map(afancorpus, stripWhitespace); #
remove whitespace, \n
afancorpus ← tm_map(afancorpus, removeWords,
stopwords(“russian”));
afancorpus ← tm_map(afancorpus, stemDocument); #
expectations → expect
dtm ← DocumentTermMatrix(afancorpus); # “” “”
findFreqTerms(dtm, 5);
# finds the five most frequent terms in dtm
```

From this point, knowing what we know about the *XML* and *tm* packages, we can create document term matrices from document collections. These document term matrices can then be processed into topic models using LSA, LDA, and other techniques.

3.2 *HMM*

Hidden Markov models can be used to predict what function is most likely to follow next in a sequence given the known current state of the AutoPropp Russian magic tale parser. What I have done so far to this end, is to create training data from Appendix III of the *Morphology of the Fairy Tale* (1928). Unfortunately, this is not an exhaustive list, but it does give a window into how the syntax in R's hmm package on CRAN can be used to take observations and to turn those observations to train a model.

This is what some of the sample code and data look like:

```
library(HMM);
afan <- initHMM(c("A", "B", "C", "depart", "D", "E",
"F", "G", "o", "L", "H", "M", "Pr", "J", "I", "N", "Rs",
"K", "return", "Q", "Ex", "T", "U", "W", "X",
"MOVE"),
c("A", "B", "C", "depart", "D", "E", "F", "G", "o", "L",
"H","M", "Pr", "J", "I", "N", "Rs", "K", "return", "Q",
"Ex", "T", "U", "W", "X", "MOVE"),
startProbs=NULL, transProbs=NULL,
emissionProbs=NULL);

observations <- c(c("A", "B", "depart", "D", "E", "F",
"MOVE"),
           c("A", "depart", "Pr", "Rs", "H", "W"),
           c("A", "B", "D", "E", "F", "MOVE"),
           c("A", "depart", "Pr", "Rs", "H", "I", "W"),
        c("A", "B", "D", "E", "F", "return", "MOVE"),
           c("A", "B", "C", "depart", "D", "E", "F",
"return"),
      c("A", "B", "C", "depart", "D", "E", "F", "D", "E",
"F", "return", "MOVE"),
           c("A", "B", "depart", "D", "E", "F", "D", "E",
"F", "return"),
      c("A", "D", "E", "F", "M", "N", "W"));
posteriorProbabilities <- posterior(afan, observations);
logForwardProbabilities <- forward(afan, observations);
logBackwardProbabilities <- backward(afan,
observations);
print(exp(logBackwardProbabilities));
bw <- baumWelch(afan, observations,
maxIterations=100, delta=1E-9, pseudoCount=0);
print(bw$afan);
```

[2] For the latest version of AutoPropp, the data, and related code snippets, visit: http://www.maleclabs.com/Propp.

[3] Should the stemmers included in *tm* prove not to be compatible with one's needs, one can extend R's capabilities by using the *rJava* package to interface other stemmers in other languages, such as these (an aggressive Russian stemmer: http://members.unine.ch/jacques.savoy/clef/RussianAggressiveJava.txt and a light Russian stemmer: http://members.unine.ch/jacques.savoy/clef/RussianLightJava.txt).

```
afanHMM ← initHMM(afan, observations,
startProbs="posteriorProbabilities");
bwafan <- baumWelch(afanHMM, observations,
maxIterations=100, delta=1E-9, pseudoCount=0);
```

3.3 Semantic Analysis

The two methods that AutoPropp will experiment with will be variations of LSA and LDA.

3.3.1 lsa

This is a brief synopsis of the *lsa* package according to CRAN: “The basic idea of latent semantic analysis (LSA) is, that text do have a higher order (=latent semantic) structure which, however, is obscured by word usage (e.g. through the use of synonyms or polysemy). By using conceptual indices that are derived statistically via a truncated singular value decomposition (a two-mode factor analysis) over a given document-term matrix, this variability problem can be overcome.” Below is some sample code from CRAN for *lsa* which plots a distribution of topics.

```
# make a space, reconstruct original
landauerOriginalSpace = lsa(dtm,
dims=dimcalc_raw());
X = as.textmatrix(landauerOriginalSpace);
# X should be equal to dtm (beside rounding errors)
all( (round(X,2) == dtm) == TRUE);
# reduce dimensionality (Y shall be
# the recalculated 'reduced' matrix)
landauerSpace = lsa(dtm, dims=2);
Y = as.textmatrix(landauerSpace);
round(Y,2);
# now read in again the landauer sample (but
# with the vocabulary of the existing matrix)
pdocs = textmatrix(ldir, vocabulary=rownames(dtm));
# now calc a pseudo SVD on the basis of dtm's SVD
Y2 = fold_in(pdocs, landauerSpace);
round(Y2,2);
# Y and Y2 should be the same (as well as
# dtm and pdocs should be equal)
all( (round(Y,2) == round(Y2,2)) == TRUE);
# calc pearson doc2doc correlation
rawCor = cor(dtm);
lsaCor = cor(Y);
round(rawCor,2);
round(lsaCor,2);
# clean up
plot(landauerSpace$dk[,1:2]*landauerSpace$sk[1:2],
pch=17, col="darkgreen");
points(landauerSpace$tk[,1:2]*landauerSpace$sk[1:2],
pch=23, col="darkred");
text(landauerSpace$tk[,1:2]*landauerSpace$sk[1:2],row
names(landauerSpace$tk), col="darkred");
text(landauerSpace$dk[,1:2]*landauerSpace$sk[1:2],ro
wnames(landauerSpace$dk), col="darkgreen");
```

This renders a graph. I have excluded the preprocessing work.

3.3.2 lsa and topicmodels

This is a synopsis of lda (a package that implements latent dirichlet analysis) from CRAN: “These functions use a collapsed Gibbs sampler to fit three different models: latent Dirichlet allocation (LDA), the mixed-membership stochastic blockmodel (MMSB), and supervised LDA (sLDA). These functions take sparsely represented input documents, perform inference, and return point estimates of the latent parameters using the state at the last iteration of Gibbs sampling.”

```
library("tm");
setwd(“your/working/directory/for/Afanas'ev/”);
txt ← system.file("texts", "txt", package = "tm");
library(tm);
plato ← Corpus(DirSource(), readerControl =
list(language = "eng"));
plato ← tm_map(plato, removePunctuation);
plato ← tm_map(plato, tolower);
plato ← tm_map(plato, stripWhitespace);
plato ← tm_map(plato, removeWords,
stopwords("english"));
plato ← tm_map(plato, stemDocument);
dtm ← DocumentTermMatrix(plato);
dtm ←DocumentTermMatrix(plato, control =
list(stemming = TRUE, stopwords = TRUE,
minWordLength = 3, removeNumbers = TRUE));
dtm ← removeSparseTerms(dtm, 0.99);
p_LDA ← LDA(dtm[1:500,], control = list(alpha =
0.1), k = 10);
p_CTM ← CTM(dtm[1:500,], k = 10);
post ← posterior(p_LDA, newdata = dtm[-c(1:500),]);
round(post$topics[1:5,], digits = 2);
get_terms(p_LDA, 5);
```

This renders a 5 x 5 matrix of topics from the input.

3.4 *bayesm* **and** *RWeka for Co-Occurrence*

One solution, would be to use the findAssoc() function to loop through the terms in a text segment in the AutoPropp parser's current window and run the following to determine the level or weight of similarity between the window and what is known about the distribution of terms and topics in a particular function (once library(“RWeka”); has been loaded):

```
findAssoc(fDtm, “term_in_window”, .5);
# this returns a list of terms from the document term
# matrix for function D
# where is is a probability of .5 or greater that there is a
# co-occurrence
```

Other techniques would be Levenshtein edit distance and Bayes with the *bayesm* package.

4. Progress

Accurate, consistent data is the most rare and most precious commodity in an endeavor such as this.

4.1 PFTML – the Training Data Set

Accurate, consistent data, to reiterate, is the most rare and most precious commodity in an endeavor such as this.

4.1.1 PFTML – the training data set

PFTML is a markup language that describes the structure of Russian magic tales as described by Vladimir Propp in his *Morphology of the Folktale* (1928). Each function is conceived as an XML element and each folktale is conceived as consisting of one or more moves. Each move consists of at least one cardinal function (villainy or lack) which may be preceded by an initial situation (such as an admonition) and is followed by such elements as departures, donor functions, pursuits, rescues, and weddings. Each of these functions may be represented by a topic model and is usually in sequence for the given corpus. The PFTML versions of the Afanas'ev text were OCRed between 1999 and 2001 from a 1957 Soviet imprint. These are available at this URL: http://clover.slavic.pitt.edu/sam/propp/theory/propp.html.

4.1.2 Afanas'ev

For this corpus, I copied the each text into its own file. As a source, I am using the following: http://narodnye-russkie-skazki.gatchina3000.ru/ . Since the training set is in Russian, I am stuck for now using Russian as the target set.

4.2 AutoPropp .01

At this point, AutoPropp performs basic functions such as it preprocesses plain text, creates a document term matrix, generates elementary topic models using LSA and LDA, and analyzes sets of observations to create a hidden Markov model of Proppian functions for a subset of Afanas'ev. This code and data have been included on my website at http://www.maleclabs.com/Propp.

4.3 Problems to Solve, Questions to Answer and some Proposed Solutions, where Solutions Exist

In this section, I discuss some of the problems that I encountered or questions that came up since I began this project. These questions will hopefully generate some interesting discourse as I think that they could be alternately perceived as both general and specific enough to provide hours of entertainment and debate among researchers, myself included, in this field.

4.3.1 Size of the Corpus

Is the corpus too small? What is the risk of overfitting the data?

4.3.2 Consistency of the Original Markup

Was I consistent with the original markup? .

4.3.3 ([I]n)fallable Propp

Did Propp get things right? Was Propp consistent with himself in his Appendix III that was used to model the sequence for the hidden Markov trainer?

4.3.4 NULL Data

What if data is not extant for a given function? It may be necessary to focus on a particular function or set of functions (e.g., the "donor" functions D-E-F) for which there is an abundance of data by way of a proof of concept. Dividing and conquering may be the way forward.

4.3.5 Noisy Data

How does one cope with textual noise? R makes it easy to remove sparse terms that fall below a given threshold of frequency within a corpus. How should AutoPropp deal with repetition?.

4.3.6 Learning and Scoring

How will AutoPropp "learn" rules for determining the assignment of text segments as members of a Proppian function? Can it learn incrementally?

4.3.7 What Role might Genetic/Evolutionary Programming Play?

After all, all that genetic/evolutionary programming (GP) is a search algorithm that uses some fitness function to optimum results. What if the windowing function were stochasticized? Might this be a way to turn this entire endeavor from a supervised learning experiment to an unsupervised one?

4.3.8 N-grams

Why not n-gram topic models?

4.3.9 Wordnet

What about Wordnet?

4.3.10 Annotated PFTML (APFTML)

Adding lexico-semantic features to the AutoPropp would help to resolve word and hence topic disambiguation.

5. DISCUSSION

Ultimately, should the proof of concept prove to be successful, a veritable wish list of methods should be included for the user to include in its own package whereby the analyst may activate a feature at will as if it were a black box. At that time, a tender balance will have to be made between elegance of interface and scrappy robustness, but I should not get ahead of myself.

In this paper, I have skirted many difficult issues; for example, what significant does the parsing of Proppian functions, assuming its validity for a moment, have for human communication, how mental story grammars influence the behavior of story tellers in real life, much less pragmatics, the cognitive comprehension (text processing) of stories, how these methods relate to the paradigmatic analyses proposed by Claude Levi-Strauss and Pierre Maranda, or what the implications for this might be for commercial applications such as sentiment analysis.

6. CONCLUSION

I have given what I hope is a useful picture to test (or, conversely), to falsify the epistemic validity of a story grammar, in this case, Propp's. I hereby conclude this paper, but not my research with AutoPropp.

ACKNOWLEDGMENTS

I am indebted to Dr. Sándor Darányi and Dr. Piroska Lendvai, the organizers of the first annual Amicus Workshop for their patience and lively discussion.

REFERENCES

[2] A. Afanas'ev. **Russian fairy tales**. Pantheon Books: New York, 1945. (Transl. Norbert Guterman).

[3] S. Bringsjord and D. Ferrucci. **Artificial Intelligence and Literary Creativity: Inside the Mind of BRUTUS, a Storytelling Machine**. Lawrence Erlbaum Associates. Mahway, New Jersey. 2000.

[4] D. Blei., A. Ng, M. Jordan. **Latent Dirichlet Allocation**. In *Journal of Machine Learning Research*, Volume 3, 2003, pages 993-1022. DOI= http://jmlr.csail.mit.edu/papers/volume3/blei03a/blei03a.pdf

[5] A. Celikyilmaz, D. Hakkani-Tur, G. Tur, **LDA Based Similarity Modeling for Question Answering** in *NAACL 2010 – Workshop on Semantic Search*, 2010. DOI=http://www.eecs.berkeley.edu/~asli/asliPublish_files/naaclhlt2010.pdf

[6] B. N. Colby. **A Partial Grammar of Eskimo Folktales.** *American Anthropology*, Vol. 75, Issue 3, pages 645-662, June 1973.

[7] K. Hornik, and D. Meyer. **Text Mining Infrastructure in R**. In *Journal of Statistical Software*. March 2008, Volume 25, Issue 5. DOI= http://www.jstatsoft.org/v25/i05/paper

[8] P. Lendvai, T. Declerck, S. Darányi, S. Malec. **Propp revisited: integration of linguistic markup into structured content descriptors of tales**. In *Digital Humanities 2010*. London, United Kingdom, Oxford University Press, July 2010. DOI= http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/pdf/ab-753.pdf

[9] P. Lendvai, T. Declerck, S. Darányi, P. Gervás, R. Hervás, S. Malec, and F. Peinado**. Integration of linguistic markup into semantic models of folk narratives: The fairy tale use case**. In *Proceedings of LREC*, 2010. DOI=http://www.dfki.de/web/forschung/iwi/publikationen/renameFileForDownload?filename=Lendvaietal_DH2010_final%5B1%5D.pdf

[10] Maranda, P. and E. K. Maranda. **Structural Models in Folklore and Transformational Essays**. Mouton. The Hague. 1971.

[11] S. A. Malec. **Proppian structural analysis and XML modeling**. In *Proceedings of CLiP, Duisburg, Germany*, December 6-9, 2001.

[12] V. J. Propp. **Morphology of the folktale**. University of Texas Press: Austin, 1968. (Transl. L. Scott and L. A. Wagner).

[13] Rumelhart, D. **On Evaluating Story Grammars** in *Cognitive Science*. Volume 4. pages 313-316. 1980.

[14] Rumelhart, D. **Notes on a schema for stories** in Representation and Understanding: Studies in *Cognitive Science* (Bobrow, D. G. and Collins, A., eds.), pp. 211--236, New York: Academic Press, Inc., 1975.