Narrative Information Extraction with Non-Linear Natural Language Processing Pipelines

A Thesis

Submitted to the Faculty

of

Drexel University

by

Josep Valls Vargas

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

December 2017

Acknowledgments

First and foremost, I would like to thank my advisors, Dr. Santiago Ontañón and Dr. Jichen Zhu for all their guidance and support over the last few years. Then, the members of my PhD dissertation committee, Dr. David Elson, Dr. Mark Riedl, and Dr. Dario Salvucci. Additionally, I would like to thank Dr. Marcello Balduccini, Dr. Christopher Geib and Dr. Rachel Greenstadt for their involvement and valuable feedback in the candidacy and proposal stages of my PhD, and Dr. Michael Mateas and Dr. Luc Lamontagne for their mentorship and support earlier in my PhD. I would also like to thank Dr. Mark Finlayson, who kindly shared his translated corpus and annotations with us, and all the authors, researchers and scholars whose work I referred to throughout this dissertation.

I should not forget all my colleagues, friends and family who supported and encouraged me through this journey. They all contributed to this dissertation and I would not had been able to finish it without them.

Finally, I would also like to thank all the administrative personnel who supported me and my fellow graduate students; the social organizations that kept us sane; and the amazing staff from the recreation athletics center that kept us healthy. All their work is often forgotten but has been an integral part of the last few years of my life.

Contents

List of Tables

List of Figures

Abstract

Narrative Information Extraction with Non-Linear Natural Language Processing Pipelines

Josep Valls Vargas
Santiago Ontañón and Jichen Zhu

Computational narrative focuses on methods to algorithmically analyze, model, and generate narratives. Most current work in story generation, drama management or even literature analysis relies on manually authoring domain knowledge in some specific formal representation language, which is expensive to generate.

In this dissertation we explore how to automatically extract narrative information from unannotated natural language text, how to evaluate the extraction process, how to improve the extraction process, and how to use the extracted information in story generation applications. As our application domain, we use Vladimir Propp's narrative theory and the corresponding Russian and Slavic folktales as our corpus. Our hypothesis is that incorporating narrative-level domain knowledge (i.e., Proppian theory) to core natural language processing (NLP) and information extraction can improve the performance of tasks (such as coreference resolution), and the extracted narrative information. We devised a non-linear information extraction pipeline framework which we implemented in *Voz*, our narrative information extraction system. Finally, we studied how to map the output of *Voz* to an intermediate computational narrative model and use it as input for an existing story generation system, thus further connecting existing work in NLP and computational narrative. As far as we know, it is the first end-to-end computational narrative system that can automatically process a corpus of unannotated natural language stories, extract explicit domain knowledge from them, and use it to generate new stories. Our user study results show that specific error introduced during the information extraction process can be mitigated downstream and have virtually no effect on the perceived quality of the generated stories compared to generating stories using handcrafted domain knowledge.

Chapter 1: Introduction

Computational narrative is an emergent field of research at the intersection of traditional narratology, artificial intelligence, natural language processing and cognitive science. Computational narrative focuses on methods to algorithmically analyze, model, and generate narratives. Moreover, computational models of narrative have applications to tasks such as machine translation, summarization or information extraction in the contexts of reporting, education, and entertainment. For example *Narrative Science* is an enterprise services company leading the transition to artificial intelligence systems that ingest data and automatically generate written reports and provide high level insights[1]. Computational narrative is also a key element in some of the latest trends around digital assistants such as Apple's *Siri*[2], Amazon's *Alexa*[3] and Facebook's chatbots[4] as a novel communication mechanism to engage with their customers. A notable problem in computational narrative is that most of the aforementioned systems and implementations rely on either large amounts of annotated data in a machine readable format for training or, handcrafted models in some knowledge representation formalism or human authored text templates. Moreover, these models and annotations tend to use complex specifications and need to be authored by knowledgeable technical people making their development tedious and expensive. This represents an even heavier burden when considering that the input annotations and models for one system typically cannot be reused for another, leading to the well known authorial bottleneck problem[11;12].

Recent advances in natural language processing (NLP), information extraction (IE) and related disciplines have recently led to a renewed interest in automatically extracting structured information from text. Toward alleviating the aforementioned authorial bottleneck problem, in recent years, the field of computational narrative has experienced cross-pollination with NLP and IE and seen substantial advancements in tasks such as automatically identifying characters[13;14], narrative and plot

[1]https://narrativescience.com/Offers/Outlook-on-AI-Research-Report
[2]http://www.apple.com/ios/siri/
[3]https://developer.amazon.com/alexa
[4]https://developers.facebook.com/docs/messenger-platform

structure[15;16], and character relationships[17–19]. This efforts could enable computational narrative systems to exploit vast amounts of existing material in the form of natural language that is currently available. This would not only allow the automatic processing of the existing body of human literature or text in the Internet, but, would also allow non-technical people to input knowledge into computational narrative systems by using natural language.

In this dissertation we explore how to automatically extract information for computational models of narrative from text, how to evaluate the extraction process, how to improve the extraction process, and how to use the extracted information in story generation applications. We focus our work on several individual tasks that allow for automatically acquiring structured narrative information directly from natural language text. At the same time, we also look at the evaluation of information extraction pipelines and how to improve the performance of narrative information extraction tasks and off-the-shelf general-purpose natural language processing (NLP) systems in the particular domain of fictional stories. We also survey computational models of narrative and how to use the extracted narrative information in existing computational narrative applications. Our hypothesis is that by leveraging existing work in NLP and computational narrative we can ease the development of end-to-end computational narrative systems that can use natural language text as their input.

In the next section we review some open problems in the fields of computational narrative and natural language processing related to the research questions we address in this dissertation. Then we provide a brief outline of this dissertation.

1.1 Open Problems

The main research question addressed in this work is wether we can automatically extract information for computational models of narrative from text and use them in other computational narrative applications. There is a collection of open research questions that are intimately related to the work described in this dissertation. These are open problems that we will discuss and try to address in the following chapters.

Authorial Bottleneck: Some computational narrative systems, such as story generators[4;20;21], or experience managers[22–24], require a significant amount of domain knowledge. Authoring such knowledge is complex and time consuming, leading to the well known authorial bottleneck problem[11;12]. How to ease the generation and encoding of such domain knowledge is still an open problem.

Models of Computational Narrative: Two of the main areas of research within the field of computational narrative are: research focused on modeling existing narratives in order to study existing literature or validate narrative theories; and research focused on expressing narrative spaces in order to study computational creativity and generate stories or interactive experiences. Despite a long history of work on formal models to characterize narratives, starting from the seminal work of Vladimir Propp[25], or more recent AI-based representations such as plans[20;21], frames[4], or plotpoints[22–24], the problem of how to computationally model narratives remains open. Large parts of what constitutes the plot and discourse of narratives, such as authorial intent, conflict, or character subjectivity are not properly captured by existing models[26]. Given the complexity of the current approaches, system builders tend to favor developing customized ad-hoc models to suit their particular implementation needs, leading back to the authorial bottleneck problem previously identified.

Narrative Information Extraction: With recent advances and with a limited scope to standard English language, several key tasks in NLP such as part-of-speech tagging and syntactic parsing are mostly solved, tasks such as coreference resolution or semantic role labeling have seen great progress over the recent years, but other tasks that depend on deep understanding of the text such as question answering, dialog systems, paraphrasing or automatic summarization still pose important problems. Moreover, with the goal of extracting narrative information, when applying existing NLP techniques to specific literary domains such as folktales, we observe higher error rates than previously reported in the literature for other more general domains. We attribute this phenomena to the specific complexities in literary text (which includes fantastic situations, anthropomorphic characters, specific rhetorical figures and different cultural backgrounds) which differs vastly from to the standard corpora used in the NLP community for training and testing NLP techniques. Besides some domain-specific approaches[27], there are no general methodologies to adapt and reuse general-purpose NLP tools for specific domains without requiring corpus re-annotation and retraining the NLP models.

Evaluation of Information Extraction Systems: Information extraction systems usually integrate several modules ranging from off-the-shelf parsers to specialized domain knowledge databases into a pipeline. Basic modules used in NLP and IE pipelines have been studied extensively, often within the context of shared task competitions (e.g., CoNLL coreference resolution shared task[28;29]). These studies are typically conducted on a single module isolated from a pipeline since no standard methodology exists to evaluate how error propagates in pipelined architectures. However, when the information extracted by the system is not accurate, it is hard to pinpoint which is the module responsible since there is no general methodology that can be applied to arbitrary pipelined information extraction systems in order to evaluate how error propagates down the pipeline and affects the final output of the system.

1.2 Thesis Statement

The central idea of this thesis is:

We can automatically extract structured narrative information from text and use the extracted narrative information in computational narrative applications. Moreover, our hypothesis is that incorporating narrative domain knowledge into the extraction process will improve the performance of certain tasks within the extraction process and the final output of the process.

With this research statement to frame our work, in this dissertation we will try to answer the question: *How do we automatically extract structured narrative information from text?*. In order to answer this question we need to look at the narrative elements we are interested in extracting and how to extract them. In our work we focused on several individual tasks related to natural language processing and information extraction which we will assemble together using a pipeline architecture.

Once we have a system capable of automatically extracting narrative information from text, we will address follow-up questions such as *How do we evaluate and improve narrative information extraction pipelines?*. Finally, and in order to tie our work with the first and second open problems described in the previous section, we will also address the question *How do we use the extracted information in computational narrative systems, specifically, story generation applications?*.

1.3 Dissertation Organization

The rest of this dissertation is organized as follows:

Background and Related Work: In this chapter we first introduce some background concepts relevant to our research and that will be used throughout the rest of the document, specially related to natural language processing (NLP). Then we will present the fields of narratology and computational narrative, review related work in the areas of information extraction, computational narrative, how to model and annotate narratives, and finally some applications that illustrate how computational narrative models can be used.

Proppian Stories: In this chapter we expand on describing the work of Vladimir Propp, a Soviet folklorist and scholar who studied Russian and Slavic folk tales. We also talk about the folk tales collected by Alexander Afanasyev, the Russian counterpart to the brothers Grimm and describe the corpus and datasets used throughout our work.

Automated Narrative Inf. Extraction: In this chapter we describe *Voz*, our narrative information extraction pipeline that has become the infrastructure of our experimental evaluation and the individual contributions that we incorporated into that pipeline. We first describe the individual tasks we addressed and then we describe our proposed framework to assemble those task into a non-linear information extraction pipeline. Finally we describe additional tasks we worked on and incorporated as modules of the pipeline.

Evaluation of IE Pipelines: In this chapter propose a methodology designed to assess how error is introduced and propagated in information extraction pipelines and apply it to *Voz*, our narrative information extraction pipeline.

Non-linear IE Pipelines: In this chapter we revisit the feedback loop introduced in Chapter 4 and describe our proposed general purpose framework for defining non-linear pipelines. We then present an empirical study of this methodology applied to *Voz* for two different tasks.

Bridging the Gap: In this chapter we present a taxonomy of computational models of narrative reported in the literature and outline our approach to connect *Voz* to an existing story generation system in order to implement a full-fledged end-to-end, text-based, story generation system. We report our analytical findings and the results of a user study on the system.

Conclusions and Future Work: In this chapter we summarize our work and revisit our contributions and how we addressed the open problems described before. We conclude the dissertation with some directions for future work.

In the appendices we provide additional information about the Proppian stories used for our work, detailed results from our work and early work that motivated the research presented in this dissertation.

Chapter 2: Background and Related Work

In this chapter we first introduce some background concepts relevant to our research and that will be used throughout the rest of the document and review related work in the areas of information extraction and computational narrative, how to model and annotate narratives, and finally some applications that illustrate how computational narrative models can be used.

2.1 A Brief Introduction to Natural Language Processing

The field of natural language processing (NLP) has received a lot of attention since the early days of computing and artificial intelligence. NLP has had successful applications in areas such as information extraction or machine translation. The history of NLP generally starts in the 1950s. Some notably successful NLP systems were developed in the 1960s but up to the 1980s, most NLP systems were based on complex sets of hand-written linguistic rules. Starting in the late 1980s, however, there was a revolution in NLP with the introduction of statistical methods and machine learning algorithms for natural language processing[30].

Work in NLP in the last decade has been dominated by unsupervised and semi-supervised learning algorithms on large annotated corpora (i.e. treebanks)[15;31] and statistical work on large datasets and exploiting the web[10]. The latest trends involve using deep and or recurrent neural networks to tackle specific NLP tasks. We will discuss these at the end of this section.

NLP is a broad research area addressing a wide range of different problems. Some of the most prominent subfields and concentrations within NLP, relevant to this dissertation, are:

- **Information Extraction (IE)**: focuses on automatically extracting structured information from unstructured textual documents. IE plays a big role in document classification, indexing and information retrieval[32].
- **Text Understanding**: also known as machine reading, text understanding is related to IE but differs in that its scope goes beyond information storage and seeks to extract a structured

representation that enables further computation, such as problem solving applications to operate on natural language input[33]. Text understanding implementations usually complement the information obtained from text with additional common sense knowledge used to perform inference and address problems such as question answering.

- **Text Generation**: also known as natural language generation (NLG), this field studies methods to convert computational representations of information into natural language that can be used to interface with a human audience[34]. Dialogue agents or interactive fiction systems use NLG to communicate an internal state to the user. Paraphrasing and summarization are text generation tasks heavily related to text understanding.

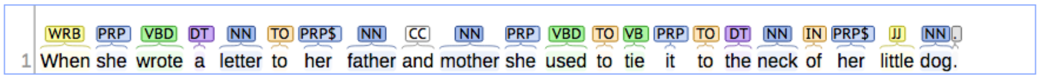
Work on the aforementioned fields shares a collection of tasks and subtasks commonly referred to "core" NLP tasks. In the remaining of this subsection we describe and provide an overview on some of these core NLP tasks we will discuss later in the context of automatically extracting computational models of narrative.

2.1.1 Segmentation, Chunking and Tokenization

Given a string of characters representing a chunk of text (or document) these tasks involve segmenting words, sentences or other elements like paragraphs or individual morphemes from compound words. Sentence segmentation and document chunking are usually implemented using a combination of rule-based methods and heuristics. Tokenization usually involves punctuation splitting and separation of some affixes like possessives. For English, state-of-the-art, high-performing implementations rely on finite-state machines[35]. Other languages with different spacing rules such as Chinese rely on statistical approaches[36]. Other languages such as Arabic require more extensive token pre-processing[35].

2.1.2 Morphological Analysis

Related to the previous tasks, given one or more tokens or words, the goal of this task is to identify individual morphemes and recognize the lemma, stem or root for inflected words along with inflection information (e.g., "singing" $\Rightarrow$ sing $\mapsto$ *lemma*, -ing $\mapsto$ *present continuous*). This task includes other


WRB PRP VBD DT NN TO PRP$ NN CC NN PRP VBD TO VB PRP TO DT NN IN PRP$ JJ NN .

When she wrote a letter to her father and mother she used to tie it to the neck of her little dog.

Figure 2.1: Part-Of-Speech (POS) tagging example.

operations such as word segmentation, lemmatization or stemming. Implementations for this task are usually language dependent. Rule-based and heuristic approaches have been developed for English[37] and have been improved by adding dictionaries and contextual information to resolve ambiguities[35].

2.1.3 Part-Of-Speech (POS) tagging

Given a sentence, this task determines the part-of-speech each word plays for a specific language grammar (e.g., "the red truck" $\Rightarrow$ the $\mapsto$ *determiner*, red $\mapsto$ *adjective*, truck $\mapsto$ *noun*). Implementations for this task are also language dependent and a mixture of statistical and rule based approaches have demonstrated high accuracies for English[35;38]. Figure 2.1 illustrate this task for processing English using the Stanford CoreNLP.

2.1.4 Grammatical and Syntactic Parsing

Given a sentence, the goal of this task is to generate the structures of the syntactic and grammatical roles and relationships between the words and phrases (e.g., "My truck is red." $\Rightarrow$ *(S (NP (PRP$ My) (NN truck)) (VP (VBZ is) (ADJP (JJ red))) (. .)))*. Approaches to syntactic parsing typically combine formal grammars with statistical models to deal with the complexity of natural language. The use of simple grammars such as context free grammars is not feasible due to the ambiguity and size of the lexicon of any typical natural language. Common approaches use probabilistic context-free grammars (PCFG), which extend context-free grammars by assigning probabilities to production rules. The probability of a derivation (parse) is the product of the probabilities of the productions used in that derivation. A trained model including the production rules and the probabilities is typically computed by machine learning algorithms over large annotated corpora (called treebanks). Lexicalized PCFGs (where head words annotate phrasal nodes) are used for high performance PCFG parsing. Great success in terms of parse disambiguation and language modeling has been achieved by various lexicalized PCFG models[39]. Additional work combined lexicalized and

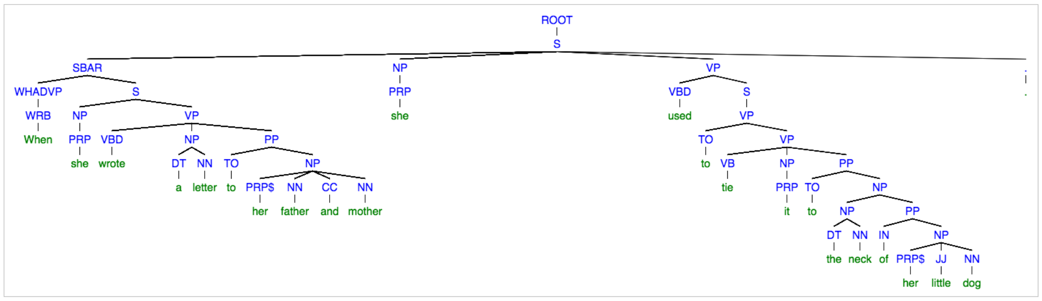


Figure 2.2: Syntactic parse example.

unlexicalized PCFG for phrase structure and lexical dependency parsing[39]. Recent work has focused on languages other than English, parsing ungrammatical sentences (e.g., microtext) and semantic parsing. These approaches use combinatory categorial grammar[40], conditional random fields and neural networks[41;42]. Figure 2.2 illustrate the output of this process using the Stanford CoreNLP.

2.1.5 Named Entity Recognition (NER) and Information Extraction (IE)

Given a document, the goal of named entity recognition is to identify items in the text that map to specific entities and recognize the type of each such entity (e.g., "Barack Obama is the president of the United States" $\Rightarrow$ Barack Obama $\mapsto$ *person*, United States $\mapsto$ *country*). Broadly speaking, information extraction is tasked with extracting other items of interest such as relationships between entities. Common approaches use patterns and regular expressions matched against the text of the document in order to identify the entities or items of interest. For example, regular expressions can define character level patterns for identifying named entities in specific domains of bioinformatics (e.g., protein and gene names)[43]. Similar to regular expressions, *Tregex*[44], are a collection of utility classes that allow matching patterns in trees, based on tree relationships and regular expression matches on nodes from different annotation layers added by different tasks such as POS-tags or parse information. Another prominent example of such approach is the *Sundance Parser*[45]. Sundance uses patterns that match certain tokens and grammatical categories and because those patterns may be complex or not straightforward, the system is complemented with the *AutoSlog* utility to extract such patterns from a given set of examples. Another approach for extracting information from text is to use noun-verb tuples derived from the dependency parse of the document[43;46;47].

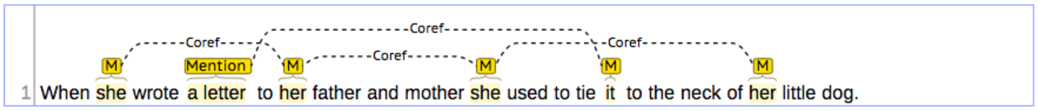

Coref
Coref
Coref
Coref
M
Mention
M
M
M
M
1 When she wrote a letter to her father and mother she used to tie it to the neck of her little dog.

Figure 2.3: Pronominal coreference resolution example.

2.1.6 Coreference Resolution

Given a document, the goal of this task is to determine which referring expressions (i.e. mentions in the text) refer to the same entities (e.g., "Alice is a beautiful girl. She is also intelligent." ⇒ *She* ↦ *Alice*). Automatic identification of corefering entities and events has been a difficult problem in NLP because in many situations may require inference and common sense knowledge, and because of the limited amount of annotated training data[48]. State of the art techniques use entity-centric, precision-ranked rules learned using machine learning[29;49]. Moreover, some systems tend to incorporate ad-hoc domain-specific rules and heuristics to solve coreference[16] due to the difficulty of this task beyond pronominal coreference. Figure 2.3 illustrate this process using the Stanford CoreNLP.

2.1.7 Semantic Role Labeling (SRL)

Given a verb with multiple arguments (i.e. subject, direct object and multiple indirect objects), the goal of this task is to map the arguments to semantic roles (e.g., "Bob bought a computer from Alice." ⇒ *Bob* ↦ *buyer*, *Alice* ↦ *seller*, *computer* ↦ *good being sold*). Some of the best performing implementations use hybrid approaches of machine learning, inference and constraint satisfaction[50]. Recent work also uses convolutional neural networks[41].

2.1.8 Word Sense Disambiguation, Common Sense and Domain Knowledge

Given a token or multi-word expression with multiple possible senses, the goal of this task is to identify the intended sense in the context it appears (e.g., "He was sent to the pen for burglary." ⇒ *pen* ↦ *penitentiary*) and map it to a database with additional knowledge that can be used for further inference (e.g., *penitentiary* ↦ *state institution*). Although there are some hybrid statistical and rule-based approaches[51], this task usually involves mapping to rich databases of common sense and domain knowledge[52;53] using contextual information and inference. Common sense and domain knowledge are studied in the field of knowledge representation which seeks methods for encoding general common sense or specific domain knowledge so it can be exploited by NLP applications[54].

2.1.9 Natural Language Processing Pipelines

Most of NLP applications, and specially information extraction systems, involve one or more of the NLP tasks presented in the previous enumeration. Moreover, some tasks such as parsing, typically depend on part-of-speech tags which in turn depend on properly tokenized text. Because of this interdependency, a pattern for a pipeline has emerged and pipelined architectures are used in many NLP tasks[55], and are specially prominent in text understanding, information extraction[56;57] and natural language generation[34]. Figure 2.13 shows a workflow diagram for this pipeline[9;58].

There are several projects readily available that implement different modules in an NLP pipeline and are widely used in both industry and academic environments[35;55]. For example, *Stanford CoreNLP* is a package that includes Java implementations of several Stanford NLP tools including the POS tagger, the named entity recognizer (NER), the Stanford parser, the coreference resolution system, and the sentiment analysis tools, and provides model files for processing English text [1]. *Natural Language Tool Kit (NLTK)* is a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning written in Python with interfaces to external tools and corpora. A book is also provided to introduce technical and non-technical people to developing applications that feature NLP and computational linguistics [2]. *Apache OpenNLP* is another Java toolkit for the processing of natural language text supporting the most common NLP tasks, such as tokenization, sentence segmentation, POS tagging, named entity extraction, chunking, parsing, and coreference resolution. OpenNLP also includes maximum entropy and perceptron based machine learning algorithms and models [3]. *General Architecture for Text Engineering (GATE)* is an integrated development environment for building NLP applications. It provides a user interface for several text processing workflows (including annotation and visualization), and includes ANNIE (A

[1]Available: http://nlp.stanford.edu/software/corenlp.shtml
[2]Available: http://www.nltk.org/
[3]Available: http://opennlp.apache.org/

Nearly-New Information Extraction System) focused on IE tasks on top of many common NLP tasks and can be used programmatically using its Java API [4]. *The Curator (U. of Illinois)* is a Java suite integrating several natural language processing tools including a semantic role labeling (SRL) tool using verb frame information [5]. Combining several third-party systems, toolkits like *OpinionFinder* (U. of Pittsburg, Cornell and U. of Utah)[6] provide means of identifying subjectivity and sentiment analysis along various other information extraction tasks. It is worth noting that recent work avoids pipelines altogether for some tasks by using neural networks. Fonseca et al. proposed an architecture based on deep convolutional neural networks that accomplish several NLP tasks and implemented NLPNET[7] that implements POS tagging, dependency parsing and semantic role labeling[42].

2.1.10 Natural Language Generation

Related to natural language processing, the field of natural language generation (NLG) focuses on methods to convert computational representations of information into natural language that can be used to interface with a human audience[34]. Story generation and interactive fiction systems use NLG to communicate an internal representation of the story to the user. Other related areas of application include dialogue agents, machine translation paraphrasing and text summarization.

There are several techniques to approach different NLG tasks; focusing on computational narrative and story generation systems, usually there is a specialized module often referred as a *realizer* or *surface realizer* that produces the final output from the given intermediate representation. For this task, simple template-based approaches have been widely used with successful results[59]. These template-based solutions (often referred to as *canned text*) are easy to implement and appropriate for some situations, however, have several limitations; the main drawback is the necessity of authoring a large database of annotated templates. The resulting systems are relatively inflexible and their expected output has to be anticipated ahead of time. Additionally, these tend to work best when the expected output is brief[60]. Figure 2.4 shows a simple synthetic example of a template that could be used to generate natural language text for a flight reservation system using substitutions.

[4]Available: http://gate.ac.uk/
[5]Available: http://cogcomp.cs.illinois.edu/page/software_view/Curator
[6]Available: http://mpqa.cs.pitt.edu/opinionfinder/
[7]Available: https://github.com/erickrf/nlpnet

```
Searching flights from :SRC to :DEST on :DATE. One moment please.
```

Figure 2.4: Example of a template that could be used to generate natural language text. Tokens starting with a colon are placeholders for substituting the relevant information to be shown to the final user.

Modern text generation systems are designed to produce fluent multiparagraph text in response to a goal presented to the system. A commonly used architecture implements a pipeline with four major modules: a domain-specific knowledge module that completes gaps given an internal data representation and a communication goal; a high-level text planning module that organizes the relevant information and decides how to present the content to the reader; a sentence generation module based on a large systemic grammar of the desired output language; and an evaluation and plan-perturbation module which revises generated text generation plans and adapts them as required[34]. Recent research has focused data-driven and trainable systems that can learn directly from a corpus of text[61].

2.1.11 Neural Networks for Natural Language Processing

As mentioned earlier in this chapter, the latest trends in NLP involve the use of deep and recurrent neural network approaches. Instead of pipelines of individual core NLP tasks that build upon one another to achieve higher-level tasks (as described in Section 2.1.9), the latest developments exploit access to large datasets or corpus of text which are processed directly by statistical methods such as deep and recurrent neural networks. These networks are usually fine-tuned for task specific approaches but have shown high performance and great flexibility in a number of areas.

Neural networks were first used to find novel encodings for text information and phrase and word representations that could then be operated onto for different tasks (e.g., word embeddings)[62]. These have been used to beat the state-of-the-art on other methods discussed earlier in this chapter in tasks such as coreference resolution[63] but also to address higher-level composite tasks such as document modeling and sentiment analysis [64] and all the way to the implementation of fully-fledged machine understanding and question answering systems, directly from text[65].

The same family of approaches have been used in other areas such as natural language generation (described in Section 2.1.10), where pipeline-based approaches have yield to neural network architectures as well. Again, here we see different neural network architectures to tackle specific high-level tasks that involve natural language generation ranging from generating captions for images[66] to fully fledged machine-translation systems[67] and conversational agents for question answering[68]. Relevant to our research, at the end of the next section we will discuss work on story generation using recurrent neural networks[69;70].

2.2 Narrative

Narratives are found in all kinds of everyday situations: from video games to movies or news reports to scientific articles and informal conversations with friends. Narratives are used to communicate complex ideas, deliver precise accounts of events or argument about one's beliefs. Narratology work in the early XIX century focusing on folkloric fantasy stories[25] provided ground work for narrative theory[71] and has been revised and updated for the study, systematic analysis and criticism of contemporary media[72]. From a computer science perspective, since the 1970s, narratives and storytelling have been a prominent test bed for evaluating human cognition theories and validating the scientific method in the field of artificial intelligence[73;74]. Rooted in narratology, computational narrative bridges humanities and computing fields by analyzing and modeling narratives, narrative understanding in terms of human cognition, and computer-readable representations of narrative spaces that enable computers to become storytellers [75;76].

2.2.1 Narratology

Narratology is a discipline dedicated to the study of the logic, principles, and practices of narrative representation and storytelling, including common themes, conventions and symbols. Core elements and ideas for narratological modeling of narrative were introduced by Greek philosophers. Narratology has been dominated by structuralist approaches since the 1900s, and has developed into a variety of theories, concepts, and analytic procedures[26]. There are several branches and theories but for the scope of this dissertation we will focus on thematic narratology, tightly coupled with the study of the plot of a narrative; and cognitive narratology that focuses on the reader's interpretation of a story.

Thematic narratology focuses on the semiotic formalization of the sequences of the actions told in the narrative. One of the main influences was the structuralist work of Propp[25]. Propp focused on an empirical analysis of Russian folk tales and presented a model of the elementary components of narratives and the way they are combined. Propp describes Russian and Slavic fairy tales in terms of a sequence of abstract "functions" that are fulfilled by characters with specific narrative "roles". Lakoff[71] operationalized Propp's theory and presented a story grammar based on Propp's functions providing ground work to structuralist approaches to narratology that in turn influenced some of the computational approaches described in the next section. Similar to Propp's work but with a broader scope, Joseph Campbell's monomyth, or the hero's journey[77] has been used widely in the study of narratives from around the world and the criticism of contemporary media.

In contrast to thematic narratology that focuses on the plot of a story (i.e., what is being told), modal narratology[78] focuses on the discourse, that is, the way a story is told during a storytelling session, stressing voice, point of view, transformation of the chronological order, rhythm and frequency. Chatman[72] presented a unified study of narrative that defines a dual model that separates but relates the plot and the discourse of a narrative. Furthermore, Chatman provides a taxonomy of characteristic elements from the plot of a narrative and the means in which the narrative is communicated (Figure 2.5) that we will reference later in this thesis.

Cognitive narratology[79] focuses on the human intellectual and emotional processing of narratives and looks at the interpretation of the character's purposes and motivations in order to infer specific narratology primitives like volitional cause and logical sequences of actions. Related to cognitive narratology, we will later make reference to the concept of the hermeneutic circle, developed in the age-old field of hermeneutics as the study of human interpretation of complex text. When making sense of the text, scholars argue that we move back and forth between its individual parts and the larger whole — our understanding of the text as a whole is hinged on our understanding of the individual parts, which in turn is shaped by how we understand the larger whole. This circular

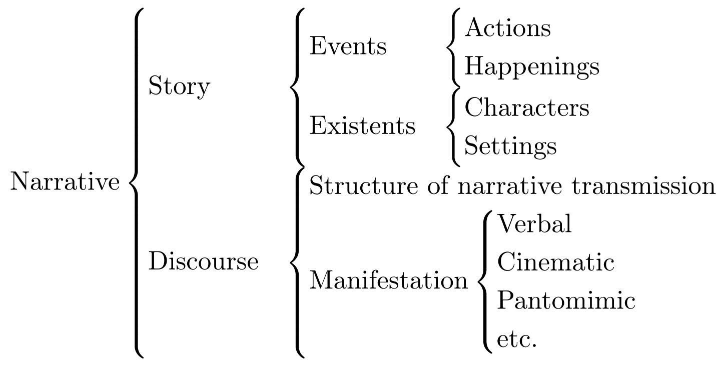


Figure 2.5: Chatman's taxonomy of narrative elements. In Chatman's work, plot is also referred as *story* or *content*; discourse is also referred as *expression*.

process of understanding text is the most basic form of *the hermeneutic circle*[80]. Work in the area of story understanding is grounded in the early work of Roger Schank[81], focused on the higher level constructs humans use to understand, memorize and reason about stories. In their work they introduce the concept of *scripts* as a method of representing chains of events or episodic knowledge, such as the sequence of events in the plot of a story.

2.2.2 Computational Narrative

Computational narrative is an emergent field at the intersection of traditional narratology, artificial intelligence and natural language processing. Computational narrative studies how to algorithmically represent, understand, and generate stories. Computational narrative has applications in areas of digital entertainment such as digital storytelling[4;20;21] or experience management[22–24;82] and can provide insights into computational creativity[83] and the analysis and understanding of literature[8;84]. The field of computational narrative itself is coarsely divided in two groups: research focused on modeling existing narratives in order to study literature or validate narrative theories[8;26;84;85]; and research focused on representing narrative spaces and algorithms to generate stories or interactive experiences and study computational creativity[12;21;86].

2.3 Computational Models of Narrative

Generative storytelling applications such as story generation systems and interactive fiction require computational models abstracted from a specific story and that instead define a story space. On the other hand, other applications rely on machine readable versions of a story, often annotated over an existing text. Regardless of the application, different computational models of narrative usually focus on specific elements of a narrative depending on the domain and application and although there are some efforts toward holistic models of narrative[26], usually there is a separation between modeling plot (i.e. what happens in the story) and the discourse (i.e. how the story is told).

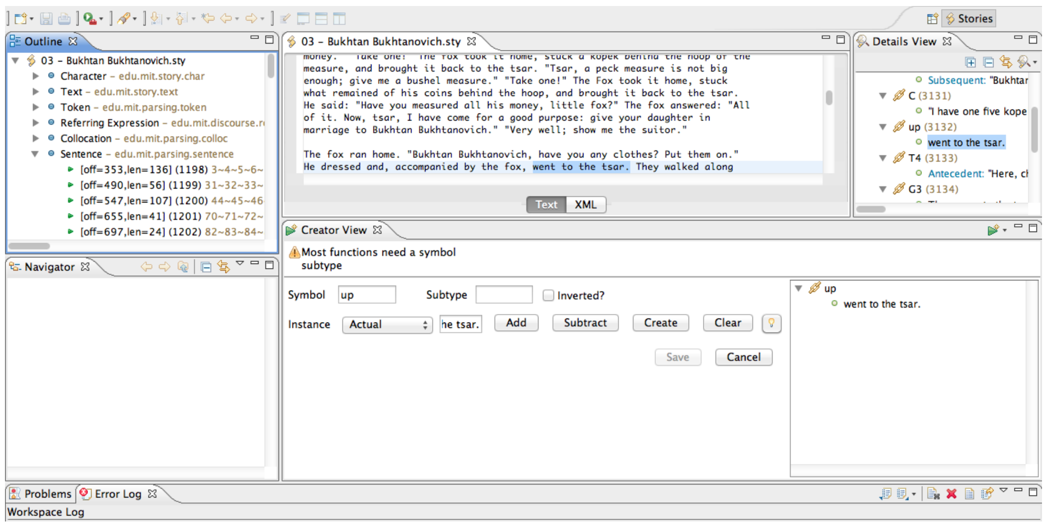
Discourse models deal with high-level narrative primitives, sequence, point-of-view (i.e., focalization), the audience mental model and embedded narratives.

When looking at computational models for plot, several different elements such as the elements in Chatman's taxonomy, have also been modeled separately. Considering a minimal model of the plot of a story as a chronologically ordered set of plot points, ideas from the field of automated *planning*, have been used to model events in the plot of a story with preconditions and postconditions for defining temporal and causal link relationships between events. *Tale-Spin*[5] is an early example that used a planning approach. More recently, planning approaches have been used to encode authorial intent[87] and encode a story space around an event[10].

Annotated narratives and plot sequences have also been used in *case-based reasoning* approaches to drive the plot of story generation systems and storytelling in interactive systems[6;88;89]. Models of the *existents* or the *space* of a narrative have also been used in planning, simulation and multi-agent systems to generate virtual environment[76;89;90]. Also related is the use of a model of characters and a social model of interaction to drive an interactive storytelling system[91].

2.3.1 Annotating Narratives

A common step to model a narrative is to author a computational model or machine readable version of a story from an existing text. Several annotation schemes and annotation environments have been proposed that combine layers of discourse and plot annotations on top of natural language



03 – Bukhtan Bukhtanovich.sty
- Character – edu.mit.story.char
- Text – edu.mit.story.text
- Token – edu.mit.parsing.token
- Referring Expression – edu.mit.discourse.r
- Collocation – edu.mit.parsing.colloc
- Sentence – edu.mit.parsing.sentence
  - [off=353,len=136] (1198) 3~4~5~6~
  - [off=490,len=56] (1199) 31~32~33~
  - [off=547,len=107] (1200) 44~45~46
  - [off=655,len=41] (1201) 70~71~72~
  - [off=697,len=24] (1202) 82~83~84~

```
measure, and brought it back to the tsar. "Tsar, a peck measure is not big
enough; give me a bushel measure." "Take one!" The Fox took it home, stuck
what remained of his coins behind the hoop, and brought it back to the tsar.
He said: "Have you measured all his money, little fox?" The fox answered: "All
of it. Now, tsar, I have come for a good purpose: give your daughter in
marriage to Bukhtan Bukhtanovich." "Very well; show me the suitor."

The fox ran home. "Bukhtan Bukhtanovich, have you any clothes? Put them on."
He dressed and, accompanied by the fox, went to the tsar. They walked along
```

- Subsequent: "Bukhtar
- C (3131)
  - "I have one five kope
- up (3132)
  - went to the tsar.
- T4 (3133)
  - Antecedent: "Here, cl
- G3 (3134)

Most functions need a symbol subtype

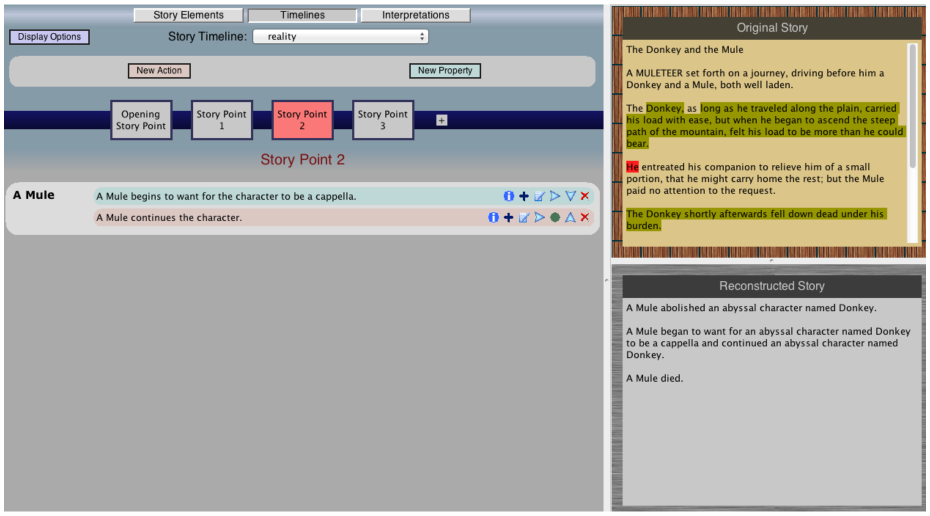
Symbol: up Subtype: Inverted? [ ]
Instance: Actual he tsar.

- up
  - went to the tsar.

Figure 2.6: Screenshot of the Story Workbench annotation tool by Finlayson [1].

text. These annotation schemes add different layers of computer readable information representing different parts of the narrative. The annotations can then be used to extract different models of the plot or discourse of the narrative for further processing. For example a model of a sequence of states from a story can be extracted in the form of a script-like representation for plot points [92]. The *Story Workbench* project[8] by Finlayson [1] provides an integrated framework supporting several annotation layers. It provides a number of common text annotation operations, including representations (e.g., tokens, sentences, parts of speech), functions (e.g., generation of initial annotations by algorithm, checking annotation validity by rule, fully manual manipulation of annotations) and tools (e.g., distributing texts to annotators via version control, merging doubly-annotated texts into a single file). It was designed to support multiple annotation workflows and it integrates with version control systems. Figure 2.6 shows a screenshot of the Story Workbench. A related approach is the Scheherazade system[9] by Elson [2]. The Scheherazade system is a platform for symbolic narrative encoding. It provides tools to define common sense knowledge regarding a story and then a graphical interface for manually encoding narrative semantics such as timelines, states, events, characters and goals. Although narrative primitives can be mapped to text, the system can be used for generating symbolic representations of a story independent from the text. Figure 2.7 shows a screenshot of

[8] Available: http://projects.csail.mit.edu/workbench/
[9] Available: http://www.cs.columbia.edu/~delson/software.shtml



A Mule

A Mule begins to want for the character to be a cappella.

A Mule continues the character.

The Donkey and the Mule

A MULETEER set forth on a journey, driving before him a Donkey and a Mule, both well laden.

The Donkey, as long as he traveled along the plain, carried his load with ease, but when he began to ascend the steep path of the mountain, felt his load to be more than he could bear.

He entreated his companion to relieve him of a small portion, that he might carry home the rest; but the Mule paid no attention to the request.

The Donkey shortly afterwards fell down dead under his burden.

A Mule abolished an abyssal character named Donkey.

A Mule began to want for an abyssal character named Donkey to be a cappella and continued an abyssal character named Donkey.

A Mule died.

Figure 2.7: Screenshot of the Scheherazade annotation tool by Elson[2].

the Scheherazade. Despite both Story Workbench and Scheherazade aiming to be general purpose tools, with no consensus on computational models to represent narrative, researchers tend to develop ad-hoc approaches to suit their particular implementation needs that are time-consuming and expensive to generate. For example the *Riu* system (described in the next section) uses a complex encoding format to represent the story space and annotate the text of the story. Figure 2.8 reproduces an excerpt of Riu's story format showing the story space structure (under *:structure*) and text annotations (under *:templates*).

2.4 Applications of Computational Narrative

Computational models of narrative have many applications. In this section we summarize some applications from the digital entertainment research community.

Narrative Generation and Storytelling

Narratives are complex intellectual products. Additionally, storytelling is an activity that requires a wide range of skills and cognitive abilities intrinsic to humans. Despite that, there have been research efforts aiming at achieving computers inventing and telling stories. In the context of storytelling, creativity implies inventing a satisfactory story in terms of believable characters, their personalities, goals, feelings and emotions; interesting situations and events; and a discourse that facilitates the

```
\resizebox{\textwidth}{!}{
(setf *riu-STORY-DAG*
 '(
  (STORY-6-FACTORY-JOB
   (:discourse
    (:clauses
     (c0 (:s phase1 phase2))
     (phase1 (:s t1a t2a t3a t4a (:m phase1)))
[...]
    (:templates
     (t1a "After counting street numbers and puzzling over the scrawl of a 4,
      which turned out to be a 9,")
     (t2a (s6-notice (ales "Ales") " found a " (s6-in-door (s6-small-door
      "small " (door "door")) " in the side of an " (s6-big-factory "immense "
      (factory "factory"))) " that matched the Aristobots description."))
     (t3a "Soon after knocking, " (s6-open "a solid, " (workbot "grubby
      workbot") " opened up") " and greeted " (ales "Ales") " with a scowl of
      pure skepticism.")
     (t4a (s6-explain (ales "Ales") " stammered out his situation, describing in
      too great of detail the recent " (theft "theft incident,")))
[...]
   (:structure
    (common
     (:entities (ales :type robot) (robot :type animate) (workbot :type robot)
      (factory :type inanimate))
     (:expressions
      ((big factory) :name s6-big-factory)
     )
    )
    (phase1
     (:entities (door :type inanimate) (theft :type inanimate))
     (:expressions
      ((small door) :name s6-small-door)
      ((big factory) :name s6-big-factory)
      ((notice ales door) :name s6-notice)
      ((tell ales workbot theft) :name s6-explain)
      ((in door factory) :name s6-in-door)
      ((open workbot door) :name s6-open)
      ((fd-agonist ales phase1) :name s6-p1-agonist)
      ((fd-antagonist workbot phase1) :name s6-p1-antagonist)
      ((fd-stronger s6-p1-agonist phase1) :name s6-p1-strong-agonist)
[...]
}
```

Figure 2.8: An excerpt of the story format used by *Riu*[3;4].

Joe Bear was hungry. He asked Irving Bird where some honey was. Irving refused to tell him so Joe offered to bring him a worm if he'd tell him where some honey was. Irving agreed. But Joe didn't know where any worms were, so he asked Irving, who refused to say.

Figure 2.9: An excerpt of an example story generated by *Tale-spin*[5].

reader to understand the characters' motivations, associate characters intentions with feelings, and develop empathy towards the characters and situations[12].

Tale-spin[5] is a story generation system that generates stories of woodland creatures using backward chaining for resolving goals and subgoals and forward chaining to compute consequences of narrative events. Tale-spin simulates a small world populated with characters, each with their own problems and motivations. Each character uses planning in order to satisfy their own goals given the current world state. Complex relations between characters are modeled using simulation over the events and the outcome is used to select plot points with satisfied preconditions. After the simulation and problem-solving phase, a separate component is used to generate a textual description of the generated narrative. Figure 2.9 shows an excerpt of an example story generated by *Tale-spin*.

Minstrel[83] tells stories about the Knights of the Round Table. Minstrel uses case-based reasoning instead of planning, and introduces concepts such as author goals in order to guide the plot generation. Minstrel implemented a general theory of creativity based around the concept of Transform Recall Adapt Methods (TRAMs). TRAMs are used to query an episodic memory and retrieve schemas or scripts that define the story. In the case that a query fails, the query process allows TRAMs to be modified and additional queries issued recursively, the output of which can be combined and adapted to match the initial query constraints. *ProtoPropp*[93] generates stories step by step by using a simplified Case-Based Reasoning approach with a Proppian ontology that transforms annotated stories using a simulator and explicit domain knowledge. *Riu*[3;4] is a text-based interactive storytelling system. Riu uses computational analogy with a force-dynamics based story representation to create stories. The narrative oscillates between two worlds: the main story world and the memory world that share parts of the structure and influence each other. Riu uses

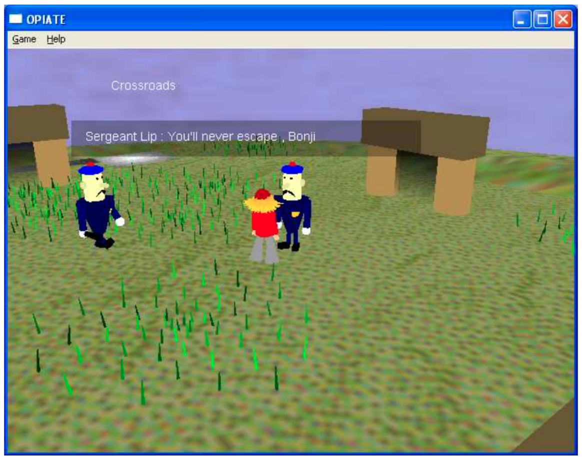


Figure 2.10: Screenshot from an interactive storytelling session in OPIATE by Fairclough[6] (reproduced with permission).

two repositories of authored data encoding the story world and the protagonist's memories and an analogy module based on the *structure-mapping engine*[94] finds mappings between scenes.

Storytelling not only happens in written stories but also in other media and realizations such as computer games. Computational models of narrative have been used to achieve systems that automatically adapt the plot of a game to maximize engagement of the player and maintain a dramatic arc defined by the author. For example, *OPIATE*[6;89] is an interactive storytelling engine that generates new stories using multiagent system similar to Tale-spin. Opiate incorporates a drama manager agent that uses a Case-Based Reasoning (CBR) approach with a case base of tales analyzed in terms of Proppian functions in order to guide plot generation. Figure 2.10 shows a screenshot from an interactive storytelling session generated by OPIATE. The *Automated Story Director*[95] uses a planning approach to model a narrative space. It automatically considers all possible breaks in the original story that can be caused by the actions of the player and proposes contingency narratives for each rupture, in order to allow the game narrative to proceed. *PAST*[96] builds on top of the *Automated Story Director* and uses it in combination with player modeling to generate player-specific game narratives while still maintaining authorial intent on the original narrative space.

Computational narrative may play other support roles within digital entertainment applications such as a part of a procedural content generation system. Procedural content generation (PCG)

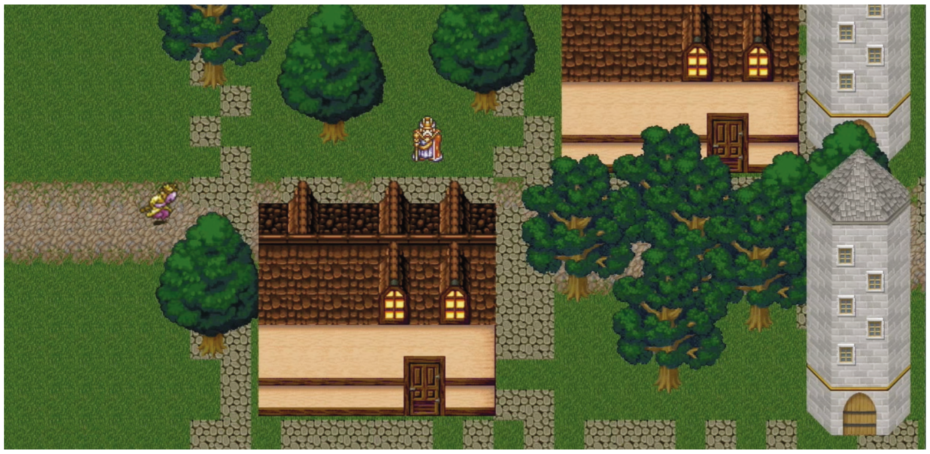


Figure 2.11: Virtual environment generated by Game Forge by Hartsook et al.[7].

refers to the creation of content automatically through algorithmic means[97]. PCG has been applied to computer games, specially for "map" (i.e., the virtual environment) generation for computer games in order to increase variability and replayability[98].

In addition to generating stories or engaging players, computational models of narrative can be embedded in other game elements. Game maps are an important storytelling element in computer games but most procedural map generation techniques typically neglect the role of the story in the construction of the map. Recent work has started to integrate narrative elements into map generation. *Game Forge*[7], is a system capable of generating a map given a linear story represented by a sequence of plot points. It uses a genetic algorithm approach to infer the spatial relationships between the locations annotated in the given story and generates a map genotype as a space tree. In a second step, the system maps the genotype to a phenotype where the space tree has been embedded on a grid. Then, the virtual world is graphically realized as a 2D map by instantiating predefined image tiles and handed to a game engine so that it can be navigated by the player's avatar. Figure 2.11 illustrates the output of this system. Dormans and Bakkes[99] introduce the idea of stories and spatial environments (referred to *missions* and *spaces* in their work) as separate structures; the first holding the logical causal relationship between the sequence of events and the second one the spatial description of the playable map where the story will unfold (or where the player will execute their *mission*). In our own research[90], we proposed a framework which, given

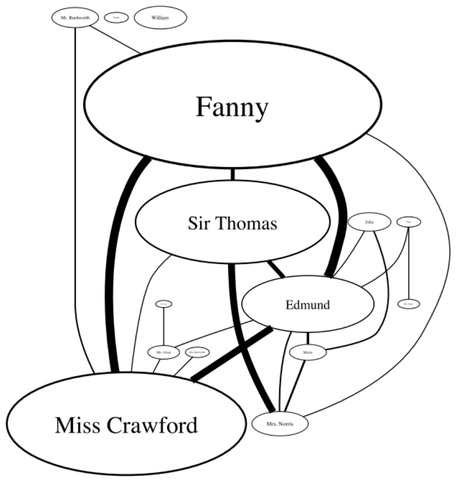


Figure 2.12: A social network automatically extracted from Jane Austen's Mansfield Park by Elson[8]. The size of the nodes represent the relevance of a character and the width of the edges the relative number of interactions (reproduced with permission).

the specification of a *story space*, represented as a collection of *plot points* and their dependencies, can generate maps that support one or more stories from that story space.

Narrative Analysis and Narrative Information Extraction

Besides uses in story generation and storytelling, computational narrative systems using automatic and semi-automatic narrative information extraction have been used to study literature and analyze narrative theories. Finlayson[84] uses a semi-automatic annotation procedure and a machine learning algorithm to extract plot patterns from a set of Russian fairy tales. In his work he reports results that aim to approximate manual narratology models via automatic analysis of the annotations. Elson[8] proposed the use of an automatic system to extract and study character interactions in classic literary works. Figure 2.12 shows a social network automatically extracted from Jane Austen's Mansfield Park by Elson. Bamman et al.[57] proposed an unsupervised approach to automatically extract characters, their attributes and the actions in which they participate in order to identify latent character archetypes or personas by clustering their stereotypical actions and attributes. In more recent work, Reagan et al.[85] studied emotional arcs in a big corpus of literature and elicited recurrent structures in books. They perform hedonometric analysis (to quantify sentiment or overall happiness over time[100]) and apply unsupervised machine learning techniques related to principal component analysis in order to elicit 6 emotional arc patterns predominant in literature[10].

2.5 A Taxonomy of Computational Models of Narrative

The wide range of different computational narrative applications, the disparity between the requirements of each system implementation, and the fact that narrative is such a broad and complex subject, makes it difficult to settle on a unified model to represent a narrative. Many approaches exist, such as scripts, plans or grammars but each of those approaches can only capture certain parts of the narrative. Large parts of what constitutes the plot and discourse of narratives, such as authorial intent, conflict, or character subjectivity are not properly captured by existing models. On the one hand, there are several competing proposed standards on text annotation for narrative, each focusing on different parts of the narrative and favoring different approaches. On the other hand, story generation systems require computational narrative models that enable the generation of multiple stories and provide affordances for interactive narrative. Moreover, given the complexity of the current approaches, system builders tend to favor developing customized ad-hoc models to suit their particular implementation needs.

In this section we present our survey on both narrative generation and analysis and the computational models of narrative used in such work.

2.5.1 Narrative Generation and Storytelling

Storytelling has been an integral part of digital entertainment, especially in interactive fiction and games in the genres of adventure and role playing[6;7]. Research in storytelling is tangentially related to fields such as text summarization, machine translation, or intelligent assistants but in our survey we focus on areas related to digital entertainment and computer games. Specifically we look at the following broad lines of work:

Story Generation: Related to computational creativity, research on story generation explores approaches for the generation of stories by computers[12]. Story generation ranges from *off-line* story generation to *interactive storytelling* work such as interactive fiction[101;102]. Most of these

[10]Visualizations available online: http://hedonometer.org/books/v1/

systems have *text output* in the form of a human-readable story whereas some fulfill a *support role* in mixed-initiative approaches to help authors write (e.g., by providing plot outlines or story space elements such as character profiles) [103;104]. We also find work where the generated stories are actually told by storyteller agents or characters within a game or other virtual environment [105]. In general, work can be classified into plot generation and discourse generation (i.e., the plot's realization) with some recent work integrating them [106]. Usually there are approaches focusing on the text realization of the stories [107–109] whereas other generate output to be consumed by a second system [110–112] or a specialized text realization component [113;114]. Another lens with which to look at research in story generation is from the computational creativity perspective. We refer the reader to an overview by Gervás [12] for more information in this regard.

Drama Management: Relevant to *interactive applications* and games (and to some extent to other educational and training tools), drama management focuses on techniques that shape and modify the plot of a game or interactive experience on-line based on both authorial intent and the users' actions. In the context of computational narrative, drama management has been used in interactive fiction applications and games to either maintain tension or an author's desired dramatic arc [22;24;82]; or maintain story coherence and/or enable/prevent player actions given some authorial goals [115]. There several other interesting systems and we refer the reader to some of the previous overviews on this topic [116;117].

Narrative-based Procedural Content Generation: Computational narrative systems have also been used in a support role to generate other game content such as side quests or maps, levels and virtual environments [7;90]. There are also examples that integrate computational narrative in several game components [6].

Other: Out of the scope of our work but relevant to this area of research, the reader may be interested in an overview on emergent narrative [118] and how narratology informs other applications of computational narrative [119].

When looking at the underlying computational models to encode narratives and narrative spaces used as input for applications included in the previous enumeration, we often observed hybrid approaches or applications that use different models to handle different parts of the generation and storytelling, still there are a few key approaches when it comes to computational models for narrative generation and storytelling. These are described below:

Planning-based: These models include logic and plan-like models representing a *story space* or the rules defining a simulation, narrative theory, agent behavior or author's goals [82;101;115;120]. Despite some existing popular representations (such as extensions to PDDL), because of the specialization of the different systems, these tend to be ad-hoc, manually authored, and not reusable.

Frame-based: These models include a wide variety of frame-based representations of a story such as description logics or semantic networks to represent events in a story. These are used often in analogy-based and modification approaches [83;102;121], and also tend to be manually authored and not reusable.

Plot-points: Plot-points usually represent the different events in a story into a branching or graph-like structure, where each node represents a *plot-point* (an important event in the story). These models are also usually domain-dependent and have been mostly used for drama management [22;24].

Rewriting Systems: Similar to the previous, these models encode narratives in a series of hierarchical abstractions. These models implement rules similar to those of a formal grammar, which have been used to study cognitive and narratology theories [73], and can be sampled for story generation [122;123].

Shallow Annotations: These take the form of a templates or involve shallow annotations over an existing text or story. These are typically used along other models in the final realization of the narrative. Used mostly for text generation [121;122], can be used by instantiating assets in other mediums (e.g., side-quests [124]).

Other Specialized Models: There are many other types of models focusing on modeling specific features or dimensions of a narrative such as the characters and their social relationships or the locations and the spatial configuration of the story environment[6;90;125]. These specialized models tend to be manually tuned for specific applications and may include rule-based systems or functions that maximize or minimize some desired target. These can also be used in conjunction with other models, usually through annotations and extensions that bind the models together[22;121].

2.5.2 Narrative Analysis and Narrative Information Extraction

Automated narrative analysis and narrative information extraction has been used to study folklore and sociocultural phenomena. This area of research, related to the digital humanities focuses on modeling narratives in order to study literature or validate narrative theories. We identified several distinctive areas of research and application:

Annotation and Markup Languages: From plot to discourse, there are several components of interest in a narrative. Moreover, natural language is ambiguous and may encode several overlapping features such as focalization, reader's mental model, embedded narratives, character affect states, beliefs, desires and goals. There have been efforts to standardize the process of adding computer-readable annotations to natural language to allow computational narrative systems to process narratives[26;126]. There have been also work on annotation tools[127;128]. Despite these aiming to be general purpose, there is no consensus on annotation formalisms, and researchers develop ad-hoc solutions to suit their particular needs that are time-consuming and expensive to generate.

Narrative Information Extraction: A second body of work focuses on automatically extracting narrative information from text using natural language processing techniques. Most of this work focuses on extracting particular components of a narrative from a previously defined ontology. These include automatically identifying characters, narrative and plot structure, and character relationships[15;18;129]. Related work also used crowdsourcing approaches[10] and/or commonsense databases and domain-specific knowledge to extract specific information[130–132]. Applications range from those tangentially related to computational narrative (such as document indexing and retrieval) or study, summarization and visualization of stories (using graphical and animated content)[9;58].

Note that the work reported in chapters 4, 5 and 6 falls within this area of research.

Story Understanding: Story understanding (a.k.a. machine reading or story comprehension) extends the idea of narrative information extraction but goes beyond extracting information from stories and strives to understand the entire story in order to reason and answer questions about it[133;134]. These efforts usually involve linking the extracted information to ontologies and common-sense databases[135]. To the best of our knowledge, despite some early interest in this area[136;137], recent work has focused on either story reasoning (ignoring natural language processing[138]), question answering from text (with limited inference and without a focus on broad understanding[139]) or knowledge representations for extracted story information[134;135].

Automated Literature Analysis: This work builds upon narrative information extraction but is intended to visualize, summarize or capture, often not a single story but a specific set of stories or texts. For example, comparing the works of different authors[128], validating narrative theories[84] or analyzing trends and recurring patterns (such as dramatic arcs in popular fiction[85], story structure[140] or character interactions[18;128]).

Let us now describe the computational models of narrative used in narrative information extraction and analysis work:

Shallow Annotations: Equivalent to the *shallow annotation* models described in Section 2.5.1, these are close to the text representation of the story, dependent on the discourse of the story but relatively straightforward to acquire automatically. These are the representation that narrative information extraction pipelines similar to *Voz* can extract and/or annotate and those can be used either as some intermediate/hidden model (for example, for indexing and retrieval[13]) or to provide a summary or visual representation of a story[85;141].

Symbolic Semantic Models: These are rich models that often include semantic information that has been extracted and/or inferred from the text, such as those described when discussing *annotation and markup languages*. The main difference from shallow models is that semantic models usually employ comprehensive ontologies that tie the different annotations together. Despite some limited work [9;15;58], these are challenging to automatically extract from text and the most feasible alternative is to use manual or semi-automatic annotation tools [127;128].

Catalogs and Other Specialized Models: These focus on modeling specific features or dimensions of a narrative (e.g., it is characters and social networks [8;13]) and are acquired by narrative information extraction pipelines that exploit off-the-shelf NLP. Despite being difficult to extract mostly because of open problems in NLP these are used in a breadth of applications related to narrative information extraction and automated literature analysis. These models can take the form of *catalogs* (a list of locations [7;142]) or graph-like structures (a social network based on character interactions [8;19;128]). The biggest difference between these and semantic models is that these do not intend to be comprehensive and unlike *shallow annotations*, these are disconnected from the discourse (i.e., text of the narrative).

2.5.3 Statistical Models for Computational Narrative

As in many other areas of research within computer science, there is an emergent trend within computational narrative that involves applying statistical methods in different computational narrative tasks. For the specific task of story generation, Markov-models have been used to learn the probability distribution of a specific corpus and have then been used to replicate the style of a specific genre or author [70]. Similarly, some examples in recent research explore the use of recurrent neural networks to both encode narrative events in a similar fashion as word embeddings encode text content and then use long short-term memory (LSTM) recurrent neural networks for learning the event structure of stories from a corpus and generating new stories by sampling the learned model [69]. More recent work combine word embeddings, Markov models and recurrent neural networks to explore additional training and sampling methods for learning from a corpus and automatically generating stories[70;143].

The key difference between the approaches in this class and the approaches described in the previous sections is that the models described previously are interpretable and explicit, that is, human authors and system designers specifically choose which features of the narrative to model and how to populate the models' contents (e.g., if we want to model the social network between characters, we have an explicit representation that we can look at that includes the characters in the story and a set of edges to define their social interactions). For the systems in this class of black-box approaches, however, the models that are used are not easily inspectable, it is hard to know what is being modeled, the author does not explicitly decide what will be modeled (besides defining the network structure) and it is up to the algorithms to find an internal representation of what needs to be modeled based on training data.

With the increase of computational power and the access to huge corpus of information, neural networks have seen a drastic increase in their applications with successfully to many areas of research. Related to computational narrative, natural language processing and natural language generation, recurrent neural networks trained from text are being used in for tasks such as summarization, machine translation and conversational agents.[67;67;68;144]. The use in computational narrative systems in the context of digital entertainment is limited but there are some examples in recent research that explore the use of recurrent neural networks (e.g., LSTMs and word embedding representations)[69;70].

2.6 Information Extraction

In the previous sections we first enumerated common tasks involved in natural language processing applications and then we discussed computational narrative, its applications and the different models used. In this section we delve in the intersection of these efforts and present the state-of-the-art in information extraction from narrative and discuss the most common architecture used in related implementations.

2.6.1 Automatically Extracting Narrative Information

Automatically extracting narrative information can be seen as a specialized case of information extraction. Except for a few recent exceptions, automatically extracting narrative information from unannotated text has not received a lot of attention. Character identification, related to named entity recognition is a crucial step in extracting and identifying narrative information. Goyal et al.'s AESOP system [16] explored how to extract characters and their affect states from textual narrative in order to produce plot units [145] for a subset of Aesop fables. The system uses both domain-specific assumptions (e.g., only two characters per fable) and external knowledge (word lists and hypernym relations in WordNet) in its character identification stage. Chambers and Jurafsky [15] proposed using unsupervised induction to learn what they called "narrative event chains" from raw newswire text. In order to learn Schankian script-like information about the narrative world, they use unsupervised learning to detect the event structures as well as the roles of their participants without pre-defined frames, roles, or tagged corpora. Regneri et al. [146] worked on the specific task of identifying matching participants in given scripts in natural language text using semantic (WordNet) and structural similarities. Schank and Abelson [147] use scripts as a formalism drawing from cognitive science as an attempt to natural language understanding. Calix et al. [13] proposed an approach for detecting characters in spoken stories. Based on features in the transcribed textual content using common sense information and speech patterns (e.g., pitch), their system detects characters through supervised learning techniques. Bamman et al. [56] extract characters, their attributes and the actions in which they participate and propose an unsupervised approach to identify latent character archetypes or personas by clustering their stereotypical actions and attributes. More recently, Li et al. [10;148] proposed a combination of NLP techniques and crowdsourcing to acquire narrative and procedural information about a specific situation.

Some of the aforementioned work focuses on capturing and identifying interactions and character relationships in the narratives using formalisms such as verb frames and logic clauses based on verbs tuples [8;15;17–19]. A specific case of narrative information extraction is the information about interactions encoded or implicit in dialogue. In this context, Elson and McKeown worked on the

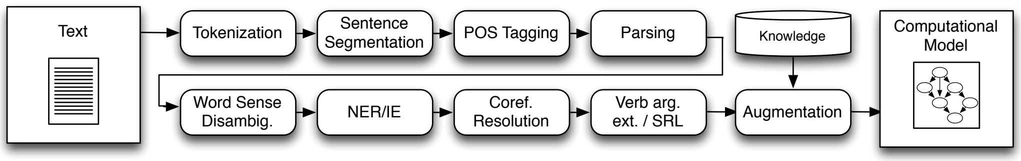


Figure 2.13: Typical NLP pipeline, specially prevalent in applications related to information extraction and text understanding.

problem of quoted speech attribution[149] from text[8]. In their work, they defined *syntactic categories* for quoted speech instances and then derived *rules* to assign the speaker for each of them. O'Keefe et al. [150] treated the problem of quoted speech attribution as a sequence labeling task. They were able to remove the use of gold-standard information and achieve similar performance on newswire text. More recently, Muzny et al.[151] also tackled the problem and proposed supervised method using a sieve approach. Their method greatly improved the performance of previous state-of-the-art. In our work, we extend their approach by automatically learning the *syntactic categories* and the *rules* to identify the speaker and intended listener for each single instance of quoted speech. Other work on extracting information from text include the use of regular expressions and specialized structures such as Tregex[44] which use a set of hand-authored extraction patterns. Riloff and Philipp[45] developed an IE system that was able to learn extraction patterns from examples.

NLP pipelines are typically embedded in larger systems used for information extraction and text understanding applications. In such applications, usually the linguistic tasks are complemented with augmentation steps that combine and augment the extracted information with common sense or domain specific knowledge in order to enable further processing and inference. For example, the *WordsEye* system[11] by Coyne et al.[58;130] is a text-to-scene system that creates 3D scenes from natural language descriptions. The system implements an NLP pipeline similar to the one presented in Figure 2.13 which is combined with a lexical database of common sense knowledge (WordNet, FrameNet and a custom scenario-based lexical knowledge resource) to extract a symbolic representation of the scene. The system converts dependency structures into semantic nodes and roles representing spatial relationships and visual attributes. The system relies on a large database of 3D models and poses

[11] More info: http://www.wordseye.com

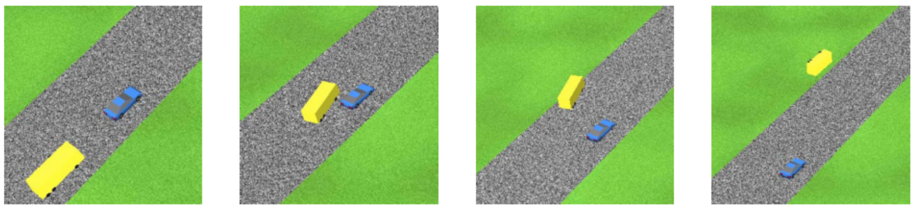


Figure 2.14: Frames from an animated sequence generated from a text report in natural language by CarSim[9].

for entities and actions. The extracted structure is mapped to the database and finally rendered into an image. Similarly, *CarSim* by Johansson et al.[9] is a system that automatically converts narratives in the traffic domain into animated 3D scenes. It is intended to be a tool for visualizing traffic situations from text reports in natural language. It also implements an NLP pipeline similar to the one presented in Figure 2.13 and extracts entities, events, relations and environment attributes separately. Then infers implicit information using a spatio-temporal planning and inference module that produces a full geometric description of the extracted symbolic representation from the text. Time descriptions and the output of the planning module are used to compute trajectories and generate an animation. Figure 2.14 shows four frames from an animated sequence generated from a text report in natural language by CarSim.

2.6.2 Evaluation of NLP Pipelines

The pipelined architectures described so far have been used extensively in NLP and information extraction applications. These systems usually integrate several natural language preprocessing modules (e.g., the Stanford CoreNLP[35]). Basic modules used in NLP pipelines have been studied extensively, often within the context of shared task competitions (e.g., Stanford's coreference resolution system at the CoNLL coreference resolution shared task[29]). These studies are typically conducted on a single module isolated from a pipeline since no standard methodology exists to evaluate how error propagates in pipelined architectures. Margaretha & DeVault[152] tackle the issue of automated evaluation of pipeline architectures in natural language dialogue systems using a Wizard-of-Oz approach and simulations of the pipeline process. In related work, Punyakanok el al.[50] combine different systems as modules of a single pipeline and study the quality of the information contributed by two different NLP systems for the task of semantic role labeling. When evaluating the performance of specific tasks or modules, most approaches agree on using a ground truth dataset and common counting metrics such as precision and recall[153]. In some tasks, such as coreference resolution, evaluating the accuracy is still contentious. Various alternative metrics have been proposed which weight different features of the coreference problem[48]. Punyakanok el al.[50] also discuss using alternative task-specific evaluation instruments on top of counting metrics.

Chapter 3: Proppian Stories

In this chapter we describe the corpus and datasets used thorough our experimental evaluation throughout this dissertation. In order to evaluate the different techniques and algorithms we will present in the next chapter, we first needed stories in our domain of application. Specifically, we collected a corpus of Russian and Slavic folk tales. This chapter presents an overview of this corpus and the datasets derived from these stories.

In the introduction and background chapters we mentioned the established and well known work of Vladimir Propp (1895-1970). Vladimir Propp was a Soviet folklorist and scholar who studied Russian and Slavic folk tales in the 20th century. His seminal work, *Morphology of the folktale* was originally published in Russian in 1928 and later translated to English[25]. In his work, Propp used folk tales collected by Alexander Afanasyev (1826-1871). Afanasyev published one of the largest folk tale collections in the world and is considered the Russian counterpart to the brothers Grimm. Propp analyzed the basic plot components of folk tales and presented a model of narratives based on *narrative functions* and *character roles*. His work influenced early western thematic narratology[71;72] and recent research in the fields of narratology and artificial intelligence[84;92].

In the context of narratology, a narrative function is a fundamental building block of storytelling: *"an act defined in terms of its significance for the course of the action in which it appears; an act considered in terms of the role it plays at the action level"*[154]. Propp described a series of narrative functions which he claimed represent canonical and invariant acts that constitute the underlying structure of Slavic and Russian fairy tales. Propp's work ultimately reduces a fairy tale to a sequence of variations of his functions (which he called subfunctions and are grouped in functions). We will be revisiting Propp's work on narrative function in Section 4.6 where we will try to predict them automatically. Propp also identified a set of broad roles for characters recurrent across stories. These are also explored further in Section 4.4.

When we incorporate narrative domain knowledge into our approaches, we use Propp's work. In order be able to use his narrative theory, we work with a corpus of Russian and Slavic folk tales that is related to the collection of stories Propp used to derive his work. In the next section we describe the corpus of English translations we used as a basis for our work. We also describe the annotations present in the dataset that accompanies the stories and a synthetic dataset derived from the original corpus.

3.1 A Corpus of Annotated Russian Stories

Throughout our work we compiled a corpus of 28 Russian and Slavic folk tales translated to English text. We selected stories from the Afanasyev collection studied by Propp. 6 of the digitalized versions of the English text were collected by Malec[155], 21 by Finlayson[84] and 1 is included in the translated version of Propp's work. Table 3.2 lists the stories in the corpus with their English titles and their numeric identifier in the Afanasyev collections. We provide additional information from Propp's own work on these tales in Appendix C and a list of sources for these stories in Appendix D.

Below is an example of one of the variations of the story *Morozko*, Afanasyev's story number 95 and 96 collected by Scott Malec[1].

[1]Available online: http://www.cedargallery.nl/engrussia_stories.htm

Once there lived an old widower and his daughter. In due time, the man remarried to an older woman who had a daughter herself from a previous marriage. The woman doted on her own daughter, praising her at every opportunity, but she despised her stepdaughter.

She found fault with everything the girl did and made her work long and hard all day long.

One day the old woman made up her mind to get rid of the stepdaughter once and for all. She ordered her husband: "Take her somewhere so that my eyes no longer have to see her, so that my ears no longer have to hear her. And do not take her to some relative's house. Take her into the biting cold of the forest and leave her there."

The old man grieved and wept but he knew that he could do nothing else; his wife always had her way. So he took the girl into the forest and left her there. He turned back quickly so that he wouldn't have to see his girl freeze.

Oh, the poor thing, sitting there in the snow, with her body shivering and her teeth chattering! Then Morozko, leaping from tree to tree, came upon her. "Are you warm, my lass?" he asked.

"Welcome, my dear Morozko. Yes, I am quite warm," she said, even though she was cold through and through.

At first, Morozko had wanted to freeze the life out of her with his icy grip. But he admired the young girl's stoicism and showed mercy. He gave her a warm fur coat and downy quilts before he left.

In a short while, Morozko returned to check on the girl. "Are you warm, my lass?" he asked.

"Welcome again, my dear Morozko. Yes, I am very warm," she said.

And indeed she was warmer. So this time Morozko brought a large box for her to sit on.

A little later, Morozko returned once more to ask how she was doing.

She was doing quite well now, and this time Morozko gave her silver and gold jewelry to wear, with enough extra jewels to fill the box on which she was sitting!

Meanwhile, back at her father's hut, the old woman told her husband to go back into the forest to bring back the body of his daughter. He did as he was ordered. He arrived at the spot where had left her, and was overjoyed when he saw his daughter alive, wrapped in a sable coat and adorned with silver and gold. When he arrived home with his daughter and the box of jewels, his wife looked on in amazement. "Harness the horse, you old goat, and take my own daughter to that same spot in the forest and leave her there," she said.

The old man did as he was told.

Like the other girl at first, the old woman's daughter began to shake and shiver. In a short while, Morozko came by and asked her how she was doing.

"Are you blind?" she replied. "cannot you see that my hands and feet are quite numb? Curse you, you miserable old man!"

Dawn had hardly broken the next day when, back at the old man's hut, the old woman woke her husband and told him to bring back her daughter, adding: "Be careful with the box of jewels." The old man obeyed and went to fetch the girl.

A short while later, the gate to the yard creaked. The old woman went outside and saw her husband standing next to the sleigh. She rushed forward and pulled aside the sleigh's cover.

To her horror, she saw the body of her daughter, frozen by an angry Morozko. She began to scream and berate her husband, but it was all in vein.

Later, the old man's daughter married a neighbor, had children, and lived happily. Her father would visit his grandchildren every now and then, and remind them always to respect Old Man Winter.

Although the stories are relatively short, fully understanding them often requires significant inference based on commonsense knowledge and contextual information. For example, in one of our stories, a magical being called Morozko (also known as Jack Frost in some western translations) gave a young girl "a warm fur coat and downy quilts." In order to understand Morozko is *helping* her, the context of the forest in the winter is important. Furthermore, some actions need to be inferred. In the same story, the text only describes how the step-sister of the hero (the anti-hero) answered

Morozko's question rudely. In the next scene, her mother "saw the body of her daughter, frozen by an angry Morozko," leaving out Morozko's direct actions (killing the step-sister) to inference.

This corpus poses challenges to several NLP tasks. For example, specific to character identification and related to named entity recognition, the text includes a range of different types of characters: humans, anthropomorphic animals (e.g., a talking mouse) and anthropomorphic objects (e.g., a magical oven, a talking river). There are also fantastical creatures (e.g., goblins) and characters specific to the Slavic folklore (e.g., Morozko and Baba Yaga). In the work described in the next chapter, among other tasks, we seek to identify these characters from the text without hand-coding specific knowledge such as knowing that "Morozko" is a traditional Slavic character. Coreference is also difficult as it is very common that a character's referring expression change from "daughter" to "sister" or "girl" throughout the story. Moreover, in one of the stories, there are two young female characters. Besides the obvious pronominal coreference problems that may arise, they are both referred as "daughter" and "maiden" in different parts of the story. Besides the aforementioned coreference resolution challenges, the particular translations in the stories pose additional challenges to most NLP tasks. For example, we noticed long sentences with several subordinate phrases that cause failures in tasks such as syntactic parsing, a task on top of which other tasks depend. For example, consider the following excerpt:

> When she opened her coffer and hung out her things on a rope that stretched from the house to the gate, the old woman, who had opened her mouth to greet her in her customary abusive way, pursed up her lips, seated the welcome guest under the icon, and said to her civilly: "What is your pleasure, madam?"

That single sentence is found at the conclusion of one of the stories. Besides the multiple instances of the word "her" used as either possessive or a personal pronoun for either of the two female characters in the story, that is a single sentence. At the same time, notice how there is a lof of subtext encoded in the adjectives and adverbs used.

The stories also exhibit a wide range of rhetoric figures. A particular rhetoric figure is quoted

Table 3.1: Annotation present in Finlayson's 15 story dataset that we use in our work and the number of annotations for each.

Annotation Layer	μ	σ	Total
Sentences	176.60	60.31	2649
Tokens and Parts of Speech	3108.47	1004.65	46627
Referring Expressions	857.93	277.40	12869
Coreference	219.27	78.96	3289
Syntactic Parse	176.60	60.31	2649
Semantic Roles	489.53	170.52	7343
Narrative Functions	28.33	7.16	425

speech (i.e., phrases and sentences surrounded by quotation marks) that contain direct speech as spoken by a character of the story. These quotes are often arranged in succession representing a conversation or *dialogue*. Processing this embedded direct speech surrounded by text in third person (i.e. reported speech) introduced errors in several natural language processing tasks. In order to mitigate these, we annotated the spans of text where direct speech or dialogue was present (that is, the quotes themselves and the surrounding *speech attribution cues*) in order to filter it out. Most of the work reported next chapter between Sections 4.2 and 4.6 ignores this text. Then in Section 4.7 we will focus specifically on processing it separately in order to extract the information about character interactions encoded or implicit in dialogue.

Besides the dialogue, 15 of the stories in the corpus had been previously deeply annotated with syntactic, semantic and narrative information by Finlayson[84], who kindly shared his dataset with us. Table 3.1 lists the features annotated (i.e. annotation layers) in Finlayson's dataset that we use in our work and the number of annotations for each. Finlayson also shared with us some translations from his corpus, 6 of which we annotated ourselves with a subset of the information layers in his original dataset.

Throughout our work we have used datasets including different subsets of the corpus with different annotations. In each subsection in the next chapter we describe the relevant subset and annotations used. Also note that both Malec and Finlayson provided different levels of annotations with their original collection of stories. Part of our work consisted on interpreting, converting and complementing their annotations. We also developed some of our own annotation schemes on the text.

At the end of our work, we have a dataset of 20 fully annotated stories. We added our own annotations for specific tasks, specifically:

- We annotated 15 entity types over the existing referring expressions and coreference groups. The class labels we used were inspired by Chatman's taxonomy of existents[72] and include: *happening* (e.g., rain), *male character*, *female character*, *anthropomorphic animal character*, *anthropomorphic object character*, *group or abstract set of characters* (e.g., people, pirates, all the devils), *magical being character* (e.g., Morozko, the devil), *part of a character* (e.g., her soul, her fingers), *animal* (non-character), *object* or *prop*, *locations* that the characters visit (e.g., the hill), *scenery* that is mentioned (e.g., the mountains in the distance, the fields surrounding the hill), *temporal references* (e.g., the day after, Winter), *part of a non-character* (e.g., the bed's blankets, the horse's back), and an additional "N/A" class label used for labeling parsing errors. These annotations and their use are described in Section 4.3.4.

- We annotated 7 character role labels over the existing characters (a subset of the entities). Propp's defined 7 broad roles for characters: *Hero*, *Villain*, *Dispatcher*, *Donor*, *(Magical) Helper*, *Sought-for-person*, and *False hero*. When annotating our class labels we merged the roles of *Donor* and *(Magical) Helper* since they mostly correspond to the same character, specially in our dataset. Additionally, we introduce a role *Other* that includes the *Dispatcher* and includes other recurring roles such as family members. As in the previous case, we use an additional "N/A" class label for classification parsing errors. These annotations and their use are described in Section 4.4.

- We annotated verb argument triplets (verb, subject, object) describing interactions between characters. These were initially annotated over the stories without associating them to specific spans of text and we used symbols to refer to the characters in order to manually resolve coreference. These annotations and their use are described in Section 4.4.

- We annotated quoted speech, direct speech and dialogue over the existing sentences. These annotations include 4 components: First we annotate the span of text for instances of quoted speech or direct speech. Most of the dialogue is realized in quotes but we found instances of direct speech outside of quotes when the narrator was addressing the audience directly, these instances are annotated with a different class label. For each of the quotes we also annotate which (if any) *speech attribution cue* relates to the quote (such as "said" in "he said to her") and the referring expressions (if any) that refer to the speaker and intended listener for the quote (such as "he" and "her" in "he said to her"). These annotations and their use are described in Section 4.7.

This 20 story dataset contains 1931 reported speech sentences plus 718 sentences in quoted speech (29.5% of the text). Within the reported speech sentences considered on most of our experiments, there are 3029 mentions to 236 unique characters (46% common nouns, 43% proper nouns, 11% pronouns) and 5131 annotated verbs.

Working with this dataset and given the considerations outlined earlier, we quickly realized that it posed difficult challenges that made it difficult to process with off-the-shelf natural language processing tools. In our work we tried to address some of these challenges but we also worked on authoring a simpler dataset derived from these stories which is described in the next section.

3.2 A Synthetic Dataset of Annotated Russian-inspired Short Stories

We authored a dataset of 100 Russian-like short stories. We wanted to be able to experiment with larger number of stories and we wanted the stories to be in the same application domain in order to continue our current line of work. Moreover, given the issues encountered when processing the original text of the stories, we also wanted the new stories to use simpler text and avoid some rhetoric figures like dialogue. In order to satisfy all of those requirements we decided to hand-craft a synthetic dataset of short stories inspired by the stories in the original corpus and Propp's work.

In order to author these stories we followed three approaches; first, we looked at the 20 stories in the previous dataset and we rewrote each of them in a simpler form, without dialogue, and ignoring certain characters and events when necessary. We carefully analyzed the Proppian functions in the

Table 3.2: Stories in our dataset. The second column lists the numerical identifier of the stories in the new Afanasyev collections.

Translated Title	Afanasyev #
Witch and Solntseva Sister	93
Morozko/Jack Frost	95, 96
The old woman Govorukha	97
Daughter and Daughter in law	98
Kroshechka-Havroshechka	100
Burenushka	101
Baba Yaga	102, 103
Vasilisa the Beautiful	104
The Witch	108
The Magic Swan Geese	113
Prince Danila Govorila	114
The Merchant's Daughter and the Maidservant	127
Frolka Stay-at-Home	131
Ivan Popyalov	135
Dawn, Evening and Midnight	140
The Seven Simeons	145
Nikita the Tanner	148
The Serpent and the Gypsy	149
Shabarsha the Laborer	151
Ivanko the Bear's Son	152
The Runaway Soldier and the Devil	154
The Crystal Mountain	162
Bukhtan Bukhtanovich	163
The language of the Birds	247
The princess who wouldn't smile	297
Girl and Bear	557

annotations and ensured that all the characters required to realize the functions were present. Unlike the original stories where some functions are implicit or out of order according to Propp's rules (refer to Section 4.6 for more details), in the reimagined story we made sure that every Proppian function was explicitly realized and when possible, in the order specified in Propp's rules.

This is an example of one of the stories in our synthetic dataset that reimagines the story of *Morozko* reproduced in the previous section.

An old widower lived with his daughter in a farm. The widower eventually married another woman who had a daughter of her own. The stepmother and stepdaughter despised the girl. They asked her father to get rid of her. The father took the girl to the forest and left her there to die. A goblin saw the girl and asked her how she was doing. The girl was kind to the goblin and they talked all night long. The goblin thanked the girl for their conversation. As a token of appreciation, he gave her new clothes and jewelry. The next morning, the father went back to the forest to retrieve the girl's corpse. He was very happy to see his daughter still alive and they went back to the farm. When the stepmother saw the clothes and jewelry she got very excited. She asked her husband to take her own daughter to the forest. The husband took his stepdaughter to the forest and left her in the same spot. The goblin saw the girl and approached her. The girl was repulsed by the goblin and was mean to him. The goblin killed the girl as punishment. The next morning, the father went back to the forest to retrieve the girl. He went back home with her corpse. When the mother saw her dead daughter she died of sadness herself. The girl and her father lived happily and she eventually married a good lad from a nearby farm.

For the remaining stories, we selected 30 sequences of Proppian functions described in Propp's work and for each of the sequences we authored stories from scratch that included the target sequence of functions. The functions are defined in two levels, the top level including 31 functions that group 215 more specific subfunctions (which include some negations). To author these stories we tried to avoid biases and followed a systematic procedure: First, we selected one of the stories listed in Propp's work[25]; these stories are described using a table that includes the identified functions. With a selected story, we would then try to read the original story when possible, to familiarize ourselves with the specific elements of such a story. Finally, for each story we went back to the sequence of functions and wrote it from scratch using for reference Propp's function definitions. We also omitted some functions, avoided duplicates or alternated subfunctions within the same group as the parent function when necessary to suit creative decisions of the story. We repeated this process to author at least two stories from each initial sequence until we had a total of 100 stories in the dataset (including the previous 20); for the second story written for each initial sequence we tried alternating as many subfunctions from each group as possible and we interleaved additional functions. In Appendix C

Table 3.3: Annotation present in Finlayson's 15 story dataset that we use in our work and the number of annotations for each.

Metric	μ	σ	Total
Sentences	14.54	4.38	1454
Tokens	131.01	60.58	13101
Narrative Functions	9.97	3.12	997

we reproduce Propp's work on function identification that we used as reference for this task.

When crafting these stories we tried to refer to the original sources to replicate Russian and Slavic themes from these traditional stories. In order to be faithful to these original stories, we searched for freely available sources of English translations of traditional Russian and Slavic folk tales. We compiled, read and manually analyzed these in order to familiarize ourselves with them and gain a better understanding of the themes and recurring elements. While doing this work we also used for reference an English translation of one of the collections by Alexander Afanasyev. We provide a reference of these sources in Appendix D.

The final dataset[2] is significantly larger in terms of stories compared to the corpus described in the previous section but the stories are significantly shorter. Table 3.3 provides some statistics for this dataset.

3.3 Discussion

In this chapter we described the corpus and datasets we collected and annotated. This work was necessary since we needed a corpus of Russian and Slavic folk tales to match the application domain of the work of Vladimir Propp. We use these stories and Propp's work thorough our experimental evaluation described in the upcoming chapters.

We want to thank Mark Finlayson[84], who kindly shared his corpus and annotations with us.

We first presented a corpus of English translations of Russian and Slavic folk tales which we annotated and used to derive several datasets for our work. This corpus poses challenges to several NLP tasks. We decided to manually author a second synthetic dataset, inspired by the same stories, but using shorter and simpler language, and a larger number of stories.

[2]Available: https://sites.google.com/site/josepvalls/home/voz

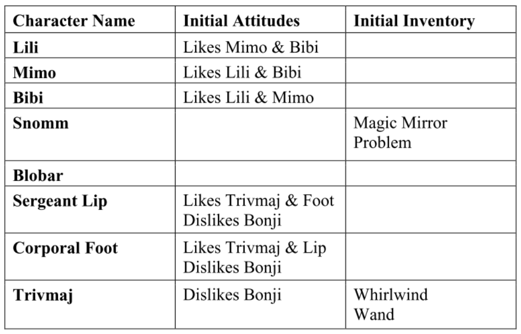
Chapter 4: Automated Narrative Information Extraction

The *authorial bottleneck* problem (see Section 1.1) is an important open problem in the field of computational narrative. Although there has been some work towards easing this problem using techniques such as computer assisted annotation (see Section 2.3.1) and automatic information extraction (see Section 2.6.1), this is a relatively under explored area of work, with a significant number of open research questions.

Let us start with an example. Let us consider the OPIATE system by Fairclough[6;89] described in Section **??** and shown in Figure 2.10. In order to operate, OPIATE requires several models of the narrative which need to be authored manually. First, it requires a list of characters present in the story world, each annotated with initial attitudes towards one another and some initial inventory, as shown in Figure 4.1 left. Second, it requires a description of the spatial relations between different environments in the story world, each annotated with the initial locations of characters and objects, as shown in Figure 4.1 right. Third, it requires an annotated case base of sequences of Proppian story functions, as shown in Figure 4.2. Moreover, it also requires additional knowledge about common sense and social practice in order to simulate the story world.

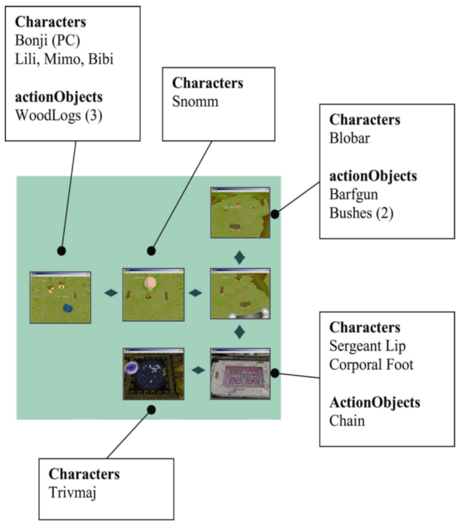
Creating these knowledge representations is a tedious and expensive process. Our goal is to automatically generate that structured knowledge. For example, instead of providing the system with the hand-crafted knowledge structure shown in Figure 4.1, our goal is to allow the user to provide information about the story world in natural language, such as the fragment shown in Figure 4.3. Being able to describe a story world in such a way would allow non-technical people to author content for computational narrative systems. Moreover, one could use existing stories to create virtual story worlds. In order to do so, one would need a narrative information extraction system capable of building computational models such as the ones used by OPIATE, or other computational narrative systems, from text.

Extracting narrative information from natural language text poses several challenges to state-



Character Name	Initial Attitudes	Initial Inventory
Lili	Likes Mimo & Bibi	
Mimo	Likes Lili & Bibi	
Bibi	Likes Lili & Mimo	
Snomm		Magic Mirror Problem
Blobar		
Sergeant Lip	Likes Trivmaj & Foot Dislikes Bonji	
Corporal Foot	Likes Trivmaj & Lip Dislikes Bonji	
Trivmaj	Dislikes Bonji	Whirlwind Wand

(a) List of annotated characters



(b) Spatial description

Figure 4.1: Example story world information used in OPIATE by Fairclough[6] (reproduced with permission).

villainy capture (family); donor request; reaction; provision; pursuit; rescue hide; departure; donor; reaction; provision take; return; pursuit; rescue (object); unrecognized; recognition; wedding;

Figure 4.2: Example of one Proppian narrative sequence used by OPIATE[6] to guide the interactive storytelling session (reproduced with permission).

Once upon a time, Bonji ran into Lili, Mimo and Bibi, 3 friends who lived in a hut. In a field nearby lived Snomm who had a Magic Mirror. Past the field and further into the woods lived Blobar. In the other side of the woods there was a little town where Sergeant Lip and Corporal Foot had lived. Both had a resentment towards Bonji. [...]

Figure 4.3: Fragment of a possible natural language description of part of the story world represented in Figure 4.1.

of-the-art NLP techniques. These challenges are even more prominent when dealing with fictional stories that exhibit non-standard patterns and features. In order to improve the accuracy in several NLP tasks for our application domain, in this chapter we explore one key idea: incorporating narrative information into core NLP and information extraction tasks.

In this chapter we present our work towards automatically extracting narrative information from natural language text. In the next section we describe our experimental infrastructure, that is,

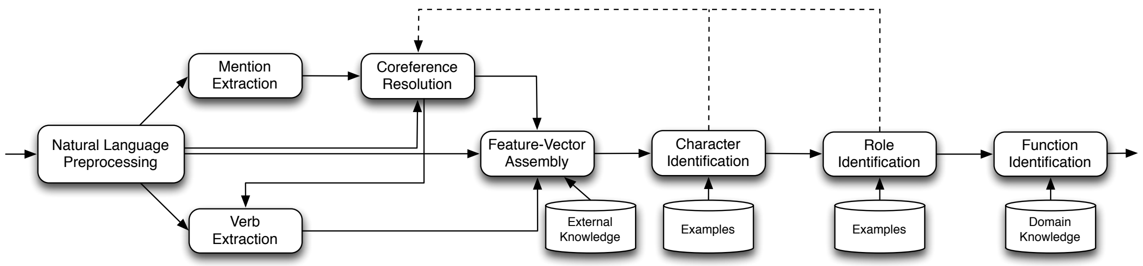


Figure 4.4: Overview of the architecture of *Voz*.

the narrative information extraction system we built in order to test our ideas and in sections 4.2 through 4.7 we describe our individual contributions in detail.

4.1 Experimental Evaluation Infrastructure: Voz

In order experiment with the different ideas and test the different hypothesis we present in this dissertation we developed a narrative information extraction system which we called *Voz*[1]. We will first describe the overall architecture of this system which we will build upon later in this chapter.

Voz is a narrative information extraction system that, given a story in natural language, can automatically extract narrative information such as the characters and their roles in the story, the actions those characters perform to one another, and additional information about the structure of the story. *Voz* implements a modular NLP architecture that combines off-the-shelf natural language processing tools, common sense knowledge and domain knowledge. Figure 4.4 illustrates the architecture of *Voz*. The rounded boxes represent the modules of the system, solid arrows the information flow and dashed arrows represent information feedback. The overall workflow is as follows:

1. **Natural Language Preprocessing:** This first step performs sentence segmentation, lemmatization, part-of-speech tagging, coreference resolution, and grammatical and dependency parsing. The *Natural Language Preprocessing* module wraps several off-the-shelf NLP toolkits in a common interface[2]. This module is self-contained and can be used independently from *Voz*. It includes the *Stanford Parser*, the *Stanford CoreNLP System* (which includes a version of the *Stanford Parser* itself), the syntactic parser from the *Apache OpenNLP* toolkit, the *MALT*

[1]Available: https://sites.google.com/site/josepvalls/home/voz
[2]Available: https://github.com/josepvalls/parserservices/tree/stable

Parser and the syntactic parser from the *ClearNLP* toolkit. In our work we have experimented with other systems (i.e., *NLPNET* and the Charniak Parser) and other components (i.e. the coreference resolution system in the *Apache OpenNLP* toolkit) but we focus mostly on the output from the *Stanford CoreNLP System* for the aforementioned tasks.

2. **Mention Extraction:** In this step, we use syntactic information to identify entities via their mentions in the text. The algorithm used by the *Mention Extraction* module relies on the part-of-speech tags and syntactic parse from the previous module and is described in Section 4.2.

3. **Coreference Resolution:** This step compiles and enhances the coreference information from the *Natural Language Preprocessing* module. We first describe how we use the information from the *Coreference Resolution* module in Section 4.3.5. We identified coreference resolution as one of the bottlenecks in the process of extracting narrative information from stories, and thus part of our work focuses on approaches to improve this step for the particular case of narrative text. In our work we introduce a feedback loop on our pipeline architecture (shown as dashed arrows in Figure 4.4) This is described in Section 4.5 and revisited in Chapter 6.

4. **Verb Extraction:** This step is in charge of identifying verbs and their arguments form text. For the extraction step, we use syntactic and dependency parse information. This module is described in Section 4.3.1 and later we describe how we use the extracted information in Sections 4.3.2 and 4.4.

5. **Feature-Vector Assembly:** This step encodes previously extracted information from mentions and verbs into numerical vectors that combine it with commonsense information and make it suitable for processing downstream. The module responsible for this step is described in Section 4.3.2.

6. **Character Identification:** Using the feature vectors described in the previous step, this step uses a case-based reasoning inspired approach to identify animacy, that is, which of the extracted mentions refer to characters. This module is described in Section 4.3.

7. **Role Identification:** Similar to the previous step, now focusing on the characters, this step identifies which narrative role the previously identified characters fulfill in the narrative. This module is described in Section 4.4.

8. **Function Identification:** This last module identifies higher-level narrative information from stories, specifically, Proppian narrative functions from segments of text using local features and global narrative information. We describe this module is described in Section 4.6.

9. **Dialogue Processing:** A parallel module currently not incorporated in *Voz* is tasked with processing quoted text (i.e., direct speech in quotes) in order to extract specific character interactions encoded in the dialogue. This module uses *Voz*'s infrastructure and could be incorporated as part of the pipeline. It is described in Section 4.7.

The remainder of this chapter details the work on the modules outlined above and describes our main contributions in several automated narrative information extraction tasks using this infrastructure.

4.2 Extracting Mentions from Unannotated Text

This module focuses on identifying the entities by extracting their mentions in the text. An *entity* (or existent in Chatman's taxonomy[72]) is something, physical or abstract, that exists as a particular unit. Entities in a narrative include physical existents such as characters, locations or props, and abstract existents such as events or time expressions. Each entity may be realized in a text using different *referring expressions*. Each instance of these referring expressions is commonly known as a *mention*. For example, in Figure 4.5 "the boy" is a mention or referring expression to *Vasili*. *Vasili* is the canonical referring expression of an entity that has been previously introduced and happens to be a character and the hero of that story. Throughout this chapter we will use the term *mention* to refer to each of the specific mentions of an entity. When we later talk about characters, we will be referring as the entities

In order to extract mentions, we developed an algorithm that uses syntactic parse trees (as generated by the Stanford CoreNLP). Our algorithm traverses each of the sentence parse trees

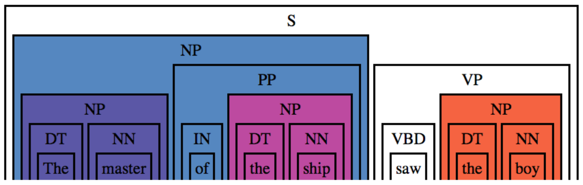


Figure 4.5: Syntactic parse of a sentence annotated by *Voz* after the mention extraction process. Mentions or referring expressions are colored. Note how there is a compound mention (shaded in blue) that is recursed and mentions to 3 entities are found (shaded in purple, pink and orange).

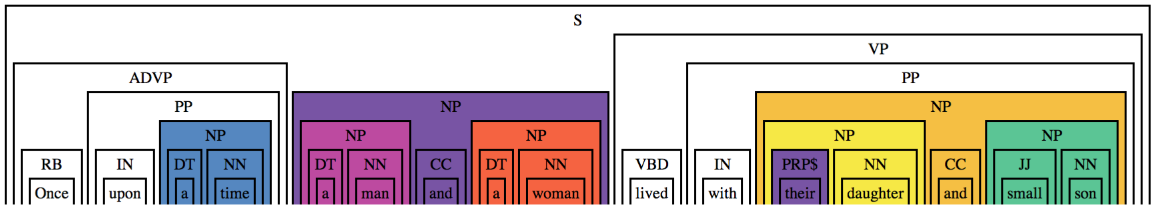


Figure 4.6: Syntactic parse of a sentence annotated by *Voz* after the mention extraction process. Lists are identified (shaded in purple and light orange) and references to individual mentions extracted.

looking for a "noun phrase" (NP) node. For each NP node, Our algorithm does the following: If the subtree rooted at the current NP node contains nested clauses (such as verb phrases or prepositional phrases) or the leaves of the subtree contain an enumeration (a conjunction or a list separator token) then our algorithm traverses its associated subtree recursively. Otherwise, if any leaf in the subtree is a noun, personal pronoun or possessive pronoun, the node is marked as a *mention*, and its subtree is not explored any further. Using this process with the input sentence *"The captain of the ship saw the boy"* (illustrated in Figure 4.6), our algorithm detects three individual mentions (shaded in purple, pink and orange). After finding the compound "The captain of the ship," the algorithm recursively detects two nested mentions, "the captain" and "the ship".

A special case is considered for enumerations: the deepest node containing an enumeration token (indicated by a comma or the *and* or *or* conjunctions) is marked as a list. Lists can be used later to match plural pronouns with the mentions in the list. Figure 4.6 shows the list "a man and a woman" shaded in purple. In a later stage, the coreferenced pronoun "their" in the following phrase (also shaded in purple for illustrative purposes) can be used to obtain the individual "man" and "woman" mentions.

Experimental Results

In order to evaluate this module, we used a dataset of 21 stories from our corpus of Russian and Slavic folk tales. Please refer to Chapter 3 for more information on our dataset. To reduce preprocessing issues at the discourse level, we manually removed quoted and direct speech (i.e., dialogues and passages where the narrator addressed the reader directly). The edited input dataset contains 914 sentences. The stories range from 14 to 69 sentences ($\mu = 43.52$ sentences, $\sigma = 14.47$). There is a total of 18126 tokens (words and punctuation; $\mu = 19.83$ words per sentence, $\sigma = 15.40$). To evaluate the task of mention extraction, we annotated 4280 noun phrases (NP) representing referring expressions.

Our algorithm identifies 4791 individual mentions, including all of the annotated noun phrases and 511 of which are not actual referring expressions but parsing errors, mostly adjectival phrases identified as nominal phrases by the off-the-shelf NLP tools used. For example, in the sentence "And indeed she was warmer.", *warmer* was wrongly identified as a noun phrase. Our method has a recall of 100% (all of the annotated mentions were found) but a precision of 89.3% ($f = 0.944$).

4.3 A Machine Learning Approach to Identifying Characters from Extracted Mentions

Once we have identified the mentions to entities in the text, an important task for our approach to modeling a narrative is to identify the characters that participate in a story. This is task is counterintuitively difficult in the domain of fictional stories because of a number of reasons, mainly: 1) the use of specific proper names for characters specific to the Slavic folklore (e.g., Morozko or Baba Yaga), 2) the presence of other fantastical creatures not commonly found in other domains (e.g., dragons or goblins), and 3) the presence of animals and even anthropomorphic objects that play a character and fulfill a narrative role (e.g., a talking mouse or a magic oven). In the second and third case, the problem is magnified by the fact that these may use the pronoun "it" which may confuse them with other animals or props. Related work in this area has acknowledged the difficulty of this task for domains like literary fiction[13] and movie plot summaries[56].

In order to address this task we frame it as a binary classification problem over the previously extracted mentions. We solve this classification problem using a machine learning approach inspired by case-based reasoning. Specifically, we try to classify the extracted mentions as characters (animated sentient beings in the story) and non-characters (remaining existents and happenings defined in Chatman's taxonomy[72] such as locations, animals, props or happenings).

Our character identification method uses case-based reasoning (CBR)[156], a family of algorithms that reuse past solutions to solve new problems. The previously solved problems, called cases, are stored in a case-base. In our approach, each case is an extracted mention, represented as a feature-vector, annotated as either character or non-character by a human. Given a problem or a query, case-based reasoning uses a retrieval step similar to *k-nn*[157] where instances from the case base are selected based on a similarity metric. Then a solution to the problem or query at hand is adapted from the retrieved cases. In terms of a classification problem, the case base contains labeled examples and when retrieved, the labels in the examples are used to predict the label of the query. We address the following problem: given a mention, extracted from an unannotated story in natural language determine wether it is a character or not.

In this work we present two contributions: 1) a set of features used to compute a feature vector describing each mention, and 2) a novel similarity measure used to determining the most similar cases for retrieval that we called the *weighted continuous Jaccard distance*[14].

4.3.1 Verb Extraction

This module runs in parallel to the previous modules and is tasked with extracting verbs and their arguments from the text. These verbs will be used to identify interactions and relationships between the previously extracted mentions.

To identify the verbs in a sentence, *Voz* uses the typed dependencies from the *Stanford CoreNLP* output. Specifically *Voz* looks at dependencies of type "nominal subject" or "passive nominal subject" where the *head word* is POS-tagged as a verb (this excludes linking verbs). In the case of "nominal subject", the dependent of the typed dependency is considered the subject of the verb. All of the remaining dependencies of the verb are explored and if a mention is found in any of the dependencies (i.e., direct object, indirect object and prepositional objects) it is extracted and tagged as one of the verb arguments and a new triplet extracted. For "passive nominal subject", subject and direct object are reversed. The spans of text where arguments have been identified will be used later with the extracted mentions from the *Mention Extraction* module.

4.3.2 Computing Features from Extracted Mentions

After extracting the mentions from the text (described in the previous section) and collecting information about coreference information (initially from the output of the *Stanford Coreference resolution system* and further described in Section 4.5), i

In this step, *Voz* converts each extracted mention (from the *Mention Extraction* module) to a feature vector. These feature vectors are used downstream to represent these mentions and include additional information from the output of the *Stanford CoreNLP* (in the *Natural Language Preprocessing* module) and the *Verb Extraction* module. When using the output of the *Stanford CoreNLP*, we use the parse tree of the sentence where the mention is found, the subtree representing the mention, the leaves of the subtree (e.g., word-level tokens with POS tags) and the dependency lists that contain a reference to any node in the mention's subtree, including verb arguments from the *Verb Extraction* module). We also query readily available knowledge bases such as WordNet[158], ConceptNet[159] and word lists (also known as dictionaries or gazetteers in the literature).

Building upon the features proposed by Calix et al.[13] we use an extended set of features that improve the overall performance of the system[14]. We use 194 features which can be grouped in 7 categories. These are described below:

- **Sentence Parse Tree Features.** *Voz* looks at the parse tree of the sentence where a mention is found and extracts features related to the nodes containing the mention, such as its depth and the presence of adverbial, adjectival or prepositional phrases. These features may indicate how relevant a mention is in a sentence or the semantic role it plays. There are 11 features in this category. For example, the *fromNodeS* feature captures whether the mention is found directly under a sentence node (S). *fromNodePP* describes whether the mention is found in a nested prepositional phrase node (PP).

- **Mention Parse Subtree Features.** *Voz* traverses the subtree representing the mention and extracts features related to the types (i.e., syntactic labels) and number of nodes found indicating nested noun phrases. These features may indicate the presence of nested noun phrases, typically found in proper nouns and proper names composed of common nouns. There are 43 features in this category. For example, *hasNodeNP* captures that the referring expression has a nested noun phrase node (NP) or *fromNodePP* indicates that the subtree is found within a prepositional phrase (PP).

- **POS Tag Features.** *Voz* enumerates the leaves of the mention's subtree and extracts features related to the POS tags assigned to the word-level tokens. The features account for the presence of common nouns, proper nouns, pronouns, adjectives or existentials (e.g., "there") that may indicate specific types of entities. There are 26 features in this category. For example, *hasTokenJJ* captures that there is an adjective in the mention and *hasTokenPRP$* captures that there is a possessive pronoun in the mention.

- **Dependency List Features.** *Voz* enumerates the lists of dependencies and extracts several types of dependencies. A dependency, in this context, is a relation between two elements in a sentence (the *dependent* and the *governor*). For example, it can be the relation between a verb with its subject, direct object or indirect object, or, the relation between a determiner and the noun it is referring to. For verb dependencies, we look at both active and passive voices for each identified verb in a sentence. Overall, *Voz* records if 1) an mention appears as a subject or object of any verb, 2) if it appears as subject or object of specific verbs (e.g., "have" or "tell") and 3) the similarity of the verb where a mention appears to predefined clouds of concepts (e.g., one such cloud is "reward," "pay," "give") or pertains to predefined verb clusters. The features computed from these account for typological semantic roles for the verb cloud arguments. Typically for verbs like "talk" or "tell", the subject and direct object arguments are likely to be characters. For the verbs like "have" or "give", the object is less likely to be a character than the subject. There are 90 features in this category. For example, the *isVerbSubject* feature captures whether the mention appears as the subject of one (any) verb. *isVerbSubjectCloud-*

Move captures that the mention appears as the subject of a verb, this feature accounts for the similarity to a cloud of verbs like “move”, “go” or “arrive.” *depHeadPrepOf* captures that the mention is used as the head of a dependency with the preposition “of.” *depHeadPoss* captures that the mention is the governor of a possessive dependency indicating that it is possessed by some other mention. *Voz* also looks at other dependencies, specifically the relationships of nouns with prepositions, possessives and determiners. *Voz* computes features to identify when a mention has a determiner or wether a mention possesses something or is possessed by something else. These features indicate relationships between entities and a mention possessing something is more likely to be a character than a mention that is possessed by some other mention.

- **WordNet-based Features.** We defined several clouds of concepts, where each “concept” is a set of WordNet *synsets*, which are in turn a set of synonyms. For example, one of these clouds is formed by synsets related to the concept of “youth” while another is related to the concept of “villainy”. The similarity between the cloud and each noun (word with a noun POS tag) in the mention is computed and the average value is the final value of the feature (if there are no nouns in the mention, then the feature takes value 0). To assess the similarity of a noun with a cloud of synsets, we use the measure proposed by Wu & Palmer[160;161] to compute similarity between the noun and each of the synsets in the clouds. This measure returns a value between 0 and 1. Among the similarity values for each sysnet, the maximum value is used. There are 8 features in this category. An example feature is *hasWordnetAgeYoung*. This feature is based on the similarity of words in the mention to synsets related to “youth.” The cloud contains adjectives like “young” and nouns like “baby”, “child”, “boy” and “girl”. These features may indicate similarities with certain character archetypes and when building our clouds we used Propp’s work as a reference. We defined 8 clouds with each containing between 3 and 17 synsets.

- **ConceptNet-based Features.** Following the work of Calix et al.[13], we query ConceptNet and look for properties in the relationships of the returned concepts. We look at edges of certain types (e.g., “IsA” and “RelatedTo”) connecting to specific nodes (e.g., magic, wizard) to compute the 9 features in this category. For example, the feature *hasConceptnetHuman-Capabilities* captures whether any nouns in the mention have ConceptNet relationships of the type “CapableOf” to concepts such as “laugh”, “feel” or “love”. There are 9 features and each checks between 3 and 9 edges.

- **Word List Features.** Our word lists are equivalent to dictionaries or gazetters. The main difference between clouds and word lists is that *Voz* uses a similarity measure for clouds (with a continuous value in $[0; 1]$) and exact matching for word lists (i.e., if *Voz* finds a word from the mention inside of the given word list, the feature will have value 1, otherwise it will have value 0). There are 6 features from lists of nouns and 11 features from lists of other types words, with each feature defining its own word list and also the set of POS tags to filter the words that can be matched. We have a list of names including 2943 male and 5001 female names (an external list without modification based on birth records) and another with 306 common nouns including titles and professions. We have lists of words for identifying number (e.g., they, them), gender (e.g., he, she) and gender neutrality (e.g., another, other, this, that...). For example, the *hasWordInCommonNamesFemale* feature checks a word list of common nouns conceptually related to female gender or with female inflection like “witch”, “nun”, “lady” or “actress”. The *hasWordInSingularThirdPersonGeneric* feature captures words identifying singular, third person objects such as “one”, “another”, “this” or “which.” There are 7 additional lists and each has between 3 and 27 words.

At this point, the mentions extracted previously by the *Mention Extraction* module are encoded into a numeric vector representation that can be used by the rest of the modules of the pipeline (see Figure 4.4). Additionally, this representation allowed us to easily experiment with different machine learning techniques from off-the-shelf packages such as Weka and scikit-learn.

4.3.3 Case-Based Reasoning for Classification

The *Character Identification* module performs a classification task using Case-Based Reasoning (CBR)[156]. In our approach, each case in the CBR case base is another mention, represented as a feature-vector, already annotated as either *character* or *non-character*. In a nutshell, for each new target mention to be classified, its feature-vector representation is compared with the annotated mentions in the case-base. The most similar entity in the case-base is *retrieved* and used to predict whether the target given mention is a character or not. As in many related *k-nn* approaches, we need a similarity or distance function to retrieve (i.e. compare) the mentions in the case-base.

Many similarity or distance functions exist in the machine learning and CBR literature. However, after initial experimentation, none of them performed adequately in our domain because of the particulars of our features, namely, our features represent wether a mention satisfies a property or not but take real values between 0 and 1 to indicate to which degree a property is satisfied. For that reason, we defined a novel distance measure for case retrieval that can properly handle our feature vectors. We use a variant of the Jaccard index[162] that we call the *Weighted Continuous Jaccard distance*, described in the next section.

Weighted Continuous Jaccard Distance

The Jaccard index[162] is a very well-known similarity function between two sets (A, B) defined as the "size of their intersection, divided by the size of their union":

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Most of the features computed by *Voz* represent whether a mention satisfies a given property or not (e.g., whether the entity if the subject of a verb or not). If the entity satisfies the property, then the feature has value 1, and if it does not, then the feature has value 0. Thus, we could see an entity e as a set of properties (that contains each property for which its corresponding attribute has value 1), and thus, we should be able to apply the Jaccard index for assessing similarity between entities. However, some of the features are actually continuous (e.g., how similar is an entity to a given concept) and thus can take intermediate values like 0.5. Therefore, the standard formulation of the Jaccard index cannot be used directly. We hence generalized the notion of the Jaccard similarity by borrowing the notion of a t-norm form the field of Fuzzy Logic[163]. Intuitively, a t-norm (triangular norm) is a function that generalizes the concepts of intersection in set theory and conjunction in logic. By interpreting the "intersection" and "union" in the Jaccard index as a t-norm and a t-conorm (the equivalent of the "union"), we replaced them by appropriate t-norms and t-conorms for numerical values: the *min* and *max* operators, resulting in the following measure, that we call *Continuous Jaccard*:

$$D_J(e_1, e_2) = 1 - \frac{\sum_{i=1}^{m} min(f_i(e_1), f_i(e_2))}{\sum_{i=1}^{m} max(f_i(e_1), f_i(e_2))}$$

where we write $f_i(e)$ to represent the value of the ith feature of the entity e. When the two entities are identical, D_J is 0. If they are completely disjoint, D_J is 1.

In our work[14], we reported an experimental evaluation where we compared this metric with two other standard distance measures:

- *Euclidean distance*: a standard euclidean distance between the feature vectors representing each mention:

$$D_E(e_1, e_2) = \sqrt{\sum_{i=1\ldots m} (f_i(e_1) - f_i(e_2))^2}$$

- *Cosine distance*[164]: a standard distance measure used in text retrieval:

$$D_C(e_1, e_2) = 1 - \frac{\sum_{i=1}^{m} f_i(e_1) f_i(e_2)}{\sqrt{\sum_{i=1}^{m} f_i(e_1)^2}\sqrt{\sum_{i=1}^{m} f_i(e_2)^2}}$$

Additionally, given that different features might contribute more or less to the classification of each entity, we experimented with two variants of each of the previous distance metrics: *standard* (as presented above) and *weighted*.

In the weighted versions of the similarity measures, we computed a numerical weight for each feature by using *Quinlan's Gain*[165]. In our previous work[166] in the context of distance measures, we observed that it achieved good results in computing feature weights. Weights are computed based on the cases in the case-base. For each feature f, the case-base is divided in two sets: C_1, with those cases where $f \leq 0.5$ and C_2, with those where $f > 0.5$. Then, Quinlan's Gain is computed as:

$$Q(f) = H(C) - \frac{H(C_1) \times |C_1| + H(C_2) \times |C_2|}{|C|}$$

where $H(C)$ represents the entropy of the set of cases C (with respect to the distribution of class labels).

The weighted versions of the three distance measures above result from multiplying the contribution of each feature to the similarity by the corresponding weight. Specifically, in each distance, each term in each of the summations (the sum in the Euclidean distance, the three sums in the Cosine distance, and the two sums in the Jaccard distance) is multiplied by the weights of the corresponding feature. The weighted continuous Jaccard measure is thus defined as:

$$D_{wJ}(e_1, e_2) = 1 - \frac{\sum_{i=1}^{m} Q(f_i) min(f_i(e_1), f_i(e_2))}{\sum_{i=1}^{m} Q(f_i) max(f_i(e_1), f_i(e_2))}$$

Experimental Results

For our experimental evaluation in this work we used an annotated dataset derived from 8 stories containing 1122 mentions, 615 labelled as character and 507 as non-character. We also performed feature selection (by groups of features) and the results reported use a set of 193 features described in the previous section. We evaluated the performance of the above described six distance measures: Euclidean, Cosine and Continuous Jaccard (S_E, S_C, S_J), and their corresponding weighted versions (S_{wE}, S_{wC}, and S_{wJ}). For the evaluation, we follow a leave-one-story-out protocol: the 1122 instances in the dataset come from 8 differentrent stories, we split the 1122 instances into 8 different sets according to the story they come from, then, we predict the labels for each story using the annotations on the remaining 7 stories in the case-base. Notice that this procedure is needed, since a standard leave-one-out procedure over the individual instances would yield deceivingly high results, since some mentions are similar inside a given story.

Table 4.1: Performance of the different distance measures over the complete dataset (1122 instances) measured as classification accuracy (*Acc.*), Precision (*Prec.*) and Recall (*Rec.*).

Distance	Acc.	Prec.	Rec.
S_E	86.45%	0.91	0.83
S_{wE}	88.41%	0.92	0.86
S_C	87.08%	0.91	0.85
S_{wC}	89.22%	0.93	0.87
S_J	86.45%	0.91	0.84
S_{wJ}	91.27%	0.93	0.91

Table 4.1 shows the performance of the 6 distance measures on our complete dataset. Performance is reported as accuracy, precision and recall. The first thing we can observe is that all the weighted variants of the distance measures outperform their non-weighted versions. For example, the weighted Euclidean distance (S_{wE}) can classify 88.41% of the instances correctly into characters/non-characters, whereas the non-weighted Euclidean distance (S_E) only classifies correctly 86.45%. Moreover, we can see that the best performance is achieved with our weighted-Continuous Jaccard distance measure (S_{wJ}), which correctly classifies 91.27% of the instances. This is also reflected in the precision and recall values.

We compared our results against Stanford's *Named Entity Recognition* (NER), which achieved a precision of 0.98, but an extremely low recall, of 0.09, since it only recognizes, as *PERSON*, entities that have capitalized names. We also compared our results against standard machine learning classifiers, using the WEKA software. AdaBoost obtained precision and recall of 0.79/0.92. This unexpected low performance is due to the sensitivity of boosting classifiers to noise. Surprisingly, from the classifiers available in WEKA, J48 achieved the best performance with P/R scores of 0.92/0.91, slightly below that of S_{wJ}.

Moreover, we noticed that in our dataset, many characters are often referred to with personal pronouns (i.e., "he" or "she"), clearly indicating that they are characters (non-character entities, such as props or scenario elements are never referred to using these personal pronouns). In order to perform a rigorous evaluation, we repeated our experiments removing all the instances that consisted of personal pronouns, excluding the pronoun "it" since it is used for both characters (anthropomorphic objects and animals) and non-characters (objects or settings). In this scenario, we use a dataset with

Table 4.2: Performance of the different distance measures over the filtered dataset removing the instances that contain personal pronouns measured as classification accuracy (*Acc.*), Precision (*Prec.*) and Recall (*Rec.*)..

Distance	Acc.	Prec.	Rec.
S_E	83.18%	0.86	0.72
S_{wE}	88.26%	0.90	0.82
S_C	83.74%	0.85	0.76
S_{wC}	88.71%	0.89	0.84
S_J	83.41%	0.86	0.74
S_{wJ}	90.18%	0.91	0.86

886 instances. Results are reported in Table 4.2. As expected, performance decreased (specially for the non-weighted distance measures). However, our S_{wJ} distance measure maintains classification accuracy over 90%.

We performed additional experiments related to feature selection to determine the relevance of the different types or groups of features described in the previous section. These experiments use the complete dataset (1122 instances) where we removed different sets of features. Table 4.3 reports the performance of the S_{wJ} measure in the following scenarios: each row in Table 4.3 represents a different set of features (from the types of features described before); each row reports how many features are in each set, the performance of S_{wJ} when *only* using features of in the given set, and also when using all the features *except* the ones of in the given set. For example, the first row of Table 4.3 (*WordNet*) reports the classification accuracy that S_{wJ} obtains when only using the 8 features coming from the parse tree containing the mention (81.11%) and also when using all the features except those 11 (87.88%). Our results indicate that the features that contribute the most to the performance of our approach are those coming from WordNet: using only the 8 features coming from WordNet, S_{wJ} achieves a classification accuracy of 81.55%. Other types of features that are very important are the *Lists of Words*, those coming from *Entity Parse Subtree*, and from the *Sentence Parse Tree*. Also, notice that there are some features that when removed performance actually increases (e.g., *Determiners*). Surprisingly, a feature that we hypothesized would be very helpful (whether an entity is the subject of a verb or not) does not actually help in this classification task but are kept for other tasks described further in this dissertation. Finally, notice that the most

Table 4.3: Performance of the S_{wJ} distance measure with different feature subsets: *Acc. only* reports the accuracy only using features of a given type, and *Acc. all except* reports the accuracy using all the features except the ones of the given type. N reports the number of features of each type.

Feature Subset	N	*Acc. only*	*Acc. all except*
WordNet	8	81.11%	87.88%
Entity Parse Subtree	9	70.59%	90.81%
POS Tags	26	68.27%	91.09%
Sentence Parse Tree	11	66.84%	90.55%
Lists of Nouns	6	66.04%	88.15%
Verb Clouds	20	56.33%	91.00%
Lists of Words	11	54.72%	91.44%
Verb List	62	54.37%	91.18%
Prepositions	14	54.01%	91.09%
ConceptNet (Calix)	9	52.85%	90.46%
Verb Subject/Object	3	51.43%	91.89%
Other Dependencies	4	50.89%	90.91%
Verb Argument	6	50.45%	91.18%
Determiners	4	48.13%	91.53%

important set of features (*WordNet*) are continuous, justifying the need for our new Continuous Jaccard measure.

If we compare these results with work reported in the literature, Calix et al.[13] report 84.3% accuracy using an Euclidean Distance, and 86.1% using Support Vector Machines. Performance is not strictly comparable, since they used a different dataset (with almost 5000 instances), but the numbers seem to indicate that our Continuous Jaccard measure, combined with the set of features we determined above, outperforms these results.

4.3.4 Extending Mention Classification

The previous module performs a binary classification task over two classes: *character* and *non-character*. For the purpose of extracting additional narrative information and enrich the extracted information for mentions, we then proceeded to extend the previous approach to classify each mention into a set of classes inspired by Chatman's taxonomy[72]: *happening* (e.g., rain), *male character*, *female character*, *anthropomorphic animal character*, *anthropomorphic object character*, *group or abstract set of characters* (e.g., people, pirates, all the devils), *magical being character* (e.g., Morozko, the devil), *part of a character* (e.g., her soul, her fingers), *animal* (non-character), *object* or *prop*, *locations* that the characters visit (e.g., the hill), *scenery* that is mentioned (e.g., the mountains in the distance, the fields surrounding the hill), *temporal references* (e.g., the day after, Winter), *part of a non-character* (e.g., the bed's blankets, the horse's back), and an additional "N/A" class label used mostly for parsing errors.

Experimental Results

A 21 story dataset was used and the 4791 mentions were annotated with the additional fine-grained labels. We developed visualization tool for assisting the annotation and evaluation of the extended classification task. This tool embeds the high-dimensional vectors representing mentions into a 2D visualization using a force-directed layout. The weighted continuos Jaccard distance is used to control the attraction between vectors therefore clustering similar mentions which help visualize the dataset and debug outliers.

For the experimental evaluation we followed the same leave-one-story-out protocol. *Voz* achieves a micro-averaged (i.e. weighted average by the number of instances in each of the 15 class labels) precision of 0.567, recall of 0.507 and an overall classification accuracy of 0.462. The confusion matrix for this classification is shown in Table 4.4. In the results table we can see how there are a few class labels such as male (MA), female (FE) and locations (SS) that have a much higher accuracy. When considering only whether the entity is correctly classified as a character or non-character (that is, AA, AO, MA, FE, GR, MB), the micro-averaged precision is 0.929 and recall is 0.934. Our approach is successful at identifying which mentions are characters and which are not.

4.3.5 Improving Mention Classification using Coreference Information

In the work reported so far, the instance class labels predicted from the case-base of examples use a nearest neighbor approach with a variant of the Jaccard distance to retrieve cases and classify new instances. This approach achieves relatively high classification accuracy but we experimented with potential refinements. In this work we improve the classification results by using the previously identified coreference resolution information (see Figure 4.4).

Coreference information can be seen as a graph where the nodes are referring expressions or

Table 4.4: Confusion matrix for predictions in the 15 class labels in our classification process with counts for all the 21 stories using the leave-one-story-out protocol. The two letter labels stand for (from top to bottom): "N/A" for parsing errors, AA: anthropomorphic animal character, AN: animal (non-character), AO: anthropomorphic object character, FE: female character, GR: group of characters, HA: happening, MA: male character, MB: magical being character, OB: object or prop, PA: part of characters, PO: part of non-characters, SC: scenery that is mentioned, SS: locations that the characters visit, and ST: temporal references. Bold face indicates correct predictions (diagonal) and the color gradient illustrates the normalized value over the total count of instances for each class.

	N/A	AA	AN	AO	FE	GR	HA	MA	MB	OB	PA	PO	SC	SS	ST	Recall	Prec.
N/A	**0**	24	1	7	8	17	30	37	4	166	11	0	9	150	47	0	0
AA	0	**39**	1	2	31	10	1	29	13	22	2	0	0	3	5	0.247	0.151
AN	0	4	**2**	6	0	2	2	8	2	49	0	0	1	3	7	0.023	0.133
AO	0	1	0	**0**	0	22	1	7	2	20	1	0	0	7	0	0	0
FE	0	14	0	0	**510**	3	8	9	0	24	0	0	0	17	4	0.866	0.765
GR	0	10	3	34	62	**56**	9	55	0	120	2	0	0	5	0	0.157	0.308
HA	0	3	2	1	2	2	**4**	7	1	60	4	0	6	21	10	0.033	0.033
MA	0	72	1	1	30	37	17	**799**	17	71	3	0	0	69	11	0.708	0.76
MB	0	34	1	9	1	5	0	57	**52**	58	0	0	21	1	2	0.216	0.433
OB	0	36	3	11	13	13	30	14	26	**375**	48	0	16	119	50	0.497	0.318
PA	0	5	1	8	7	5	5	4	0	56	**33**	0	1	16	0	0.234	0.308
PO	0	0	0	0	0	0	0	0	1	2	1	**0**	1	1	0	0	0
SC	0	8	0	0	0	2	2	1	1	19	0	0	**2**	21	6	0.032	0.032
SS	0	4	0	4	1	6	9	13	1	94	1	0	4	**283**	14	0.652	0.387
ST	0	5	0	0	2	2	2	12	0	42	1	0	2	16	**57**	0.404	0.268

mentions and there exist edged between nodes when two mentions refer to the same entity. It is therefore intuitive to assume that given that there is an assignment of one class label per entity, all the mentions of the same entity should have the same class label. Therefore, we use a majority voting approach among the predictions for each coreference group, that is, given a mention e, *Voz* identifies its coreference group $coref(e)$, i.e., all the other mentions that are linked to e in the coreference graph G. Then, a final classification is generated by assigning to each mention e the majority class in the coreference group $coref(e)$. For example, if three mentions in $coref(e)$ were labeled as *character* and only one as *non-character*, then all the mentions in $coref(e)$ will be labeled as *character*. In the (unusual) case of tie, the class of the earliest occurring mention in the text is chosen.

Once all the mentions have been classified, the output of the coreference resolution is used to refine the results. Given a mention $e \in E$, we identify its *coreference group* $coref(e)$, i.e., all the other mentions that are linked to e in the coreference graph G. Then, the class assigned to e is replaced by the majority among all the classes of all the mentions in the coreference group $coref(e)$. For example, given a mention e labeled as *non-character*, if two mentions in $coref(e)$ were labeled as

Table 4.5: Effect of coreference information on the majority voting processes. Rows report results without coreference information, using the automatically computed coreference graph and using the coreference from the ground truth. The columns report the accuracy, precision and recall on the binary classification task. Note that the dataset used for this experimental evaluation is larger than the one used in the results reported in Tables 4.1 and 4.2.

Approach	Acc.	Prec.	Rec.
Without Voting	0.86	0.844	0.876
Auto Coref.	0.868	0.859	0.878
GT. Coref.	0.868	0.896	0.839

character and only one as *non-character*, then the label assigned to e will be replaced by *character*.

Experimental Results

For the experimental evaluation we used both a ground truth coreference graph and an automatically extracted coreference graph (from the Stanford CoreNLP). With either approach we observed an improvement on accuracy and f-measure from 0.860 to 0.868. Table 4.5 reports detailed accuracy, precision and recall comparison between these approaches. Note that the dataset used for this experimental evaluation is larger than the one used earlier in Section 4.3.3.

4.4 A Machine Learning Approach to Identifying Narrative Roles from Characters

In this section we focus on a method for identifying a particular high-level feature of narrative theory, namely, narrative roles for characters. Narrative roles for characters are a recurrent feature present in several narrative theories (e.g. Propp's[25] or the Monomyth[77]). These roles identify prototypical character actions for specific narrative roles. Typically, the protagonist of the story takes the role of the *hero*. Another character roles is that of the *villain* that triggers or motivates the heroe's actions.

Let us consider the except shown in Figure 4.7. In Propp's theory, the dragon (*character*) fulfills the specific function of villain (*role*). This structure-level narrative information of roles is important to understand the story as well as its relation to others in its domain. However, as the word "villain" or "hero" rarely appears explicitly in the text, extracting the role information requires combining NLP and narrative theory. A key Proppian insight that we use is that each role has a "*sphere of action*." It defines the core actions of whatever characters fulfilling that role. For example, no matter

> One day, somewhere near Kiev, a dragon appeared, who demanded heavy tribute from the people. He demanded every time to eat a fair maiden: and at last the turn came to the Tsarevna, the princess. But the dragon would not eat her, she was too beautiful. He dragged her into his den and made her his wife. [...]
> When she wrote a letter to her father and mother she used to tie it to the neck of her little dog. [...]
> The Tsarevna got every day on more intimate terms with her dragon in order to discover who was stronger. At last he owned that Nikita, the tanner at Kiev, was the stronger. [...]
> The Tsarevna at once wrote to her father [...] So the Tsar looked for Nikita, and went to him himself to beg him to release the land from the cruelty of the dragon and redeem the princess. [...]

Figure 4.7: An excerpt of a story from our dataset.

whether the villain is a dragon or a wizard, its sphere of action centers on villainy, struggle, and pursuit.

In order to automatically identify a character's narrative role we rely precisely on the recurring actions and interactions with other characters as described in the narrative theory; Propp's in our case. In the work described in this section we introduce the idea of representing the "sphere of action" of a character role (their prototypical actions in Propp's narrative theory) as a matrix encoding interactions indicated by verbs linking characters with different roles[167]. Initially we used an annotated dataset to compute a matrix from a story and compare it against a reference matrix using Wordnet to find similarities. Then we folded this approach into the automated mention and feature extraction process described earlier and we integrated it with the machine learning approach used to classify characters described in the previous section[129].

In our initial work on modeling narrative character interactions using "spheres of action", we hypothesize that, *given a particular narrative convention, information about how characters behave towards one another can help identify their roles.* This is based on our observation of recurring patterns in the relationship between different roles. To test our hypothesis, in this work we used function definitions from Propp's narrative theory and an annotated dataset of 8 Russian and Slavic folktales to build a knowledge base of common actions and interactions between different character roles.

Table 4.6: One of the role-action matrices used in our experiments. The largest one is 7×7, with 506 verbs.

	Hero	Villain	Other	N/A
Hero	talk	fight	rescue, marry	depart
Villain	fight		kidnap, lock	plot
Other	marry			cry
N/A	summon, reward			

Propp categorized 7 broad roles for characters: Hero, Villain, Dispatcher, Donor, (Magical) Helper, Sought-for-person, and False hero. From the example in Figure 4.7, a dragon (villain) kidnaps the princess (the Sought-for-person), then *Nikita the Tanner* (hero) is summoned by the king (the dispatcher) and sent to fight the dragon and rescue the princess. Such interactions or "spheres of action" can be encoded into what we call the *reference role-action matrix R* (Table 4.6). This matrix is provided as part of the narrative domain knowledge. Each cell $R_{i,j}$ contains the set of actions that a character (subject) with a role r_i executes over a receiver (object or patient) of the role r_j . The additional row and column are for actions performed without a known subject or object. Also, note the diagonal of the matrix need not be empty since there may be more than one character playing a certain role and some actions performed upon oneself may also be encoded in the diagonal (e.g., the hero talks to himself or herself). Each cell contains a set of verbs that have been manually authored. Table 4.6 reproduces one of our matrices. In our experiments we used three ways to construct this matrix which we describe below under the experimental results section.

Narrative role identification for each character is performed by 1) assigning one of the m roles to each of the n characters (including an additional *no-role* role for those characters that play no clear role in the story), 2) comparing the assignment with the reference role-action matrix and selecting the assignment that better matches with the reference matrix. These processes are described below.

- **Role Assignment.** Given n characters and m roles, there are m^n possible assignments of roles to characters. The number of characters in a given story may range from a handful up to several dozen. Thus, systematically evaluating all the possible assignments has a prohibitive cost. We use a genetic algorithm with an initial random population of 80 individuals (role assignments encoded as lists of integers), and we perform 1000 iterations of a simple ranked selection with a mutation rate of 2%, and a crossover rate of 90%. We use a swap mutator and a single point crossover as recombination operator. In order to speed-up the assignment process, we incorporated some constraints. Specifically, we force exactly one character to be assigned the role of the hero. Each assignment during the search process is matched with the reference role-action matrix to obtain a *fitness score*, as detailed below.

- **Matching.** In order to compare character/role assignments with the reference matrix, *Voz* proceeds by constructing a new character role-action matrix A computed from the character action matrix C, by replacing the character labels by the roles from a character role assignment. Then, A is compared to the reference matrix R by comparing each cell in A with the cell in R with the matching role labels. Each cell in the matrices contains a list of verbs $c_1 = \{v_1, ..., v_r\}$ and $c_2 = \{w_1, ..., w_s\}$. We use the measure proposed by Wu & Palmer[160;161] to calculate the similarity between each pair of verbs $v_i \in c_1$ and $w_j \in c_2$, which we note by $S(v_i, w_j)$. This measure calculates the similarity between two verbs by determining the *least common subsumer* (LCS) verb in the verb taxonomy in WordNet, and then using its depth in the taxonomy to determine the similarity between the two input verbs:

$$S(v_i, w_j) = \frac{2 \times depth(LCS(v_i, w_i))}{depth(v_i) + depth(w_j)}$$

Then, assuming $r \leq s$, we aggregate the values as follows:

$$S(c_1, c_2) = \begin{cases} \frac{\sum_{v_i \in c_1} \max_{w_j \in c_2} S(v_i, w_j)}{s} & \text{if } c_1, c_2 \neq \emptyset \\ 0 & \text{if } c_1 = \emptyset \vee c_2 = \emptyset \end{cases}$$

Intuitively, this measure matches each verb in c_1 with the most similar verb in c_2 according to Wu & Palmer's measure, and then normalizes by the size of the largest set of verbs, s. Finally, the values for each cell comparison are added together to obtain a numeric similarity measure for the current character role assignment A and the assignment that maximizes similarity is selected.

This process allows us to use a general search-based method to explore the space of possible assignments and evaluate them.

Experimental Results

In order to test our hypothesis we conducted experiments with an annotated dataset that represented character interactions in terms of verb triplets and we manually crafted 3 versions of the reference role-action matrix.

- R_1: This reference matrix was developed by reading the English translation of the role descriptions written by Propp [25] and extracting all the actions (i.e. verbs) described in the 31 functions and subfunctions. Moreover, we merged the roles of Donor and Helper since, in our dataset, they mostly correspond to the same character. Additionally, the roles of Dispatcher and Prize (and others, such as "victim" or "family member") are unclear, and thus we grouped them into an "other" role. This resulted in a 7×7 matrix with 506 verbs.
- R_2: This reference matrix was manually created and captures our own common sense of the actions that the different roles perform upon each other. This is a 7×7 matrix with 32 verbs.
- R_3: Finally, we created a simpler matrix, with only three roles (Hero, Villain and Other) manually designed to capture only the relation of these three roles. This is the matrix shown in Table 4.6.

Table 4.7 shows the classification accuracy (average percentage of actors with the correct role assigned) obtained using each of the role/action reference matrices. The first column (*Top*), shows the results obtained by selecting the best role assignment found by the genetic algorithm. As we can see, the performance is very low (accuracy around 30%). As an alternative approach, we experimented with a method that estimates, for each character, the probability of each role. For estimating this probability, we used the entire population in the last iteration, and weighted each one by the fitness score. The role with the highest probability is selected for each character. The second

Table 4.7: Averaged performance results of our role assignment using the topmost assignment versus weighting all the assignments in the last iteration.

	Top	*Weighted*
R_1	15.10%	48.75%
R_2	31.03%	64.58%
R_3	46.35%	78.99%
Avg.	30.83%	64.11%

column (*weighted*) shows the results obtained with this method. Results improved significantly, reaching classification accuracies of up to 78.99% for reference matrix R_3.

Our results indicate that the fitness function used in the genetic algorithm (matching against a reference matrix that captures the narrative domain knowledge) is effective in determining character roles. However, we observed that, in some situations, the actual ground truth had lower fitness than some of the solutions being found. This indicates that while useful, either the reference matrices, or the matching procedure being used introduce some noise in the results. Another source of noise are very common verbs, that can be performed by all the different character roles.

Additionally, the role definitions are ambiguous in some aspects and sometimes open to interpretation. For example, Finlayson, from whom we borrowed some of the annotations used for our work, reports an inter-annotator agreement between a team of 12 annotators of $F_1 = 0.7$ for the role annotations (or *Proppian Dramatis Personae* in his work[84]).

4.5 Improving Coreference Resolution Using a Narrative Information Feedback Loop

The goal of coreference resolution is to group extracted mentions into *coreference groups*, where each coreference group is a set of mentions that refers to the same character or entity in the story. Our dataset and application domain poses two challenges to the task of coreference resolution. First, pronominal coreference resolution (resolving that a pronoun refers to a specific named entity mentioned earlier) requires commonsense and inference in some scenarios. Second, full coreference resolution needs to associate mentions using different referring expressions for a single entity. For example, in one of the stories, there are two young female characters. Besides the obvious pronominal coreference problems that may arise when we encounter a single female pronoun "she", they are both referred as “daughter” and “maiden” in different parts of the story.

There are two key ideas we explore in order to improve coreference information. First, we want to use narrative information to inform the coreference resolution process, specifically, information about the narrative roles of different mentions. Since this information is not available yet when the coreference resolution step is computed, the second idea we explore is introducing a feedback loop that feeds the character and narrative role information extracted later in the pipeline back into this module. This is illustrated by the dashed arrows in Figure 4.4.

Sourcing Coreference Information

We identified some issues with the output of the output of the *Stanford Coreference Resolution* system we were using, namely, there were some split coreference groups, that is, there were edges missing in the coreference graph. In order to address this, we explored alternative sources of coreference information that could complement the initial coreference assignment and then we devised a method to aggregate different sources of coreference information. In order to be able to aggregate the matrices we defined a common representation with we call a *coreference preference matrix* (CP matrix). Note that the coreference can be understood as a graph $G = \langle A, L \rangle$ where the A is the set of mentions, and $L \in A \times A$ contains those pairs of mentions that refer to the same character or entity. In this graph, cliques (fully connected sets of nodes) represent coreference groups. A CP matrix m is an $l \times l$ matrix. $m(i, j) = 1$ means that the corresponding method believes that e_i and e_j are expressions for the same referent; $m(i, j) = -1$ means the method believes they are not; and $m(i, j) = 0$ means the method is unsure (intermediate values are allowed, to indicate degrees of confidence).

For this work, we used six coreference resolution methods, represented as six CP matrices:

- m_1 is derived directly from the output of the *Stanford Coreference Resolution* system. Only yields -1s and 1s.
- m_2 computes coreference by assigning a 1 between two mentions when there are matching common nouns and proper nouns in the leaves of the extracted parse trees for each mention.

Yields 0s and 1s. m_2 intends to consolidate mentions that use similar referring expressions (e.g., “a girl” and “the girl”).

- m_3 computes coreference based on 15 lexical and syntactic features, separating mentions that do not have gender or number agreement between noun-noun, noun-pronoun and pronoun-pronoun pairs (e.g., “girl” and “him”). This matrix only contains 0s or -1s
- m_4 computes coreference by considering 8 lexical and syntactic features and computing the similarity between these features in the two mentions. This captures semantic similarities between mentions (e.g., “girl” and “sister”), and yields preferences in the interval $[0, 1]$.
- m_5 computes coreference based on the narrative roles for the mentions, fed back via a feedback loop described below. It assigns a 1 between two mentions when they have the same predicted role (except for the *other* or *non-character* roles). Yields 0s and 1s. m_5 intends to consolidate mentions to the same character, assuming not too many characters share the same role.
- m_6 computes coreference based on the roles and *character/non-character* labels fed back via the feedback loop. It assigns a -1 between two mentions when they have different labels (either different roles, or if one is a character and the other is not). Yields 0s and -1s. m_6 intends to prevent coreferencing mentions that clearly do not refer to the same character or entity, based on the character and role predictions.

Notice that the last two of the sources m_5 and m_6 depend on information that is not available yet at this stage of the pipeline. In order to incorporate this information into the coreference resolution process what we will do is run several iterations of the pipeline and after the first one, we use a feedback loop to feed the extracted information into these modules. The first iteration over the pipeline is illustrated by the solid arrows in Figure 4.4.

Feedback Loop

Voz implements an architecture inspired by the idea of hermeneutic circle[80] by incorporating a feedback loop from the output of the system to the input of the coreference resolution process.

$$w_1 \underset{m_1}{\begin{array}{c|c|c|c|c} & e_1 & e_2 & e_3 & e_4 \\ \hline e_1 & 1 & 1 & 0 & -1 \\ e_2 & 1 & 1 & 0 & 0 \\ e_3 & 0 & 0 & 1 & 0 \\ e_4 & -1 & 0 & 0 & 1 \end{array}} + w_2 \underset{m_2}{\begin{array}{c|c|c|c|c} & e_1 & e_2 & e_3 & e_4 \\ \hline e_1 & 1 & 0 & 1 & 1 \\ e_2 & 0 & 1 & 0 & 0 \\ e_3 & 1 & 0 & 1 & 0 \\ e_4 & 1 & 0 & 0 & 1 \end{array}} + \ldots + w_6 \underset{m_6}{\begin{array}{c|c|c|c|c} & e_1 & e_2 & e_3 & e_4 \\ \hline e_1 & 0 & -1 & 0 & -1 \\ e_2 & -1 & 0 & 0 & 0 \\ e_3 & 0 & 0 & 0 & 0 \\ e_4 & -1 & 0 & 0 & 0 \end{array}} \longrightarrow \underset{G = \langle E, L \rangle}{\begin{array}{c|c|c|c|c} & e_1 & e_2 & e_3 & e_4 \\ \hline e_1 & 1 & & 1 & \\ e_2 & & 1 & & \\ e_3 & 1 & & 1 & \\ e_4 & & & & 1 \end{array}}$$

Figure 4.8: Example pair-wise coreference restrictions and preferences adjacency matrix. Independent weights are assigned to each, aggregated, and a threshold used to determine pair-wise coreference grouping between mentions.

After the first iteration is complete and for every subsequent iteration, the output of the *Character Identification* and *Role Identification* modules are feed back and encoded into m_5 and m_6 within the *Coreference Resolution* module. This is illustrated by the dashed arrows in Figure 4.4.

Aggregating Coreference Information

Once we have identified potential methods to obtain coreference information, we have to aggregate the output of these methods. The output from the modules described in the previous section is in the form of an adjacency matrix. These matrices are aggregated cell by cell into a joint *coreference assignment* using the method described below. Figure 4.8 illustrates the entire process.

Once the available CP matrices are computed for an iteration, they are aggregated in the following way. First, a new matrix m_{merged} is generated as:

$$m_{merged}(i,j) = \begin{cases} 1 & \text{if } 0 \leq \sum_k w_i \times m_k(i,j) \\ 0 & \text{otherwise} \end{cases}$$

This new m_{merged} matrix represents a new CP matrix encoding a coreference assignment. However, this matrix might not be fully consistent (e.g., if $m_{merged}(1,2) = 1$ and $m_{merged}(2,3) = 1$, then $m_{merged}(1,3)$ must be also 1). This matrix is thus made consistent by adding the missing 1s, and turned into a graph G, where each clique represents a coreference group. This new coreference graph G is then used downstream in the pipeline in place of the original output from the *Stanford Coreference Resolution* system.

Experimental Results

For our experimental evaluation we analyzed the effect of the feedback loop in the performance of the different modules of the system, and the effect of varying the weight applied to the CP matrices that come form the feedback loop during coreference resolution. In this work, our dataset contains 21 stories and we manually edited the text to remove quoted and direct speech (i.e., dialogues and passages where the narrator addressed the reader directly). Our edited dataset contains 914 sentences. The stories range from 14 to 69 sentences ($\mu = 43.52$ sentences, $\sigma = 14.47$). Despite their relatively short lengths, understanding these stories requires significant commonsense knowledge and contextual inference. For example in one of the stories, there are two young female characters, who are both referred as “she”, “daughter” and “maiden” throughout the story while they fulfill different narrative roles.

Our experimental evaluation uses annotations for all noun phrases (NP) representing referring expressions (4791 mentions) and coreference information for the characters (2781 mentions). The characters were also annotated with the 6 character role labels described in the previous section. Additionally, we created an even coarser classification including only *Hero*, *Villain* and everything else as *Other*.

The experimental results below are reported using both the set of six, and the set of three roles. The weights used for the different CP matrices during coreference resolution were $w_1 = 1.0$, $w_2 = 1.1$, $w_3 = 10$ and $w_4 = 0.9$ (specific values are not important, and the only important aspect is that $w_2 > w_1$, and that w_3 is sufficiently large as for canceling all other weights out). Our dataset only contains coreference annotations for characters, and thus we only evaluate the performance of coreference resolution on the 2781 mentions that are annotated as characters. Our coreference process groups those 2781 mentions into to 1359 coreference groups. To evaluate the accuracy of the process, based on our ground truth annotations, we compute the average number of different characters found per coreference group (C/Gr), and the average number of different groups a single character is spread across (Gr/C). Perfect coreference would score C/Gr = 1.00, and Gr/C = 1.00 meaning that each group only contains mentions to one character and a character is mentioned in

Table 4.8: Performance of the first three iterations of *Voz* using 6 role classes. Last two rows display the theoretical upper bound using the ground truth and a random baseline.

	Coref. Resolution			Char.	Role	
	$\|Gr\|$	C/Gr	Gr/C	Acc	Acc	Acc (F)
It 1	1359	1.07	6.00	0.87	0.64	0.44
It 2	888	1.21	4.36	0.87	0.64	0.44
It 3	888	1.21	4.36	0.87	0.64	0.44
GT 6	642	1.24	2.90	0.88	0.73	0.46
Rnd.	642	1.93	3.11	-	-	-

Table 4.9: Performance of the first three iterations of *Voz* using 3 role classes. Last two rows display the theoretical upper bound using the ground truth and a random baseline.

	Coref. Resolution			Char.	Role	
	$\|Gr\|$	C/Gr	Gr/C	Acc	Acc	Acc (F)
It 1	1359	1.07	6.00	0.87	0.67	0.48
It 2	952	1.18	4.56	0.87	0.67	0.48
It 3	952	1.18	4.56	0.87	0.67	0.48
GT 3	788	1.18	3.44	0.88	0.73	0.47
Rnd	788	1.75	3.21	-	-	-

only one group respectively. Errors in coreference resolution will make these values higher. Our method achieves a C/Gr = 1.07, and Gr/C = 6.00. This means that while *Voz* is relatively good at separating mentions from different characters, it does not work so well at merging different mentions of the same character. The mentions each character are grouped, on average, in 6 different coreference groups.

We then assess the efficiency of the feedback loop where the coreference resolution process has access to all six CP matrices. This results in a different coreference assignment that will be used in the next iteration, yielding a different set of characters and potentially different roles for each. Weights for the first four matrices are the same as in the previous experiments, $w_5 = 0.9$, and $w_6 = 10.0$. The rationale for these weights is that $w_4 + w_5 > w_1$, thus when both m_4 and m_5 predict that two mentions should be grouped, that can overrule the decision made by the Stanford parser (m_1). w_6 is just set to a very large weight, since mentions with different roles cannot refer to the same character. Specific values for the weights do not have a strong impact as long as these relationships are kept.

Tables 4.8 and 4.9 show the obtained results when trying to predict the six-role annotations and the three-role annotations respectively. Each of the first three rows of the tables shows the performance of the system during the first three iterations: “It 1” just runs *Voz* as a pipeline, “It 2” runs the feedback loop once, and “It 3” runs it twice. Accuracy for character and role identification is reported after voting, and “Acc (F)” is the accuracy when measured only for those mentions classified as characters. We observed how the number of coreference groups decreases over the first iterations and remains stable afterwards. From the initial 2781 mentions, the Stanford deterministic coreference system yields 1359 groups. Considering results in Table 4.8, a second iteration, using the six CP matrices further reduces the number of coreference groups to 888 while improving coreference performance. The average groups per character is greatly reduced to Gr/C $= 4.36$, while the average characters per group only grows slightly to C/Gr $= 1.21$. Moreover, in order to calculate the upper-bound for the improvement in performance achieved by the feedback loop, we experimented by feeding back the ground truth for role labels. This is shown in the row labeled “GT 6” in Table 4.8, showing an even better coreference resolution performance: reducing Gr/C further to 2.90 while C/Gr increased only slightly to 1.24, and reducing the number of groups from 1359 to 642, which is a significant improvement. Character and role predictions also improved (from 0.87 to 0.88 for characters and from 0.67 to 0.73 for roles), showing further potential of our feedback loop approach. Finally, in order to validate that the role information is useful, we tried joining coreference groups randomly (starting with the coreference groups obtained in iteration 1) until the same number of groups (642) were reached. This resulted in really worsening results, further indicating that role information is indeed useful in improving coreference resolution. Table 4.9 shows similar trends in the three-role prediction setting. Although still far from the ideal 173 coreference groups in the ground truth, the feedback loop clearly increases the performance of coreference resolution. Moreover, we observed no significant impact of the feedback loop on character and role predictions, which improved only marginally (an improvement smaller than the precision shown in the tables). The output of the system stabilized in the third iteration which obtained the same exact results as the second.

Finally, we experimented with different values for weight w_5, which is the weight given to the

Table 4.10: Performance comparison with different weights for m_5, built form the feed back role information.

	Coref. Resolution			Char.	Role
	$\|Gr\|$	C/Gr	Gr/C	Acc.	Acc.
$w_5 = 0.1$	933	1.17	4.46	0.87	0.65
$w_5 = 0.25$	894	1.20	4.36	0.87	0.65
$w_5 = 0.5$	890	1.21	4.36	0.87	0.65
$w_5 = 0.9$	888	1.21	4.36	0.87	0.65
$w_5 = 1.5$	879	1.22	4.34	0.87	0.65

CP matrix generated from the fed-back roles. This has the expected effect: higher w_5 results in role predictions having a stronger effect, and thus result in a lower coreference group count. Table 4.10 shows the results of different values for w_5 after the second iteration using the 6 roles. As the table shows, the main effect of increasing w_5 is reducing the number of coreference groups. From 1359 (with $w_5 = 0.1$) to 879 (with $w_5 = 1.5$) while only increasing the ratio of characters per group (C/Gr) slightly (from 1.17 to 1.22). Effects on character and role prediction are smaller than the accuracy precision shown in the table (<0.01).

Our experiments confirm that the idea of feedback loops can increase the performance of some of the modules of the system, specifically coreference resolution, but not others, such as character or role identification. The feedback loop introduced in the NLP pipeline of our system has been demonstrated to be particularly successful in reducing the number of coreference groups in the output of the system.

4.6 A Machine Learning Approach to Predicting Proppian Narrative Functions from Stories in Natural Language

In the work described so far we have been using our narrative information extraction pipeline (see Figure 4.4) to build upon previously extracted information and compute higher-level narrative information. At the forefront of Propp's work[25] are his narrative functions that he defines as the building blocks for the structure of a story. In this section we present an approach to identify those narrative functions automatically from text segments.

In the context of narratology, a narrative function is a fundamental building block of storytelling:

"an act defined in terms of its significance for the course of the action in which it appears; an act considered in terms of the role it plays at the action level"[154]. Specifically, in this work, we draw from Vladimir Propp's theory of narrative functions described in his Morphology of the Folktale[25].

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

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

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

In order to address this problem, we present an approach that integrates supervised learning, narrative domain knowledge and probabilistic inference, described below. In this work, we use a set $\mathbb{F}$ of 34 narrative functions derived from Propp's original collection of narrative functions. Table 4.11 enumerates the 34 narrative functions in $\mathbb{F}$ and the number of instances in the dataset used in our experiments.

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

- S_i is divided into a sequence of contiguous, non-overlapping, potentially empty text segments

Table 4.11: Enumeration of the narrative functions for our experimental evaluation ($|\mathbb{F}| = 34$). The table includes the symbols Propp used to refer to them, and the total number of instances in our dataset of 15 stories. The third column is the total number of instances annotated, the fourth column, the number of those that are actually realized in the text (i.e., explicit) and the fifth column the number of instances once the direct speech (e.g., dialog) is filtered out.

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

$[s^i_1, ..., s^i_{m_i}]$ where narrative functions are realized (i.e., if a part of a story does not realize any function, then that part might not be present in any of the text segments).

- Each text segment s_j^i is automatically converted to a feature vector x_j^i, and manually associated to a narrative function $f_j^i \in \mathbb{F}$ (a description of the annotations and features in the feature vector is provided in the experimental evaluation section).

This results in two training datasets:

- A text segment to narrative function mapping dataset:

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

- A function sequences dataset:

$$D_s = \{[f_1^1, ..., f_{m_1}^1], ..., [f_1^n, ..., f_{m_n}^n]\}$$

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

D_f and D_s are used to train our system. At run time, in order to predict the functions associated with a given sequence of text segments $[s_1, ..., s_m]$, we employ our system *Voz*[3] to automatically extract the necessary narrative information (characters, narrative roles, and actions) from the natural language text and translate each text segment s_i into a feature vector x_i which is then used for prediction.

Our approach is based on a search process over the space of possible function sequences, returning the sequence with highest probability as predicted by a set of predictors and aggregated by a module we call *joint inference*. In the following itemization we describe our proposed probabilistic predictors. Then we will describe the search process in the *joint inference* module. Figure 4.9 illustrates the overall workflow of the system.

- **Local Predictor (k-nn).** Our first predictor is a supervised machine learning predictor that uses the D_f training dataset mapping feature vectors to narrative functions. Given a

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

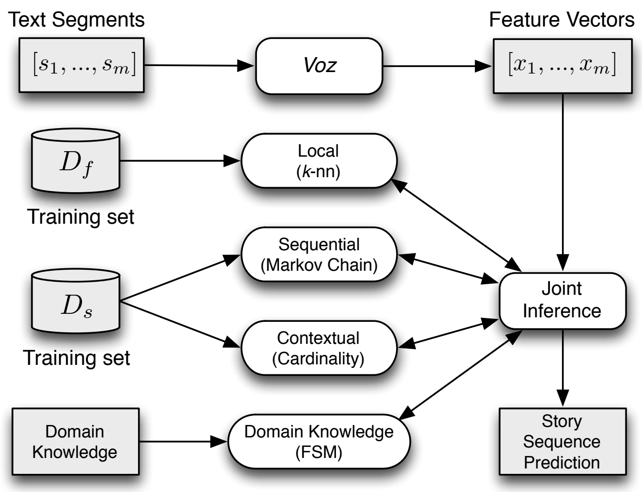


Figure 4.9: Overall system diagram. Both databases indicate automatically learned training sets from our annotated dataset. *Voz* is our previous narrative information extraction system which we use to automatically compute feature vectors from text segments in a given story. The output is a sequence of narrative function predictions such as: $\langle \alpha, \beta, \delta, A, B, C, \uparrow, H, I, K, \downarrow, W \rangle$

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

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

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

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

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

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

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

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

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

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

- **Domain Knowledge Predictor (FSA).** This last predictor uses domain knowledge and encodes our interpretation of the rules in Propp's narrative theory.A subset of the rules in Propp's theory was manually encoded in a probabilistic finite state machine, namely:
  - The precedence relationships defined by Propp's ordering may not be violated.
  - Each function may only appear once (our dataset only includes stories with a single *move*,

as described below).

– The first function must be one of the introductory functions ($\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \lambda$), villainy ($A$) or lack ($a$).

– Return ($\downarrow$) should not appear without prior departure ($\uparrow$).

– Rescue (Rs) should not appear without prior pursuit (Pr).

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

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

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

Search

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

Beam search evaluates the nodes at depth i using the four predictors (except the contextual predictor that can only be used for complete sequences, i.e., only for the leaf nodes of the search tree), and discards all but the top N candidates (where N is the size of the beam, set to 10000 in our experiments) before expanding the next level of depth $i + 1$. The assignment with highest likelihood at the end is returned as the sequence of narrative functions for the input sequence of feature vectors $[x_1, ..., x_m]$ representing a story S.

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

Experimental Results

For our experimental evaluation, in this work, we use a dataset of 15 stories. The dataset contains 23291 tokens (words and punctuation) and includes annotations for referring expressions, coreference, verbs, semantic roles, narrative roles and narrative functions. There is a total of 190 annotated narrative functions. In this work we ignore narrative functions that are implicit or do not have an explicit text realization. The dataset was manually filtered to include text segments where narrative functions are realized. The experiments reported in this work use 167 text segments corresponding to the explicit functions covering a total of 3915 tokens. The sequence of segments is not altered in any other manner and it is processed in the order it is mentioned in the text of the story. All the stories in our dataset were identified by Propp as single *move* (a higher-level narrative structure defined by Propp that contains narrative functions). We report results with 3 different scenarios:

- *Automatic*: In this scenario, the input to the system are the sequences of text segments in natural language without any further annotation. Our system uses *Voz* to automatically process the text and build the feature vectors. *Voz* is not currently able to process direct speech (e.g., dialog) and thus narrative functions that appear solely in it are removed from the dataset before the experiments. The dataset used to run experiments in this scenario contains 130 text segments which span over 2342 tokens.

- *Annotated*: This scenario, instead of using *Voz* to generate the feature vectors, they are computed directly from the dataset annotations contained within each text segment and includes the 167 text segments. This scenario provides an upper bound that ignores any error introduced by *Voz* due to inaccuracies in the natural language processing and information extraction.

- *Filtered Annotated*: This scenario replicates the *Annotated* scenario, except that removes text segments contained within direct speech (e.g., dialog) in order to provide results comparable to the *Automatic* scenario (i.e., using the annotations from the filtered 130 text segments).

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

We performed an exhaustive analysis of the different combinations of the predictors described earlier. For simplicity we report a representative subset of the experimental results. Moreover, we tested using the FSA domain knowledge predictor in two different ways: 1) using it during the search process as all other predictors, and 2) only using it once we reach the leaves of the search process. We call these two scenarios "FSA Search" and "FSA Leaves" respectively. This is done since the FSA predictor used slightly more computational resources than the others, and using it during the search increases execution time significantly. The number of neighbors k in k-nn, the neighbor distance metric, the value a of the Laplacian smoothing pseudocount, the number of features to include during selection, the size of the beam and the verb grouping were set experimentally.

To validate our hypothesis and evaluate the usefulness of the individual predictors and their combination we repeated the application of our methodology to several combinations of predictors in the three different scenarios described above. Table 4.12 reports the accuracies of the different combinations.

To provide a reference to our results, the bottom row of Table 4.12 shows the accuracy obtained

Table 4.12: Classification accuracy obtained in different scenarios. The first five columns describe which predictors were used. The remaining three columns report the classification accuracies for narrative function predictions in three different scenarios. The last row shows the accuracy obtained by predicting the most common function in the dataset.

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

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

The first five rows of Table 4.12 show the performance obtained by each of the four predictors in isolation (and also when we used the FSA during the search or only at the leaves). As we can see, when used in isolation, the k-nn local predictor achieves the best performance (unsurprising, since it is the only one that considers the actual text features to make predictions). The Markov Chain predictor achieves the second best accuracy, since it is accurately able to predict the first functions of a story (stories regularly start with the α function and are often followed by A). Even with low performance, notice the local predictor by itself doubles the performance of the informed baseline in all three scenarios.

The rest of the rows in Table 4.12 show the performance of our approach with an increasing number of predictors (all the way to using all four predictors, in the bottom rows). As we can see, the highest performance in all scenarios is achieved when combining at least three predictors. We do not include rows when the k-nn predictor was not used, since none of those produced good results. This indicates that integrating information from multiple sources (including Propp domain knowledge) significantly improves performance. The highest performance achieved by our system is between 0.271 and 0.287 depending on the scenario, which is close to three times that of the informed baseline.

In our experimental evaluation we can compare the performance of the local supervised machine learning predictor (k-nn) and the different combinations with sequential (Markov), contextual (Cardinality) and domain knowledge (FSA) predictors. Starting from an informed baseline, a predictor predicting one of $\uparrow$ or $\downarrow$ would have an accuracy of 0.084 in the first scenario (*Annotated*) and up to 0.109 in the other two filtered scenarios (*Automatic* and *Filtered Annotated*). After a close inspection of the automatic feature selection process, we can identify a single feature, the position of the function, that contributes 0.149 in the *Annotated* and 0.159 in the filtered scenarios. Adding the remaining features to the local supervised machine learning predictor the accuracy increases to 0.210 in the *Automated* scenario and 0.202 in the *Automatic* scenario. These numbers indicate that our choice of features, although potentially incomplete, has a significant contribution to the prediction of the local predictor. Although there may be other annotations in the annotated dataset that could be better indicators, our choice of features can be computed automatically and we observe how the accuracy in the *Automatic* scenario also increases. Since this second scenario comprises a separate set of function annotations, the third scenario of the *Filtered Annotated* dataset gives us an upper bound of the performance of the narrative function identification should we be able to extract perfect information using *Voz*, our narrative information extraction system.

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

Focusing on the *Annotated* scenario, we can observe how every combination of predictors improves the performance of the predictions with respect of the local predictor (k-nn). Including the contextual (Cardinality) and domain knowledge either during search (FSA Search) or at the end of the search (FSA Leaves) our approach achieves 0.281 accuracy.

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

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

In conclusion, these results show that our approach significantly outperforms an informed baseline. Our experimental results confirm our initial hypotheses that combining top-down narrative theory and bottom-up statistical models inferred from an annotated dataset increases prediction accuracy with respect to using them in isolation.

4.7 Identifying Dialogue Participants

Storytellers use a wide variety of rhetoric figures when telling a story. A common element in written stories told in third person is the use of *quotes* (i.e., phrases and sentences surrounded by quotation marks) that contain direct speech as spoken by a character of the story. These quotes are often arranged in succession representing a conversation or *dialogue*. Also, they are usually accompanied by *speech attribution cues* (such as “he said to her”) in order to help the reader identify the *speaker* and the intended *listener* for the quote. Although common, these cues are some times missing (such as in dialogue sequences) or partial (“he said” only) and the information about speaker and listener is left implicit for the reader to infer. In this work, we focused on the problem of identifying the speaker and the intended listener of the different quotes in a given story in order to use this information for tasks within our narrative information extraction pipeline (see Figure 4.4). Please note that the dialogue or quoted speech in our corpus was annotated and removed from the datasets used for all the work reported so far in Section 4.2 through Section 4.6. There are interactions that appear solely within dialogue and those were lost. This was specially problematic for the work on identifying narrative functions described in Section 4.6 where some functions would exist only within dialogue. Additionally, the speech act information encodes interesting interactions that were missing in the output of *Voz*. Let us illustrate these interactions with an example. Given the following text snippet:

Then the dragon began to implore Nikita: “Do not put me to death [...]. Let us divide all the earth [...]” “Very well,” said the tanner, “let us draw a boundary line.”

From the previous example, after processing the story we were able to extract three mentions to two characters: *The dragon* and *Nikita the tanner* (*Nikita* and *the tanner* are both mentions to the same character). Additionally, we are interested in extracting the interaction between the two, specifically, *The dragon* asking *Nikita* and *Nikita* replying to *The dragon*. Specifically, we would like to identify that the *speaker* of the quote *“let us draw a boundary line.”* is *the tanner* and the *dragon* is the *listener*. Moreover, we aim at extracting information about character interactions in terms of *speech acts* (e.g., asking or answering) and use the coreference information extracted previously to identify the canonicak participants (e.g., resolve *Nikita* and *the tanner* as the same character). We finally encode this interaction as a list of triplets such as $[\langle \textit{ask}, \textit{dragon}, \textit{Nikita}\rangle, \langle \textit{reply}, \textit{Nikita}, \textit{dragon}\rangle]$

We use these triplets as a basis for representing character interactions and we use them for further processing downstream, for example, for compiling the character interaction matrices described earlier in Section 4.4.

In the rest of this section we describe a module that pre-processes natural language stories that include quoted text using off-the-shelf NLP tools and uses machine-learning and narrative domain knowledge to extract the aforementioned pieces of information. Specifically, we focus on interactions between characters in quoted speech and dialogue which we previously ignored in our work. We propose a method to identify participants in dialogue in stories using pattern induction. We evaluate the extracted patterns using both our annotated corpus of 20 Russian stories translated to English and the Quoted Speech Attribution Corpus[4]. We also assess the accuracy with which we can use the extracted patterns to predict speaker and listener and how much this extra information helps improve the accuracy of other narrative information extraction tasks such as narrative role prediction.

The core of our approach is to translate the input text into a sequence of *items* or specialized tokens (where the items represent parts of the text such as a quoted utterance or a mention to an entity) and then use an induction algorithm to extract patterns that can predict the participants in each quote given the corresponding sequences of items.

Therefore, the problem statement that we are addressing in this work can be described as: Given a story written in natural language which contains a set of quotes Q and a set of mentions representing characters P, the goal is to identify, for each quote $q_i \in Q$ its most likely *speaker* (i.e., the mention speaking the quote) $q_i^s \in P$ and it is most likely *listener* (i.e., the mention the quote is intended for or is being spoken to) $q_i^l \in P$. The output is a set of tuples $Q' = \{\langle q_i, q_i^s, q_i^l \rangle, ...\}$, where each $q_i \in Q$, and q_i^s and q_i^l and the predicted speaker and listener.

In the remainder of this section we assume that the set of quotes Q and the set of mentions representing characters P have already been identified in the text. Note that for our experimental evaluation we first use annotated sets for Q and P in our dataset and then we report our results

[4]Available: http://www.cs.columbia.edu/nlp/tools.cgi#Quoted%20Speech%20Attribution%20Corpus

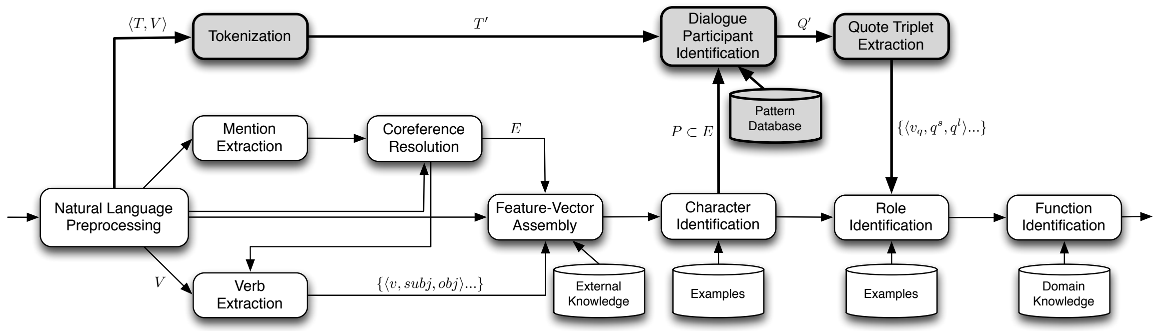


Figure 4.10: Overview of our narrative information extraction system, the shaded boxes represents the dialogue participant identification modules introduced in this work. The new modules are highlighted in grey. The feedback loop is still present but has been removed from the figure for clarity.

using sets automatically identified by *Voz* (which may include errors in e.g., coreference resolution). Additionally we also report results on the Quoted Speech Attribution Corpus used by Elson and McKeown[149]. This last dataset only contains annotations for the speaker of the quote therefore in the training and evaluation we ignore the listener.

In order to identify the *speaker* and *listener* mentions for each of the quotes in Q, we build upon the work by Elson and McKeown[149] on quoted speech attribution (speaker identification). Specifically, we base our work on identifying syntactic categories for quotes by using a specialized pattern matching procedure. We extend their work in two ways: First, we attempt to identify both the *speaker* and *listener*, and extract a limited amount of *speech act* information to enrich our downstream narrative information extraction. Second, we introduce a pattern definition language and propose an automatic induction process from example data that extracts patterns that can be examined and tweaked. Finally, we apply our approach to a new dataset that features a substantial use of different rhetoric figures compared to their dataset (e.g., we observed a much higher frequency of mentions using common nouns instead of proper nouns compared to their dataset).

Pre-Processing and Tokenization

Before proceeding to dialogue participant identification, *Voz* extracts a set of verbs V using part-of-speech information and a set of character mentions P. Figure 4.10 shows an overview of the new dialogue participant identification modules (shaded boxes) in the context of *Voz*.

The information extracted by the *Natural Language Preprocessing* module is used to convert the story into a sequence of items or specialized tokens. This process is similar to first step in the work by Elson and McKeown. Specifically, we take a sequence of tokens (such as words and punctuation) generated by an off-the-shelf NLP tool (the Stanford CoreNLP in our case), and then perform a series of replacements using previously extracted information and discarding irrelevant tokens. Note that these tokens are contain additional information such as part-of-speech tags, lemmatized words and offset information from the original text.

The input to this process is a sequence of tokens $T = [t_0, \ldots, t_n]$ representing a story, a set of verbs V, and a set of mentions P, where each verb $v_i \in V$ and each $p_i \in P$ corresponds to a subsequence of tokens from T.

Given this input, T is scanned for quotation marks, and a set of quotes Q is extracted where each $q_i \in Q$ is a subsequence of tokens $t_a, \ldots, t_b$ that starts and ends with a quotation mark, and where there are no quotations marks in between. Finally we extract a set of punctuation tokens K where each $k \in K$ is a token t that matches either a *dot* (.) or a *colon* (:) outside of quotation marks. All the verbs and character mentions that appear within any of the quotes in Q are ignored (they are removed from V and P). Therefore the union of V, P, Q and K is a set of non-overlapping subsequences of tokens in T.

A string of items T' is then generated by replacing each of the subsequences of tokens representing verbs in V, mentions in P, quotes in Q and punctuations tokens in K by items or specialized tokens representing them, all the other tokens in T are ignored, and are not included in T'; therefore the output is a sequence of items, each of which wraps a subsequence of the tokens in T and maintains the order in which these appear in T. We consider the following types of items:

v**:** A verb and the lemma of the verb.

p**:** A mention and the annotated *character* or the automatically extracted coreference group (i.e., the character(s) this mention refers to, such as *Nikita* and *the tanner* in the example fragment from the introduction).

k**:** Either a *dot* (.) or a *colon* (:).

q: A quote and the text inside the quotation marks, including punctuation, such as exclamation marks, question marks, commas, etc. If the speaker or listener of this quote has already been identified, this information is also stored here. Which is referred to in the patterns below as $speaker(q)$ and $listener(q)$.

Using the same example fragment from the beginning of this section, this process outputs the following sequence of items; in parenthesis we illustrate relevant information contained within the items that will be used later, specifically: the lemma of the verbs, the coreference group of the characters, the text in punctuation or the text of the last token before the closing quotation mark (mostly punctuation), we use subscripts to differentiate items in the sequence:

p_1(Dragon) v_2(begin) v_3(implore) p_4(Nikita) k_5(:) q_6(.) q_7(,) v_8(say) p_9(Nikita) q_{10}(.)

Pattern Definition Language

In order to predict the speaker and listener of each quote in the sequence of items, we use a pattern matching procedure. Let us now describe the proposed pattern definition language. Each pattern $r = \langle m, f_s, f_l \rangle$ is composed by a matching pattern $m = [s_1, ..., s_n]$ defined as a sequence of symbols in a symbol alphabet S, a mapping function f_s which maps each quote in m to its speaker (which could be a character mention in m, of the speaker or listener already identified in a previous quote), and a mapping function f_l, which maps the quotes to their listeners. These patterns represent the different syntactic categories, such as those identified by Elson and McKeown. The matching pattern M is a sequence of symbols m that match items in the sequence of items T'. We consider the following symbol alphabet for the matching patterns:

V: matches any verb v.

T: verbs v with the lemma *think*.

A: verbs v with the lemmas *ask*, *question*.

R: verbs v with the lemmas *answer*, *reply*.

S: verbs v with the lemmas *say*, *tell*.

E: verbs v with any of the lemmas matched by T, A R S or *call*, *beg*, *cry*, *yell*.

Q: matches any quote q.

Q!: a quote q where the second-to-last token is either a dot (.), an exclamation mark (!) or a question mark (?).

Q,: all other quotes (in our dataset, these mostly end with a comma).

P: matches any character mention p.

K: matches any punctuation mark k.

K.: matches dots (.).

K:: matches colons (:).

Moreover, by prepending a question mark (?) to any symbol we can make it optional.

The verbs in the previous enumeration items (T, A, R, S, E) were selected by analyzing the speech acts defined in Propp's narrative functions[25].

Pattern Application

With a given set of patterns, either manually crafted or automatically extracted (as described in the next section), pattern application works as a two step process. In the first step, a set of patterns R_1 is used to identify candidates for speakers and listeners for each quote. In the second step, a set of patterns R_2 uses the candidates identified by the patterns in R_1 to propagate them to other quotes. For example, a pattern in R_1 might identify a certain mention to be the speaker of a quote, and a second pattern in R_2 might use that prediction to identify the speaker of follow-up quote (i.e., R_1 does a first pass of identification, and R_2 expands it).

Given a set of patterns R (either R_1 or R_2), the pattern application process iterates the items in the sequence T' and tries to find matching patterns for each quote q_i in the sequence. When a matching pattern is found, the speaker or listener indicated in the pattern is stored as a candidate for the quote. Conflicts (different patterns making different predictions for the same quote) are resolved by assigning weights to patterns (the pattern with highest weight has preference). Weight calculation is described later in the next section. In our implementation, for each pattern in R, we initialize a non-deterministic finite state machine to ingest the items in the sequence T' one at a time and store the candidates for the quote's matched by a given pattern.

Given the following sets of patterns:

$R_1 : \{\langle[\mathrm{Q}_1\ \mathrm{S}_2\ \mathrm{P}_3], f_s = \{\mathrm{Q}_1 \rightarrow \mathrm{P}_3\}, f_l = \{\}\rangle\}$

$R_2 : \{\langle[\mathrm{Q},_1\ ?\mathrm{V}_2\ ?\mathrm{P}_3\ \mathrm{Q}!_4], f_s = \{\mathrm{Q}!_4 \rightarrow \mathit{speaker}(\mathrm{Q},_1)\}, f_l = \{\}\rangle\}$

And the example sequence from the previous section (again, we use subscripts to differentiate each symbol when they appear more than once in a sequence):

p_1(Dragon) v_2(begin) v_3(implore) p_4(Nikita) k_5(:) q_6(.) q_7(,) v_8(say) p_9(Nikita) q_{10}(.)

the application of the pattern in R_1 yields the following:

$Q' = \{\langle \mathrm{q}_7, \mathrm{p}_9, -\rangle$

then, applying the pattern in R_2 would expand the predictions to the following output:

$Q' = \{\langle \mathrm{q}_7, \mathrm{p}_9, -\rangle, \langle \mathrm{q}_{10}, \mathrm{p}_9, -\rangle\}$

which means the system has identified p_9 (*Nikita*) to be the speaker of both quotes, but it has not identified which is the listener of either of them yet. Furthermore, since P has been already preprocessed, there should be coreference information available linking p_9 and p4, therefore patterns in R_2 should be able to exploit this information.

We considered creating these patterns manually, which proved impractical. For example, those defined by Elson and McKeown only covered a small percentage of our dataset, given the variety of ways in which quotes appear in folktale text. We report results with the manually authored patterns in the experiments below for comparison.

Automatically Learning Patterns for Dialogue Participant Identification

In order to extract the patterns, our approach requires a training set containing sequences of items T' where each quote q contains annotations for the speaker and listener. Then, the overall idea is to first select a subsequence around a quote and for each mention $p \in P$ that matches either the annotated speaker or listener, we generate a pattern. Next, we generalize each pattern iteratively using a series of pattern generalization operations storing each of the generated generalizations. Finally, weights are assigned to each pattern based on their performance on the training set. This process is described in detail below.

1. **Pattern Extraction**. Given a window of items $[t_a, ..., t_b]$ of T', of length between 1 and W (a predefined maximum window size), if a window contains any quote $q \in w$ for which the speaker or listener of q appears in any of the other items in the sequence (either a mention or another quote with the same speaker or listener), then a pattern is extracted. This process is shown in Algorithm 1 (where we use subindexes to indicate the position of each item in the input sequence T'). The *toSymbols* function turns a sequence of items into a sequence of the most specific symbols that match those items, as defined in the previous section. For example, for the first sentence in the example used in the previous section, one of the extracted patterns would be as follows:

   $r_1 : \langle[\mathrm{P}_1\ \mathrm{V}_2\ \mathrm{V}_3\ \mathrm{P}_4\ \mathrm{K}{:}_5\ \mathrm{Q!}_6], f_s = \{\mathrm{Q!}_6 \rightarrow \mathrm{P}_1\}, f_l = \{\}\rangle$

2. **Pattern Generalization**. We define a list of possible generalizations for each symbol in a given pattern. Then, a systematic process generates all the possible generalizations of a given pattern by application of one or more generalizations. For example, we define the possible generalizations of the symbol $\mathrm{K}{:}_i$ to be the set $\{\mathrm{K}_i$, $?\mathrm{K}{:}_i$, $?\mathrm{K}_i$, $\emptyset$ $\}$ (where $\emptyset$ means removing

Algorithm 1 Pattern Extraction Algorithm

```
1: procedure ExtractPatterns($T', W$)
2:     $R_1 \leftarrow \emptyset$    ▷ these patterns will be used in the first step of pattern application
3:     $R_2 \leftarrow \emptyset$ ▷ these will be used in the second step to propagate predictions from the first set of patterns
4:     for $q_i \in Q$ do    ▷ for each quote $q_i$ in the set of quotes
5:         for $w \in [1, \ldots, W]$ do
6:             for $s \in [s - w, i]$ do
7:                 $window \leftarrow subsequence(T', s, s + w)$    ▷ $window$ is a sequence of symbols around $q_i$
8:                 for $p \in \{t \in window | t \in P\}$ do    ▷ for each mention $p$ in the current $window$
9:                     if $speaker(q_i) = p$ then    ▷ if the mention $p$ is the speaker of the current quote $q_i$
10:                        $R_1 = R_1 + \langle toSymbols(window), f_s = \{q_i \rightarrow p\}, f_l = \{\}\rangle$
11:                    if $listener(q_i) = p$ then
12:                        $R_1 = R_1 + \langle toSymbols(window), f_s = \{\}, f_l = \{q_i \rightarrow p\}\rangle$
13:                for $q_j \in \{t_j \in window | j < i \ \wedge \ t \in Q\}$ do ▷ now, iterate other quotes within the $window$
14:                    if $speaker(q_i) = speaker(q_j)$ then    ▷ if there is a match between speakers or listeners
15:                        $R_2 = R_2 + \langle toSymbols(window), f_s = \{q_i \rightarrow speaker(q_j)\}, f_l = \{\}\rangle$    ▷ add pattern for propagating predictions
16:                    if $listener(q_i) = listener(q_j)$ then
17:                        $R_2 = R_2 + \langle toSymbols(window), f_s = \{\}, f_l = \{q_i \rightarrow listener(q_j)\}\rangle$    ▷ idem
18:                    if $speaker(q_i) = listener(q_j)$ then
19:                        $R_2 = R_2 + \langle toSymbols(window), f_s = \{q_i \rightarrow listener(q_j)\}, f_l = \{\}\rangle$    ▷ idem
20:                    if $listener(q_i) = speaker(q_j)$ then
21:                        $R_2 = R_2 + \langle toSymbols(window), f_s = \{\}, f_l = \{q_i \rightarrow speaker(q_j)\}\rangle$    ▷ idem
22:    return $\langle R_1, R_2\rangle$
```

the symbol from the pattern). Considering only the generalizations of K:$_i$, the following patterns would be generalized from the extracted pattern shown above:

$r_2 : \langle[\mathrm{P}_1\ \mathrm{V}_2\ \mathrm{V}_3\ \mathrm{P}_4\ \mathrm{K}_5\ \mathrm{Q!}_6], f_s = \{\mathrm{Q!}_6 \rightarrow \mathrm{P}_1\}, f_l = \{\}\rangle$

$r_3 : \langle[\mathrm{P}_1\ \mathrm{V}_2\ \mathrm{V}_3\ \mathrm{P}_4\ ?\mathrm{K:}_5\ \mathrm{Q!}_6], f_s = \{\mathrm{Q!}_6 \rightarrow \mathrm{P}_1\}, f_l = \{\}\rangle$

$r_4 : \langle[\mathrm{P}_1\ \mathrm{V}_2\ \mathrm{V}_3\ \mathrm{P}_4\ ?\mathrm{K}_5\ \mathrm{Q!}_6], f_s = \{\mathrm{Q!}_6 \rightarrow \mathrm{P}_1\}, f_l = \{\}\rangle$

$r_5 : \langle[\mathrm{P}_1\ \mathrm{V}_2\ \mathrm{V}_3\ \mathrm{P}_4\ \mathrm{Q!}_5], f_s = \{\mathrm{Q!}_5 \rightarrow \mathrm{P}_1\}, f_l = \{\}\rangle$

We report experiments with two different generalization procedures; the first only considers one type of generalization: making symbols optional by prepending a question mark, which we call *limited generalization* (as in r_3); and the second that produces all the possible generalizations described above which we call *full generalization*.

3. **Weighting and Pruning**. The extraction and generalization processes may produce thou-

sands of patterns, making prediction inefficient, and introducing potential conflicts, where different patterns make different predictions. Thus, we assign each pattern a weight to resolve conflicting predictions by assessing its performance in the training set. The weight $w(r)$ of a pattern r is set as the classification accuracy of the pattern in the training set, corrected using Laplacian smoothing: $w(r) = \frac{correct+1}{total+2}$ (where *total* is the number of times r matched in the training set, and *correct* is, out of those, how many times the prediction was correct). Any pattern with weight below 0.5 is pruned.

The final output of the dialogue participant identification module is a set of quotes Q' that, for each quote $q_i \in Q$, includes a prediction for the quote's speaker $q_i^s \in P$, the quote's listener $q_i^l \in P$, and which pattern matched q_i^r.

We then convert this information into the formalism used downstream in our narrative information extraction pipeline (as seen in Figure 4.4). Specifically, we use triplets that represent interactions between entities (i.e., characters and other characters or props). These are extracted from the verb and the verb's arguments in the form: $\langle$ verb, subject, object $\rangle$. We use the information of the matched pattern to identify the actual verb (e.g., *beg* in the example in the previous section) or a representative verb for each *speech act* (e.g., *ask* for the pattern symbol A) and the speaker and listener as the subject and object respectively. The output of this triplet extraction module is a set of triplets containing one triplet for each $q \in Q'$ of the form $\langle v_q, q^s, q^l \rangle$, where v_q is the verb identified by the speech act of the pattern or *say* if no speech act information is available.

4.7.1 Experimental Results

For our experimental evaluation we studied three scenarios using two corpora:

- *Russian Stories*: The first set of experiments use a corpus of 20 Russian stories translated to English. We use a dataset consisting of annotations for verb and semantic role labels, mentions, coreference, Proppian narrative roles for characters, Proppian narrative functions, speaker and listener for quoted speech and other quoted speech. The dataset contains 1931 reported speech sentences plus 718 sentences in quoted speech (29.5% of the text). Within the reported speech sentences, there are 3029 mentions to 236 unique characters (46% common nouns, 43% proper nouns, 11% pronouns) and 5131 annotated verbs. In the next section we report experiments on this corpus using *Voz* to automatically extract this information.

- *Quoted Speech Attribution Corpus*: The second dataset is the quoted speech attribution corpus used by Elson and McKeown[149]. This dataset only contains annotations for the speaker of the quote, therefore in the training and evaluation we exclude the listeners. Regarding the syntactic categories we observed a very different distribution compared to the ones identified by Elson and McKeown[149]. The $\langle[\text{Q E P}], f_s = \{Q \to P\}, f_l = \{\}\rangle$ pattern covers only 9% of our dataset and has an accuracy of 97% (compared to 19% and 99% respectively for Elson and KcKeown) and the analogous Q P E pattern covers 4% with accuracy of 80% (compared to 2% and 92%). For the evaluation of the dialog participants we evaluate independently the speaker and the listener. For evaluation, we skip 2.8% of the quote's speakers and 8.2% of the quote's listeners in cases where there isn't a single obvious speaker or listener such as in: *"Well, brothers," said one of the three [to the other two brothers].*

Unless stated otherwise, all the experiments reported in this section use a leave-one-story-out protocol where we perform cross-validation using 19 stories in the training set and the other one in the test set. All results are microaveraged by the number of instances in each story.

- **Russian Stories**. We first evaluated several baselines on the annotated *Russian Stories* dataset:
  - *baseline*: assigning the speaker to the most common character and the listener to the second most common character (reported in row 1 of Table 4.13);
  - *manual*: using a set of 44 manual patterns derived from Elson and McKeown's work and including additional patterns for listeners (row 2 of Table 4.13);
  - the rules in *manual* with fallback to *baseline* when none matched (row 3 of Table 4.13);
  - *permutations*: permutations of the 44 manual patterns (row 4 of Table 4.13); and;
  - *permutations* with fallback to *baseline* (row 5 of Table 4.13).

Table 4.13: Results on the *Manual Russian Stories* dataset for several approaches concerning: number of patterns generated, the number of patterns after pruning, the accuracy in predicting the *speaker*, *listener* and their weighted average for each quote and the coverage for the same slots.

		# Patterns		*Accuracy*			*Coverage*		
		Bef.	Aft.	Spk.	List.	Both.	Spk.	List.	Both.
1	Baseline		1	0.403	0.246	0.325	1.000	1.000	1.000
2	Manual		44	0.822	0.897	0.838	0.414	0.115	0.265
3	Manual+Baseline		45	0.573	0.326	0.450	1.000	1.000	1.000
4	Permutations	30K	947	0.703	0.604	0.670	0.756	0.372	0.564
5	Permutations+Baseline	30K	948	0.663	0.431	0.547	1.000	1.000	1.000
6	Induction	3.8K	1.2K	0.773	0.487	0.585	0.319	0.609	0.464
7	Ind.+Generalization	8.5K	3.5K	0.773	0.546	0.648	0.782	0.962	0.872
8	Ind.+Full Generalization	60K	38K	0.717	0.581	0.645	0.871	0.982	0.927
9	Ind.+Full Gener. (0.4)	60K	55K	0.708	0.576	0.642	0.976	0.983	0.980
10	Ind.+Full Gener. (0.6)	8.5K	2.9K	0.933	0.533	0.578	0.107	0.842	0.475

Table 4.13 summarizes the results. In the case of the manual patterns, we achieve an overall accuracy of 83.9% with a coverage of 26.5% on the whole data set. By using the informed baseline for fallback we achieve a coverage of 100% with an accuracy of about 44.9%. Automatically generating permutations of these 44 patterns we generate 30805 patterns and after pruning patterns with an accuracy below 50% we are left with 947 patterns that cover 56.4% of the test set with an overall accuracy of 67.0%.

We then ran a first experiment training on the whole data set and observed that our pattern induction process, without generalization, extracts about 1400 useful patterns and covers 96.4% of our data set. it is worth noting that we ran our procedure without generalization on a cross-validated scenario and the coverage fell to 45.9%. We report results of our experiments using our automatic induction method in 4 scenarios: using induction without any generalization (row 6 of Table 4.13); using the *limited generalization* (row 7 of Table 4.13); using the *full generalization* (row 8 of Table 4.13); and; using the *full generalization* and lowering the threshold for pruning patterns to 0.4 (row 9 of Table 4.13). Intuitively, in the cross-validation evaluation, using induction without generalization lowers the coverage to 46.4% with an accuracy of 58.5%. Using generalization increases the coverage to 87.1% of the dataset with an accuracy of 64.8% (77.3% for speakers). Using *full generalization* and lower threshold, we observed greater number of patterns and increased coverage at the expense of accuracy confirming our hypothesis that 0.5 is a good threshold for our desired output. In these scenarios, we attribute the ceiling in the coverage of our approach at around 98% to the fact that in our experiments we used a window with a maximum size of 8 around the quotes. Increasing the window size without a bigger dataset, however, would lead to overfitting. We performed a final experiment increasing the threshold to 0.6 (row 10 of Table 4.13) and we observed how pruning additional rules increases the accuracy for identifying speakers to 93.3% but coverage for speakers is reduced to 10.7% and the overall coverage drops to 47.5%.

- **Quoted Speech Attribution Corpus**. We applied our approach to the dataset used by Elson and McKeown[149]. This dataset only contains annotations for the speaker of the quote therefore in the training and evaluation we ignore the listener. We first applied the 44 manually defined patterns targeting their syntactic categories: *Quote-Said-Person trigram*, *Quote-Person-Said trigram* and *Anaphora trigram*; and including the same permutations described in their work. Our results (on speaker only) match their reported values (accuracy 95%, coverage 29%). We applied our proposed pattern induction method using their original files to perform 16-fold cross validation. Our results in this dataset are an accuracy 65.6% and coverage 67.2% (on speaker only).

In conclusion, after this work we can observe how our approach performs similarly than state-of-the-art work for speaker identification but can also identify the listener and does not require manually defined syntactic categories.

4.7.2 Evaluating Automatic Narrative Information Extraction with Dialogue Participant Interactions

Now we present results where we compare the performance of the *Narrative Function Identification* module described in Section 4.6 with and without the dialogue interaction information extracted in this section.

We performed experiments using the pipeline described in Figure 4.10. Notice that the new modules introduced for this section are bolded. We focused the evaluation the performance of the

Narrative Function Identification module, which attempts to identify Proppian narrative functions in the stories. In this setting, the input of the Dialogue Participant Identification module (the sets P and Q) is generated automatically from the natural language text by running the Mention Extraction, Coreference Resolution and Character Identification modules. The character interactions are described using triplets of the form: $\langle \mathit{say}, \mathit{Nikita}, \mathit{dragon} \rangle$. *Voz* extracts these triplets from the text outside quoted speech.

When running without Dialogue Participant Identification, out of the 5131 verbs annotated in the dataset, our system is capable of identifying 1240 interaction triplets (with subject and object). Using this information the system is capable of identifying narrative functions with an accuracy of 23.4%. The information from the Dialogue Participant Identification contributes 560 additional triplets (with some duplicates). Using these additional triplets, the overall performance for identifying narrative functions improves to 31.3%. Note that narrative functions identification is a notoriously difficult task, since there are 34 different narrative functions, some of which have some degree of ambiguity and where an informed baseline only identifies 10.9% of the functions. We refer the reader to the previous section in this chapter for more information.

In addition to an improvement in function identification accuracy, an added benefit of processing dialogue is that a 28% of the functions in our dataset appear in dialogue (67 out of 239 functions), which are not identifiable without extracting triplets from dialogue (23.8% of the narrative functions in dialogue were identified correctly). Table 4.14 provides a comparison of the results of the *Narrative Function Identification* module with and without using the dialogue participant information extracted by the work described in this section.

4.8 Discussion

In this chapter we presented a collection of techniques to extract different narrative elements from text. In order to evaluate our ideas, we incorporated them into an automated narrative information extraction pipeline called *Voz*. We use *Voz* for experimental evaluation throughout this dissertation.

In Section 4.1 we described the overall architecture of *Voz*. Then in sections 4.2 through 4.7 we describe our individual contributions that we incorporated into *Voz*.

Table 4.14: Comparative between the scenarios without using and using the dialogue participant information extracted by the work described in this section. The triplets column reports the number of triplets available for the *Narrative Function Identification* module to use. The remaining two columns are the accuracy and number of functions identified. The last row reports the total annotations present in the dataset.

Scenario	Triplets	N.F. Acc.	N.F. #
w/o Dialogue	1240	23.4%	56
w/ Dialogue	1800	31.3%	75
Annotated	5131		239

Through our work, we realized that our corpus poses specific challenges to several NLP tasks even when using state-of-the-art NLP techniques. These challenges are more prominent in our application domain of fictional stories that exhibit non-standard patterns and features. For example, specific to character identification and related to named entity recognition, the text includes a range of different types of characters: humans, anthropomorphic animals (e.g., a talking mouse) and anthropomorphic objects (e.g., a magical oven, a talking river). There are also fantastical creatures (e.g., goblins) and characters specific to the Slavic folklore (e.g., Morozko and Baba Yaga).

We worked on improving these tasks and we realized that in order to improve the overall accuracy of the system, we could incorporate narrative domain information into core NLP and information extraction tasks. In the next two chapters we will focus on how we evaluated our narrative information extraction pipeline in order to identify areas of improvement and how we worked to improve it by incorporating narrative domain knowledge by incorporating non-linear features to our pipelined architecture.

Chapter 5: Evaluation of Information Extraction Pipelines

Information extraction pipelines (such as *Voz*) are usually composed of a number of modules, each of which is in charge of extracting a certain type of information. Evaluating each module in the pipeline in isolation provides useful information but it is often the case that while some modules generated inaccurate predictions, those inaccuracies had little effect on the predictions of other dependent modules downstream. At the same time, modules could have yield very accurate predictions, but the small number of errors they incur may have a large impact on other modules downstream and on the overall performance of the system. This was also observed when evaluating the work presented in the previous chapter. Since coreference information is used in several voting processes, small errors in the coreference graph had a bigger impact on multiple modules downstream. We observed the opposite case with the error in the verb extraction module. Despite it having a large error, intrinsic noise in the data and the inner workings of the role and function identification tasks meant that the misidentified verb arguments had a smaller impact than anticipated on those modules' output. In general, we observed that it was difficult to assess the contribution of each of the modules in the pipeline to the final error in the output of the system.

In our attempts to analyze our system, we realized that, despite a large body of work on empirical methods for AI[168] including the design of factorial experiments, there were limited known methods to evaluate how errors propagate in a pipeline. For example, there are no established methodologies on how to assess the contribution of each module, or how the error introduced by one module impacts the performance of later modules in the pipeline. Thus, in this section we present a methodology designed to assess how error is introduced and propagated in information extraction pipelines. Specifically, the proposed methodology aims at answering the following questions:

Q1: What is the error introduced by each module?

Q2: How much does the error introduced by one module affect a later module in the pipeline?

Q3: How much does the error introduced by one module affect the final output of the pipeline?

In order to address these questions, we propose a systematic methodology which we evaluated by applying it to *Voz*, our narrative information extraction pipeline.

5.1 Systematic Methodology for Evaluating Information Extraction Pipelines

Our methodology is based on 4 steps. The first step encodes a pipeline as a graph and is designed to abstract the underlying system in order to apply our methodology to arbitrary pipelines. The second step helps system builders generate an annotated dataset to use for the rest of the evaluation. In the third step we use the annotated dataset to evaluate individual modules and finally, in the fourth step we use the annotated dataset to study error propagation.

The four steps of our methodology are detailed below:

1. *DAG and Topological Ordering.* First, formalize the information extraction pipeline as a directed acyclic graph (DAG). The modules of the pipeline are represented as graph nodes and the edges represent dependencies between the modules. Then, compute one *topological ordering*[1] of the nodes in the DAG. A module m_i is the ith node in the topological ordering of the DAG. Figure 5.1 illustrates the representation of *Voz* (described in Chapter 4 Section 4.1) as a DAG.

2. *Incremental Dataset.* Then, annotate an *incremental ground truth dataset* that contains annotations for each of the modules m_i at each intermediate step of the pipeline. We use GT_i to denote the ground truth for the output of the module m_i. In order to alleviate the burden of annotation, it is often possible to run the automated pipeline and extract an automatically generated dataset from the output of each module m_i which then is manually corrected and stored as GT_i.

3. *Individual Module Evaluation.* Next, evaluate each module m_i using as input GT_{i-1} and comparing the output against GT_i. We use m_1 to refer to the first module of the system and

[1] The topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge uv from node u to node v, u comes before v in the ordering. Any DAG has at least one topological ordering, which can be computed in linear time[169].

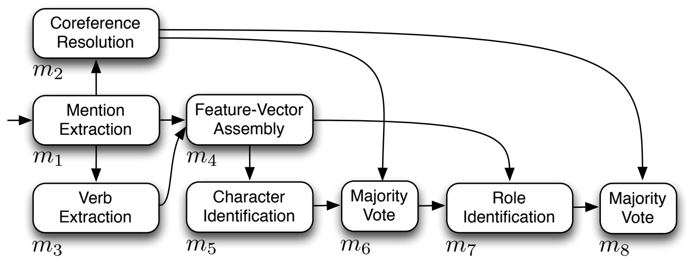


Figure 5.1: Directed acyclic graph encoding a representation of the dependencies in *Voz*, our narrative information extraction pipeline. Note that the DAG includes two *Majority Vote* modules not included in Figure 4.4 but described in Chapter 4 Section 4.3.5. Also, the *Function Identification* module was not considered in this work.

GT_0 as the original input to the system, which, for example, in our case is just one Russian folktale in natural language without any additional annotation. Note that the individual modules may have module-specific parameters that can be fine-tuned during this step to study the effect of those parameters on the module's error contribution.

4. *Error Propagation.* Finally, given a pair of modules of interest m_a and m_b where $m_a \prec m_b$, we compare the performance of m_b in two settings: first, by feeding GT_{a-1} to m_a then running the pipeline from m_a to m_b, and second, by feeding GT_a to m_{a+1} then running the pipeline from m_{a+1} to m_b. In this way, we can measure the change of error in m_b, as a result of the error introduced by m_a.

In order to study the error, our methodology uses the standard metrics of precision, recall and f-measure (P/R/F), and defines error as 1 minus f-measure. The reported P/R/F values in the remainder of this section are the weighted averages of the P/R/F for each different solution classes a module can predict. For example, when reporting P/R/F for role prediction, we compute P/R/F for each individual role, and then report the average weighted by the number of times each role appears in the ground truth. However, we make an exception for the coreference resolution module where the f-measure was not very informative, and we use a different measure, characterizing the spread of a single character: the average number of coreference groups with a reference to a given character; and the misgrouping of different characters: the average number of distinct characters referred to in a single coreference group. These two metrics will help us explain the impact of the error of coreference resolution. In this work we discuss how error propagates down the pipeline, and compare the modules' error to two baselines:

- *Random*: which generates predictions randomly.
- *Informed Baseline*: which always predicts the most common solution in the training set.

Besides the metrics described above, we also apply the bias and variance decomposition framework described in Kohavi and Wolpert[170] to further identify the error introduced by the training data in the character and role identification modules. We only report the bias and variance estimators, since, as reported by Kohavi and Wolpert, intrinsic error is always zero when estimated from standard training sets where all instances are unique. To perform the bias and variance decomposition, for each of the stories in the dataset we perform 50 iterations learning a classifier and using it to classify the mentions of the selected story. The classifier is trained on a randomly sampled subset from the rest of the stories with a size equal to half the total size of the dataset available.

5.2 Evaluating Voz

In this section we move on the application of the proposed methodology to evaluate the performance of *Voz* using our corpus of Russian folk tales. We will first describe how we built the incremental dataset used for the evaluation, how the modules described in the previous chapter were evaluated individually, and finally, the evaluation of the pipeline itself focusing on the interactions between select modules.

5.2.1 Incremental Dataset

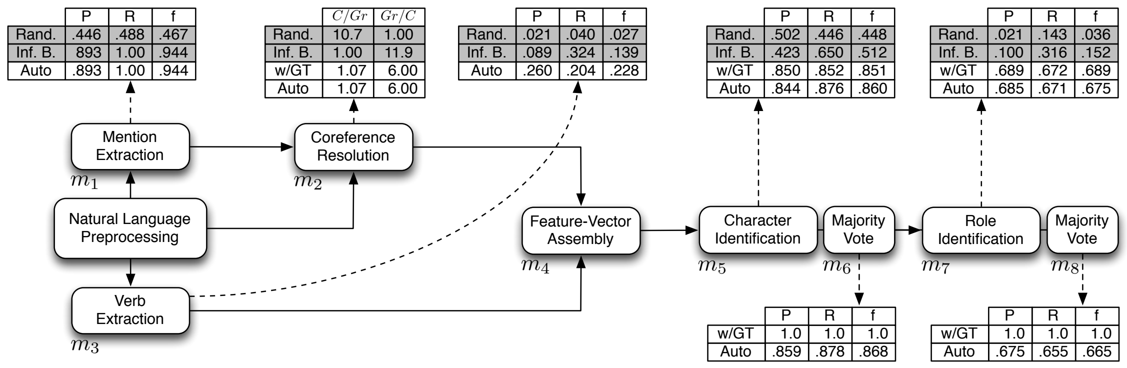
Our input dataset (GT_0) contains 21 unannotated Russian folk tales translated to English text, a subset of the 28 stories described in the previous chapter in Section 3.1. To reduce preprocessing issues at the discourse level, we manually removed quoted and direct speech (i.e., dialogues and passages where the narrator addressed the reader directly). The edited input dataset (GT_0) contains 914 sentences. The stories range from 14 to 69 sentences ($\mu = 43.52$ sentences, $\sigma = 14.47$). There is a total of 18,126 tokens (words and punctuation; $\mu = 19.83$ words per sentence, $\sigma = 15.40$).

To quantify the performance of the different modules in *Voz*, we added a series of annotations on the dataset as our ground truth $GT_1 \ldots GT_8$. Following our proposed methodology, we used GT_0 as the input to our pipeline and proceeded to extract and correct the output for each module following the topological order enumerated in the previous subsection. We started running the *Mention Extraction* (m_1) module and from the output, we annotated 4,280 noun phrases (NP) representing referring expressions (GT_1). These were used as a basis for annotating the coreference (GT_2), character labels (GT_5, GT_6) and role labels (GT_7, GT_8). The coreference groups are annotated for characters and mentions to groups of characters (e.g., "her parents" $\mapsto$ {"father", "mother"}, "they" $\mapsto$ {"king", "queen"}). Coreference annotations for non-characters were not included. The characters have been annotated with the character role labels described by the Proppian functions[25]. We merged the roles of *Donor* and *Helper* since, in our dataset, they mostly correspond to the same character and *Dispatcher* into an *Other* along with several other minor roles. This resulted in the 6 role labels: *Hero*, *False Hero*, *Villain*, *Sought-for-person*, *Tester* (which includes *Donor* and *Helper* exhibited by the same character in our dataset) and *Other* (which includes *Dispatcher* and other minor roles). The annotations were performed by two researchers independently and disagreement resolved manually by consensus.

For the *Verb Extraction* module (m_3) we annotated verbs and the character's actions as triplets of the form $v = \langle \text{verb, subject, object} \rangle$, where subject (i.e., executor) and object (i.e., receiver) may be empty. In the ground truth (GT_3) there are a total of 1,586 annotated verbs that correspond to 3,029 triplets across the 21 stories. There are more triplets than verbs because when a verb had multiple arguments that could be considered receivers, multiple triplets were annotated. The triplets may also refer to a list of characters (i.e., they) which are expanded with its individual mentions. For example, the sentence "they went to work" where *they* refers to "the father and the mother", involves one verb annotation that yields two triplets: $\langle$*go,father,work*$\rangle$ and $\langle$*go,mother,work*$\rangle$).

5.2.2 Individual Module Evaluation

To evaluate the error introduced by each of the modules we used the dataset presented above. Each of the following experiments evaluates the performance of module m_i by feeding the ground truth



	P	R	f
Rand.	.446	.488	.467
Inf. B.	893	1.00	.944
Auto	.893	1.00	.944

	C/Gr	Gr/C
Rand.	10.7	1.00
Inf. B.	1.00	11.9
w/GT	1.07	6.00
Auto	1.07	6.00

	P	R	f
Rand.	.021	.040	.027
Inf. B.	.089	.324	.139
Auto	.260	.204	.228

	P	R	f
Rand.	.502	.446	.448
Inf. B.	.423	.650	.512
w/GT	.850	.852	.851
Auto	.844	.876	.860

	P	R	f
Rand.	.021	.143	.036
Inf. B.	.100	.316	.152
w/GT	.689	.672	.689
Auto	.685	.671	.675

	P	R	f
w/GT	1.0	1.0	1.0
Auto	.859	.878	.868

	P	R	f
w/GT	1.0	1.0	1.0
Auto	.675	.655	.665

Figure 5.2: Architecture of the *Voz* system and a summary of the error for each module. The first two rows report performance baselines for each module. *Rand.* reports a random baseline and *Inf. B.* an informed baseline using the most common class in the training set to cast predictions. Then, we report the performance of each module running the fully automated narrative extraction pipeline (*Auto*) and applying our methodology to isolate the error introduced by each module using our incremental dataset (*w/GT*). In this scenario, module m_i is given as input the annotated output of the previous module (GT_{i-1}).

GT_{i-1} as input, and evaluating against GT_i.

The experiments follow the topological order presented before. In the experiments we skip the *Feature Vector Assembly* as it involves aggregating the output from other modules and external knowledge (e.g., Conceptnet, Wordnet) which we do not target in our current work.

Mention Extraction

There are a total of 4,791 individual mentions identified by our algorithm, 4,280 correspond to noun phrases and 511 of which are not actual referring expressions but parsing errors, mostly adjectival phrases identified as nominal phrases. Our method has a recall of 100% (all of the annotated mentions were found) but a precision of 89.3% ($f = 0.944$) matching the informed baseline. The full 4,791 mentions were included in the annotated dataset with specific labels for the 511 non-referring expressions in order to use the annotations for the remaining of the experiments. The results of this module are summarized in Figure 5.2.

Coreference Resolution

Our GT_2 only contains coreference annotations for characters, and consequently we can only evaluate the performance of coreference resolution on the 2,781 mentions that are annotated as characters.

Our coreference process groups those 2,781 mentions into to 1,359 coreference groups. We found that the overall precision, recall and f-measure metrics for this process are not representative of the module's performance due to class unbalance (it is much more common for a pair of mentions not to refer to the same character, than the other way around). To compute the f-measure for coreference resolution, we represent the coreference graph as a binary adjacency matrix (where a 1 in column i and row j represents that mentions i and j belong to the same coreference group, and a 0 means otherwise). The f-measure of the coreference resolution module is 0.757, which is artificially high, since the adjacency matrix consists mostly of zeroes (precision/recall for zeroes in this matrix are 0.839 are 0.962 respectively), masking the precision and recall for ones (which are 0.752 and 0.342 respectively). Thus, we used two additional metrics: the average number of distinct characters referred to in a single coreference group (C/Gr) and the average number of coreference groups with a reference to a single character (Gr/C). Perfect coreference would score $C/Gr = 1.00$ and $Gr/C = 1.00$ meaning that each group only contains mentions to one character and a character is mentioned in only one group respectively. Errors in coreference resolution will make these values higher. *Voz* obtains $C/Gr = 1.07$ and $Gr/C = 6.00$. This means that while *Voz* is relatively good at separating mentions from different characters, it does not work so well at merging different mentions of the same character. The results are summarized in Figure 5.2. The random baseline reported in the first row joins mentions at random and to maintain consistency ends up grouping all the mentions together (10.7 is the average number of characters per story). On the other hand, the informed baseline will not join mentions and create singleton coreference groups (11.9 is the average number of mentions for each character). The third row (w/GT) reports performance of m_2 when the ground truth (GT_1) is used as input. The fourth row (*Auto*) reports the performance of *Voz* feeding GT_0 to m_1 and its output to m_2. Since *Mention Extraction* (m_1) has perfect recall and no mentions of non-characters are merged with mentions of characters, there was no difference in performance between w/GT and *Auto*.

Table 5.1: Character's actions: Performance of the verb extraction when checking the verb, the verb and its subject, and the verb, subject and object arguments. *# in GT* is the number of expanded triplets in the ground truth.

	# in GT	P	R	f
Verb	1,586	0.802	0.666	0.728
V+Subject	2,627	0.534	0.383	0.446
V+S+Object	3,029	0.260	0.204	0.228

Verb Extraction

Our method identifies 1,335 verbs out of the 1,586 annotated ones in the ground truth across the 21 stories. Considering only the verb part of the triplet, our current method has an average precision of 0.802, recall of 0.666 and $f = 0.727$. The missing verbs are not extracted mainly due to parsing issues (e.g., a verb phrase parsed as a noun phrase) and a few action nouns that cannot be identified as verbal nouns (e.g., "the arrival"). When comparing the full triplet, our current method has an average precision of 0.260, recall of 0.204 and $f = 0.228$, as shown in Table 5.1. This is a difficult task given the big space of possible combinations as highlighted by the random (randomly assigning mentions to the subject and object slots for each verb) and informed baselines (randomly assigning a subject but no object). Although the overall performance of verb argument extraction is very low we will later see that this does not affect much the global performance of *Voz*. The results of this module are summarized in Figure 5.2.

Character Identification

There are 2,781 characters annotated in GT_5. Figure 5.2 contains the performance of the predictions for the character identification process before voting. The random baseline reflects a binary classification and the informed baseline predicts the majority class of *non-character*. The third row (w/GT) reports performance of m_5 when feeding the ground truth (GT_4) as input. The fourth row (*Auto*) reports the performance of this module when *Voz* is run completely automated from the beginning up to this point. When feeding ground truth the module yields a precision of 0.850, recall of 0.852 and $f = 0.851$, which is slightly worse than the performance achieved when *Voz* runs automatically because an additional set of mentions are identified by the *Mention Extraction* module which are easily classified as *non-characters*, thus improving the resulting f measure.

Analyzing the error of this module using the bias and variance decomposition (error = bias^2 + variance) indicates that the error is decomposed as $0.145 = 0.109 + 0.035$. Note that to compute the bias and variance decomposition, we generate subsamples of the training data, and thus error might be slightly different than reported in Figure 5.2. While bias represents intrinsic error, it is well known that the error caused by variance can be reduced by ensemble learning techniques and by increasing the size of the dataset, pointing out some direction for future work.

Character Identification Majority Voting

Figure 5.2 shows the precision, recall and f-measure after the voting process. Again, the first row (w/GT) uses GT_5 as input to this module, and the second row (*Auto*) reports the results of the output of *Voz* running completely automated up to this point. As Figure 5.2 shows, the majority voting process in module m_6 does not contribute to any error in the pipeline (it has perfect precision and recall when fed the ground truth GT_5).

Role Identification

Figure 5.2 shows the performance of the predictions for the role identification process. The reported precision, recall and f-measure values are averages weighted by the total number of annotated mentions in each class in the ground truth (GT_7). As the results show, considering the difficulty of this task, character role identification is very accurate with a precision of 0.689, recall of 0.672 and $f = 0.689$ compared to random and informed the baselines (recall there are 7 roles with *Hero* being the most common and used for the informed baseline). Also, notice that performance of this module when fed with ground truth is only slightly higher than when fed with the automatic output computed by *Voz*. This means that most of the final error of this task comes from the module itself (or from noise in the data), and not from error introduced by previous modules.

For this module we also study the source of error using the bias and variance decomposition analysis replicating the process used in the *Character identification* module. The error is decomposed as $0.533 = 0.342 + 0.097$ (bias^2 + variance). Upon closer inspection of our results, we noticed that the high intrinsic error of this module (bias) is due to the fact that our approach is biased towards the most recurrent roles (*Hero* or *Other*) and is neglecting smaller but still important roles (*Villain* or *Sought-for-person*).

Role Identification Majority Voting

Figure 5.2 shows the precision, recall and f-measure after the voting process, where w/GT and *Auto* have the same meaning as before. As reported for m_6, the majority voting process does not contribute to any error in the pipeline (as we can see from the perfect f-measure score when fed with ground truth values).

Individual Module Evaluation Summary

Figure 5.2 summarizes the performance of our narrative information extraction pipeline. With the exception m_1 and m_3 that receive as input the output of the Stanford CoreNLP, we report performance for the rest of the modules using our methodology and feeding the ground truth GT_{i-1} to each module (w/GT). For comparison, we also report the performance for each module when *Voz* runs completely automated using GT_0 (*Auto*). Looking at the performance results for each module we can quickly identify a significant error introduced by m_3.

5.2.3 Error Propagation

In this section we present our study on how the error propagates through the narrative information pipeline by pairing different modules as described in the last step in our methodology. In this step, we select pairs of modules m_a and m_b, where $m_a \prec m_b$, and we assess how the error introduced by m_a affects the performance of m_b. We report the results of the pairs of modules that yield most insight regarding the introduction and propagation of error.

$m_1 \rightarrow m_5$ (effect of the error introduced by Mention Extraction on Character Identification)

The *Mention Extraction* module (m_1) has a recall of 100% and a very high precision. Still there are some parsing errors and misidentified mentions that may introduce error further down the pipeline. Table 5.2 reports the performance of the character identification process when module m_1 is bypassed by substituting its output directly by the ground truth (GT_1) , and when m_1 is actually

Table 5.2: Results of the character identification process (m_5) using the ground truth GT_1 and the automatic process of processing the input with the mention extraction module ($GT_0 + m_1$).

	# Mentions	P	R	f
GT_1	4,280	0.858	0.884	0.871
$GT_0 + m_1$	4,791	0.844	0.876	0.860

used ($GT_0 + m_1$). Since the recall of m_1 is 1.00, and the additional set of mentions extracted by m_1 that should not be extracted are easily identified by m_5 as not characters, the error introduced by m_1 is actually significantly reduced by m_5. The 5% of error introduced by m_1 contributes to less than 0.2% of the error of m_5, and thus the error of m_1 is reduced by a factor of 36 by the time it reaches m_5.

$m_3 \rightarrow m_5$ **(effect of the error introduced by Verb Extraction on Character Identification)**

To evaluate how the error of m_3 propagates down the pipeline to m_5, we report experiments under three conditions: 1) measuring the performance of m_5 when fed with the output of m_3 2) measuring the performance of m_5 when fed directly with GT_3, and 3) measuring the performance of m_5 if we completely remove m_3 from the pipeline. Table 5.3 (*Module:* m_5) summarizes these results. Our results indicate that having verb information improves the performance of the character identification module. Even when using the underperforming automated verb extraction, error is reduced by 7% with the precision, recall and f-measure improving from 0.832, 0.851 and 0.841 to 0.844, 0.876 and 0.860 respectively. Counterintuitively, performance when using ground truth (GT_3) is slightly lower than using the automatically extracted verb information but still with an overall error less than 15%. This artifact is due to the fact that one of the features in our data (whether a mention appears as the subject of a verb or not) is a significant contributor toward character classification. When using the ground truth, many entities that are not characters, and that are not recognized by the verb argument extaction module, are annotated as the subject of verbs (e.g., “the door creaked”), which is having a negative impact in character classification performance.

Table 5.3: Effect of the verb features (from m_3) on the character and role identification processes (m_5 and m_7). Rows report results using the automatically extracted verb features (m_3), using the verb features from the ground truth GT_3, and completely removing the verb features (GT_4 w/o Verbs).

Input	Module	P	R	f
m_3	m_5	0.844	0.876	0.860
GT_3	m_5	0.850	0.852	0.851
GT_4 w/o Verbs	m_5	0.832	0.851	0.841
m_3	m_7	0.685	0.671	0.675
GT_3	m_7	0.689	0.672	0.689
GT_4 w/o Verbs	m_7	0.618	0.595	0.602

Table 5.4: Effect of coreference information (from m_2) on the majority voting processes ($m6$ and m_8). Rows report results without coreference information, using the automatic coreference (m_2) and using the coreference from the ground truth GT_2.

Voting	Module	P	R	f
Without Voting	m_6	0.844	0.876	0.860
m_2	m_6	0.859	0.878	0.868
GT_2	m_6	0.896	0.839	0.868
Without Voting	m_8	0.685	0.671	0.675
m_2	m_8	0.644	0.624	0.629
GT_2	m_8	0.728	0.713	0.714

$m_3 \rightarrow m_7$ (effect of the error introduced by Verb Extraction on Role Identification)

To evaluate this effect, we used the same three conditions as with the previous experiment. Table 5.3 (*Module:* m_7) summarizes the results. What we observed is that the *Verb Extraction* module error has a significant impact on the output of the *Role Identification* module, yet much smaller than the impact on the previous *Character Identification* module. We also observed is that not using the verb features at all (last row in Table 5.3) shows a large impact on the output of about 10% in precision, recall and f-measure. Our hypothesis for this is that verb information play a large role in identifying the roles of characters since they capture the actions that characters exert upon each other. Not having this information at all hinders the performance of the *Role Identification* module but the difference between having some partially incorrect information or the ground truth has a small effect because of intrinsic noise in the data and the inner workings of the role identification task.

$m_2 \rightarrow m_6$ (effect of the error introduced by Coreference Resolution on Character Majority Voting)

We compare the automated pipeline feeding GT_1 to module m_2 and then we repeat the process, but bypassing m_2 and directly using the ground truth GT_2. Table 5.4 (*Module:* m_6) summarizes these results. We observe how the voting process in m_6 reduces the error from m_5 from 14% to 13.2% to regardless of wether we use m_2 or GT_2. This results indicates that the current character predictions are correlated with the groups already found in the coreference resolution information and therefore, the error in m_2 does not affect the error in m_6. Notice that this is expected, since coreference resolution tends to be conservative when making coreference groups, and creates small groups that contain mostly mentions to the same character ($C/Gr = 1.07$ and $Gr/C = 6.00$), and thus error does not affect the voting process as much as if errors were made in the other direction (few groups with mentions to different characters).

$m_2 \rightarrow m_8$ (effect of the error introduced by Coreference Resolution on Role Majority Voting)

Table 5.4 (*Module:* m_8) summarizes these results. In this case, the voting process using the automatically computed coreference hinders the overall performance of the module (with respect to not using coreference resolution at all) whereas the ground truth improves the performance for the module significantly. In this case, the 32.5% of error (f-measure of 0.675) at the output of m_8 could be reduced to 28.6% (f-measure of 0.714) with perfect coreference. Upon close inspection of our results, we saw that the few times coreference resolution grouped mentions that did not refer to the same entity together, they were always characters (and that's why the performance of m_6 is not affected), but it would sometimes group mentions to different characters in the same coreference group, thus affecting m_8.

Experimental Results

In our experimental evaluation we used the proposed methodology to produce an incremental dataset using and updated version of *Voz*. Our methodology elicited three phenomena: the error between some modules is mitigated (i.e., *Verb Extraction* to *Character Identification*), the error introduced by certain modules has a considerable impact in later modules (i.e., *Coreference Resolution* to *Role Identification*), and the error between some modules is independent (*Character Identification* and *Role Identification*). Figure 5.3 provides a visual representation of the error distribution through the pipeline, showing how the error introduced by each module propagates down the pipeline.

The main implication of the results presented in Figure 5.3 is that working toward improving modules such as verb extraction will have no immediate impact on the performance of the character and role identification tasks. The only module that if improved would significantly improve role identification is coreference resolution. Notice that this does not mean that verb extraction does not contribute toward character and role identification, but that the current performance of verb extraction is enough to improve results of character and role identification to the same extent as if verb extraction produced perfect predictions.

Overall, our automatic role identification results exhibit a 32.5% of error (f-measure of 0.675), 8.5% of which can be directly attributed to coreference resolution since the majority voting process could be improving the results of m_7 (f-measure of 0.675) by 13.5% (to have an f-measure of 0.714) but is actually hindering the results by 4.6% (obtaining an f-measure of 0.629). Of the remaining error introduced by the role identification module, less than 0.1% can be attributed to the verb extraction error and the rest comes from external knowledge error in the feature-vector assembly (m_4) and other biases in our processes and the training data.

On the one hand, using the *Verb Extraction* information (with a 55.4% of error when identifying the subject of the verb) is enough to correctly predict role labels for a few mentions referring to magical beings (such as a talking oven) and decrease the error for the role identification task by 7.3%. On the other hand, our methodology indicates that reducing the error of the *Verb Extraction* module would not decrease the error of the character and role identification tasks.

5.3 Discussion

The *Coreference Resolution* module had been already identified as a bottleneck in our previous research[167], but thanks to the methodology presented in this work, we can now assess exactly its contribution to overall error: when used for majority voting, the automatic coreference information

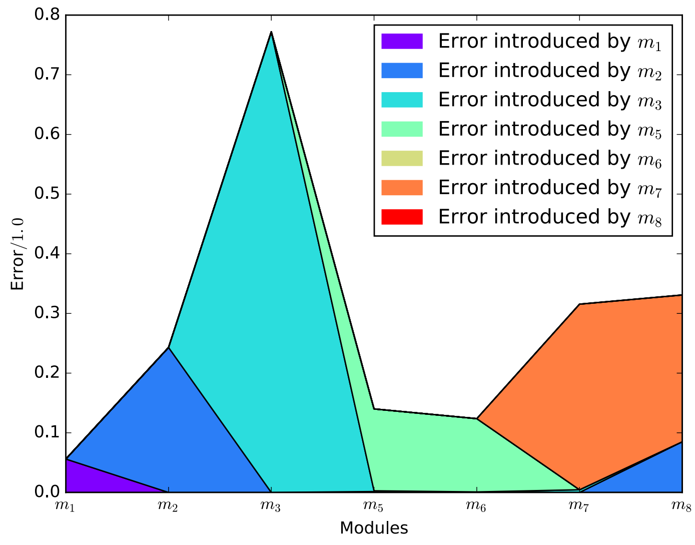


Figure 5.3: Summary of error distribution through the pipeline when running *Voz* using GT_0 as input. The error of a module goes to 0 when a subsequent module does not use the output of the module. m_4 is not included since we did not evaluate the error introduced by the external knowledge sources.

worsens the performance of the role identification process by 8.5%. Our methodology elicits the greater benefits of improving this module instead of the *Verb Extraction* module. The *Verb Extraction* module will be more prominent for other tasks such as identifying affective relationships between characters. If we pursue this task we can apply our methodology again.

Moreover, in order to properly understand the phenomena behind the low impact of coreference resolution in the voting process it was necessary to find the proper metrics to assess the performance and error. The two metrics we introduced (Gr/C and C/Gr) proved more informative through the process, specially as our system achieves a low misgrouping (C/Gr) which would otherwise have a much greater effect in voting.

Finally, when assessing the performance of the individual modules of our pipeline, we observed considerably higher error rates than in previously reported results of the NLP tool we currently use (specially for the coreference resolution task [29]). The discourse and rhetoric figures used in the stories in our dataset, the variability of the referring expressions and some of the scenarios described make it challenging to identify all the mentions of a single character. This leads to identifying multiple characters that are in fact the same one. This contrasts with the corpora typically used in NLP work such as the *CoNLL shared tasks*[28;29] or the *Penn Treebank Project*[171] which contain a much more regular language (i.e., Wall Street Journal text) used for training the models provided by the off-the-shelf systems and the test sets in the NLP shared tasks.

Through the application of our methodology, we learned that part of the error of certain NLP tasks may be mitigated by later modules in the pipeline. In our study we found that even though verb extraction has a low performance when studied in isolation, its output is sufficient to reduce the overall error of our system. This novel methodology can be applied to a wide variety of information extraction pipelines in order to elicit error contribution information that may not be obvious. It can inform decisions related to determining what modules may be preferred to invest improving and which may not yield major performance gains.

Chapter 6: A Probabilistic Framework for Non-linear Information Extraction Pipelines

The problem we address in this work is how to define a framework that allows for integrating results computed in different stages of an information extraction pipeline into a decision processes within the same pipeline, ideally without requiring modification of the internal function of these modules. The motivation for this problem is that in pipelined architectures, the data flow goes through a sequence of modules implementing different tasks. In a linear pipeline, the output of one module m_k is the input of the next one m_{k+1}. However, it might be the case that for two modules m_a and m_b, knowing the output of one can improve the performance of the other and the other way around. Since we need to choose an order in the pipeline, if m_a will be executed first, the output of m_b will not be known at the time m_a runs.

In Chapter 4, specifically, Section 4.5, we outlined work that exploits a feedback loop on an information extraction pipeline in order to improve the task of coreference resolution using narrative information. Incorporating narrative information for different tasks and in different stages of our narrative information pipeline is one of our key contributions. In this chapter we expand on that work by generalizing our approach and expanding it to a second task: verb argument identification. In order to do so we propose a probabilistic framework that relies on two key ideas: 1) representing the output of each module in the pipeline as a probability distribution, and 2) defining a new type of module we call a *decision maker* that can aggregate the outputs of multiple modules.

6.1 Probabilistic Inference in Natural Language Processing and Information Extraction Pipelines

The goal of the work described in this section is to incorporate predictions generated by a later module m_b in the pipeline into the decision processes of an earlier module m_a without having to modify the internals of those modules.

We achieve this by introducing two key components in the pipeline:

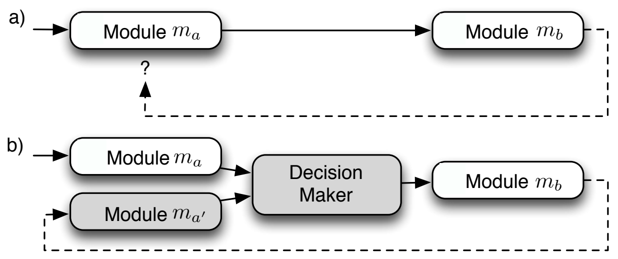


Figure 6.1: a) Illustrates a non-linear scenario where the output of module m_b is expected to inform the task of module m_a. b) Illustrates our proposed solution which solves this scenario by introducing a new auxiliary module $m_{a'}$ and a *decision maker*, without modifying modules m_a nor m_b.

- First, we introduce a new module $m_{a'}$, tasked with mapping the prediction of m_b into a probability distribution for the task addressed by m_a. Note that m_b can be a module that already operates in parallel with m_a or a module later in the pipeline.
- Second, we introduce a new *decision maker* module to aggregate the predictions of m_a and $m_{a'}$ and forward the aggregated output downstream.

The aggregation scheme described above is shown in Figure 6.1. In the figure m_b comes after m_a but that is not strictly necessary as we will demonstrate later.

6.1.1 Module Outputs to Probability Distributions

Our approach requires different modules in the pipeline to yield a probability distribution as their output, that is, consider a prediction task T, where the goal is to predict the value of certain variable Y, given some situation X, which takes a fixed set of values $\mathcal{Y} = \{y_1, ..., y_n\}$. In our framework, a *predictor* is a function $f_k(X)$ which generates a probability distribution p_k over the values $\mathcal{Y}$ can take. Since we do not want to modify the internals of existing modules, we instead propose adding an intermediate module ($m_{a'}$ in Figure 6.1) that will transform the original module's output into a probability distribution.

Given a predictor f_k for a variable X, which can take n values, and a training set $T = \{(x_1, y_1), ..., (x_m, y_m)\}$ (a collection of labeled instances), intuitively, we should be able to translate the prediction of f_k into a probability distribution p_k in the following way:

$$p_k(Y = y_i | f_k(X) = y_i) =$$

$$= \frac{|\{(x,y) \in T | y = y_i \wedge f_k(x) = y\}|}{|\{(x,y) \in T | f_k(x) = y\}|}$$

The intuition on that formula is that, given a predictor, we only know the value of the current prediction but since we know the set of values $\mathcal{X}$ and we have access to a training dataset, we can use the distribution in the dataset for the context we are interested in to draw a probability of the current prediction.

However, we found that this estimation method biases the probability distributions toward the raw distribution of labels in the training set, which in our case are very unbalanced. Thus, in order to correct for the unbalance, we applied a correction factor, $\frac{|T|}{|\{(x,y) \in T | y = y_i\}|}$, and then renormalized. This results in probability distributions that are more resilient to biased training sets. For example, when a module just predicts outputs at random, the correction results on probability distributions that approximate uniform distributions, rather than approximating the raw distribution of the data, which is preferable in our framework. We may need to calculate a probability distribution for a predictor of a certain variable X which takes a fixed set of values $\mathcal{X} = \{x_1, ..., x_n\}$. To do so, we can use the training set to compute a probability distribution p'_k of the values in the training set for each of the predictions in $\mathcal{X}$ by defining a mapping function where $p_{k_{x_i}} = p'_{k_{x_i}}$.

For example, let us consider the task of coreference resolution where we have a boolean variable that determines wether two mentions refer to the same entity or not. In this case we have two values in our training set (yes/no) and since most mentions will not be to the same entity we will have a skewed probability distribution like $\{0.2, 0.8\}$.

6.1.2 Aggregating Probability Distributions

Intuitively, we will model each module in the pipeline as a predictor for a given task. Now, consider we have a set $F = \{f_1, ..., f_n\}$ of predictors for the same task. Each prediction f_k generates a probability distribution p_k over the possible values of X. For example, in the task of coreference resolution we may have two predictors f_1 and f_2 using different approaches (e.g., matching nouns or narrative roles) that are trying to predict the value of a boolean variable X. Each of the predictors yields a probability distribution over the two values (yes/no) of the prediction task such as $\{f_1 \mapsto k_1 = \{0.5, 0.5\}, f_2 \mapsto k_2 = \{0.9, 0.1\}\}$ where we can see that the first predictor is not confident in it's prediction and the second one is very certain that the two mentions refer to the same entity. The first problem we need to solve is how to aggregate such probability distributions in order to obtain a single probability distribution over the possible values of X taking into account all the predictions. In order to solve this problems, we make the following assumptions:

1. For a variable X in the pipeline, we assume the existence of a set of modules implementing k predictors $F = \{f_1, ..., f_n\}$. The prediction of each predictor f_k, is encoded as a probability distribution p_k over the set of values $\mathcal{X}$ the variable X can take.

2. Assume the predictors in F are independent.

3. Any prior information on the predictors' reliability, is represented as a set of weights $w_1, ..., w_n$, where $w_k \geq 0$ for any k between 1 and n. These weights do not have a probabilistic interpretation, and it can be the case that $\sum_{k=1}^{n} w_k \neq 1$. If no information is available, uniform weights can be used.

4. Any prior information on the value of X is encoded as a prior distribution P_0 over the possible values the variable X can take. If no information is available, a uniform distribution can be used.

Given the previous assumptions, the predictions $p_1, ...p_n$ can be aggregated into a single probability distribution P_F by the Bordley's aggregation method[172]:

$$P_F = \frac{\prod_{k=1}^{n} (\frac{p_k}{P_0})^{w_k} P_0}{\prod_{k=1}^{n} (\frac{p_k}{P_0})^{w_k} P_0 + \prod_{k=1}^{n} (\frac{1-p_k}{1-P_0})^{w_k} (1 - P_0)}$$

The final value for X is determined by the aggregated probability distribution P_F as follows:

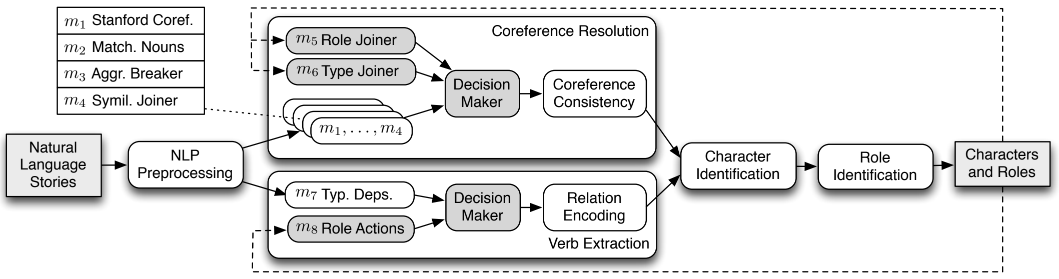


Figure 6.2: Architecture of *Voz*. Arrows describe information workflow and dashed arrows highlight the feedback loops.

$$x^* = argmax_{x_k \in \mathcal{X}} P_F(X = x_k)$$

In case of a tie, an ad-hoc tie breaker is used (the tie breakers used in our work are described below). With the aggregation scheme defined, predictions from any number of parallel modules working on the same task can be aggregated trivially. In the remaining of this section, we will use the term *decision maker* to refer to the additional modules we add to the pipeline to implement this probability distribution aggregation scheme.

6.2 Applying the Methodology to Introduce Feedback Loops in Voz

Let us now we discuss how to use this aggregation scheme in order to define non-linear pipelines. We show two use cases where we apply it to two tasks within *Voz*, our information extraction pipeline, specifically, we feed the output of the *Character Identification* and *Role Identification* tasks back to the *Coreference Resolution* and *Verb Identification* tasks. Let us now describe the modules used in this instance of our pipeline.

Coreference Resolution. As described earlier in this dissertation, the coreference resolution task consists of the grouping of mentions in E into a set of non-overlapping *coreference groups* where each coreference group is a subset of mentions that refers to the same referent (e.g., to the same character in the story). This is a generalization and extension of the work described in the previous section; specifically, *Voz* exploits two non-linear pipeline features:

- In the first run through the pipeline, *Voz* includes four parallel modules $m_1, \ldots, m_4$ that predict coreference. Their predictions are represented as a probability distributions and aggregated using a *decision maker*. The four parallel modules provide different information for the coreference resolution task: m_1 encodes the output of the *Stanford coreference resolution system*. m_2 links mentions with matching common and proper nouns. m_3 uses lexical and syntactic features to break groups with no gender or number agreement. m_4 links mentions with similar syntactic and semantic features. These first 4 modules were used in our previous work and are described in further detail in Section 4.5.

- In subsequent iterations, *Voz* incorporates a feedback loop. As shown in Figure 6.2, once the last two modules of the pipeline has been executed, their predictions are fed back and incorporated into the coreference resolution prediction as two additional auxiliary modules (providing a total of 6 modules aggregated for coreference resolution). After the first run of the pipeline, the final predictions are fed back and incorporated into the coreference resolution task as two additional auxiliary modules (providing a total of 6 modules to aggregate). These modules provide additional information to the decision maker: m_5 joins mentions with the same predicted roles, and m_6 joins mentions with the same predicted character types. These last 2 modules are also analogous to m_5 and m_6 described in further detail in Section 4.5.

The box *Coreference Resolution* in Figure 6.2 illustrates this setup. $m_1, \ldots, m_4$ are available before the feedback loop and m_5 and m_6 are available for subsequent iterations.

Each module m_k implements a predictor and encodes its output as an $|E| \times |E|$ matrix p_k of probability distributions, where $p_k(i, j)$ is a distribution representing the probability with which m_k considers mention e_i and e_j belong to the same coreference group or not.

The outputs of each module m_k are translated into probability distributions using the estimation described in the beginning of this section. Then, the matrices are aggregated cell by cell into a single $|E| \times |E|$ matrix P_F using a *decision maker*. The prior P_0, the set of weights, and the tie breaking mechanism are set in the following way:

- An uniform distribution is used for P_0.

- The reliability w_k of each module is computed from the module's performance on training data (F-score for identifying mentions that belong to the same group)
- If there is a tie in any of the cells in P_F (same probability for the two mentions to belong and not belong to the same group), then the mentions are considered not to belong to the same coreference group.

Matrix P_F is then translated to a $|E| \times |E|$ adjacency matrix G where $G(i,j) = 1$ indicates mentions e_i and e_j belong to the same coreference group when $P_F(i,j)$ is higher than the probability that they do not belong to the same group. $G(i,j) = 0$ otherwise. This matrix G is then made consistent by setting any cell $G(i,j) = 1$ if $G(j,i) = 1$ or if there is a mention e_k such that $G(i,k) = 1$ and $G(k,j) = 1$.

Verb Extraction. Specifically, we focus on a subset of the standard *semantic role labeling*[28] task, since we are only interested on identifying which are the mentions that are the subject and object of each verb.

This task uses a single module m_7 in the first run. The output of role prediction is fed as a second parallel module m_8 for subsequent iterations creating a feedback loop.

- m_7 first identifies a set of verbs $V = \{v_1, ..., v_n\}$ by their part-of-speech tags and uses the typed dependencies from the Stanford CoreNLP to extract their arguments. The dependency tree is explored and for each mention and verb in a given sentence, this module predicts whether the given mention is the subject, the object, both subject and object of the verb, or neither. The output is translated to a probability distribution over the four possible predictions (subject, object, subject and object, or none) using the training set with the same procedure described before.
- In subsequent iterations, *Voz* incorporates a feedback loop using an additional module m_8 that attempts to predict relations between mentions and verbs given the narrative roles of the mentions. For example, characters with the role "villain" are more likely to be the subject of verbs expressing violence. We use this methodology:

- Verbs are classified into a set of classes $\mathcal{A}$ (after exploring several alternatives, we used the set of *verb frames* in FrameNet[173] as our set of verb classes).
- Then, for each possible role (villain, protagonist, etc.), we compute which relation among subject, object, both or none is the most common for each verb class. For each mention/verb pair in a sentence, this module predicts this most likely relation.
- Finally, the predictions are translated to a probability distribution using the described procedure.

The box *Verb Extraction* in Figure 6.2 illustrates this setup. m_7 available before the feedback loop is shaded in white whereas m_8 available for subsequent iterations is shaded in gray.

Both modules in this task encode their output as a $|E| \times |V|$ matrix p_k of probability distributions where $p_k(i,j)$ is a probability distribution representing the prediction of module m_k for the relation between mention e_i and verb v_j. Moreover, notice that only those cells where the corresponding mention and verb appear in the same sentence have a value. In the first run there is no decision making process involved since there is only one module m_7 implementing the predictor for the task. In subsequent iterations, matrices are aggregated cell by cell to produce a single matrix P_F, and then predictions cast by selecting the value with highest probability in each cell.

The prior P_0, the set of weights, and the tie breaking mechanism are set in the following way:

- An uniform distribution is used for P_0.
- The reliability w_k is computed from the module's performance on the training dataset; specifically, we use the F-score for identifying verb arguments (subject and object).
- Should there be a tie in any of the cells of matrix P_F, preference is given to "subject and object".

The result of the aggregation process is an $|E| \times |V|$ matrix of mention/verb relations, this information is then encoded into the set of features representing each mention that are passed on to the next module of the pipeline.

Table 6.1: The first three rows report the performance of the *coreference resolution* task on three aggregation scenarios: 1) the first run through the pipeline ($m_1, \ldots, m_4$), the aggregation used in the iterative feedback loop (using $m_1, \ldots, m_6$), and 3) an aggregation where we replace m_5 and m_6 with the ground truth variants $m_{5_{GT}}$ and $m_{6_{GT}}$. The next eight rows report the individual performance of the parallel modules. The goal is to achieve $|Gr| = 258$ with $C/Gr = 1$ and $Gr/C = 1$.

	$\|Gr\|$	C/Gr	Gr/C	P	R	F	MUC	B^3	CEAF$_e$
First run before iteration	280	1.116	4.742	0.293	0.261	0.276	80.11%	57.34%	16.37%
Feedback loop	253	1.197	3.862	0.311	0.272	0.282	92.92%	38.63%	14.32%
Feeding ground truth	262	1.139	4.186	0.302	0.278	0.290	93.18%	53.84%	20.79%
m_1 Stanford coreferences	1091	1.061	5.433	0.758	0.103	0.181	79.9%	56.65%	39.7%
m_2 Matching nouns	1198	1.058	4.914	0.944	0.099	0.179	51.97%	42.02%	21.5%
m_3 Agreement breaker	1981	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
m_4 Similarity joiner	291	1.938	2.186	0.187	0.937	0.312	0%	22.61%	12.48%
m_5 Role joiner	188	2.25	1.63	0.261	0.399	0.316	93.5%	36.86%	9.27%
m_6 Type joiner	94	2.745	1.357	0.112	0.598	0.189	0%	22.61%	12.48%
$m_{5_{GT}}$ Ground truth for m_5	191	2.022	1.638	0.255	0.454	0.327	95.36%	82.94%	44.49%
$m_{6_{GT}}$ Ground truth for m_6	94	2.745	1.357	0.112	0.598	0.189	0%	22.61%	12.48%

The rest of the modules used in this work, specifically **Character Identification** and **Role Identification** exploit the work described in Sections 4.3 and 4.4 respectively.

We performed an empiric analysis of our approach with the purposes of determining 1) whether feedback-loops can improve performance with respect to linear pipelines, and 2) whether our aggregation approach is robust to feeding back noisy or irrelevant information.

The dataset used in the experiments in this section is described in Section 3.1. It was generated from 20 Russian folktales from which we removed quoted and direct speech (i.e., dialogues and passages where the narrator addressed the reader directly). The edited dataset contains 14879 words and annotations for 5131 verbs with 11321 argument relations (4105 of which involve individual characters), 7410 mentions (4356 of which are mentions to individual characters) and 1350 coreference groups (258 of them representing characters). Extracting information such as coreference from the stories in either dataset requires significant inference based on commonsense knowledge and contextual information.

All the reported results employ a leave-one-story-out methodology, where when testing on one story, the remaining stories are used as the training set.

6.2.1 Evaluation of the Coreference Resolution Feedback Loop

These experiments evaluate the performance of the feedback loop in the *coreference resolution* task. We compare the performance of three different scenarios: 1) *First run before iteration*: The first run through the pipeline (where only predictors $m_1, \ldots, m_4$ are available), without feeding back any information, 2) *Feedback loop*: feeding back the character and role predictions after one execution of the pipeline, and then executing two iterations of the feedback loop (where $m_1, \ldots, m_6$ are used), and finally 3) *Feeding ground truth*: we replace m_5 and m_6 with the ground truth $m_{5_{GT}}$ and $m_{6_{GT}}$ (this last scenario is for evaluation purposes only and represents the upper-bound on the performance that can be gained with the feedback loop). The first three rows in Table 6.1 report coreference resolution performance in these scenarios; role prediction accuracy is reported in the text.

To evaluate each scenario we report the number of character coreference groups $|Gr|$ (the closer to the actual number of characters in the ground truth, 258, the better), the average number of different characters in each group, C/Gr (the closer to 1 the better), and the number of groups the mentions of a character are spread across, Gr/C (the closer to 1 the better). In the table we report precision (P), recall (R) and F-value of the links identified between mentions in our adjacency matrix. For comparison, we also include the standard MUC, B^3 and $CEAF_e$ metrics computed using a reference implementation[174].

We can see how m_1 and m_2 alone fail to group all the characters leaving 1091 and 1198 groups respectively. On the other hand, we can see how m_4, m_5 and m_6 group too many characters in each group (C/Gr). m_3 does not represent a coreference graph as it is intended to break groups instead during the decision making process. It is an adjacency matrix where a 0 represent edges that needs to be explicitly removed (a coreference group broken) and 1 otherwise.

Concerning role prediction for characters, classification accuracy is 0.342, 0.394 and 0.354 for each of the three scenarios respectively, showing that the feedback loop does indeed improve the system performance. In the third scenario we expected the ground truth of roles and characters to give us an upper bound for the role identification task. Contrary to our expectations, we observe how the high recall of $m_{5_{GT}}$ creates bigger groups which in turn impact the performance of the system slightly lowering the role prediction accuracy.

Concerning the performance of coreference resolution, in the first scenario, the number of groups improves from 1091 when using only m_1 (Stanford) to 280. This is because the Stanford coreference resolution system has a very high precision, but low recall. By aggregating more information, precision lowered, but recall increased slightly. In the second scenario, the number of coreference groups improve even more, to 253, thanks to an even higher recall. In the third scenario, the number of coreference groups was 262, where we see some of the mis-groupings have been fixed but there are still not aligned with the 258 annotated characters. We conclude that feeding back role prediction information helps coreference resolution, thus confirming that the use of feedback loops can improve the performance of some NLP tasks.

6.2.2 Evaluation of the Verb Argument Extraction Feedback Loop

In this second set of experiments we evaluate the performance of the feedback loop in the *verb argument extraction* task. We experiment with three scenarios: 1) the first run through the pipeline (which is equivalent to running m_7 alone). 2) *Feedb. loop*: iterating two times using the feedback loop and 3) *Feeding GT.*: feeding back the ground truth for the roles (again, this scenario is for testing purposes only, to evaluate the potential gains that can be achieved by feeding back role information). The first three rows in Table 6.2 correspond to these scenarios. To evaluate each scenario and module we report the number of verb-argument relations identified and the precision (P), recall (R) and F-value of the relationships in the verb-argument matrix.

In this case, we can see no significant change in performance between feeding back role information and not feeding back information. This indicates that narrative role information is not very useful in identifying verb arguments. This can be seen by the low performance achieved when using m_8 (F measure of just 0.107 and only 483 relations found) as it only finds new relationships with mentions to characters. As a result, we observed that the probability distributions that our approach translates the output of m_8 to are very close to uniform distributions, and thus, they barely affect the decision making process when aggregated with m_7. This indicates that our approach is robust enough to handle irrelevant information fed back and not hinder the performance of tasks where other modules

Table 6.2: The first three rows report the performance of the *verb extraction* task on three aggregation scenarios: 1) the first run through the pipeline (m_7), 2) the aggregation used in the iterative feedback loop ($m_7 + m_8$) and 3) an aggregation where we replace m_8 with the ground truth variant $m_{8_{GT}}$. The next three rows report the individual performance of the modules. $|Rel| = 11321$ is the target.

	$\|Rel\|$	P	R	F
First run bef.	2737	0.353	0.506	0.416
Feedb. loop	2689	0.357	0.498	0.416
Feeding GT.	2548	0.359	0.490	0.414
m_7 Deps.	2737	0.353	0.506	0.416
m_8 Role act.	483	0.300	0.096	0.145
$m_{8_{GT}}$	505	0.301	0.102	0.152

provide accurate predictions.

6.3 Discussion

In this chapter we presented a probabilistic framework that allows the addition of non-linear workflows on top of traditional pipeline architectures typically used in information extraction tasks. Our framework is based on representing the output of each module of the pipeline as a probability distribution and then a *decision maker* module capable of aggregating such distributions. Our results confirm that some feedback loops can improve the performance of certain tasks, such as coreference resolution. Additionally, our results for verb argument extraction show that the aggregation mechanism is robust enough to handle situations where the information that is fed back does not contain relevant information for the task at hand (e.g., narrative roles, which do not contribute to verb argument extraction) and not hinder the performance of the task. One practical aspect of the proposed approach is that it does not require internal changes on existing modules, even for those which do not naturally represent their output as probability distributions.

As part of our future work we would like to explore adding additional feedback loops, both reusing the same narrative information described in this chapter (e.g., characters and their roles) for other NLP and IE tasks (e.g., word sense disambiguation for verbs involving character's actions) and other high-level narrative information (e.g., narrative functions) for the current tasks in our pipeline (e.g., using narrative function information for verb argument identification).

Chapter 7: Bridging the Gap between Narrative Information Extraction and Story Generation

The long term goal of our work is to enable the automatic extraction of narrative information from natural language text and use the extracted information for storytelling and story generation applications. Traditionally, the body of research around storytelling and story generation applications has relied on handcrafted models in some knowledge representation formalism and/or hand annotated text templates. These models and annotations tend to use complex specifications and need to be authored by knowledgeable technical people making their development tedious and expensive. We want to bridge work from narrative information extraction with that in story generation so that the structured narrative information extracted by the first can be used as input to the other.

In this chapter we present our approach for bridging the gap between work on narrative information extraction and work on story generation in order to implement end-to-end computational narrative systems. Then we present our work on using the narrative information extracted by *Voz*, our automated narrative information extraction pipeline for an existing story generation system: *Riu*[3;4]. In order to do that we describe the steps necessary to map the structured narrative information output from *Voz* into the input required by *Riu*. This enables us to implement a full-fledged end-to-end, text-based, story generation system. Finally we report our analytical findings and the results of a user study in which we intended to evaluate the human perceived quality of the generated stories given the system's shortcomings described in the previous chapter.

7.1 End-to-end Computational Narrative Systems

Earlier in this dissertation, in Sections 2.4 and 2.6, we outlined two of the main lines of research within computational narrative: 1) storytelling and story generation applications; and 2) analyzing or model existing narratives in order to study existing literature or validate narrative theories. Despite both being actively developed, there are very few instances of work across both areas of research. In Section 2.5 we surveyed the different computational models used within different approaches. Figure

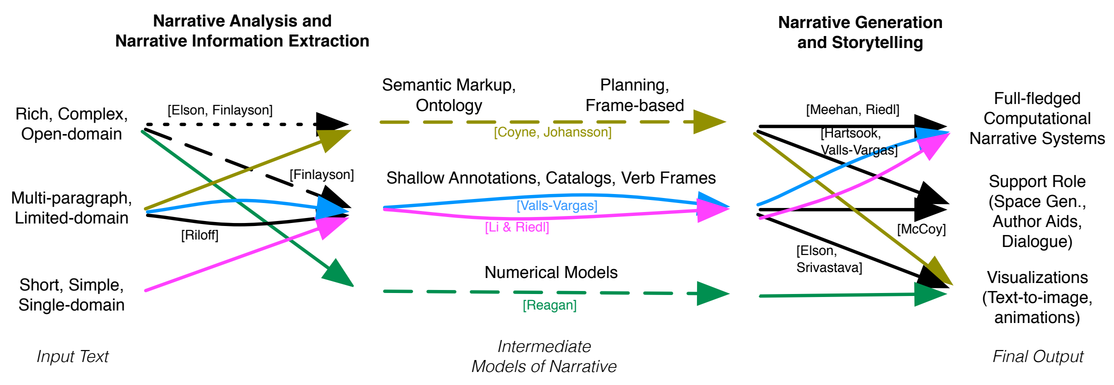

Narrative Analysis and Narrative Information Extraction
Narrative Generation and Storytelling
Rich, Complex, Open-domain
Multi-paragraph, Limited-domain
Short, Simple, Single-domain
Input Text
[Elson, Finlayson]
[Finlayson]
[Riloff]
Semantic Markup, Ontology
Planning, Frame-based
[Coyne, Johansson]
Shallow Annotations, Catalogs, Verb Frames
[Valls-Vargas]
[Li & Riedl]
Numerical Models
[Reagan]
Intermediate Models of Narrative
[Meehan, Riedl]
[Hartsook, Valls-Vargas]
[McCoy]
[Elson, Srivastava]
Full-fledged Computational Narrative Systems
Support Role (Space Gen., Author Aids, Dialogue)
Visualizations (Text-to-image, animations)
Final Output

Figure 7.1: Overview of input and output in computational narrative applications in the main two areas of research within computational narrative. Black arrows indicate existing approaches (solid arrows indicate automated approaches, dashed arrows indicate manual and semi-automatic approaches). The pink arrow shows the related work by Li et al.[10], and the blue arrow indicate our work for text-based end-to-end computational narrative systems as described in Section 7.2; these are highlighted since they are end-to-end approaches combining both areas of work.

7.1 provides a brief visual summary of this work. The left-hand side of the figure represents work on narrative analysis and the right-hand side represents work on narrative generation. The four vertical axes represent the complexity of the input and output of these systems. The center of the figure contains a set of computational models of narrative employed by the different pieces of work shown in the figure and the existing or potential mappings between them that we discuss in the next section.

7.1.1 Mapping Computational Narrative Models

Despite the lack of consensus on the most appropriate models to encode narratives or represent different features (or even human narrative mental models)[75], we observed some commonalities between the underlying models used in these different areas of computational narrative (those used in narrative generation and those used in narrative analysis described in the previous section), which could lead to the construction of end-to-end systems (where by "end-to-end" systems, we mean computational narrative systems that extract narrative models directly from text and use those to generate new narratives) that use narrative analysis techniques to generate the models required by narrative generation techniques.

For example, mappings between models based on shallow annotations has been explored in specific support modules within computational narrative applications (such as text generation using text modification[3;4;175]) but do not truly enable the creation of end-to-end systems. We can propose a few other unexplored intuitive mappings, that is, we can establish a mapping between the output models acquired by information extraction systems (shown on the left in Figure 7.1) and the models required as input by story generation and storytelling systems (shown the second from the left in Figure 7.1), instead of having to hand-author them. For example, we could use other algorithms to extract specialized models such as: social networks and character relationship information[19;128] for a social simulation[125] or character-centric multi-agent story generation system[104;176]; or use a sequence of events and extracted location information to generate virtual environments for a story to happen[7;90]; or; use automatically annotated text as templates for natural language generation systems.

The most coveted mapping would be between automatically extracted rich semantic annotations or even symbolic models of a narrative to planning/frame-based models. Since these planning and frame-based models are often manually crafted, the required features for storytelling or story generation applications are manually crafted as well. There are, however, a few examples of work that attempt to build end-to-end systems. An example is the work of McIntyre et al.[114]. They attempt to automatically extract semantic frame knowledge from a corpus of stories, sample their knowledge base, assemble a frame-based story representation and use a text realization component to generate text. They then use a generate-and-and-rank iterative process with an interest and a coherence model to refine the generated output. More recent work by Li et al.[10] relies on a set of crowd-sourced short reports describing a given theme (e.g., bank robbery). They extract Shankian script-like structures[15] from each and assemble them into a *plot graph* that joins common events. These graph-like structures are related to a set of plot points or planning operators used to describe a story space. Despite not being full-fledged symbolic representation nor completely eliminating human intervention, this approach could exploit existing text reports.

Other examples include text-to-image systems that extract a semantic representation from short, simple and limited domain text. Similar than the previous work, there is a mapping step that compiles and links the different extracted representation components (e.g., the catalog of participants and the sequence of events) with commonsense and domain-specific databases. Then, instead of handing this internal model to a storytelling system, it is used to generate images and/or animations for the final user[9;58]. We identified some limited work related to interactive learning systems and digital entertainment with similar characteristics[104;176].

In our survey, we have seen some research efforts in the direction towards end-to-end systems. Currently, these seem to be limited in terms of the kind of text they can process (i.e. only short simple text), their scope (i.e. only domain-specific applications) and/or their output (can only render a limited inventory of objects or generate a small specific set of stories)[9;10;58]. Another prominent class of examples implementing end-to-end computational narrative systems includes de statistical approaches described in Section 2.5.3 which include Markov models and neural networks. These approaches and models can be used to implement end-to-end systems that connect narrative analysis and generation tasks, specially in text-based systems. Some common downsides of these methods are the often large training data requirements[68] and the fact that there are still open research challenges in keeping character, word, and sentence level neural networks coherent for story generation[69]. Moreover, the lack of explainability and the fact that the learned models are often difficult (if possible at all) to tweak by human authors is in conflict with the often desired property of authorial control over the output[177].

7.2 Connecting Voz and Riu

Towards our goal of developing a prototype of an end-to-end story generation system that can take their input in natural language with minimal human intervention; given the findings presented in the previous section, in this section we present our work on mapping models that can be automatically extracted using narrative analysis work to those models used in narrative generation work.

Specifically, we worked on an end-to-end system that combines *Voz*, our automated narrative information extraction pipeline focused on identifying narrative elements from a set of Russian folk tales, with *Riu*, an analogy-based story generation system[102]. The end goal is to create an analogy-based narrative generation system that works directly from natural language, without requiring the human user/author to craft any specialized narrative model. Thus, the human user should be able to provide a partially written story in natural language, and the system would automatically complete it. The background knowledge required by the system is provided in the form of narrative domain-knowledge, specifically, a formal representation of a subset of Proppian narrative theory that describes the structure of character roles and narrative functions, and a case of annotated examples for classification. Additionally, the system takes as input a corpus of natural language stories, which are automatically analyzed in order to populate the content for the internal computational models of narrative used.

We selected these two systems because compatibility between the knowledge structures that *Riu* requires and the structures that *Voz* can extract from text. Specifically, *Voz* extracts a catalog of entities and uses verb argument information to extract a graph-like structure. *Voz* also identifies high-level narrative information from the text such as the narrative roles different characters fulfill and narrative structure information, specifically Propp's narrative functions[25]. All this information is used in the analogy and mapping modules within *Riu*. The final system uses the extracted information for then generating natural language and implements a fully realized story generation system that utilizes a given corpus of stories for story analogy and natural language generation.

7.2.1 Riu

Riu is an interactive story generation system that uses the *Story Analogies through Mapping* (SAM) algorithm[102]. Analogy-based story generation emulates the human cognitive process of analogy-making by identifying similarities and a mapping between two domains. This general idea is that if two domains are similar in a certain way, they are likely to be similar in another and we should be able to find a mapping which we can exploit to transfer knowledge from one another.

Specifically, *Riu* operates over a repository of complete stories. The story generation process starts with a given story fragment as input *target story*. Riu then identifies which of the complete stories in the repository has a stronger analogy with the provided story fragment (*the source story*). *Riu* then calculates an analogical mapping between the target and the source stories, and completes

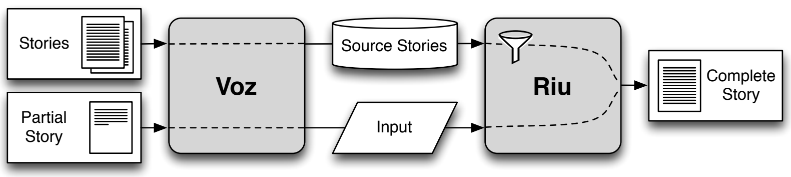


Figure 7.2: Overview of our end-to-end story generation system. The knowledge structures required by *Riu* are generated automatically by *Voz* from natural language.

the target story by analogy with the source. *Riu* requires all the input stories to be encoded using a frame-based symbolic knowledge representation formalism. In all the existing literature on the *Riu* system, each of the stories in the repository and the given story fragment have been manually authored. In this work, we will study the performance of *Riu* when the representation of such stories is automatically generated using the *Voz* system, as illustrated in Figure 7.2.

Riu's Story Representation

Riu uses the *Story Analogies through Mapping* (SAM) algorithm which internally uses the *Structure Mapping Engine* (SME) algorithm described by Falkenhainer et al.[94]. Thus, given SEM is a symbolic analogical mapping algorithm, it requires a symbolic representation of the stories.

Riu represents stories as *scenes* and *phases*. A *scene* is a small encapsulated segment of a story involving a limited set of characters, actions and locations. Furthermore, each *scene* is broken into several *phases* (which represent specific story states the relations between the characters and props in the story at that particular state, and the actions the characters are performing). For the purposes of this paper, we will see a story as a scene which is in turn represented as a sequence of phases (more complex stories are represented in *Riu* as collections of scenes, but we will focus on single-scene stories in this paper). Each of these phases contains two representations: a *Computer-Understandable Description* (CUD) and a *Human-Understandable Description* (HUD). The CUD is a symbolic, frame-based representation of the phase that includes the different entities present in the phase and a graph-like structure that defines links between them using *expressions*. The original authors of *Riu* reported experiments using different representations in the CUD and have shown that the representation formalism has a substantial impact on computational analogy[4].

On the other hand, the HUD is a collection of annotated natural language phrases and sentences. The CUD and HUD are linked, so that during the analogical reasoning process, SAM can manipulate the CUD and use the HUD to realize the final text of the story. An example of such representation is shown in Figure 7.3. For more detail on these representations, we refer the reader to the previously published work on *Riu*[102].

Analogy-based Story Generation

Given an incomplete input target story, *Riu* compares it with the stories in its repository and selects the most similar story (the source story). In order to evaluate similarity, *Riu* uses both the *Computer-Understandable Description* (CUD) and the *Human-Understandable Description* (HUD) to compute structural and surface similarities[4]. Then *Riu* invokes the SAM algorithm for finding an analogical mapping between source and target stories and generating the output story by completing the partial target story[1].

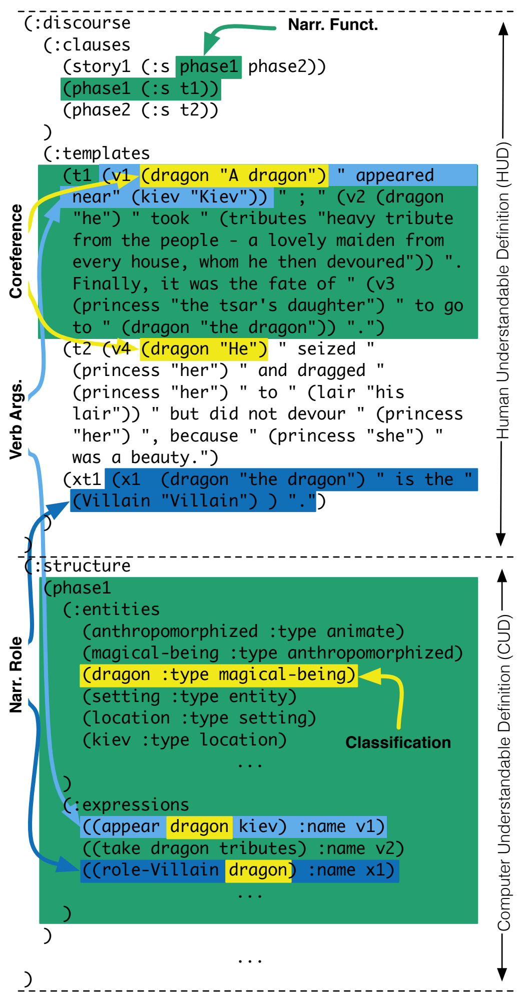
SAM takes two input parameters: a source story S from the story repository and a target T in place from the given story segment. Note that both S and T are story representations encoded as a sequence of phases, each with a CUD and HUD. It generates a set of all possible consistent injective mappings M between S and T. Then, SAM finds a mapping $m* \in M$ that maximizes a numerical similarity between the entities and relationships defined in the CUD for the phases in S and T using the mapping $m*$. With this mapping $m*$, SAM can construct a new story R by applying the mapping $m*$ to the phases of the source S, and then bringing them to T. For each element in the CUD that is brought from S to T, the corresponding elements from the HUD are also brought to T (applying the appropriate transformations given the analogical mapping) in order to realize the output in natural language.

Mapping the Output of Voz to Riu's Input

In this section we describe how we mapped the output of *Voz* to the dual representations used by *Riu*.

There are three distinctive parts to consider: segmentation (breaking a story in phases), entities

[1]Available online: https://sites.google.com/site/santiagoontanonvillar/software


Narr. Funct.
Coreference
Verb Args.
Narr. Role
Classification
Human Understandable Definition (HUD)
Computer Understandable Definition (CUD)

```
(:discourse
  (:clauses
    (story1 (:s phase1 phase2))
    (phase1 (:s t1))
    (phase2 (:s t2))
  )
  (:templates
    (t1 (v1 (dragon "A dragon") " appeared
     near" (kiev "Kiev")) " ; " (v2 (dragon
     "he") " took " (tributes "heavy tribute
     from the people - a lovely maiden from
     every house, whom he then devoured")) ".
     Finally, it was the fate of " (v3
     (princess "the tsar's daughter") " to go
     to " (dragon "the dragon")) ".")
    (t2 (v4 (dragon "He") " seized "
     (princess "her") " and dragged "
     (princess "her") " to " (lair "his
     lair")) " but did not devour " (princess
     "her") ", because " (princess "she") "
     was a beauty.")
    (xt1 (x1  (dragon "the dragon") " is the "
     (Villain "Villain") ) ".")
    )
)
(:structure
  (phase1
    (:entities
      (anthropomorphized :type animate)
      (magical-being :type anthropomorphized)
      (dragon :type magical-being)
      (setting :type entity)
      (location :type setting)
      (kiev :type location)
                  ...
    )
    (:expressions
      ((appear dragon kiev) :name v1)
      ((take dragon tributes) :name v2)
      ((role-Villain dragon) :name x1)
                  ...
    )
  )
                  ...
)
```

Figure 7.3: Example input used by *Riu*. Bold labels identify information from *Voz*. Different colors illustrate linked parts.

(from the text's mentions) and expressions (that link entities and provide structure for the SME algorithm to use). We generate these by automatically transforming and annotating the output of *Voz* described in the previous chapters.

Segmentation. *Riu* uses two levels of segmentation; a story is divided in scenes and each scene is divided in phases. Given the significant amount of subjectivity involved in determining the scenes and phases of a story, we used the notion of *narrative functions* as defined by Propp[25] for this purpose. Thus, we use these functions to segment each story in 5 phases: phase 1) the introductory and setup functions (alpha – lambda in Propp's work); phase 2) villainy/misfortune, mediation, counteraction and departure (A/a, B, C, ↑ in Propp's work);

phase 3) the main development of the story (including functions D – K); phase 4) return (including ↓, Pr, Rs, o, L – T, Ex); and phase 5) conclusions (including U, W and any narrator's remarks after that). This segmentation will be used for both the CUD and the HUD representations. The first two phases make up the introductory scene and in the experiments reported in this paper we use only these first two phases mainly because of performance concerns. Note that currently, *Voz* can only identify narrative functions in stories that have been previously segmented into chunks of text by hand (since it basically predicts which function is expressed in each chunk of text). In our experimental evaluation we report results with the functions automatically identified by *Voz* and with manual annotations but both use the previous manual segmentation of the text in chunks. The overall accuracy for predicting narrative function in *Voz* is 0.287 but the accuracy for the first and second phases used to segment our experiments is 0.534 and 0.487 respectively.

Entities. One of the tasks *Voz*'s development has focused on is the extraction of *referring expressions* or *mentions* from the text and their coreference information (a.k.a. coreference resolution) which groups mentions into unique entities[132]. Furthermore, the SME algorithm uses a taxonomy for the entities in the CUD for which we use the entity classification labels *Voz* provides. *Voz* classifies each entity into different classes using a taxonomy inspired on Chatman's existents[72] in order to provide a deeper structure for the SME. These include existents (e.g., characters: male, magical being, etc.; settings: locations and time anchors) and happenings (e.g., events such as rain). This information is linked in the HUD where the original text for each mention is used and annotated with their coreference group information. *Voz* has perfect recall of mentions but some false positives yield an accuracy of 0.954. The accuracy for classifying into the 15 classes of our taxonomy is 0.535. In terms of coreference, on our corpus of Russian fairytales, *Voz* achieves the following performance measured using 3 common metrics; F1 score for MUC: 0.932; CEAF$_e$: 0.208; B^3: 0.538.

Expressions. The SME algorithm within *Riu* favors deep, structural similarity during the analogy process. Besides the aforementioned taxonomy, this structure is represented in the CUD using logical predicates. Specifically, we use:

1) *Verb and verb argument information* encoded as triplets representing an interaction between two entities, the subject or actor and the object or receiver. This results in an expression for each verb in the CUD and an annotation over the span of text where the verb and mentions appear in the HUD. Note that we define a *null* entity placeholder for intransitive verbs where there is no object. Additionally, to provide a closed language for the SME, we automatically abstract the verbs and group them in 191 semantic classes based on the Levin verb classification[178] used for the expression's predicates. *Voz* has an accuracy of 0.807 for verb identification, 0.229 when considering the verb and arguments.

2) *Narrative role information* identified for the extracted characters is used to add additional expressions in the CUD for characters (a subset of the entities). For the HUD we add the template "{character} is the {narrative role}" so it can be used by SAM if necessary during text realization. The accuracy of *Voz* for identifying characters is 0.931 and then 0.394 for identifying their narrative roles; but that accuracy is 0.540 and 0.622 for the first two phases.

3) *Narrative function information* is used to add a layer of structure on top of verb expressions that span complete sentences within a phase. For each narrative function within a phase; and for each sentence within the span of the function from which a verb expression has been extracted; an expression is added in the CUD linking the narrative function and the expression of the root verb of the sentence. The narrative function is also annotated in the HUD for each sentence.

Ontology. When creating the expressions for the CUD, symbols from a closed, known ontology are used for the classes of the entities, the narrative roles and the narrative functions. The verbs, on the other hand, are treated as an open domain. In order to provide a language that the SME can handle, we abstract the verbs and group them in 191 semantic classes based on the Levin verb classification[178]. This is used in the CUD and the original text representation for the verbs is used in the HUD.

Figure 7.3 shows an example story representation that highlights the aforementioned parts, the links between the CUD and the HUD, and illustrates how these map to the output from *Voz*. Note that for clarity, this example uses human-readable symbols such as *dragon* and *take* whereas *Voz* uses equivalent symbols such as *E1* or *Levin-02-1*.

7.3 Evaluation of Riu's Output

In this section we evaluate the feasibility and output of the end-to-end, text-based, story generation system implemented by connecting *Voz* to *Riu*. Through this experimental evaluation we seek to answer the following questions:

1. What is the quality of the generated stories?
2. How is the quality of the generated stories affected by the knowledge structures automatically generated by *Voz* with respect to using the ground truth annotations on the stories to generate these structures?

In the next section we address these questions by running the system and using an analytical evaluation on the output. Then we present the results of a user study to provide further insights on the aforementioned questions.

7.3.1 Analytical Evaluation

For this experimental evaluation we use 20 Russian folk tales translated to English (from the corpus described in 3.1). We have annotations for each of the 20 stories and each of the tasks in the information extraction pipeline[141]. In the following experimental section we report the results in two scenarios: 1) generating *Riu*'s story representation formalism using our complete automated narrative information extraction pipeline (*Voz*), and 2) instead of using *Voz*, generating *Riu*'s knowledge structures directly from the ground truth annotations on the text (*GT*).

In order to answer the first question, using the output of *Voz*, we are specially interested in whether the current performance of *Voz* suffices to generate *Riu*'s required input. We are aware that whereas for some tasks performed by *Voz* have a high degree of accuracy (e.g., the binary classification task between character and non-character mentions), some other tasks have a much lower accuracy (e.g., verb argument identification) but in our previous work described in Chapter 5, we observed some error in narrative information extraction pipelines can be mitigated downstream.

We used a *leave-one-story-out* evaluation procedure, where we provided the first phase of one story to *Riu* as the target story, and the remaining 19 stories as the repository of stories to use as source stories. The expected output of *Riu* is to complete the target story with a second phase that continues the given input story.

Considering the two scenarios described in the previous section (*Voz* and *GT*), we launched a total of 40 experiments (one per story and scenario). Out of the 40, 22 completed successfully within a few minutes, 7 exhibited errors that prevented Riu from being executed and 11 timed out within the time allocated (48 hours). Inspecting the experiments with errors, we observed that 2 stories had segmentation issues that yield a phase without contents and therefore, 4 experiments cannot be completed in either the fully automated (*Voz*) scenario or the annotated (*GT*) scenario. Additionally, there are 3 experiments that cannot be completed because the automated scenario is unable to extract any expressions for the initial phase (that is, there are no roles identified nor verbs with arguments). Inspecting the input files (including both the human and the computer readable definition for the storie) for the experiments that didn't finish, we found that in the *GT* scenario, the segmentation was causing some stories to have unexpectedly longer phases with a great number of entities and expressions. Since SME uses a systematic search approach for computing the analogy mappings, these phases cause a combinatorial explosion that cannot be properly handled. In the *Voz* scenario we observed a similar problem but additionally, because of some coreference errors, a large number of different entities are present in the output. As part of our future work we plan to modify *Riu* to use a more efficient computational analogy algorithm.

We manually inspected the output stories of the 22 experiments that completed successfully. In this section we report the major trends we observed. Note, minor spacing and punctuation corrections were made to the output reproduced in this section and square brackets and ellipsis are used to provide interpretation remarks and shorten some story fragments. In the following examples, the span in *italics* corresponds to *Riu*'s generated continuation of the given introduction.

Successful Output. We found several of the experimental runs where *Riu* found a plausible mapping and generated output that could be considered successful. With some considerations mentioned in the following sections, we consider 6 out of 10 stories in the *Voz* scenario and 6 out of 12 stories in the *GT* scenario to have a plausible output using the top ranked mapping or any of the 3 top ranked mappings when tied. 5 out of these 6 stories are the same and share characteristics that make them suitable for being combined with other stories (i.e., they have a generic introduction). This is one example (from the *Voz* scenario):

A dragon appeared near Kiev; he took heavy tribute from the people - a lovely maiden from every house, whom he then devoured. Finally, it was the fate of the tsar's daughter to go to the dragon. *He [the king] just didn't know; there was no one suitable for the job the job. Then he remembered the seven Simeons [seven brothers who offered their services to the king]. Summoning them, he commissioned them as soldiers, with the assignment to bring him that princess.*

Natural Language Generation Issues. Currently, the natural language generation (NLG) component of *Riu* relies on SAM's mapping and performs the identified replacements directly on the *Human-Understandable Description* (HUD). These replacements lack the variety and use of pronouns found in the original text (e.g., Ivan, he, me). Consider this original text fragment:

Well, Ivan undertook to kill that Snake, so he said to his father, "Father make me a mace five poods in weight."

The following fragment from the *Voz* scenario illustrates the aforementioned problem:

Well, A fox undertook to kill that Snake, so A fox said to his his father, "Father make A fox a mace five poods in weight."

Notice a repeated “his his”. This is an error caused by the mention identification not properly capturing the full span of the mention and replacing “father” with “his father”.

Nonsensical Output. Some of the original stories use strange rhetoric figures and constructs that may be surprising for a western audience not familiar with Russian and Slavic folktales. Still, in the output of the system we found several instances that do not have continuity and with nonsensical mappings, despite these mappings exhibiting a high score by the SME. Inspecting these mappings we found that besides errors in the narrative information extraction the lack of depth of our *Computer-Understandable Description* (CUD) structure was assigning analogies between stories that a human would not consider.

An old man lived with his old wife; they had a daughter and a little son. “Daughter, daughter,” said the mother, “we are going to work. [...] Be careful , watch over your your little brother, do not leave the house.” The parents went away and the daughter forgot what they had told her; she put her brother on the grass beneath the window, ran out into the street, and became absorbed in games. *The prince flew straight into the royal palace, turned into a goodly youth, and asked the palace guards: “Will your king take me into his service?” “Why should he not take such a goodly youth?” they answered. Thus he entered the service of that king and lived in his palace for one week, then a second, then a third . [...]*

Scoring, Ranking and Bias Issues. Sometimes the input given to *Riu*, that is, the first phase in some stories, is either too short or *Voz* is not able to extract sufficient information to compile a rich CUD. This situation cascades into problems retrieving candidate sources for analogy and the analogical reasoning itself. For example, consider the following two continuations from the *GT* scenario that got tied scores for one target partial story.

This soldier got permission to go on leave. *The tsar’s word was law. This soldier was banished and This soldier left home not knowing where to go. This soldier walked for a long time; at last This soldier arrived in another kingdom, presented This soldier to the king, and asked to be taken into his service.*

This soldier got permission to go on leave. *Well, This soldier undertook to kill that Snake, so This soldier said to his his father, "Father make This soldier a mace five poods in weight".*

We observe this problem in the same 5 stories in both the *Voz* and *GT* scenarios. On the other hand, some stories in the database feature longer/richer structures which bias the algorithm towards using them more often and scoring them higher. For example, in the following case, the dragon (the villain) is intended to be "punished", so *Riu* identifies the act of trying to cook the dragon on the stove as a punishment:

A dragon appeared near Kiev; he took heavy tribute from the people - a lovely maiden from every house, whom he then devoured. Finally, it was the fate of the tsar's daughter to go to the dragon. *So Alenka heated the stove hot, ever so hot, and said to A dragon, "Come here and sit on this shovel!"*

In this second case, Prince Ivan or Ivanshko, in the original story, is being (unfairly) punished for his action and again, the same continuation is used despite there being more plausible punishments available:

For many years a certain tsar had kept under lock and key a little peasant [who was a cunning wizard ...] Prince Ivan, the tsar's son who was still a little boy [... set the prisoner free by accident ...] *So Alenka heated the stove hot, ever so hot, and said to Ivashko, "Come here and sit on this shovel!".*

This Alenka continuation is used (in some instances tied) in 6 stories in the *GT* scenario and 4 in the *Voz* scenario.

So far, we showed examples from the two scenarios evaluated in our experiments: executing *Riu* with input automatically generated with *Voz*, and executing *Riu* with input generated from the ground truth annotations in the stories. When generating short story snippets (as done in our experiments), we do not observe relevant differences between these two scenarios. We attribute this to the fact that the structures in the CUD are quite similar in terms of depth (and contain the same kinds of expressions). However, with longer examples or trying to generate full stories (not reported in this section), several shortcomings arise, specially when coreference errors accumulate and the size of the CUD increase making the analogy process take longer and fail to find mappings. Additionally, we identified several issues, specifically:

Coreference. In the results reported in this section, the continuation or second phase is usually short enough that the coreference graph is relatively small and the substitutions made by *Riu* make sense. In some longer instances, although none selected in the final output, we noticed that missing links in the coreference graph cause *Riu* to miss replacements. This was more evident if we try completing introductory segments using full stories (all 5 phases).

Segmentation. Errors in the function identification task used for segmentation cause phases to lack key information or include out-of-place information that belongs in other phases. Moreover, 2 of the stories in the dataset used for these experiments do not seem to conform to the expected Proppian structure (and hence they have some empty phases).

Shallow Structure. A recurrent problem we observe is due to the lack of structure in the CUD. Despite often due to errors in *Voz*, the choice of the mapping yields shallow structures in the CUD even when using the annotated stories. As mentioned earlier, the representation formalism has a substantial impact on computational analogy[4]. Adding additional layers of annotation (e.g., force dynamics) that could be extracted by a narrative information extraction system should have a positive effect on the analogical reasoning process.

In the next section we present the results of a user study to evaluate the *Voz*+*Riu* pipeline.

7.3.2 Empirical Evaluation on Perceived Quality

The evaluation described in the previous section motivated us to further inquire into the quality of the generated stories. In this followup we first want to isolate our biases due to our knowledge of the stories and our subjective interpretation of the results. Additionally, we are interested in the perceived quality of the generated stories by humans. This dataset is larger than the annotated corpus used in our previous experiments which may alleviate some of the issues identified in the previous experiments as there are more candidates for retrieval and the structures between stories are more uniform in terms of narrative functions and size. Additionally, the stories in this dataset are simpler in terms of natural language features which simplifies some of the core NLP tasks. To evaluate the current results, we decided to conduct a user study where we asked human subjects to score the stories and get a grasp of the overall perceived quality of the stories beyond the individual issues that we identified analytically. Specifically, we designed a study to address the following two questions: 1) How does the perceived quality of automatically generated stories compare against human authored stories? 2) Can we quantify the impact and identify the relationship between errors introduced by automatic narrative information extraction tasks and the perceived quality of the generated stories?

For the study we used the synthetic dataset described in Section 3.2. Besides the different dataset, there are no changes on how *Voz* and *Riu* are executed and the ontology used within *Riu* to guide the analogy mapping process is the same.

The stories are in this dataset are annotated with segmentation information and in the experiments reported in this section we also use the first two segments in the same manner as we did in the previous section. For each of the 100 short stories, the first two segments are processed automatically by *Voz*, then, each one story is held out of the sources set which will contain the first and second segments for the remaining 99 stories. The first segment of the held out story is therefore a *partial story* and is used as the target or query for *Riu*'s analogy mapping. After holding out each one of the 100 stories, the output is a set of 100 story *continuations* from each of the initial segments used as a query. Each of these *continuations* follows the initial segment or *partial story* used for the query and the pair is presented to the participants as a *complete story*.

In order to better isolate the potential sources of error and provide a more thorough evaluation of the generated stories by our pipeline, we considered 4 scenarios:

Voz+Riu: In this scenario, the data structures required by *Riu* are directly generated from the annotations in the dataset (i.e., instead of running *Voz*, we use the ground truth). The goal of this scenario is to measure the difference in quality in the generated stories due to potential errors introduced by *Voz* when extracting the narrative information.

LSTM: In order to have a baseline to compare our method against, we used an off-the-shelf implementation of a long short-term memory (LSTM) recurrent neural network to generate text for story continuations in a similar fashion as the previous two scenarios. We decided to use a neural-network method as our baseline given their recent rise in computational narrative [70;143] and their success in other areas of NLP and text generation [62;68]. In this scenario, we use the 99 stories in the source set to train the LSTM and then we use the held out story initial segment as a seed for the LSTM and we generate a continuation. The cutoff for the continuations is two sentences or 150 words (whatever comes first). For our LSTM we use the implementation from Tensorflow. In our experimentation the LSTM has 2 layers and 650 units per layer. Before processing, the text is normalized and we use word-level tokens. Despite this preprocessing, the dataset is still small for this approach and the sentences generated are often grammatical but difficult to parse and mostly non-sensical. See Appendix B for examples.

Original: Also included in the questionnaire are the original continuations for the stories in order to provide an upper-bound for the evaluation. This scenario also provides the study participants some contextual information about the particularities of the Russian-like stories used in the dataset.

The user study was conducted through an online questionnaire publicly posted on the Internet. The study was expected to be completed in 30 minutes. We recruited individuals using online messaging and social media to participate in the study. There was no compensation nor penalty for choosing to participate or not in the study. Participants were required to be 18 years of age or older, be able to read and understand English, and provide consent.

For our questionnaire, we first included a few demographic questions, a statement of consent and a reading comprehension test. This was intended to discriminate automated submissions and ineligible participants. Specifically we include the following questions:

- What is your age? (Numerical field).
- What is your highest level of education? (Categorical with 5 options)
- Are you currently a student at Drexel University? (Yes/No).
- Are you a native English speaker? (Yes/No).
- Are you fully competent in written English? (Yes/No).

The questionnaire then displays 24 sections corresponding to 6 *partial stories* and 4 *continuations* for each, corresponding to the 4 scenarios described above. We manually inspected both the input files used by *Riu* (including both the human and the computer readable definition) and the output text and selected a subset of 9 of the 100 generated stories that exhibit interesting properties related to the specific questions we are investigating in terms of errors from *Voz*, including at least 2 relevant examples for each of the 4 (described below). These 9 stories and continuations are included in Appendix B. Of the 9 selected stories, one was used for control. This control story was displayed in the first place to each participant and always showed the same continuations displayed in the same order (*Original, Annotations+Riu, Voz+Riu, LSTM*). We do not consider the data for this first story in the reported results. After this control story, each subject saw 5 *complete stories* drawn at random from the pool of 8 stories we selected and the continuations for those were presented in a random order. In our questionnaire we asked the following questions regarding each pair of *partial story* used as the query and the generated *continuation*:

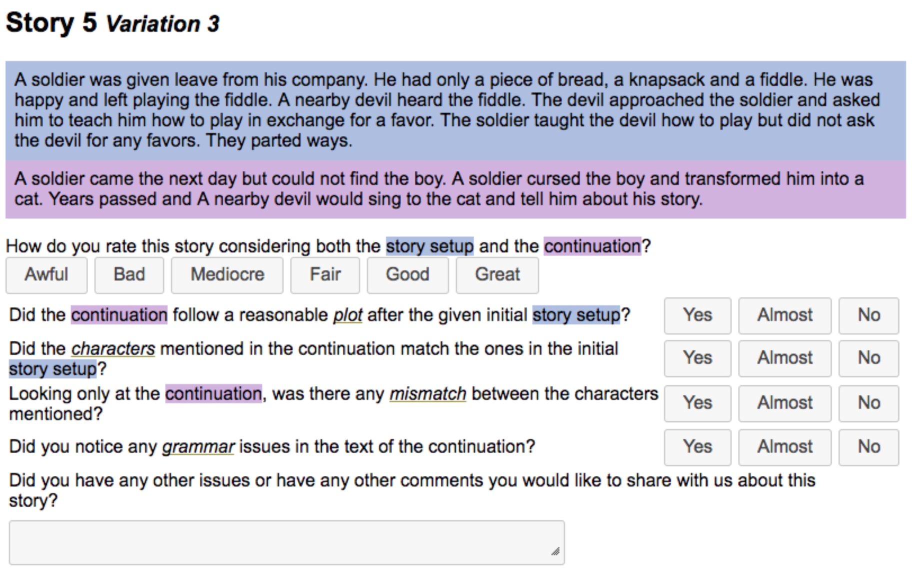
- How do you rate this story considering both the story setup[2] and the continuation? (A 6-point scale with the options: Awful/Bad/Mediocre/Fair/Good/Great).
- Did the continuation follow a reasonable plot after the given initial story setup? (A 3-point scale with the options: Yes/Almost/No).
- Did the characters mentioned in the continuation match the ones in the initial story setup? (A 3-point scale with the options: Yes/Almost/No).

[2]We used the term "story setup" in the form to refer to the *partial story* used as a query for *Riu*.

- Looking only at the continuation, was there any mismatch between the characters mentioned? (A 3-point scale with the options: Yes/Almost/No).
- Did you notice any grammar issues in the text of the continuation? (A 3-point scale with the options: Yes/Almost/No).
- Did you have any other issues or have any other comments you would like to share with us about this story? (Optional open text answer)

The first question is intended to compare the overall user perception of the stories and compare the results with and without using *Voz*, and calibrate these against the LSTM baseline and the original as an upper bound. The next 4 questions aim to isolate the potential sources of error, specifically, errors in the *plot* caused by mismatches during the retrieval and analogy mapping evaluation output in *Riu*, errors in the *character* mappings found during the analogy mapping potentially due to issues in the character classification or the taxonomy used; errors in *coreference* where a continuation has inconsistent replacements; and other errors in the grammatical structure of the continuation, potentially due to problematic mappings, misidentified verb arguments or other replacement issues after a mapping is applied. We will discuss these questions and the corresponding issues below; these 4 questions correspond to the 4 labels in Figure 7.6. Finally there is an open ended optional text field for the participants to provide any other comment they feel like sharing. Figure 7.4 shows an excerpt of the form used in our questionnaire showing an initial story setup (highlighted with a blue background), the continuation for one of the scenarios considered in the study (highlighted in purple), and the questions described above.

Over the course of 5 weeks, 179 participants started the study (that is, submitted the consent and demographic information form). For the results reported below we use the answers from 90 participants who completed the study (that is, submitted the second questionnaire form), which are eligible (are 18 or older, fully competent in written english and scored at least 5/6 in the reading comprehension test) and spent at least 5 minutes answering the questionnaire (average is 22 minutes when discarding 6 outliers: 4 participants spending more than 3 hours and 2 participants spending


Story 5 *Variation 3*

A soldier was given leave from his company. He had only a piece of bread, a knapsack and a fiddle. He was happy and left playing the fiddle. A nearby devil heard the fiddle. The devil approached the soldier and asked him to teach him how to play in exchange for a favor. The soldier taught the devil how to play but did not ask the devil for any favors. They parted ways.

A soldier came the next day but could not find the boy. A soldier cursed the boy and transformed him into a cat. Years passed and A nearby devil would sing to the cat and tell him about his story.

How do you rate this story considering both the story setup and the continuation?

Awful | Bad | Mediocre | Fair | Good | Great

Did the continuation follow a reasonable *plot* after the given initial story setup?	Yes	Almost	No
Did the *characters* mentioned in the continuation match the ones in the initial story setup?	Yes	Almost	No
Looking only at the continuation, was there any *mismatch* between the characters mentioned?	Yes	Almost	No
Did you notice any *grammar* issues in the text of the continuation?	Yes	Almost	No

Did you have any other issues or have any other comments you would like to share with us about this story?

Figure 7.4: Example section from the questionnaire showing one variation for a story consisting on a story setup, that is, the first segment of a story used to query *Riu*, and one of the continuations, in this case from the *Voz+Riu* scenario.

more than 12 hours). The participant's average age is 31.3 ($\sigma = \pm 10.8$) with 93% of the participants having at least started higher education and 27% pursuing or having completed a PhD. Notice that the results below consider only 1678 data points out of the $90 \times 5 \times 4 = 1800$ possible due to the responses being optional and some participants skipping some sections. Additionally, we received and coded 220 open text responses.

When looking at the overall rating of the 4 scenarios for the continuations, using the 1678 data points, we can clearly see how the original story continuation dominates the scores with a mean score rating of 4.7 ($\sigma = \pm 1.25$) on a scale of 1 (Awful) to 6 (Great). As expected, the LSTM scenario provides a lower bound with mean score rating of 1.35 ($\sigma = \pm 0.69$). When considering the two scenarios involving *Riu*, using *Voz*, has a slightly lower mean score rating of 2.48 ($\sigma = \pm 1.35$) than using *Riu* with the input derived from the annotations, which gets a mean score rating of 2.85 ($\sigma = \pm 1.24$). The difference between the distributions for the two scenarios using *Riu* is close but statistically significant ($p = 0.00004$). The original score distribution difference when compared to the *Annotations+Riu* scenario is statistically significant ($p = 8.107 \times 10^{-102}$) and the so is the difference between the *Voz+Riu* and *LSTM* scenarios ($p = 2.309 \times 10^{-78}$). Table 7.1 reports the

Table 7.1: Number of responses, average and spread of the overall scores for each of the scenarios included in the questionnaire. N=90 participants, 1678 data points. Figure 7.5 provides a visual representation of this data.

Scenario	Original	Anno.+Riu	Voz+Riu	LSTM
Responses	422	417	420	419
μ	4.70	2.85	2.48	1.35
σ	± 1.25	± 1.24	± 1.35	± 0.69

Table 7.2: Average score and spread for each scenario (columns) and the 4 follow-up questions (rows) included in the questionnaire. Scores have been normalized and inverted for the third and fourth question (higher is better). N=90 participants, 1678 data points. Figure 7.6 provides a visual representation of this data.

	Original	Anno.+Riu	Voz+Riu	LSTM
Char.	$0.77(\pm 0.52)$	$-0.22(\pm 0.77)$	$-0.49(\pm 0.74)$	$-0.96(\pm 0.24)$
Coref.	$0.88(\pm 0.37)$	$0.07(\pm 0.77)$	$-0.41(\pm 0.77)$	$-0.74(\pm 0.55)$
Gramm.	$0.93(\pm 0.32)$	$0.34(\pm 0.88)$	$-0.02(\pm 0.94)$	$-0.46(\pm 0.85)$
Plot	$0.85(\pm 0.48)$	$0.41(\pm 0.85)$	$0.06(\pm 0.97)$	$-0.79(\pm 0.59)$

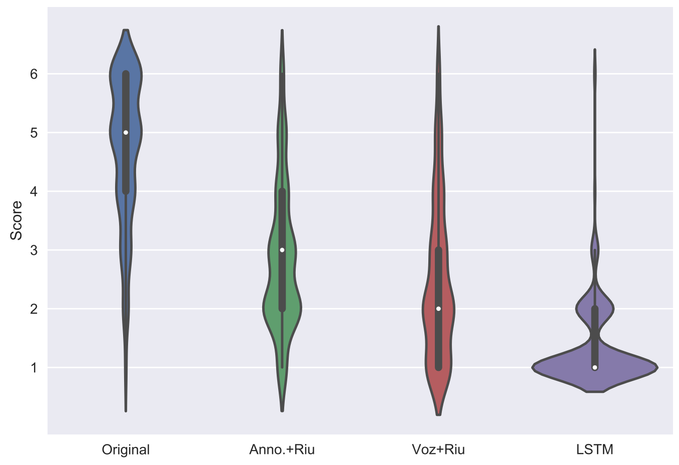


Figure 7.5: Plot comparing the 4 overall score ratings given by the users for each of the 4 continuation scenarios included in the questionnaire. N=90.

average and spread scores for each of the scenarios. Figure 7.5 shows a visual representation of the distributions for Table 7.1. Inside each of the violin plot we include a box plot with the quartiles for each scoring distribution.

Then we analyzed the 4 follow-up questions for each of the continuation scenarios. As mentioned above, these intend to elicit the specific sources of error as perceived by the study participants.

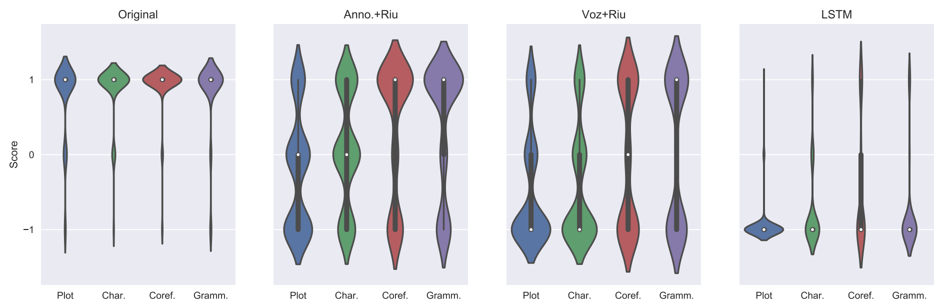


Figure 7.6: Plot comparing the 4 continuation scenarios and the scores for the 4 follow-up questions included in the questionnaire. Scores have been normalized and inverted for the third and fourth question (higher is better). N=90.

These questions were asked in a 3-point scale; note that for the first two questions, higher is better and for the third and fourth question, lower is better. We normalized the scores by inverting the last two and put them in a [-1,1] scale. When looking at the aggregated sore, we observe a high correlation between these and the overall scores discussed above where the original stories are highly rated ($\mu = 0.857$) and the LSTM exhibit the worst ratings ($\mu = -0.740$). When looking specifically at the results for the two scenarios involving *Voz*, using annotations has a higher average rating ($\mu = 0.261$) compared to using the fully automated pipeline ($\mu = 0.246$). We anticipated that there would be more coreference and grammar errors because of the replacements in the fully automated pipeline scenario as we can clearly see the participants noticed character mismatches ($\mu = -0.0217$ and $\mu = 0.0629$ for automated versus $\mu = 0.345$ and $\mu = 0.411$ for the annotated scenario), yet it was interesting to see that the perceived error related to plot mismatches and character mismatches for the annotated scenario has such large discrepancies ($\mu = -0.218$, $\sigma = \pm 0.770$ and $\mu = 0.068$, $\sigma = \pm 0.767$). Table 7.2 reports the average and spread scores for follow-up questions for each of the scenarios. Figure 7.6 provides a visual representation of the 4 data in Table 7.2. Note that Table 7.2 has been transposed to fit in the page.

Finally, we analyzed the open text questions. We performed thematic analysis by coding labels for recurring topics which then we grouped and mapped to them to the sources of error targeted by the 4 questions used previously and compiled the most relevant below. Notice that we coded multiple labels in each comment. In parenthesis we indicate the label and the count of the thematic labels found across the set of textual responses.

- The most widely reported thematic label found in the comments was regarding characters, that is, when there was a character mismatch between the story setup and the given continuation, or when there had been a coreference error that caused mismatch in the characters within the continuation. This is mentioned in 27 of the 220 comments (27 instances of *character-issues*) and is specially prevalent in the *Voz+Riu* scenario (20 instances of *character-issues*). Of special interest comments where participants notice that due to missing matches, there are new characters in the continuations that have not been introduced before (7 instances of *new-characters*).

- In terms of pronominal errors, there are several codes. First, there are pronominal errors due to coreference errors (13 instances of *pronominal*). Closely related, there are a couple stories where the initial segment introduces two girls. These were chosen purposefully when selecting the stories for the questionnaire. In the continuations, there are a few instances where there are two girls as well but in some from the *Voz+Riu* scenario there is only one. Either way, the continuation uses pronouns to refer to them (*she*) and it is then unclear to which girl the pronoun is referring. There are 8 instances (*ambiguous-pronoun-she*) of this specific case coded in the comments, included within other ambiguous pronouns issues (19 instances of *ambiguous-pronoun*).

- A very common comment for the *LSTM* scenario is that the output is either incomprehensible (18 instances of *incomprehensible*) or ungrammatical (15 instances of *ungrammatical*). Unfortunately, some users reported ungrammatical for *Anno.+Riu* continuations (2 instances) and ungrammatical for several *Voz+Riu* scenarios (5 instances).

- We also see comments referring to particularities of the Russian folk tales, which appear even for the original story continuations (18 instances of *weird*). Related to the previous there are several instances of comments about particular common sense issues (15 instances of *commonsense*), but again, these appear distributed across all the scenarios. Some recurring comments on the originals involve a story where a girl cut another's eyes out which participant found odd (7 instances of *weird-eyes*) and wonder why snakes are referred with female pronouns (4 instances of *pronominal-she-snake*).

- Another common comment, prevalent across the different scenarios but specially for the *Voz+Riu* and *Anno.+Riu* scenarios is the lack of explanations (11 instances of *lack-explanation*) or motivation (13 instances of *lack-motivation*) for the actions in the continuation. Also, several participants complain about a missing resolution (10 instances of *missing-ending*) or other missing part of the story (10 instances of *missing-other*) for most of the continuation scenarios except the *LSTM*.

- Overall the participants in the study provided thoughtful and useful comments, with some trying to guess the sources of the continuations (5 instances of *guess*), curious about the stories (4 instances of *continuation*) and providing suggestions on how to fix errors in the continuations (11 instances of *suggestion*). Unfortunately, as this study was conducted anonymously over the Internet, there were a few instances of nonsense and profanity (13 instances of *profanity*).

Table 7.3 reports the codes used, their counts and an example of the instantiation within the comments.

7.4 Discussion

In this chapter we presented our approach for bridging the gap between work on narrative information extraction and work on story generation in order to implement end-to-end computational narrative systems. Then presented our work on using the narrative information automatically extracted by *Voz* in *Riu* and developing a prototype of a text-based, end-to-end computational narrative system that can automatically process a corpus of natural language stories with minimal human intervention. We first manually analyzed the structures extracted by *Voz* and the final stories generated by *Riu*. We observed issues with the performance of narrative information extraction and some limitations of the shallow representation formalism produced by our mapping. On the other hand, we also

Table 7.3: Counts for codes and example instances for comments where they are instanciated. The numeric columns stand for *Original*, *Annotations+Riu*, *Voz+Riu*, *LSTM*, and *Total*.

Code	Or.	AR	VR	NN	T.	Example
character-issues	1	5	20	1	27	"Boy" should be replaced with "prince".
ambiguous-pronoun	0	19	0	0	19	"She" is far too ambiguous in this context.
incomprehensible	0	0	0	18	18	This is a mess.
weird	9	1	8	0	18	Stories that hint at incest do not usually begin with "once upon a time."
ungrammatical	4	2	5	4	15	Just a mess.
commonsense	4	4	4	3	15	Snakes, hungry or otherwise, cannot walk.
lack-explanation	10	3	0	0	13	Revenge doesn't make sense.
pronominal	0	6	6	1	13	Who is who?
missing-other	3	5	5	0	13	An extra sentence or two could cause it to make sense.
bad	5	4	3	0	12	Macabre.
good	6	1	4	0	11	Beautiful.
lack-explanation	1	3	7	0	11	How did the witch curse the girl when she wasn't found?
suggestion	7	3	1	0	11	the second sentence in the continuation should read "the jealous neighbor" no "a jealous neighbor"
missing-ending	7	3	0	0	10	No climax or resolution just inciting incident and rising action.
ambiguous-she	7	1	0	0	8	We are not told who "she" is untill we get to know that she stole her older sister's clothes.
weird-eyes	7	0	0	0	7	Cut her eyes is a strange attack and would not be fatal.
guess	2	0	2	1	5	Ok, this one was not written by a human.
continuation	0	4	0	0	4	Open ending. Will the bearlet be allowed to stay?
pronominal-snake	3	0	1	0	4	Is the snake characterized as a "her"?

observed that some errors in *Voz* have a smaller or negligible effect in the generated stories showing that the analogy-based story generation approach is resilient to some errors, for example related to the story structure.

We followed up on this analysis by conducting a user study using a larger dataset. We collected data from 90 participants whom were asked to score automatically generated stories in several comparison scenarios. The numerical results on the story ratings and follow-up questions for the scenario where *Riu* uses the annotations as input are statistically higher than those using *Voz*. The larger difference between human-authored originals indicate that there is potentially a problem in either the intermediate model representation we decided to use and/or within *Riu*. The detailed analysis of the open-ended responses reinforces this statement. Additionally, the results on the LSTM baseline elicit some of the current limitations of neural network approaches, specially in terms of generating coherent output when sampling and their underperformance when used with small data sets. This supports our efforts towards developing approaches and techniques that incorporate additional information and can work on smaller datasets.

We have shown that *Voz* can extract high-level narrative information in the specific domain of Russian-like folk tales and we have used this information to develop an end-to-end text-based story generation system. Currently, this end-to-end system has some limitations and underperforms compared to our expectations. We need to better isolate the sources of error, but, we believe we could improve the output with further work on enriching the extracted model so *Riu* has more information for the retrieval and analogy mapping processes. We would also like to isolate the shortcomings of *Riu* by connecting *Voz* to another story generation system. As part of our future work, we would like to do a follow up study connecting *Voz* to another story generation system to get further insights into which part of the difference in performance observed was due to *Riu* and which was due to errors introduced by *Voz*.

Chapter 8: Conclusions and Future Work

In this final chapter of this dissertation we will give a brief summary of the work presented so far, highlighting our contributions and discussing how they relate to the open problems enumerated in the introduction. Finally we provide a brief discussion and list potential lines for future research.

Computational narrative is an emergent field of research at the intersection of traditional narratology, artificial intelligence, natural language processing and cognitive science with applications that range from the study of literature to content generation for computer games. Given its novelty, there are several open problems in the field. The main problems we addressed in this dissertation are related to the disconnect between the areas of research of narrative information extraction and story generation. Specifically, we are concerned with automatically extracting structured narrative information from text and the usefulness of the extracted information for computational narrative applications. Moreover, our hypothesis is that incorporating narrative domain knowledge into the extraction process will improve the performance of certain tasks within the extraction process and the final output of the process.

With this research statement to frame our work, we addressed the following questions:

How do we automatically extract structured narrative information from text? There is information that is trivial to automatically extract from text (e.g., letter counts) but in order to extract narrative information useful for other computational narrative systems we need to focus on richer, higher-level narrative information. This question involves addressing open problems in the fields of natural language processing, information extraction, knowledge representation and computational narrative. In order to answer this question, we studied automatic narrative information extraction. Specifically, we explored what structured narrative information we can extract from natural language text and how to map this information to intermediate computational narrative models. We worked on improving the performance of off-the-shelf general-purpose natural language processing (NLP) tools in the particular domain of fictional stories by incorporating narrative domain knowledge to core NLP and information extraction tasks. In order to incorporate this information we consider alternatives to linear pipelined architectures for information extraction, specifically the idea of feedback loops that allow feeding information produced by later modules of the pipeline back to earlier modules.

How do we evaluate narrative information extraction pipelines? When studying our narrative information extraction system, there is a visibility problem related to eliciting how the error was introduced and propagated between modules of the pipeline. Most of the existing work evaluated either single modules in isolation or treated the pipeline as a black-box and only evaluated the final output. This question goes deeper into pipelined architectures and tries to identify the interactions between modules and their contributions to the final output. We worked on a methodology to evaluate information extraction pipelines and to identify and quantify sources of error different sources of error.

How do we use the extracted information in story generation applications? There is no consensus on a representation formalism for narrative or the annotation to capture the subtleties of text. We surveyed computational models of narrative and devised an intermediate representation of a narrative that allowed us to connect our narrative information extraction pipeline to an existing story generation system. We then studied the feasibility of text-based end-to-end story generation systems using this approach.

The rest of this chapter summarizes the research presented in this dissertation, our contributions and a short discussion on how our contributions address the research questions posed above and how these relate to the open problems listed in the introduction.

8.1 Dissertation Summary

Throughout our related work and literature survey, we identified a clear distinction between different areas of research within the field of computational narrative. On the one hand, we identified work related to literature analysis involving natural language processing, information extraction and story understanding, and on the other hand, work related to story generation in approaches exploring computational creativity or applications in digital entertainment. Our most notable takeaway from our survey is the disconnect between these two areas of work. This situation may be justified by the lack of consensus on how to model a narrative or represent a narrative space. On top of that, there is little holistic work and most researchers use ad-hoc approaches to suit the needs of the particular task at hand. This is the main cause of the authorial bottleneck problem where different models are hand-crafted and their content populated in an often expensive and time-consuming process.

In our work we explore how to leverage previously existing work in the fields of natural language processing, information extraction and machine learning to address and improve tasks related to automatically extracting structured narrative information from text. Then, we studied different computational models of narrative used, their advantages and disadvantages, their similarities and how they relate to other tasks within computational narrative. Our search was focused on finding ways to map between tasks addressed in previously existing work and the specifics of these computational models of narrative. In this process we looked at how to improve our the performance of our own narrative information extraction system by leveraging narrative domain-specific knowledge resulting in an improvement over the original tasks. Because of the limitations of the current technology and the intrinsic properties of our application domain, such as ambiguity in natural language, the use of specific rhetorical figures or contextual/cultural phenomena in the sources, we do not anticipate automatic approaches to achieve perfect performance, specially when considering that even humans encounter problems when dealing with these tasks. This is even more relevant if we would consider automatically acquiring even higher-level computational models of narratives. These would need to incorporate extra tasks (e.g., extract appropriate time relations and embedded narratives) where the accuracy of existing technology currently fails. And even if the technology were able to achieve human-level results, there may still be shortcomings in the selected computational models in use. Yet, through our research and engineering efforts we learned that some error may not have the anticipated effect downstream in a pipeline, that some error may actually be negligible or mitigated later and that non-perfect performance may be enough in some applications and approaches to obtain results that we as human cannot really perceive.

Focusing on our work, we implemented our ideas into *Voz*, a narrative information extraction pipeline which we used in our work in conjunction with a corpus of annotated Russian and Slavic folktales and a synthetic dataset of Russian-like stories. The choice of such dataset and application domain is manifold. First, it is a well known domain within literature and computational narrative that has been extensively studied in or influenced many existing work; including work we can compare with and borrow from (e.g., Finlayson's dataset[84]). Then, it is a relatively constrained domain that allowed us to focus our research and delimit our engineering efforts. And last but not least, it is a domain that exhibits a plethora of specific phenomena uncommon in other text domains that brings interesting challenges to the table. For example, we worked on character identification in order to identify anthropomorphic animals and magical beings, and we worked on a framework for non-linear information extraction pipelines in order to incorporate and leverage narrative information about recurring elements in this specific domain.

These efforts added capabilities for *Voz* to extract structured narrative information from the stories in our corpus. We proposed a novel framework to evaluate information extraction pipelines and we used it to identify sources of error quantify their impact in the final output of the information extraction pipeline. We were able to use this information to guide our efforts and work on incremental improvements. Currently, *Voz* can process stories in a corpus and automatically extract limited and imperfect structured narrative information from the stories. This information may already be useful for applications that generate summarized representations or visualizations of individual stories or a corpus.

Our goal is to use this extracted information in computational models for other applications in computational narrative, specifically, automatic story generation. We shifted our work towards the study of suitable computational narrative models that could be used to bridge these areas of work and we settled with a dual-model approach used for analogy-based story generation. This dual-model best suits our efforts since it uses a computer-readable structured representation of a story, which we were able to generate from the extracted information by mapping it into an arbitrary structure; and also uses annotated text templates, which again, we can generate from the original stories and using *Voz* to annotate the different identified components in the extracted structure.

The final product of our work has been a study of the output of *Riu* an existing story generation system that uses the dual-model described before for automatic story generation. Specifically, we relate this work to our previous work on pipelines and the effect of the error introduced in the pipeline, by *Voz* in this case, in the final output. This work uses a user study where we evaluated the perceived quality by human subjects given the known errors in the extracted narrative information. In this user study we gained insights into some of the current issues with the prototype. Analyzing the results we developed ideas on how to improve the prototype (e.g., enhance the structure used for analogy mapping by including additional elements) and extend the study to other story generation systems and towards a broader use for other computational narrative applications.

8.2 Contributions

Let us now provide a detailed list of our specific contributions derived from our work and described in this dissertation.

The Idea of Incorporating Narrative Information in Information Extraction Pipelines: Transversal to the rest of the contributions in this list is the idea of exploiting narrative domain information for different tasks related to narrative information extraction. For example, we use prototypical interactions described in narrative theory in our work on "sphere of action" encodings and to improve coreference information. Ultimately our individual contributions are tied in together in our narrative information extraction pipeline where we introduce feedback loops that feed the extracted narrative information back to previous modules. The idea of incorporating domain specific knowledge and feeding back this knowledge into other tasks is a critical contribution from our work that has not received enough attention in the fields of computational narrative, information extraction or natural language processing.

A Framework for Non-linear IE Pipelines: Related to the previous, we propose a general framework for introducing non-linear features, such as feedback loops or the aggregation of multiple parallel modules, in existing information extraction pipelines. We use this framework to introduce two feedback loops for two separate tasks in our information extraction pipeline which utilize extracted narrative information.

A Framework for Studying Error Propagation in IE Pipelines: We proposed a general method for studying how error is introduced and propagated though different modules in information extraction pipelines. This method can be used to find performance bottlenecks in existing information extraction pipelines and modules where the error has little effect to the final performance of the system or it may even be mitigated by later modules downstream.

A Method for Mention Extraction: We introduced a mention extraction algorithm that can handle mentions to individual entities and lists. This makes the extracted mentions able to handle split antecedents (e.g., a plural pronoun referring to multiple entities that have been mentioned in a list.

A Method for Mention and Entity Classification: We proposed a machine learning methodology for mutli-class classification which we used to classify mentions and entities (represented by a group of mentions) in terms of animacy, type and narrative role. Our methodology is inspired by case-based reasoning and uses coreference information to improve classification accuracy by performing a voting process over the coreference group including the mentions representing an entity.

A Novel Similarity Metric: We proposed a novel variant to the Jaccard index or Jaccard similarity coeficient that we call *Weighted Continuous Jaccard Distance*. This metric can be used for computing similarities between vectors of rational numbers. We showed that in our particular case this distance measure outperformed other commonly used measures like cosine or Euclidean distance.

An Encoding for Sphere-of-Action Information: We introduced a matrix-based representation to encode the "sphere of action" of a set of characters. This representation uses verbs to define interactions between characters. We showed how this matrix can be used to both represent prototypical interactions in a narrative theory, can be extracted from text, and both can be compared to one another to find assignments.

A Method for Identifying Narrative Functions from Natural Language: We propose a method for predicting Proppian functions in text segments based on aggregating often conflicting local information from features present in the segments and global information from Propp's narrative theory.

A Method for Identifying Character Interactions in Dialogue: We extended an existing method for processing dialogue in stories. Our new method allows the identification of both speaker and intended listener for quoted text or dialogue in stories. Moreover, our method can work with user-defined patterns for speaker and listener identification or can learn patterns automatically.

Voz: We implemented a narrative information extraction pipeline which we made available for researchers interested in using it for their projects. The source code for *Voz* is currently freely available online[1].

A Compilation of Sources for Russian and Slavic Folk Tales: We compiled a list of freely available sources for Russian and Slavic folk tales and mapped the numbers used by Propp to the English titles from a collection of translated folk tales by Alexander Afanasyev. This information is available in Appendix D.

A Synthetic Dataset of Russian-like Short Stories: We manually authored a collection of 100 Russian-like short stories which we annotated and made available for researchers interested in a dataset that poses less natural language complexities but still exhibits features from Russian stories like the ones analyzed by Propp.

A Prototype for End-to-end Computational Narrative Systems: Finally, in this dissertation we present a survey on the underlying models used in different computational narrative applications and show a proof-of-concept in which we map the output of *Voz* to the input of *Riu*, an existing computational narrative system. We implemented an end-to-end text-based

[1]Voz and some related components are available online: `https://sites.google.com/site/josepvalls/home/voz`

computational narrative system that can work directly from a corpus of natural language stories.

Our contributions extend previous work on different levels. At the lowest level, we have specific contributions for methods to identify mentions in text and identify or classify them into different classes. Our approach differs from other work, such as the named entity recognizer in the Stanford CoreNLP[35] that uses a statistical approach (conditional random fields) trained on a large corpus of annotated data. In contrast, our work relies on a hybrid approach that encodes domain knowledge (e.g., the concept that an entity or referring expression is realized in a noun phrase) and a limited amount of narrative domain knowledge (e.g., the labeled examples provided). For some tasks, such as character/non-character classification, our approach extends previous state-of-the art[13] and improves their results in terms of classification performance for our dataset. Another example, in the task of dialog participant identification; our method is built upon previous work[149] and achieves similar performance in our dataset but we argue our approach is more flexible as it allows easily defining patterns and rules or learn these patterns and rules automatically. Some other tasks, such as character role identification or narrative function identification, are less common and it's difficult to compare to other approaches because of the particularities of the different domains used in the literature or differences in the datasets, annotations or goals.

From a higher level perspective, most of our contributions are focused around the same idea where we encode a limited amount of domain knowledge to avoid the large training data requirements of other approaches[62]. For example, in our narrative function identification work, we rely on learning a model of the narrative function sequence from a small annotated dataset which is complemented by a model that encodes the rules defined in Proppian narrative theory. Additionally, our contributions relate to previous work in computing and NLP related to combining local and contextual information. Some previous work used global inference and contextual awareness[17;179]. Our work differs from these methods in two key areas: First, we extend the very common pipelined architecture used in many NLP and IE applications. Second, we extend this work on pipelined architectures[55] by adding non-linear features which we relate to the field of hermeneutics and specifically, the *hermeneutic circle* that suggests that for better interpretation of complex texts, one must alternate between acquiring local information from individual parts of the text and the context of the text as a whole[180].

8.3 Discussion

In this dissertation we made specific contributions to computational narrative and information extraction. Additionally, we showcased an end-to-end computational narrative system that bridges the gap in computational narrative between automated literature analysis and story generation by automatically extracting structured narrative information from text, mapping the extracted information into an intermediate model and using it with an existing application.

We describe the success of our approach in two ways: first and foremost, our proof-of-concept implementation shows that we can connect our approach with existing work and continue working to alleviate the authorial bottleneck problem in story generation applications. On the other hand, the results of our user study show that despite the known error in *Voz*, the generated stories are rated similarly to stories generated from an annotated dataset. As mentioned in the previous chapter, this points at evident limitations in the model used but at the same time, we believe that both the narrative information tasks and the model could be expanded in order to improve the generated story ratings. Richer narrative models could be suitable for other domains and applications which leads us to ask: What could be accomplished in the near future by extending our work and incorporating upcoming developments in NLP? We believe the answer to this question is to review the work discussed earlier in this dissertation related to automated literature analysis[8;18;26;84;85;140] and any of the work in storytelling[10;12;95]. There has already been scattered work addressing several tasks (e.g., identifying embedded narratives, parsing time cues to encode storylines using time relations, etc.) independently. And these tasks are the stepping stones required to assemble more complex narrative models already proposed in the literature[26]. What is needed is to homogenize and bring all these efforts together. Towards this goal, our core contributions lie in the proposed frameworks for integrating different modules into information extraction pipelines, improving the performance of individual modules within these pipelines (exploiting domain-specific information and/or feeding back extracted information to earlier modules) and evaluating the error propagation within pipelines in order to guide further work. Additionally, as we already mentioned, we expect the performance of the individual modules to continue improving which will cascade into the final results, still, we do not anticipate that the performance will ever be perfect, specially when we consider that some tasks in narrative are ambiguous even for human readers and in some circumstances, narratives are even purposefully open to interpretation. In the final stages of our work we briefly explored approaches that can potentially hide imprecision or even uncertainty of certain tasks along the pipeline from the final output of a system. For example, the analogy-based story generation approach of *Riu* can work even when there are errors in the coreference information graph.

Related to this, we would like to reiterate on some of the strengths of our approach: we have been working on extracting structured and rich high-level narrative information automatically from text. We propose that these models be augmented with additional information and transformed into intermediate models that could be fed into other system. We expect this last step to be done automatically but nothing prevents researchers from looking into the models in order to examine what has been extracted or manually tweak the extracted information into an intermediate model. We believe that the explainability property is very desirable for certain applications were being able to interpret and manually tweak the models is useful.

This can be seen as a desirable property from the user-interface perspective in which NLP tools and information extraction pipelines may be used to both automatically ingest a sizable corpus but at the same time, allow for a simpler, natural-language based input into complex systems. In either case, the pipeline can be paired with existing user-interfaces into mixed-initiative approaches to model creation and population that allow examining and modifying an extracted model. Since our framework feeds information back into the pipeline, the pipeline itself could benefit from this user-interface elements and enable an iterative/incremental cycle between automatically extracting information, presenting it to the user for tweaking and feeding it back for refining the extraction.

Also related to this UI or model-tweaking scenario, as we already mentioned, our proposed framework relies heavily on the ability of incorporating domain-specific knowledge into the pipeline in order to improve the performance of specific tasks. This is specially relevant when dealing with limited datasets, sparse annotations or low-resource domains; all situations in which other approaches that rely on large datasets (e.g., neural networks) may not handle gracefully.

8.4 Future Work

In this thesis dissertation we have looked at different areas of research related to computational narrative and natural language processing. In our research we have worked on several of the open problems in the area yet these are still far from solved and there are many related problems that could be addressed by continuing our line of work. One topic that may warrant additional research is related to modeling stories and computational models of narrative. Fortunately, this seems an active area of research that draws sufficient interest from the community with workshops such as the 10th Workshop on Intelligent Narrative Technologies in 2017 and the 7th Workshop on Computational Models of Narrative in 2016. Progress in this area will further motivate further cross-pollination between NLP, information extraction and research in areas of computational narrative such as literature analysis and story generation. We believe that this will then lead to additional research on approaches to alleviate or eliminate the authorial bottleneck problem.

When looking at the current state of the field of computational narrative and our specific contributions, there are several lines of work we would be interested in exploring further.

In this work we have described NLP and information extraction pipelines and the benefits of non-linear architectures that incorporate feedback loops. To the best of our knowledge, the idea has not been widely explored. We believe this warrants further exploration, in terms of what tasks can benefit from feedback loops and what information can be used. We believe the use of feedback loops may be beneficial beyond computational narrative applications and a framework for adapting general purpose tools for domain-specific tasks may be of interest of several communities.

Our current research is focused on narrative information extraction, more specifically, information components from Proppian narrative theory. The non-linear framework for information extraction pipelines is not constrained to specific components or a specific narrative theory. We believe that different domain-specific information can be extracted that can be then related to the narrative models that are being targeted. At the same time, other narrative domain knowledge (or not even narrative-related) may be useful for other core NLP tasks. As part of our future work, we would like to explore the *Monomyth* or *Hero's journey*[77] in place of Propp's work. As mentioned in the previous item, we believe similar approaches can be explored for other domain-specific tasks beyond narratology.

In the work presented in this dissertation we used different corpora and datasets with different layers of annotation. We focused on different tasks where we used subsets or partially annotated datasets for the experimental evaluation. Moreover, we have identified and discussed several narrative and non-narrative elements that we cannot extract at the moment and may be required or useful for other computational narrative systems. For example information about narrative tension or moral of a story may be interesting for story generation and storytelling systems. Additionally, non-narrative information such as characters' mental models (or embedded narratives) and temporal models of a story may be useful to other text processing systems beyond computational narrative applications.

One task that we plan on working on the short term is the task of automatic *scene segmentation*. This task will segment stories into a sequence of spans of text containing coherent functional units of the narrative. We believe this task can use extracted spatial information, temporal cues and patterns of verb usage to delimit transitions between scenes. This automatic segmentation will be a required step towards a goal of full automatic processing for identifying functional parts of the dramatic structure of a story based on Proppian theory and will replace the manual segmentation used for our current work on narrative function identification. Moreover, this information is required by some computational narrative systems and will enable the rendering of visual representations of a story as a sequence of scenes as in a story board. Once we address the task of *scene segmentation*, we will be able to replace the manual segmentation used for our current narrative function identification work. We will still need to work to map the new segments (i.e. *scenes*) to the locations where narrative functions are being realized (i.e. the span of text where they happen). We may also need to tweak the current module to be able to identify multiple functions per scene or scenes without functions. We believe work in this area will elicit relationships between narrative theories and data driven approaches to reproduce or validate these theories. Moreover, these narrative elements encode information about the character and segmentation. Therefore, we expect to be able to use the proposed non-linear framework to feed back this information in order to inform some of the tasks of the pipeline and further improve the overall performance of the system.

As mentioned earlier in this dissertation, as part of our future work will be on using *Voz* for different computational narrative applications. In Section 7.2 we described our work connecting *Voz* to *Riu*, an existing story generation system. We would like to expand on this work by addressing some of the issues described in Section 7.3 and incorporate additional elements to the structure used for analogy mapping in order to improve the generated stories. In order to better isolate the error contributed by *Voz*'s performance and *Riu*'s particularities we are considering connecting *Voz* to another story generation system. We would like conduct a second user study featuring another system and additional options such as baselines derived from randomly selected sentences (guaranteed to be grammatical and sensical but not necessarily consistent with the plot) and other story continuations generated using additional manual annotations (which should enhance the analogy mapping process). Along with this user study, we would like to enlist participants to manually author the required annotations. This would give us insights on the actual effort required to author the input required by *Riu*, the examples required by *Voz* and quantify the difference between the effort required by the different scenarios therefore giving us a way to evaluating actual contributions towards alleviating the authorial bottleneck problem.

Additionally, related to future work with *Voz*, in our previous work described in Appendix E, we used an annotated plan-like representation of a story space to generate spatial environment configurations that could support the encoded story. We would like to further explore the work by Li et al. on *plot graphs*[10] and use *Voz* to extract location information from stories and use the extracted information to generate spatial environment configurations and visualizations of the environment of a story.

Another example of our planned future work would be to use *Voz* to visually represent stories and corpora, yet, several questions remain open; for example: how to compare or aggregate the graph representations of a corpus of stories and how to improve our *story graphs* for processing in other computational narrative tasks. Specifically we would like to explore how to use the output of *Voz* for visualizations such as the ones used by Reagan et al.[85] or the *plot graph* used by Li et al.[10]. Related to the use of *Voz* in other applications, we would like to see how to incorporate *Voz* as a tool to enable natural language user interfaces in mixed-initiative approaches similar to the work of Finlayson et al.[1] where a user can provide their input as text and use *Voz* to automatically annotate specific narrative information and ease the overall annotation process.

Bibliography

[1] Mark A. Finlayson. The Story Workbench: An Extensible Semi-Automatic Text Annotation Tool. *Intelligent Narrative Technologies*, pages 21–24, 2011. URL http://www.aaai.org/ocs/index.php/AIIDE/AIIDE11WS/paper/viewPDFInterstitial/4091/4455.

[2] David K. Elson and Kathleen R. McKeown. A platform for symbolically encoding human narratives. *Proceedings of the AAAI Fall Symposium on Intelligent Narrative Technologies*, pages 29–36, 2007. URL http://www.aaai.org/Papers/Symposia/Fall/2007/FS-07-05/FS07-05-007.pdf.

[3] Santiago Ontanon and Jichen Zhu. Story and text generation through computational analogy in the riu system. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2010.

[4] Jichen Zhu and Santiago Ontañón. Shall i compare thee to another story? - an empirical study of analogy-based story generation. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):216–227, 2014.

[5] James Meehan. *The Metanovel: Writing Stories by Computer.* Ph.d., Yale University, 1976.

[6] Chris R. Fairclough. *Story Games and the OPIATE System.* PhD thesis, University of Dublin, Trinity College, 2007. URL http://www.citeulike.org/group/1820/article/1056489.

[7] Ken Hartsook, Alexander Zook, Sauvik Das, and Mark O. Riedl. Toward supporting stories with procedurally generated game worlds. In *Proceedings of the 2011 IEEE Conference on Computational Intelligence in Games*, pages 297–304. IEEE, August 2011. ISBN 978-1-4577-0010-1. doi: 10.1109/CIG.2011.6032020.

[8] David K Elson, Nicholas Dames, and Kathleen R McKeown. Extracting social networks from literary fiction. In *Proceedings of the Fourty-Eighth Annual Meeting of the Association for Computational Linguistics*, pages 138–147. Association for Computational Linguistics, 2010.

[9] Richard Johansson, Anders Berglund, Magnus Danielsson, and Pierre Nugues. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI*, volume 5, pages 1073–1078, 2005.

[10] Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. Story generation with crowdsourced plot graphs. In *Proceedings of the 27th AAAI Conferece on Artificial Intelligence*, 2013.

[11] John A. Bateman. Enabling technology for multilingual natural language generation: the kpml development environment. *Natural Language Engineering*, 3(1):15–55, March 1997. ISSN 1351-3249. doi: 10.1017/S1351324997001514.

[12] Pablo Gervás. Computational Approaches to Storytelling and Creativity. *AI Magazine*, pages 49–62, 2009. URL https://blog.itu.dk/MPGG-E2011/files/2011/10/2250-3077-1-pb.pdf.

[13] Ricardo A. Calix, Leili Javadpour, Mehdi Khazaeli, and Gerald M. Knapp. Automatic Detection of Nominal Entities in Speech for Enriched Content Search. *The Twenty-Sixth International FLAIRS Conference*, pages 190–195, 2013.

[14] Josep Valls-Vargas, Santiago Ontañón, and Jichen Zhu. Toward automatic character identification in unannotated narrative text. In *Proceedings of the Seventh Workshop in Intelligent Narrative Technologies*, 2014. URL http://www.aaai.org/ocs/index.php/INT/INT7/paper/view/9253.

[15] Nathanael Chambers and Daniel Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of the 2008 Anniversary Meeting of the Association for Computational Linguistics*, volume 94305, pages 789–797, 2008.

[16] Amit Goyal, Ellen Riloff, and Hal Daumé, III. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 77–86, Stroudsburg, PA, USA, 2010.

[17] Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. Technical report, DTIC Document, 2004.

[18] Snigdha Chaturvedi, Shashank Srivastava, Hal Daume III, and Chris Dyer. Modeling dynamic relationships between characters in literary novels. *arXiv preprint arXiv:1511.09376*, 2015.

[19] Shashank Srivastava, Snigdha Chaturvedi, and Tom Mitchell. Inferring interpersonal relations in narrative summaries. *arXiv preprint:1512.00112*, 2015.

[20] James R Meehan. Tale-spin, an interactive program that writes stories. In *IJCAI*, volume 77, pages 91–98, 1977.

[21] Mark O Riedl and Robert Michael Young. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research*, 39(1):217–268, 2010.

[22] Peter Weyhrauch and Joseph Bates. *Guiding interactive drama*. Carnegie Mellon University Pittsburgh, PA, 1997.

[23] Mark J Nelson and Michael Mateas. Search-based drama management in the interactive fiction anchorhead. In *AIIDE*, pages 99–104, 2005.

[24] Manu Sharma, Santiago Ontañón, Manish Mehta, and Ashwin Ram. Drama management and player modeling for interactive fiction games. *Computational Intelligence*, 26(2):183–211, 2010.

[25] Vladimir Propp. *Morphology of the Folktale*. University of Texas Press, 1973.

[26] Inderjeet Mani. *Computational Modeling of Narrative*, volume 5 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers, 2012. ISBN 9781608459827. URL http://books.google.com/books?id=kbZdAQAAQBAJ.

[27] Ekaterina Buyko, Joachim Wermter, Michael Poprat, and Udo Hahn. Automatically adapting an nlp core engine to the biology domain. In *Proceedings of the ISMB*, pages 65–68, 2006.

[28] Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164. Association for Computational Linguistics, 2005.

[29] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34. Association for Computational Linguistics, 2011.

[30] Christopher D. Manning and Hinrich Schuetze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999. ISBN 0262133601.

[31] Emily Thomforde and Mark Steedman. Semi-supervised ccg lexicon extension. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1246–1256. Association for Computational Linguistics, 2011.

[32] Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1): 80–91, 1996.

[33] Daniel G. Bobrow. *Natural language input for a computer problem solving system.* PhD thesis, Massachusets Institute of Technology, 1964.

[34] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*, volume 33. MIT Press, 2000.

[35] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[36] Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 171, 2005.

[37] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[38] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112–116. Association for Computational Linguistics, 1992.

[39] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the Forty-First Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[40] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*, 2012.

[41] Erick R. Fonseca and Joao Luis G. Rosa. A two-step convolutional neural network approach for semantic role labeling. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 2955–2961. IEEE, 2013.

[42] Erick Rocha Fonseca, Sandra Maria Aluisio, et al. A deep architecture for non-projective dependency parsing. In *Conference of the North American Chapter of the Association for Computational Linguistics-Human Language Technologies, Workshop on Vector Space Modeling for Natural Language Processing.* Association for Computational Linguistics, 2015.

[43] Mark Stevenson and Mark A. Greenwood. Comparing Information Extraction Pattern Models. In *Proceedings of the Workshop on Information Extraction Beyond The Document*, pages 12–19, 2006.

[44] Roger Levy and Galen Andrew. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, pages 2231–2234. Citeseer, 2006.

[45] Ellen Riloff and William Phillips. An introduction to the sundance and autoslog systems. Technical report, Technical Report UUCS-04-015, School of Computing, University of Utah, 2004.

[46] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 940–946. Association for Computational Linguistics, 2000.

[47] Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 602—-610, 2009.

[48] Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27. Association for Computational Linguistics, 2011.

[49] Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916, 2013.

[50] Vasin Punyakanok, Dan Roth, and Wen Tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, 2008.

[51] Hien T. Nguyen and Tru H. Cao. Named entity disambiguation: A hybrid statistical and rule-based incremental approach. In *The Semantic Web*, pages 420–433. Springer, 2008.

[52] Jon Curtis, John Cabral, and David Baxter. On the application of the cyc ontology to word sense disambiguation. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, pages 652–657, 2006.

[53] Catherine Havasi, Robert Speer, and James Pustejovsky. Coarse word-sense disambiguation using common sense. In *AAAI Fall symposium series*, 2010.

[54] Lei Shi and Rada Mihalcea. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Computational Linguistics and Intelligent Text Processing*, pages 100–111. Springer Berlin Heidelberg, 2005.

[55] James Clarke, Vivek Srikumar, Mark Sammons, and Dan Roth. An nlp curator (or: How i learned to stop worrying and love nlp pipelines). In *LREC*, pages 3276–3283, 2012.

[56] David Bamman, Brendan O'Connor, and Noah A Smith. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 352–361, 2014.

[57] David Bamman, Ted Underwood, and Noah A Smith. A bayesian mixed effects model of literary character. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 370–379, 2014.

[58] Bob Coyne, Owen Rambow, Julia Hirschberg, and Richard Sproat. Frame semantics in text-to-scene generation. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 375–384. Springer, 2010.

[59] WC Mann. *Overview of the penman text generation system.* University of Southern California, Marina Del Rey (USA). Information Sciences Inst., 1983.

[60] Eduard H Hovy and Yigal Arens. Automatic generation of formatted text. *Readings in intelligent user interfaces*, page 262, 1998.

[61] François Mairesse and Marilyn A Walker. Trainable generation of big-five personality styles through data-driven parameter estimation. In *ACL*, pages 165–173, 2008.

[62] Yoav Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)*, 57:345–420, 2016.

[63] Sam Wiseman, Alexander M Rush, and Stuart M Shieber. Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*, 2016.

[64] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432, 2015.

[65] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[66] Xinlei Chen and C Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431, 2015.

[67] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[68] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[69] Lara J Martin, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. Event representations for automated story generation with deep neural nets. In *Proceedings of the 2018 Conference of the Association for the Advancement of Artificial Intelligence*, 2018.

[70] Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. Deeptingle. *arXiv preprint arXiv:1705.03557*, 2017.

[71] George Lakoff. Structural complexity in fairy tales. *The study of man*, 1972.

[72] Seymour Benjamin Chatman. *Story and discourse: Narrative structure in fiction and film.* Cornell University Press, 1980.

[73] Sandy Andersen and Brian M Slator. Requiem for a theory: the 'story grammar' story. *Journal of Experimental & Theoretical Artificial Intelligence*, 2(3):253–275, 1990.

[74] Michael Mateas and Phoebe Sengers. Narrative intelligence. In *Proceedings AAAI Fall Symposium on Narrative Intelligence*, 1999.

[75] Whitman Richards, Mark Alan Finlayson, and Patrick Henry Winston. Advancing computational models of narrative. Technical report, CSAIL Technical Report, 2009.

[76] Ben Kybartas and Rafael Bidarra. A survey on story generation techniques for authoring computational narratives. *IEEE TCIAIG*, 2016.

[77] Joseph Campbell. *The hero with a thousand faces*, volume 17. New World Library, 2008.

[78] Gérard Genette and Jane E. Lewin. *Narrative discourse: An essay in method.* Cornell University Press, 1983.

[79] David Herman. Narratology as a cognitive science. *Image and Narrative*, 1:1, 2000.

[80] Benedictus de Spinoza. *Theological-political Treatise.* Hackett Publishing House, Indianapolis, gebhardt edition, 2001.

[81] Roger C Schank and Robert Wilensky. A goal-directed production system for story understanding. *ACM SIGART Bulletin*, 63(63):72–72, 1977.

[82] Adam Amos-Binks, Colin Potts, and R. Young. Planning graphs for efficient generation of desirable narrative trajectories. In *Proceedings of the 13th Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017. URL `https://www.aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15849`.

[83] Scott R. Turner. *Minstrel: A Computer Model of Creativity and Storytelling.* PhD thesis, University of California at Los Angeles, Los Angeles, CA, USA, 1993. UMI Order no. GAX93-19933.

[84] Mark A. Finlayson. *Learning narrative structure from annotated folktales*. PhD thesis, Massachusetts Institute of Technology, 2012.

[85] Andrew J Reagan, Lewis Mitchell, Dilan Kiley, Christopher M Danforth, and Peter Sheridan Dodds. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5(1):31, 2016.

[86] Mark O. Riedl and Vadim Bulitko. Interactive narrative: An intelligent systems approach. *AI Magazine*, 34(1):67–77, 2013.

[87] Mark Riedl, David Thue, and Vadim Bulitko. Game ai as storytelling. In *Artificial Intelligence for Computer Games*, pages 125–150. Springer, 2011.

[88] Reid Swanson and Andrew S. Gordon. Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling. *ACM Transactions on Interactive Intelligent Systems*, 2(3):16:1–16:35, September 2012. ISSN 2160-6455. doi: 10.1145/2362394.2362398. URL http://doi.acm.org/10.1145/2362394.2362398.

[89] Chris R. Fairclough and Pádraig Cunningham. A multiplayer case based story engine. *Computer Science Technical Report*, 2003.

[90] Josep Valls-Vargas, Santiago Ontañón, and Jichen Zhu. Towards story-based content generation: From plot-points to maps. In *Proceedings of the 2013 IEEE Conference on Computational Intelligence in Games*. IEEE, 2013. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.385.5440&rep=rep1&type=pdf.

[91] Joshua McCoy, Mike Treanor, Ben Samuel, Noah Wardrip-Fruin, and Michael Mateas. Comme il faut: A system for authoring playable social models. In *AIIDE*, 2011.

[92] Mark Alan Finlayson. Deriving narrative morphologies via analogical story merging. In *Proceedings, 2nd International Conference on Analogy*, pages 127–136, 2009.

[93] Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story Plot Generation based on CBR. *Knowledge-Based Systems*, 2005. URL http://www.sciencedirect.com/science/article/pii/S0950705105000407.

[94] Brian Falkenhainer, Kenneth D Forbus, and Dedre Gentner. The structure-mapping engine: Algorithm and examples. *Artificial intelligence*, 41(1):1–63, 1989.

[95] Mark O Riedl, Andrew Stern, Don Dini, and Jason Alderman. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, 4(2):23–42, 2008.

[96] Alejandro Ramirez and Vadim Bulitko. Telling Interactive Player-Specific Stories and Planning for It: ASD+ PaSSAGE= PAST. In *Proceedings of the Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 173–178, 2012. URL http://www.aaai.org/ocs/index.php/AIIDE/AIIDE12/paper/download/5455/5716.

[97] Julian Togelius, Georgios Yannakakis, Kenneth Stanley, and Cameron Browne. Search-based procedural content generation. *Applications of Evolutionary Computation*, pages 141–150, 2010.

[98] Jonathon Doran and Ian Parberry. A prototype quest generator based on a structural analysis of quests from four MMORPGs. *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games - PCGames '11*, pages 1–8, 2011. doi: 10.1145/2000919.2000920.

[99] Joris Dormans and Sander Bakkes. Generating Missions and Spaces for Adaptable Play Experiences. *Computational Intelligence and AI in Games, IEEE Transactions*, 3(3):216–228, 2011.

[100] Peter Sheridan Dodds, Kameron Decker Harris, Isabel M Kloumann, Catherine A Bliss, and Christopher M Danforth. Temporal patterns of happiness and information in a global social network: Hedonometrics and twitter. *PloS one*, 6(12):e26752, 2011.

[101] James Meehan. Tale-spin. *Inside computer understanding: Five programs plus miniatures*, pages 197–226, 1981.

[102] Santiago Ontanon and Jichen Zhu. The sam algorithm for analogy-based story generation. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[103] Michael Lebowitz. Story-telling as planning and learning. *Poetics*, 14(6):483–502, 1985.

[104] Mei Si, Stacy C Marsella, and David V Pynadath. Proactive authoring for interactive drama: An author's assistant. In *International Workshop on Intelligent Virtual Agents*, pages 225–237. Springer, 2007.

[105] Mariët Theune, Sander Faas, DKJ Heylen, and Anton Nijholt. The virtual storyteller: Story creation by intelligent agents. *Proceedings TIDSE 2003: Technologies for Interactive Digital Storytelling and Entertainment*, pages 204–215, 2003.

[106] Nick Montfort, Rafael Pérez, D Fox Harrell, and Andrew Campana. Slant: A blackboard system to generate plot, figuration, and narrative discourse aspects of stories. In *Proceedings of the 4th international conference on computational creativity*, pages 168–175, 2013.

[107] Nick Montfort. Curveship: an interactive fiction system for interactive narrating. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 55–62. Association for Computational Linguistics, 2009.

[108] Yun-Gyung Cheong and R Michael Young. Narrative generation for suspense: Modeling and evaluation. In *Joint International Conference on Interactive Digital Storytelling*, pages 144–155. Springer, 2008.

[109] Joseph A Goguen and D Fox Harrell. Style: A computational and conceptual blending-based approach. In *The structure of style*, pages 291–316. Springer, 2010.

[110] Arnav Jhala and R Michael Young. Cinematic visual discourse: Representation, generation, and evaluation. *IEEE TCIAIG*, 2(2):69–81, 2010.

[111] Isabel Machado, Ana Paiva, and Paul Brna. Real characters in virtual stories. In *International Conference on Virtual Storytelling*, pages 127–134. Springer, 2001.

[112] Ulrike Spierling, Dieter Grasbon, Norbert Braun, and Ido Iurgel. Setting the scene: playing digital director in interactive storytelling and creation. *Computers & Graphics*, 26(1):31–44, 2002.

[113] Benoit Lavoie and Owen Rambow. A fast and portable realizer for text generation systems. In *Proceedings of the fifth conference on Applied natural language processing*, pages 265–268. Association for Computational Linguistics, 1997.

[114] Neil McIntyre and Mirella Lapata. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 217–225. Association for Computational Linguistics, 2009.

[115] Mark Riedl, Cesare J Saretto, and R Michael Young. Managing interaction between users and agents in a multi-agent storytelling environment. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 741–748. ACM, 2003.

[116] David L Roberts and Charles L Isbell. A survey and qualitative analysis of recent advances in drama management. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, 4(2):61–75, 2008.

[117] Mark Owen Riedl and Vadim Bulitko. Interactive narrative: An intelligent systems approach. *Ai Magazine*, 34(1):67, 2012.

[118] Ruth Aylett. Narrative in virtual environments-towards emergent narrative. In *Proceedings of the AAAI fall symposium on narrative intelligence*, pages 83–86, 1999.

[119] Marc Cavazza and David Pizzi. Narratology for interactive storytelling: A critical introduction. In *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, pages 72–83. Springer, 2006.

[120] R Michael Young, SG Ware, BA Cassell, and Justus Robertson. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative*, 37(1-2):41–64, 2013.

[121] Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on cbr. *Knowledge-Based Systems*, 18(4):235–242, 2005.

[122] Lewis Seifert, Celeste Lim, Laura Tan, and Nicole Wee. Digital propp: Proppian fairy tale generator, 2005.

[123] Ben Kybartas and Clark Verbrugge. Analysis of regen as a graph-rewriting system for quest generation. *IEEE TCIAIG*, 6(2):228–242, 2014.

[124] Julian Horsey. The elder scrolls v: Skyrim radiant quest system, will customise your game as you play. *On-line, Geeky Gadgets*, 2011.

[125] Josh McCoy, Mike Treanor, Ben Samuel, Brandon Tearse, Michael Mateas, and Noah Wardrip-Fruin. Authoring game-based interactive narrative using social games and comme il faut. In *Proceedings of the fourth International Conference & Festival of the Electronic Literature Organization: Archive & Innovate*. Citeseer, 2010.

[126] Scott A. Malec. Proppian structural analysis and xml modeling. *Proceedings of Computers, Literature and Philology (CLiP 2001)*, 2001.

[127] Mark A. Finlayson. Collecting semantics in the wild: The story workbench. In *Naturally Inspired Artificial Intelligence, Technical Report FS-08-06, Papers from the 2008 AAAI Fall Symposium*, pages 46–53, 2008.

[128] David K. Elson. *Modeling Narrative Discourse*. PhD thesis, Columbia University, 2012.

[129] Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. Toward automatic role identification in unannotated folk tales. In *Proceedings of the Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014. URL http://www.aaai.org/ocs/index.php/AIIDE/AIIDE14/paper/viewPDFInterstitial/9004/8947.

[130] Bob Coyne and Richard Sproat. Wordseye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496. ACM, 2001.

[131] Angel X Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *EMNLP*, pages 2028–2038, 2014.

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

[133] Erik T Mueller. Story understanding through multi-representation model construction. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning-Volume 9*, pages 46–53. Association for Computational Linguistics, 2003.

[134] Ellen Riloff. Information extraction as a stepping stone toward story understanding. *Understanding language understanding: Computational models of reading*, pages 435–460, 1999.

[135] Cynthia Matuszek, John Cabral, Michael J Witbrock, and John DeOliveira. An introduction to the syntax and content of cyc. In *AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.

[136] Gerald DeJong. An overview of the frump system. *Strategies for natural language processing*, 113:149–176, 1982.

[137] Ashwin Ram. *Question-driven understanding: An integrated theory of story understanding, memory and learning.* Ph.d., Yale University, 1989.

[138] Patrick Henry Winston and Dylan Holmes. The genesis manifesto: Story understanding and human intelligence, 2017.

[139] Peter Clark and Oren Etzioni. My computer is an honor student-but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 37(1):5–12, 2016.

[140] Mark Alan Finlayson. Inferring propp's functions from semantically annotated text. *Journal of American Folklore*, 129(511):55–77, 2016.

[141] Josep Valls-Vargas, Jichen Zhu, and Santiago Ontanon. Error analysis in an automated narrative information extraction pipeline. *IEEE Transactions on Computational Intelligence and AI in Games*, PP, 2016.

[142] Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. Towards automatically extracting story graphs from natural language stories. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[143] Brent Harrison, Christopher Purdy, and Mark Riedl. Toward automated story generation with markov chain monte carlo methods and deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2017. URL https://aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15908.

[144] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

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

[146] Michaela Regneri, Alexander Koller, Josef Ruppenhofer, and Manfred Pinkal. Learning Script Participants from Unlabeled Data. *RANLP*, 2011.

[147] Roger C. Schank and Robert P. Abelson. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.* Psychology Press, 2013.

[148] Boyang Li. *Learning Knowledge to Support Domain-Independent Narrative Intelligence.* PhD thesis, Georgia Institute of Technology, 2015.

[149] David K Elson and Kathleen McKeown. Automatic attribution of quoted speech in literary narrative. In *Proceedings of the Twenty-First international joint conference on Artificial Intelligence*, pages 1013–1019. AAAI Press, 2010.

[150] Tim O'Keefe, Silvia Pareti, James R Curran, Irena Koprinska, and Matthew Honnibal. A sequence labelling approach to quote attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 790–799. Association for Computational Linguistics, 2012.

[151] Grace Muzny, Michael Fang, Angel Chang, and Dan Jurafsky. A two-stage sieve approach for quote attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 460–470, 2017.

[152] Eliza Margaretha and David DeVault. An approach to the automated evaluation of pipeline architectures in natural language dialogue systems. In *Proceedings of the SIGDIAL 2011 Conference*, pages 279–285. Association for Computational Linguistics, 2011.

[153] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *Advances in information retrieval*, pages 345–359. Springer, 2005.

[154] Gerald Prince. *A dictionary of narratology*. Univ of Nebraska Press, Lincoln, 2003.

[155] Scott Malec. Autopropp: Toward the automatic markup, classification, and annotation of russian magic tales. In *Proceedings of the First International AMICUS Workshop on Automated Motif Discovery in Cultural Heritage and Scientific Communication Texts*, pages 112–115, 2010.

[156] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994.

[157] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[158] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1999.

[159] Hugo Liu and Push Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

[160] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the Thirty-Second annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[161] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.

[162] Paul Jaccard. The distribution of the flora in the alpine zone. *New phytologist*, 11(2):37–50, 1912.

[163] Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular norms*. Springer, 2000.

[164] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[165] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[166] Santiago Ontañón and Enric Plaza. Similarity measures over refinement graphs. *Machine learning*, 87(1):57–92, 2012.

[167] Josep Valls-Vargas, Santiago Ontañón, and Jichen Zhu. Toward character role assignment for natural language stories. In *Proceedings of the Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 101–104, 2013. URL http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/view/7439/7661.

[168] Paul R Cohen. *Empirical methods for artificial intelligence*, volume 139. MIT press Cambridge, MA, 1995.

[169] Daniel J Lasser. Topological ordering of a list of randomly-numbered elements of a network. *Communications of the ACM*, 4(4):167–168, 1961.

[170] Ron Kohavi and David H Wolpert. Bias plus variance decomposition for zero-one loss functions. In *ICML*, pages 275–283, 1996.

[171] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[172] Robert F. Bordley. A multiplicative formula for aggregating probability assessments. *Management Science*, 28(10):1137–1148, 1982.

[173] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, 1998.

[174] Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 30–35, 2014.

[175] Pablo Gervás, Raquel Hervás, and Juan Antonio Recio-García. The role of natural language generation during adaptation in textual cbr. In *4th Workshop on Textual Case-Based Reasoning: Beyond Retrieval (ICCBR 2007)*, pages 227–235, 2007.

[176] Ana Paiva, Joao Dias, Daniel Sobral, Ruth Aylett, Polly Sobreperez, Sarah Woods, Carsten Zoll, and Lynne Hall. Caring for agents and agents that care: Building empathic relations with synthetic agents. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 194–201. IEEE Computer Society, 2004.

[177] Mark O Riedl. Incorporating authorial intent into generative narrative systems. In *AAAI Spring Symposium: Intelligent Narrative Technologies II*, pages 91–94, 2009.

[178] Beth Levin. *English verb classes and alternations: A preliminary investigation*. University of Chicago press, 1993.

[179] Tomasz Marciniak and Michael Strube. Beyond the pipeline: Discrete optimization in nlp. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 136–143. Association for Computational Linguistics, 2005.

[180] Hans-Georg Gadamer. Hermeneutics and social science. *Philosophy & Social Criticism*, 2(4): 307–316, 1975.

[181] Olivier Goldschmidt and Alexan Takvorian. An efficient graph planarization two-phase heuristic. *Networks*, 24(2):69–73, 1994.

[182] Frank K Hwang, Dana S Richards, and Pawel Winter. *The Steiner tree problem*, volume 53. North Holland, 1992.

[183] R Tamassia. On Embedding a Graph in the Grid with the Minimum Number of Bends. *SIAM Journal on Computing*, 16(3):421–445, 1987. URL http://epubs.siam.org/doi/abs/10.1137/0216030.

Appendix A: Publications

A.1 Journal Articles

- J. Valls-Vargas, J. Zhu, S. Ontañón (2016). Error Analysis in an Automated Narrative Information Extraction Pipeline. IEEE Transactions on Computational Intelligence and AI in Games.

A.2 Peer-Reviewed Conference Proceedings

- J. Valls-Vargas, J. Zhu and S. Ontañón (2017). Graph Grammar-based Controllable Generation of Puzzles for a Learning Game about Parallel Programming. Proceedings of the Twelfth International Conference on the Foundations of Digital Games.

- J. Valls-Vargas, J. Zhu, S. Ontañón (2016). Predicting Proppian Narrative Functions from Stories in Natural Language. Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment

- J. Valls-Vargas, S. Ontañón, J. Zhu (2015). Exploring Player Trace Segmentation for Dynamic Play Style Prediction. Proceedings of the Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment

- J. Valls-Vargas, J. Zhu, S. Ontañón (2015). Narrative Hermeneutic Circle: Improving Character Role Identification from Natural Language Text via Feedback Loops. Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence

- J. Valls-Vargas, A. Kahl, J. Patterson, G. Muschio, A. Foster, J. Zhu (2015). Designing and Tracking Play Styles in Solving the Incognitum. Proceedings of the Games+Learning+Society Conference.

- J. Valls-Vargas, J. Zhu, S. Ontañón (2014). Toward Automatic Role Identification in Unannotated Folk Tales. Proceedings of the Tenth AAAI Conference on Artificial Intelligence and

Interactive Digital Entertainment.

- J. Valls-Vargas, S. Ontañón, J. Zhu (2013). Towards Story-Based Content Generation: From Plot-Points to Maps. Proceedings of the 2013 IEEE Conference on Computational Intelligence in Games.
- Josep Valls-Vargas and Santiago Ontañón (2012). Natural Language Generation through Case-based Text Modification. Case-Based Reasoning Research and Development.

A.3 Workshops

- J. Valls-Vargas, J. Zhu and S. Ontañón (2017). Towards End-to-end Natural Language Story Generation Systems. Workshops at the Thirty-First AAAI Conference on Artificial Intelligence.
- J. Valls-Vargas, J. Zhu, S. Ontañón (2017). Towards Automatically Extracting Story Graphs from Natural Language Stories. Workshops at the Thirty-First AAAI Conference on Artificial Intelligence.
- J. Valls-Vargas, S. Ontañón, J. Zhu (2014). Toward Automatic Character Identification in Unannotated Narrative Text. Intelligent Narrative Technologies 7 Workshop in Tenth Artificial Intelligence and Interactive Digital Entertainment Conference
- J. Valls-Vargas, S. Ontañón, J. Zhu (2013). Toward Character Role Assignment for Natural Language Stories. Intelligent Narrative Technologies 6 in Ninth Artificial Intelligence and Interactive Digital Entertainment Conference

A.4 Other Peer-Reviewed Publications

- J. Valls-Vargas, J. Zhu, S. Ontañón (2017). From Computational Narrative Analysis to Generation: A Preliminary Review. Proceedings of the 12th International Conference on the Foundations of Digital Games.
- S. Ontañón, J. Zhu, B. Smith, B. Char, E. Freed, A. Furqan, M. Howard, A. Nguyen, J. Patterson, J. Valls-Vargas (2017). Designing Visual Metaphors for an Educational Game

for Parallel Programming. Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems.

- N. Lyon, J. Valls-Vargas, C. Guevara, N. Shao, J. Zhu, J. Zhu (2014). Little Newton: an educational physics game. Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play.

Appendix B: User Study on Connecting Voz+Riu

In this appendix we provide examples of the stories shown to the user participants in our user study on connecting *Voz* and *Riu*. For our user study on analyzing the perceived quality of automatically generated stories we showed the participants small excerpts of a setup or beginning and a continuation for some stories. For each of the setups or beginnings we showed the participants four continuations: first, the original continuation, a continuation generated by *Riu* using *Voz* to automatically extract *Riu*'s input (using the setup or beginning as the query or target), a continuation using hand annotated data to generate *Riu*'s input, and a continuation an LSTM trained on the corpus of stories and samples using the setup or beginning as a seed. Note that the order for these scenarios was randomized in the user study.

Story 1 Setup

Once there lived a rich merchant with two beautiful daughters. The merchant sent them to marry the kingdom's prince.

Original

The younger sister was jealous of her older sister. When her older sister was sleeping, she cut her eyes. She then left her sister to die.

Voz+Riu

The maidservant was jealous. The maidservant stole the merchant's daughter clothes and left her to be eaten by the wolves.

Anno.+Riu

She was jealous. She stole her older sister's clothes and left her older sister to be eaten by the wolves.

LSTM

The father kept his daughter had to the forest. The girl became a bull ate the father didn't take it into a bear.

Story 2 Setup

A woman was once kidnapped by a bear. The woman gave birth to a bearlet who was half-boy and half-bear. Eventually, the woman was able to escape and took the boy with her.

Original

When they got back home, the woman's husband could not accept the bearlet. He had a plan to get rid of the bearlet. He told the bearlet that he would only let him stay if he got rid of the water spirits in the lake. The bearlet left not knowing what to do.

Voz+Riu

The maidservant stole the merchant's daughter clothes and left her to be eaten by the wolves. A beautiful daughter challenged a maidservant to see who could scream the loudest. A maidservant and a beautiful daughter took turns yelling and screaming. A maidservant asked a beautiful daughter to close his eyes. Then, a maidservant smacked a beautiful daughter with a stick.

Anno.+Riu

He had a plan to get rid of the bearlet. He told the bearlet that he would only let him stay if he got rid of the water spirits in the lake. The bearlet left not knowing what to do. A bear ambushed a bearlet but a bearlet was faster than him. A bear saw he could not win and asked a bearlet for forgiveness.

LSTM

From home, the devil started tormenting the princess of a dragon set their house on. A nearby witch smelled the milk of bread and buried him.

Story 3 Setup

A soldier was given leave from his company. He had only a piece of bread, a knapsack and a fiddle. He was happy and left playing the fiddle. A nearby devil heard the fiddle.

The devil approached the soldier and asked him to teach him how to play in exchange for a favor. The soldier taught the devil how to play but did not ask the devil for any favors. They parted ways.

Original

After some time, the devil was tormenting the princess of the kingdom. The king issued a call asking for help. The soldier answered the call and interviewed with the king. He assured the king he would be able to free the princess from her torments.

Voz+Riu

A soldier came the next day but could not find A soldier. A soldier cursed A soldier and transformed him into a cat. Years passed and everyday the mother would sing to A soldier and tell him about his story.

Anno.+Riu

A soldier came the next day but could not find the boy. A soldier cursed the boy and transformed him into a cat. Years passed and A nearby devil would sing to the cat and tell him about his story.

LSTM

Baba Yaga saw them and tried it on fire but then the baby brother and the crops and when he devised a bad winter. The father did not accept the kid to meet with three heads saw them and buried him.

Story 4 Setup

Once upon a time there was a happy queen with a son and a daughter. In the kingdom lived a witch that was jealous of their happiness. The witch devised a plan to ruin their family. The witch visited them. She gave the prince a ring and told him that he would be happy as long as he married a girl who the ring would fit.

Original

Years passed but the boy could not find a girl who the ring would fit. One day, his sister saw the ring and tried it on. The ring fit perfectly and she couldn't take it off. He immediately asked her to marry him. The girl panicked and escaped the palace.

Voz+Riu

The soldier accepted the king's orders and prepared to leave. The king issued a call for help. The king responded to the call. The king told The king he would make The king a nobleman is The king could rescue a bride.

Anno.+Riu

Their father threatened to marry them if he would not find a bride. The girl was terrified. She decided to try to find a bride for her brother.

LSTM

Once kidnapped princess had to get rid of her stepsister despised Alenka was no food. The stepmother.

Story 5 Setup

A poor mother lived in a little house with her two sons. There was famine and she could barely feed her sons. One day a hungry snake walked nearby. The snake looked through the window and saw the younger son unattended.

Original

The snake crawled in and took the younger son with her.

Voz+Riu

Her brother could not find a bride. Their father threatened to marry them if he would not find a bride. The girl was terrified.

Anno.+Riu

The mother saw her younger son missing. She asked her older son to go look for his younger brother. The older brother was very responsible and agreed. He left the house and went into the woods.

LSTM

A dragon kidnapped by the cow grew up he immediately asked her father and threw him. The girl once kidnapped the father and stepmother and escaped the baby brother unattended and threw him into the bull came to eat Vasili.

Story 6 Setup

An old couple lived in a hut with their adopted son. The boy had a magical bird that could control the weather. They used the bird's predictions to make it rain on their crops and were always prosperous. A jealous neighbor learned about the bird. The neighbor told the boy he needed to borrow the bird. But the boy wouldn't let him take it.

Original

The neighbor took the bird and locked it into a cage. A mouse nearby saw everything.

Voz+Riu

The mouse came the next day but could not find the mouse. The mouse cursed the mouse and transformed him into a cat.

Anno.+Riu

A jealous neighbor came the next day but could not find the mouse. A jealous neighbor cursed the mouse and transformed him into a cat.

LSTM

The witch grabbed the family's house and their fields. A wicked neighbor took over the bull came to eat every single child if his town.

Story 7 Setup

Once upon a time, in a land, far, far away. There lived a girl with her baby brother and her mother. One day, the mother went to the market. The mother told the girl to watch her brother. The girl started playing and soon forgot what her mother had asked her.

Original

A nearby witch smelled the baby brother and took him away.

Voz+Riu After a while, the girl came back and realized her baby brother was missing. The girl suspected the witch may be behind.

Anno.+Riu

The witch came the next day but could not find the girl. The witch cursed the girl and transformed him into a cat.

LSTM

From home, the old mother hid the merchant's daughter had been ruined and her father kept his daughter Alenka.

Story 8 Setup

There once lived a poor and unlucky peasant.

Original

He never married and was very lonely.

Voz+Riu He wanted revenge and ran after the dragon but got lost.

Anno.+Riu A poor and unlucky peasant wanted revenge and ran after the dragon but got lost.

LSTM

Time passed but could not find a suitable husband could not find her room. The family and ran away.

Story 9 Setup

An old couple lived with their daughter in a farm. The mother died of illness. The widower eventually married a widow that had a daughter of her own. The stepmother was always cruel to the poor girl. One night a group of pirates came to the farm. They spied the family through the windows.

Original

Once the family was asleep the pirates kidnapped the father and stepmother. The next morning, the girl and her stepsister couldn't find their parents. They decided to go look for them. They took separate ways.

Voz+Riu

The woman's daughter was jealous of her stepsister's beauty. She wanted to learn her secret. The woman's daughter followed the girl everyday. She learned that she drank the milk of a magic cow to keep beautiful. She plotted to steal the cow's milk all for herself.

Anno.+Riu

The king kidnapped the father and threw him in jail.

LSTM

The king's second birthday, she cut her in jail. The man forgot about his lover.

Appendix C: Narrative Functions Identified by Propp

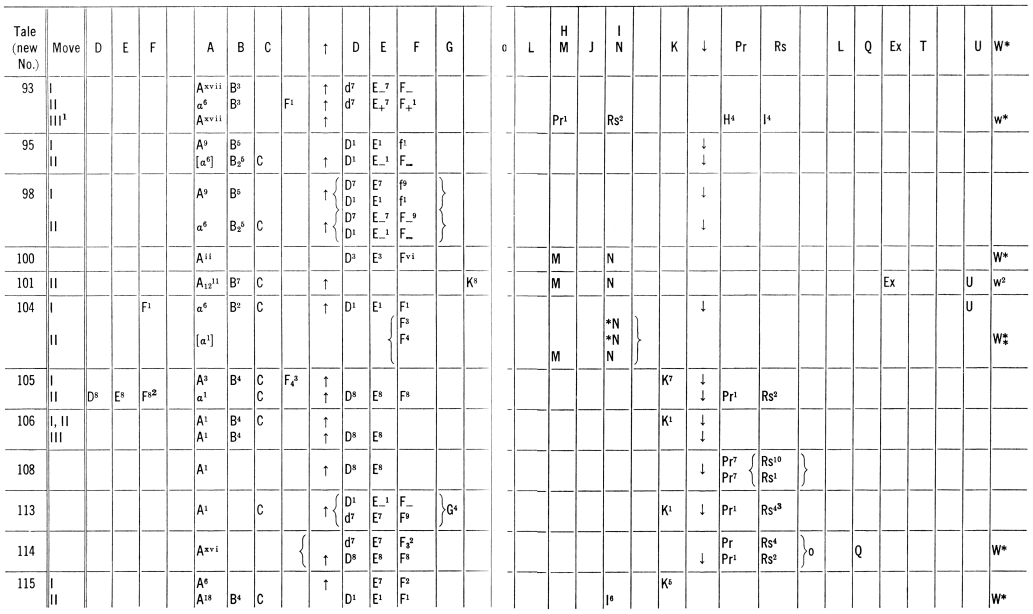
In this appendix we include some of the tables published in the English translation of Morphology of the Folktale[25]. These tables show the narrative functions identified by Propp for several of the stories he studied. The stories are identified using the number found in the collections published by Alexander Afanasyev. Please refer to Chapter 3 for the titles of the stories used in our experimental evaluation.

C.1 Table of Narrative Functions Identified by Propp

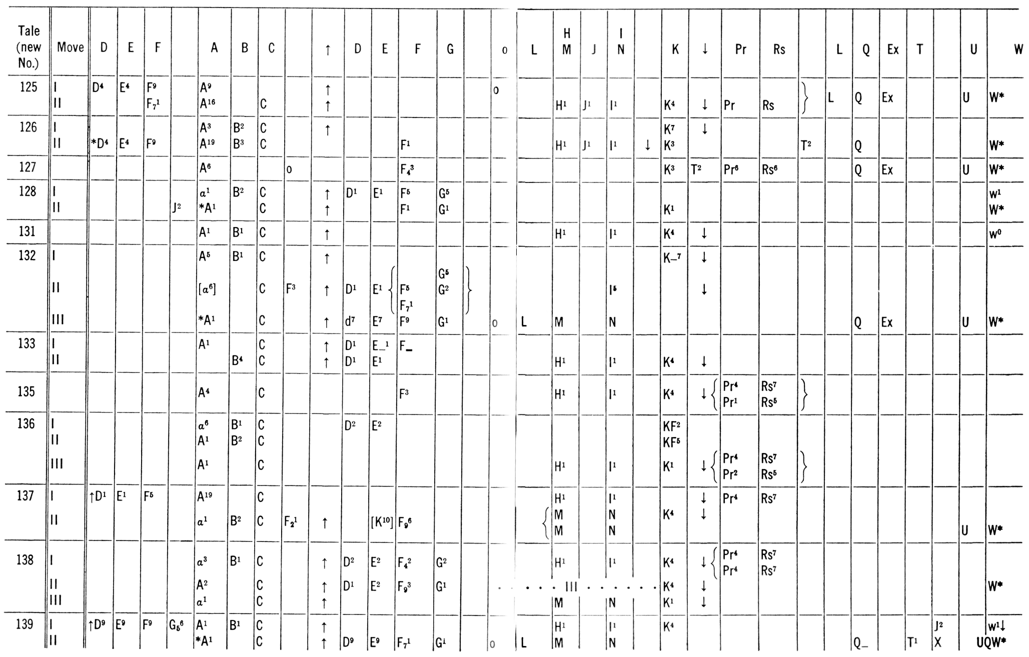
Below we include the narrative functions identified by Propp in the stories he analyzed. We include this information for reference. These tables are scanned from the appendix in the English translation of Morphology of the Folktale[25].

The 1961 Report of the Register of Copyrights on the General Revision of the U.S. Copyright Law specifically states that the fair use of a copyrighted work, including such use by reproduction for purposes such as scholarship or research, is covered by fair use and not an infringement of copyright.

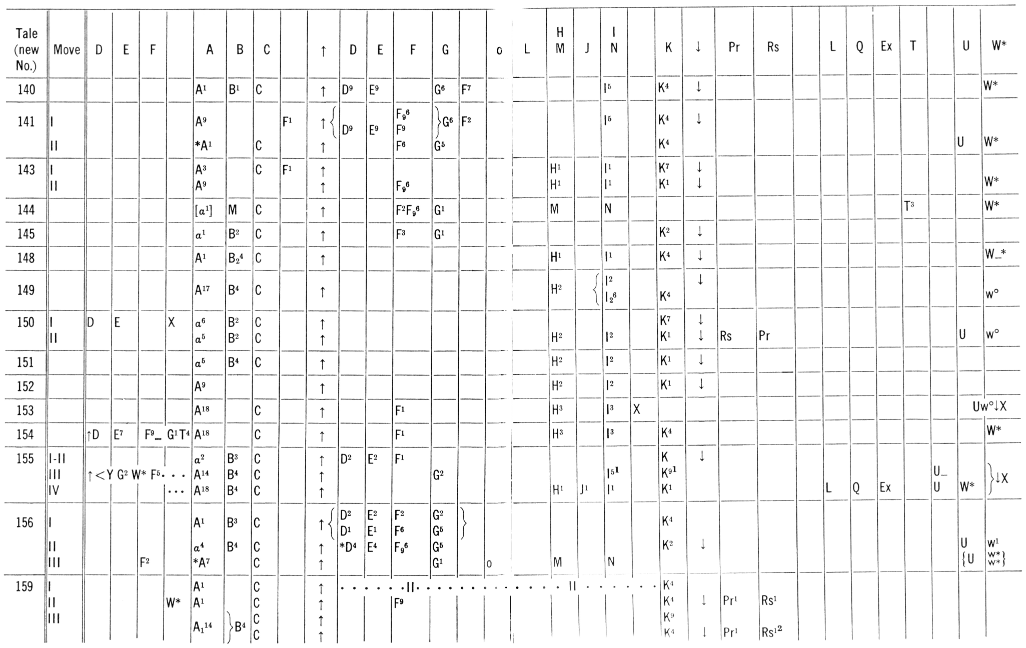
XX UNCOMMENT THESE FIGURES



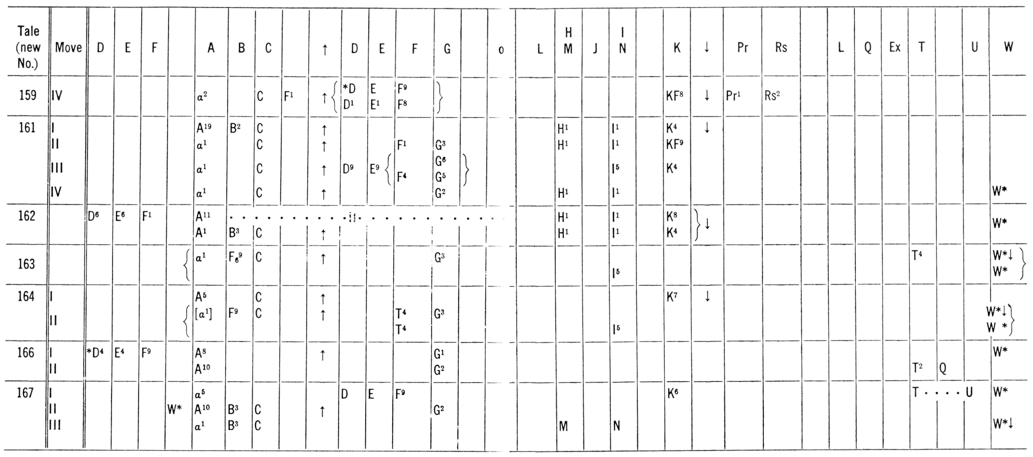
Tale (new No.)	Move	D	E	F		A	B	C		↑	D	E	F	G		o	L	H M	J	I N		K	↓	Pr	Rs		L	Q	Ex	T		U	W*
93	I					A^{xvii}	B^3			↑	d^7	E_-^7	F_-																				
	II					α^6	B^3		F^1	↑	d^7	E_+^7	F_+^1																				
	III^1					A^{xvii}				↑								Pr^1		Rs^2				H^4	I^4								w*
95	I					A^9	B^5				D^1	E^1	f^1										↓										
	II					$[\alpha^6]$	B_2^5	C		↑	D^1	E_-^1	$F_=$										↓										
98	I					A^9	B^5			↑	D^7 D^1	E^7 E^1	f^9 f^1										↓										
	II					α^6	B_2^5	C		↑	D^7 D^1	E_-^7 E_-^1	F_-^9 $F_=$										↓										
100						A^{ii}					D^3	E^3	F^{vi}					M		N													W*
101	II					A_{12}^{11}	B^7	C		↑					K^8			M		N									Ex			U	w^2
104	I			F^1		α^6	B^2	C		↑	D^1	E^1	F^1										↓									U	
	II					$[\alpha^1]$							F^3 F^4					M		*N *N N													W^*_*
105	I					A^3	B^4	C	F_4^3	↑												K^7	↓										
	II	D^8	E^8	$F^{8\,2}$		α^1		C		↑	D^8	E^8	F^8										↓	Pr^1	Rs^2								
106	I, II					A^1	B^4	C		↑												K^1	↓										
	III					A^1	B^4			↑	D^8	E^8											↓										
108						A^1				↑	D^8	E^8											↓	Pr^7 Pr^7	Rs^{10} Rs^1								
113						A^1		C		↑	D^1 d^7	E_-^1 E^7	F_- F^9	G^4								K^1	↓	Pr^1	$Rs^{4\,3}$								
114						A^{xvi}				↑	d^7 D^8	E^7 E^8	F_3^2 F^8										↓	Pr Pr^1	Rs^4 Rs^2	o		Q					W*
115	I					A^6				↑		E^7	F^2									K^5											
	II					A^{18}	B^4	C			D^1	E^1	F^1							I^6													W*



Tale (new No.)	Move	D	E	F		A	B	C		↑	D	E	F	G		o	L	H M	J	I N		K	↓	Pr	Rs		L	Q	Ex	T		U	W
125	I	D^4	E^4	F^9		A^9				↑						o																	
	II			F_7^1		A^{16}		C		↑								H^1	J^1	I^1		K^4	↓	Pr	Rs	}	L	Q	Ex			U	W*
126	I					A^3	B^2	C		↑												K^7	↓										
	II	*D^4	E^4	F^9		A^{19}	B^3	C					F^1					H^1	J^1	I^1	↓	K^3				T^2		Q					W*
127						A^6			o				F_4^3									K^3	T^2	Pr^6	Rs^6			Q	Ex			U	W*
128	I					α^1	B^2	C		↑	D^1	E^1	F^5	G^5																			w^1
	II				J^2	*A^1		C		↑			F^1	G^1								K^1											W*
131						A^1	B^1	C		↑								H^1		I^1		K^4	↓										w^0
132	I					A^5	B^1	C		↑												K_{-}^7	↓										
														G^5																			
	II					[α^6]		C	F^3	↑	D^1	E^1 {	F^5	G^2	}					I^5			↓										
													F_7^1																				
	III					*A^1		C		↑	d^7	E^7	F^9	G^1		o	L	M		N								Q	Ex			U	W*
133	I					A^1		C		↑	D^1	E_{-}^1	F_{-}																				
	II						B^4	C		↑	D^1	E^1						H^1		I^1		K^4	↓										
135						A^4		C					F^3					H^1		I^1		K^4	↓ {	Pr^4	Rs^7	}							
																								Pr^1	Rs^5								
136	I					α^6	B^1	C			D^2	E^2										KF^2											
	II					A^1	B^2	C														KF^5											
	III					A^1		C										H^1		I^1		K^1	↓ {	Pr^4	Rs^7	}							
																								Pr^2	Rs^5								
137	I	↑D^1	E^1	F^5		A^{19}		C										H^1		I^1			↓	Pr^4	Rs^7								
	II					α^1	B^2	C	F_2^1	↑		[K^{10}]	F_9^6					{ M		N		K^4	↓										
																		M		N												U	W*
138	I					α^3	B^1	C		↑	D^2	E^2	F_4^2	G^2				H^1		I^1		K^4	↓ {	Pr^4	Rs^7								
																								Pr^4	Rs^7								
	II					A^2		C		↑	D^1	E^2	F_9^3	G^1		· ·	· ·	· III ·	· ·	· ·	· ·	K^4	↓										W*
	III					α^1		C		↑								M		N		K^1	↓										
139	I	↑D^9	E^9	F^9	G_5^6	A^1	B^1	C		↑								H^1		I^1		K^4									J^2		w^1↓
	II					*A^1		C		↑	D^9	E^9	F_7^1	G^1		o	L	M		N								Q_{-}		T^1	X		UQW*



Tale (new No.)	Move	D	E	F		A	B	C		↑	D	E	F	G		o	L	H M	J	I N		K	↓	Pr	Rs		L	Q	Ex	T		U	W*
140						A^1	B^1	C		↑	D^9	E^9		G^6	F^7					I^5		K^4	↓										W*
141	I					A^9			F^1	↑ {	D^9	E^9	F_9^6	} G^6	F^2					I^5		K^4	↓										
													F^9																				
	II					*A^1		C		↑			F^6	G^5								K^4										U	W*
143	I					A^3		C	F^1	↑								H^1		I^1		K^7	↓										
	II					A^9				↑			F_9^6					H^1		I^1		K^1	↓										W*
144						[α^1]	M	C		↑			$F^2F_9^6$	G^1				M		N										T^3			W*
145						α^1	B^2	C		↑			F^3	G^1								K^2	↓										
148						A^1	B_2^4	C		↑								H^1		I^1		K^4	↓										W_{-}*
149						A^{17}	B^4	C		↑								H^2	{	I^2			↓										
																				I_2^6		K^4											w°
150	I	D	E		X	α^6	B^2	C		↑												K^7	↓										
	II					α^5	B^2	C		↑								H^2		I^2		K^1	↓	Rs	Pr							U	w°
151						α^5	B^4	C		↑								H^2		I^2		K^1	↓										
152						A^9				↑								H^2		I^2		K^1	↓										
153						A^{18}		C		↑			F^1					H^3		I^3	X												Uw°↓X
154		↑D	E^7	$F_{=}^9$	G^1T^4	A^{18}		C		↑			F^1					H^3		I^3		K^4											W*
155	I-II					α^2	B^3	C		↑	D^2	E^2	F^1									K	↓										
	III	↑<Y	G^2	W*	F^5· · ·	A^{14}	B^4	C		↑				G^2						I^{51}		K^{91}									U_{-}		} ↓X
	IV				· · ·	A^{18}	B^4	C		↑								H^1	J^1	I^1		K^1					L	Q	Ex		U	W*	
156	I					A^1	B^3	C		↑ {	D^2	E^2	F^2	G^2	}							K^4											
											D^1	E^1	F^6	G^5																			
	II					α^4	B^4	C		↑	*D^4	E^4	F_9^6	G^5								K^2	↓									U	w^1
	III			F^2		*A^7		C		↑				G^1		o		M		N												{U	w* w*}
159	I					A^1		C		↑	· ·	· ·	· II ·	· ·	· ·	· ·	· ·	· ·	· II ·	· ·	· ·	K^4											
	II				W*	A^1		C		↑			F^9									K^4	↓	Pr^1	Rs^1								
	III					A_1^{14}	} B^4	C		↑												K^9											
								C		↑												K^4	↓	Pr^1	Rs^{12}								



Tale (new No.)	Move	D	E	F		A	B	C		↑	D	E	F	G		o	L	H M	J	I N		K	↓	Pr	Rs		L	Q	Ex	T		U	W
159	IV					α^2		C	F^1	↑	*D D^1	E E^1	F^9 F^8									KF^8	↓	Pr^1	Rs^2								
161	I					A^{19}	B^2	C		↑								H^1		I^1		K^4	↓										
	II					α^1		C		↑			F^1	G^3				H^1		I^1		KF^9											
	III					α^1		C		↑	D^9	E^9	F^4	G^6 G^5						I^5		K^4											
	IV					α^1		C		↑				G^2				H^1		I^1													W*
162		D^6	E^6	F^1		A^{11}	· · ·	· · ·	· · ·	· · ·	· ⁞⁞ ·	· · ·	· · ·	· · ·	· · ·	· · ·		H^1		I^1		K^8	↓										W*
						A^1	B^3	C		↑								H^1		I^1		K^4											
163						α^1	F_6^9	C		↑				G^3						I^5										T^4			W*↓ W*
164	I					A^5		C		↑												K^7	↓										
	II					[α^1]	F^9	C		↑			T^4 T^4	G^3						I^5													W*↓ W *
166	I	*D^4	E^4	F^9		A^8				↑				G^1																			W*
	II					A^{10}								G^2																T^2	Q		
167	I					α^5					D	E	F^9									K^6								T · ·	· ·	U	W*
	II				W*	A^{10}	B^3	C		↑				G^2																			
	III					α^1	B^3	C										M		N													W*↓

Appendix D: Alexander Afanasyev's Stories

During our research we compiled several sources of Russian and Slavic folk tales, matching available English translations with the original stories referred to by Propp required some work that we are glad to be able to share with the community.

D.1 Relationship of Alexander Afanasyev's Story Titles

The following table reports a relationship of the numbered stories used in Propp's work and the English titles of the stories used in Jack V. Haney translation *The Complete Folktales of A. N. Afanas?ev.*, Univ. Press of Mississippi, 2014[1]. The ranges in the first column indicate several versions or variations of the same story. Additionally, we include other titles used in previous translations and collections. For those we look at publications freely available on Project Gutenberg. In the table the numbers in the *Sources* column refer to the following titles:

1. Folk Tales from the Russian by various authors[2]
2. Russian Fairy Tales From the Skazki of Polevoi by R. Nisbet Bain[3]
3. Russian Fairy Tales A Choice Collection of Muscovite Folk-lore by W. R. S. Ralsto[4]
4. Old Peter's Russian Tales by Arthur Ransome[5]

Table D.1: Relationship of numbered Alexander Afanasyev's stories used in Propp's work and their titles. The last column provides correspond to freely available sources from Project Gutenberg.

#	Title	Alternative Titles	Sources
1-7	Little Sister Fox and the Wolf		

[1]Preview available: https://books.google.com/books?id=IQQbBwAAQBAJ
[2]Available: http://www.gutenberg.org/files/12851/12851-h/12851-h.htm
[3]Available: http://www.gutenberg.org/files/34705/34705-h/34705-h.htm
[4]Available: https://www.gutenberg.org/files/22373/22373-h/22373-h.htm
[5]Available: http://www.gutenberg.org/files/16981/16981-h/16981-h.htm

8	For a Bast Boot a Hen-For the Hen, a Goose		
9-13	The Fox Midwife		
14	The Fox, the Hare, and the Cock		
15-17	The Fox Confessor		
18	The Fox Healer		3
19	An Old Man Climbs Up to the Sky		
20	The Old Man in Heaven		
21-22	The Fox as Keener		
23-26	The Peasant, the Bear, and the Fox		
27	Old Hospitality is Soon Forgotten		
28	The Sheep the Fox and the Wolf		
29-30	The Beasts in the Pit		
31	The Fox and the Grouse		
32	The Fox and the Woodpecker		
33	The Fox and the Crane		
34	Snow Maiden and the Fox		
35	The Fox and the Crayfish		
36	The Bun		
37-39	The Tomcat, the Cock, and the Fox		
40-43	The Tomcat and the Fox		
44-47	The Frightened Bear and Wolves		
48	The Bear, the Fox, the Gadfly, and the Peasant		
49-50	The Wolf		
51-52	The Sow and the Wolf		
53-54	The Wolf and the Goat		

55-56	The Wolf is a Fool		
57-58	The Bear		
59	The Bear, the Dog, and the Cat		
60-61	The Goat		
62	A Tale about a Shedding Goat		
63	A Tale about a Certain One-Sided Ram		
64	The Beasts' Winter Quarters		
65	The Bear and the Cock		
66-67	The Doc and the Woodpecker		
68	The Cock and the Hen		
69	The Death of the Cock		
70-71	The Hen		
72	The Crane and the Heron		
73	The Crow and the Lobster		
74	The Eagle and the Crow		
75	The Gold Fish		4
76	The Greedy Old Woman		
77-80	The Tale of Ersh Ershovich, Bristleback's Son		
81	A Story about a Toothsome Pike		
82-84	The Tower of the Fly		
85-86	The Spider		
87-88	The Bubble, the Straw, and the Bast Boot		
89	The Turnip		
90	Mushrooms		
91	Frost, Sun, and Wind		
92	Sun, Moon, and Raven Ravenson		

93	The Witch and the Sun's Sister	Witch and Solntseva Sister, Prince Ivan, the Witch Baby, and the Little Sister of the Sun	3, 4
94	The Vazuza and the Volga		4
95-96	Frost	Jack Frost, Morozko	1, 2, 3, 4
97	The Old Woman Who Griped	The old woman Govorukha	
98	Daughter and Stepdaughter	Daughter and Daughter in law	
99	The Mare's Head		
100	Little Bitty Khavroshechka	Kroshechka-Havroshechka	
101	The Little Red Cow	Burenushka	
102-103	Baba Yaga		1, 3, 4
104	Vasilisa the Beautiful		3
105	Baba Yaga and the Midget		
106-107	Baba Yaga and the Nimble Youth		
108-101	Ivashko and the Witch	The Witch	
112	Tereshechka		
113	The Swan-Geese	The Magic Swan Geese	
114	Prince Danila Govorila		

115-122	The Truth and the Lie		
123-124	The Prince and His Uncle		
125	Ivan Tsarevich and Marfa Tsarevna		
126	The Little Copper Man		
127	The Merchant's Daughter and Her Maid	The Merchant's Daughter and the Maidservant	
128-130	The Three Tsardoms-Copper, Silver, and Gold		
131	Frolka the Dropout	Frolka Stay-at-Home	
132	The Norka Beast		
133-134	Rollingpea		
135	Ivan Popialov	Ivan Popyalov	3
136	Stonn-Bogatyr, Ivan the Cow's Son		
137	Ivan the Bull's Son		
138	Ivan the Peasant's Son and the Little Man the Size of a Finger with Moustaches Seven Versts Long		2
139	Ivan Suchenko and the Belyi Polianin		
140	Dawn, Evening, and Midnight	Dawn, Evening and Midnight	4
141-142	The Bear, Moustaches, Mountain Man, and Oakman Bogatyrs		
143	Nodei, the Priest's Grandson		
144	The Flying Ship		4
145-147	The Seven Simeons		1

148	Nikita the Tanner		
149	The Serpent and the Gypsy		
150	The Hired Hand		
151	Shabarsha	Shabarsha the Laborer	
152	Ivan the Bear's Son	Ivanko the Bear's Son	
153	A Soldier Rescues a Tsarevna		
154	A Fugitive Soldier and the Devil	The Runaway Soldier and the Devil	
155	The Two Ivans, a Soldier's Sons		2
156-158	Koshchei the Deathless		3
159	Mar'ia Morevna		
160	Fedor Tugarin and Anastasia the Beautiful		
161	Ivan Tsarevich and the Belyi Polianin		
162	The Crystal Mountain		
163	Bukhtan Bukhtanovich		
164	Kozma Quickrich		
165-166	Emelia the Fool		3
167	At the Pike's Command		
168	The Tale about Ivan Tsarevich, the Firebird, and the Gray Wolf		
169-170	The Firebird and Vasilisa Tsarevna		4
171-178	A Tale about the Brave Lad, the Rejuvenating Apples, and the Living Water		
247	The language of the Birds		1

297	The princess who wouldn't smile		
557	Girl and Bear		

Appendix E: Story-based Procedural Content Generation

This Appendix describes some related work performed during my dissertation, which was not directly connected to my core contributions

E.1 Story-based Procedural Content Generation

Procedural content generation (PCG) refers to the creation of content automatically through algorithmic means[97]. Some computer game genres require meaningful stories and complex worlds in order to successfully engage players. However, most procedural map generation techniques typically neglect the role of the story in the construction of the map. In this work, we present a novel PCG approach for generating maps intended to be played in a computer game context that uses stories to guide the generation process.

Our approach is motivated by the following observations: First, procedural content generation has the potential to increase the variability and replayability of games. This in turn can lead to an increase in player interest in these games, as it will take longer for the player to see or complete everything in the game[98]. Second, game maps impose constraints to the set of possible stories that may happen in a game. For example, the order of narrative events should not be at odds with the spatial configuration of the map. For example, we may not want a player of a murder mystery game to leave the current room before the next critical narrative event can be triggered.

In this work, we intended to explore the correlation between stories, represented as sequences of plot points, and the spatial configuration of the environment where these stories unfold. Our goal is to procedurally generate story-based game maps. We propose a framework which, given the specification of a *story space*, represented as a collection of *plot points* and their dependencies, can generate maps that support one or more stories from that story space. Our system searches in the space of possible spatial configurations of the map, determining the set of stories that can unfold in each of those configurations. Then, using automatic story evaluation techniques, it determines the narrative quality of each possible story, the combination of which determines the overall quality of the map. Finally, the map is *realized* into a graphical representation.

The key technical challenges that we address here are: 1) how to automatically generate game maps from a collection of plot points, 2) how to evaluate the quality of a given map based on the different stories that it supports, and 3) how to generate a graphical realization of the map.

Story-Based Map Generation

In this work, we focus on the problem of story-based map generation. In order to address this problem, we present a system that combines story-telling and procedural map generation techniques. Specifically, our system takes as input a collection of *plot points*, which are events that are key to the game story (for example: "the player discovers the existence of a hidden door"). This collection of plot points defines the *story space* (e.g., the space of all the potential stories we could generate). The goal of our map generation system is to generate game maps that support a subset of stories from the story space which are of high-quality based on our given storytelling criteria.

Our plot point representation is based on those used in planning-based story generation systems, such as *ASD*[87]. It is extended in order to allow the use of story quality evaluation functions, such as those developed by Weyhrauch for *MOE*[22]. Additionally, we take as input a set of author goals that stories need to accomplish[87]. Specifically, the input of our system is a tuple $\langle S, L, player, I, G, P\rangle$, where:

- S is a list of symbols (e.g., *tree*, *sword*, *cave*, *warrior*, *use*, *path*, etc.).
- $L \subseteq S$ is the subset of those symbols that represent location names (e.g., *cave*).
- $player \in S$ is the symbol that identifies the player.
- I is the initial state, consisting of a list of positive literals of the form $o(s_1, ..., s_n)$, where $o \in S$, and $\forall_{i=1...n} s_i \in S$. The initial state can include information about NPCs, objects, and locations that are referenced by the plot points (e.g., $at(path, warrior)$, $has(warrior, sword)$...).
- G is the set of goals that need to be achieved to complete the game, represented as a list of literals.

- P is a set of plot points. A plot point is defined as a tuple $p = \langle \kappa, \alpha, \delta, A \rangle$ (in a very similar way to how planning operators are defined), where:
  - κ represents the preconditions of the plot point, as a list of (potentially negated) literals.
  - α (or the *add* list) are the literals that will become true after this plot point occurs.
  - δ (or the *delete* list) are the literals that will become false after this plot point occurs.
  - A is a list of additional annotations we might need for this plot point. In our particular implementation, we annotated each plot point with the set of features required to use a story evaluation function inspired by the work of Weyhrauch[22] (which we elaborate later in Section E.1).

Given the input, our system generates, as output, a graphically realized map based on a rectangular grid. Each cell of the grid is annotated with a location symbol in L, the objects or characters that should initially be in that location, and which of the four neighboring cells the current cell is connected to.

Internally, our system uses an abstract intermediate representation of maps before they are graphically realized. We call this representation a *spatial configuration* graph. Formally, a spatial configuration is defined as a graph $E = \langle L, K \rangle$, where the nodes are the locations L and $K \subseteq L \times L$ are the edges between those nodes. An edge connecting two nodes represents that the locations are accessible from each other. Later, we transform the spatial configuration into a planar, grid-based, orthogonal graph (we call this the *graph embedding* step). Once embedded, the graph can be realized as a playable game map (we call this the *graphical realization* step).

The main steps of the process can be summarized as follows: First, our system generates spatial configurations. Then, the system generates all the possible stories that can unfold in a given spatial configuration, based on the set of input plot points. After that, each spatial configuration is evaluated by assessing the narrative quality of all possible stories that could unfold in it, and finally the system graphically realizes one of the spatial configurations.

The entire process takes into account three different spaces: 1) the *spatial configuration space* (which locations are connected to each other), 2) the *story space* (sequences of plot points), and 3) the *graph embedding space* (planar graph embeddings which will become graphically realized game maps, into rectangular grids in our experiments).

The input of the system defines a set of symbols L that represent locations. However, their spatial relationships (from which ones characters can reach others) are not specified in the input of our system. The first step described in Section E.1 assumes as input a spatial configuration graph $E = \langle L, K \rangle$, describing which locations are accessible from one another. The graph may be initialized randomly. Section E.1 describes the ϵ-greedy algorithm that uses the evaluation of the graph E from Section E.1 to refine the construction of E for the next iteration.

Planning and Story Generation

From a given spatial configuration graph E and a given input $\langle S, L, \mathit{player}, I, G, P \rangle$, we want to know what stories can unfold in E. We perform an exhaustive search to generate all possible stories. We are interested in identifying spatial configurations that feature low quality stories (e.g., short circuited or without any challenges) or no stories at all (e.g., no way to reach any goal) so that can be discarded.

Our system uses a forward chaining planner in order to generate stories. The given plot points are used as the planning operators. A story consists of the list of plot points generated by the planner for reaching the goals G from the initial state. The initial state used by our planner to generate stories consists of the initial state I expanded with one literal $\mathit{path}(l_1, l_2)$, for each edge (l_1, l_2) that exists in E.

Moreover, we are interested in obtaining the set of *all* possible stories that can unfold in a given spatial configuration. In order to obtain such set, our planner performs breadth-first-search. When a state is reached where all the goals in G have been satisfied, the operator history is saved in the set of solution candidates $\mathbb{S}_E$ for the current spatial configuration graph. Then the planning process continues with the next branch. If it is not possible to find operators that add new literals to the state, the search along a branch will be terminated even if there are goals remaining, therefore preventing loops that may contribute to solutions of infinite length. Moreover, our planner gives a special treatment to movement plot points and adds a literal ($been(character, l)$) after each movement action in order to allow different locations to be explored and revisited but preventing duplicated search states (e.g., visiting a bird before and after having acquired bird food). Also, some of the literals in the state are dealt with in a special way by some modules in our system:

- $at(object, l)$: Specifies object or character locations in the spatial environment.
- $path(l_1, l_2)$: Specifies whether a path exists from a location to another.
- $locked(character, l_1, l_2)$: Specifies whether a path exists but is currently locked for a character.

The output of the planning module is a set of plans $\mathbb{S}_E$ representing all the stories found in the current spatial configuration. Each story in $\mathbb{S}_E$ consists of an ordered list of pairs $[\langle s_1, p_1\rangle, ...\langle s_m, p_m\rangle]$ with the states s_i and the executed plot points $p_i \in P$. More complex planners can be used, but our current breadth-first forward-checking planner suffices for the purposes of our experiments. Also, for large story spaces, it might not be feasible to generate the complete set of all stories that can unfold in a given spatial configuration, and we might need to sample it. Sampling should suffice as long as we can identify the lowest quality stories, which in our experiments are correlated with the shorter stories (e.g., when short circuiting the initial plot point with some goal). This idea is part of our future work.

Figure E.1 shows an example plot point dependency graph. The story is inspired by plots produced by Tale-Spin[101]. Some of the preconditions have been removed for display. Figure E.2 shows a specific story for the plot point dependency graph shown in Figure E.1. The main character in this story is *Joe Bear*, who starts at the location *cave*. The numbers in the edges in Figure E.2 represent the order in which the plot points occur.

Story Evaluation

An important component of our system is the story evaluation which has the goal of computing a numeric value representing the quality of a story according to some predefined criteria to a given story $[\langle s_1, p_1\rangle, ...\langle s_m, p_m\rangle]$.

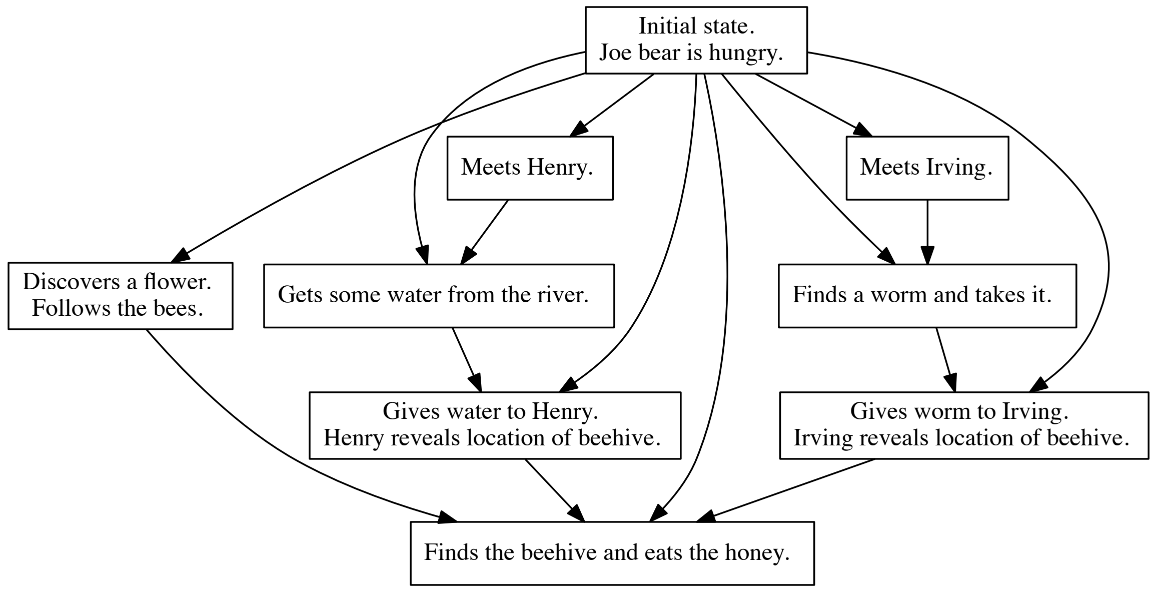

Initial state.
Joe bear is hungry.
Meets Henry.
Meets Irving.
Discovers a flower.
Follows the bees.
Gets some water from the river.
Finds a worm and takes it.
Gives water to Henry.
Henry reveals location of beehive.
Gives worm to Irving.
Irving reveals location of beehive.
Finds the beehive and eats the honey.

Figure E.1: Example story causal link graph inspired by the plots produced by TALE-SPIN. Boxes represent plot points and arrows precondition/postcondition dependencies.

The evaluation function used in our experiments, draws from the fields of experience and drama management, specifically from the work of Weyhrauch[22]. Our evaluation function is a weighted sum of a set of simpler normalized feature values. The selected features try to evaluate certain logical and aesthetic properties that characterize story quality from a storytelling point of view:

- **Thought flow**: Measures whether one event in the player's experience relates logically to the next. Each plot point may be annotated with a *thought tag* grouping logically related events (e.g., visiting the bird and acquiring bird food share one tag whereas seeing a flower and following some bees may have different tags). We implemented this feature value counting the total number of distinct thought groups in a given story normalized by the number of transitions between distinct thought groups.

- **Activity flow**: Measures boredom of the player caused by moving without unfolding new events of the story. We measure the activity flow by the total number of distinct visited locations normalized by the total number of transitions between locations.

- **Manipulation**: Measures how manipulated the player feels by the limit of meaningful choices

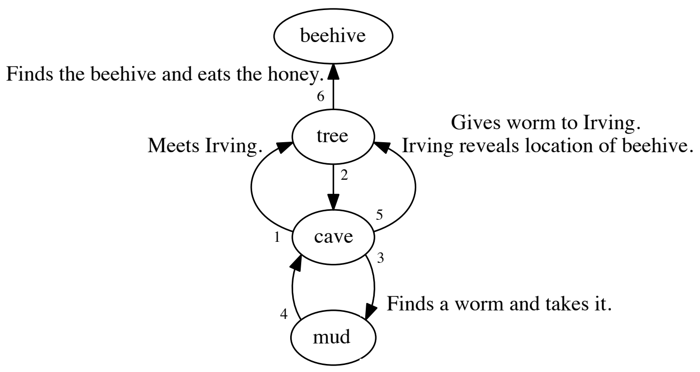

beehive
Finds the beehive and eats the honey.
6
tree
Meets Irving.
Gives worm to Irving.
Irving reveals location of beehive.
2
5
1
cave
3
Finds a worm and takes it.
4
mud

Figure E.2: Example story planned from the story space described by Figure E.1. Nodes represent locations, edges represent movement between locations and are annotated with the ordered plot points describing a story.

available. We measure the manipulation by computing the average of the number of available operators before executing each plot point in the story normalizing by the total number of operators.

- **Intensity**: Measures the player's excitement built as the story unfolds. Each plot point may be annotated with a numeric *intensity value* describing the author's intended intensity for an event. We measure the difference between a prototype dramatic arc and the dramatic arc represented by the sequence of intensities annotated in the current story. Any particular desired dramatic arc shape could be specified. In our experiments we approximate an Aristotelian arc by $f(x) = \sin(\pi x^2)$ where x is in the interval $[0, 1]$.

- **Action consistency**: Measures the player perceived logic in a sequence of actions by penalizing redundant or complementary occurrences. Each plot point may be annotated with one or more *complementary thought tags* describing logically incompatible events to the current plot point. We measure the number of violations normalized by the total number of distinct thoughts.

- **Length**: Measures how much the story length deviates from a desired story length. We implemented this feature value by comparing the current story length to a predefined value (part of the input annotations). We return a value interpolated linearly from 1.0 when the lengths match exactly to 0.0 when the current story length is ±75% or more.

Finally the individual values are added together using a weighted sum. In our implementation we defined some arbitrary weights representing our desired aesthetic values. The values used in our experiments are $0.1, 0.05, 0.05, 0.1, 0.1$ and 0.6 respectively, for each of the individual features defined previously.

Spatial Configuration Evaluation

Given the set of stories $\mathbb{S}_E$ that can unfold in a spatial configuration E (computed by our planner), and the evaluation of each of the stories in $\mathbb{S}_E$, we are interested in aggregating the evaluations and compute a numeric value representing the quality of the spatial configuration E.

In order to give a numerical value to the list of individual story evaluations we experimented with several aggregation operators with different properties and we selected the following:

- Average: We average the list of individual evaluations.
- Minimum: We compute the minimum of the list of individual evaluations (which identifies the worst story).

Graph Embedding

Once a spatial configuration E has been selected (we describe how the previous processes are combined to select a spatial configuration in Section E.1), we need to turn the graph E into a planar graph (a graph that can be drawn in a plane without having intersecting edges), so that a final map can be realized. We call this output a *graph embedding* $\mathbb{Y}_E$.

There is an infinite number of drawings or plane embeddings for a graph. When embedding a spatial configuration E, we would like to take into account a variety of factors. First, allowing two edges to cross, will result in the creation of additional paths, not in E, when the spatial configuration is realized, allowing the player to access locations that should not be readily accessible. Even though those can be tackled with multi-level environments or the use of certain elements, those may not make sense in some story domains (like teleports in a historical setting) or scales (like multi-level continental maps). Thus, in order to provide a story-agnostic graph handling we decided to enforce planar embeddings.

When embedding E into a planar graph, we consider edges to be undirected since we are assuming that locations will be accessible from each other when connected.

Several hierarchical and force-driven (e.g., rubber bands) algorithms have been discovered for finding planar embeddings, but those may not be suitable for our purpose since we impose several aesthetic and playability constrains. Thus, we decided to use a rectangular grid layout for our graph embedding step (although our algorithm would work for arbitrary reticules, such as hexagonal patterns). We employ a technique inspired by Goldschmidt and Takvorian[181].

Before embedding the graph, two preprocessing steps are executed in order to improve the chances of finding a planar embedding:

1. Inspired by the idea of the *Steiner points* in the *Steiner tree problem*[182], the graph is searched for cliques of size >= 4. If any is found, the edges between the nodes are removed and replaced by vertices to a new intermediate node.

2. In order to enable orthogonalization on a rectangular grid, nodes with a vertex degree > 4 (vertices with more than four edges) are split into several nodes with vertex degree of at most 4 following the *Giotto* approach presented by Tamassia[183].

Then, the graph is processed as follows:

1. Select a subpath from the graph, lay down the nodes consecutively on the grid and mark the nodes as processed. The selected subpath is initially the longest of all the shortest paths in the graph.

2. Select an unprocessed node (if any) that has only one neighbor already marked processed, lay it in the grid adjunct to its neighbor and mark it as processed. The neighbor position is selected from the first available position searching clockwise.

3. Select an unprocessed node (if any) and search for paths between all the neighbors already marked as processed. Select the shortest path and split the node as many items as necessary

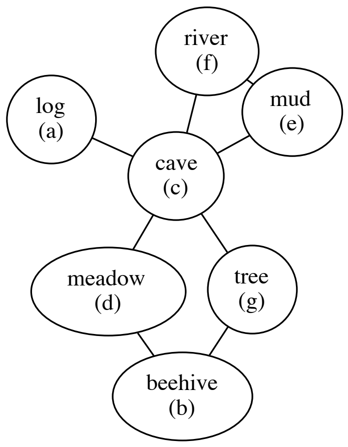


Figure E.3: Example environment configuration for an environment inspired by the plots produced by TALE-SPIN (input A in our experiments).

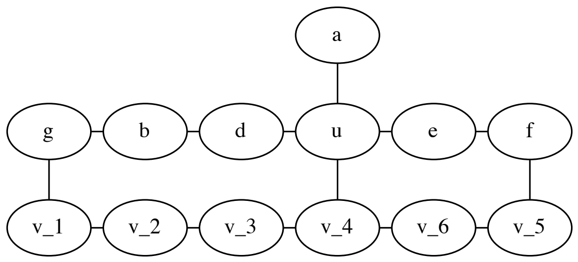


Figure E.4: Example environment configuration embedding for the environment in Figure E.3. with additional connecting nodes u and v.

to cover all the steps in the path and lay the items in the grid, marking the node as processed. The paths are searched using A* and Manhattan distance.

4. Iterate until there are no more nodes to process or the remaining nodes cannot be processed.

If any node remains after processing, our embedding algorithm returns a failure token, otherwise it returns a graph embedding defined as the tuple $\mathbb{Y}_E = \langle E, coords \rangle$ where *coords* are $\langle x, y \rangle$ coordinate pairs for each node in E.

Figure E.3 shows an example spatial configuration graph. Figure E.4 shows an example graph embedding for that spatial configuration graph. We used letters rather than the full names of the locations for space reasons (e.g., *a* for *log*, *b* for *beehive*...). It can be observed how a node (*cave*) has been split multiple times (first into u and v, and later v has been split again to form a path).

The algorithm described in this section is not guaranteed to find a planar embedding for a graph, even with the preprocessing steps defined previously. If no embedding is found, the process will backtrack to the first step and will select the subsequent longest of all the shortest paths. If no graph embedding is found a failure token will be returned that will prompt the ϵ-greedy search to continue to another spatial configuration graph.

Graphical realization

Given a graph embedding $\mathbb{Y}_E$, we want to compute a graphical realization for the spatial configuration E. Once a grid layout has been found, we proceed to the graphical realization by instantiating grid cells at the coordinates given by $\mathbb{Y}_E$ with a bitmap representation for rooms, doors for connected rooms and areas for split nodes.

Algorithm construction

This section describes the combination of the steps presented so far into an unified algorithm to generate a fully realized map, given the input to the system, $\langle S, L, player, I, G,\ P\rangle$.

Our algorithm works by performing two main tasks: first, decide on a *spatial configuration graph*, specifying which locations are connected with each other; then, embed and realize the given spatial configuration. One of the main difficulties on choosing a spatial configuration graph, is that there exist $2^{\binom{|L|}{2}}$ different spatial configurations for a given story world specification. Thus, we propose an ϵ-greedy strategy to search for a good candidate in the spatial configuration space. Spatial configuration graphs candidates with different features will be built and evaluated. Based on the features selected and the evaluation obtained, subsequent graphs will be built. Once a satisfactory candidate is identified, the algorithm will proceed with embedding and realization steps.

Using the processes defined in the previous sections, our system executes the following steps:

1. Initialize two tables, M and N:
   (a) M has $|L| - 1$ rows, one entry for each of the possible number of edges in E. All entries are initialized to zero.
   (b) N has $\frac{|L|(|L|-1)}{2}$ rows, one entry for each possible edge in E. All entries are initialized to zero.

2. Create a candidate spatial configuration E by:

   (a) Using an ϵ-greedy sampling strategy, select a number of edges m form the table M (e.g., select the maximum entry with probability $1-\epsilon$, and one at random with probability ϵ). In our experiments $\epsilon = 0.3$.

   (b) Select m edges by using an ϵ-greedy sampling strategy in table N (e.g., we use the ϵ-greedy strategy m times, to get m edges, without replication).

3. Using our planner, generate all possible sequences of plot points (e.g., stories) that could unfold in the given spatial configuration E.

4. Evaluate each individual story using our evaluation function.

5. Aggregate the individual story evaluations to obtain an evaluation for the spatial configuration E.

6. If the evaluation of E is below a threshold, update the tables M and N with the evaluation of E (each entry in M stores the average evaluation of the spatial configurations that have a given number of edges, and each entry in N stores the average evaluation of a particular edge in the spatial configurations), and go back to step 2. Otherwise, go to step 7.

7. Generate a planar embedding $\mathbb{Y}_E$. If none found, go to step 2.

8. Realize the planar embedding $\mathbb{Y}_E$ into a graphical representation (a grid in our experiments).

E.1.1 Experimental Evaluation

Our first experiment aims at evaluating the behavior of our story planner for inputs with different properties. We experimented with three different inputs to our system, which we altered in order to demonstrate the effects of adding and removing goals, plot points, locations and paths between locations:

- Input A is based on the plots produced by TALE-SPIN, specifically the *Joe Bear,* Irving Bird and *Henry Ant stories. It is composed of 8 plot points and 7 locations.*

- *Input B features a key and lock puzzle that requires revisiting some locations in order to accomplish the given goal. It is composed of 8 plot points and 6 locations.*

- *Input C is a task-based story inspired in the Tolkien universe. The player is presented with a set of 14 tasks represented as goals that must be satisfied. It is composed of 26 plot points and 11 locations.*

As we expected, the spatial configuration parameters have great impact on the number of stories found. Table E.1 shows the results from this experiment. Each row shows one different experiment using a specific variation of one of the three inputs described above, paired with a specific spatial configuration E. The first five columns show the given base input, the number of plot points, locations, the number of edges in E, and the number of goals in each experiment. The last column ($|\mathbb{S}_E|$), show the total number of stories that our planner could find in each of the experiments.

The first three rows show results for input A when using a spatial configuration with a predefined number of edges (6), with no edges (0) and with all the possible edges (21). As expected, with no edges, no stories could be found. Rows 4 and 5 show two variations of input B, where in the second one, we added an extra location and two additional edges to E. The result is that the number of stories that can unfold grows significantly (from 2 to 21). Notice that we are not measuring story quality yet. Finally, the last three rows show three variations of input C, where we varied the number of goals (from 4 to 14), and also varied the number of locations and edges. As we can see, adding goals constraints the planner to trying to satisfy each goal and therefore reducing the number of different stories found. The results also show the exponential relationship between the number of plot points and the number of stories found.

In order to validate our spatial configuration ϵ-greedy search process, we compared the configuration evaluations obtained against those obtained by some manually authored and some random generated spatial configurations. In the experiments we use the input A described in the previous experiment. Results are shown in Table E.2.

The first four rows show the results for input A, using three authored spatial configurations with 6, 8 and 3 edges respectively and the complete K_7 (with 21 edges). The spatial configurations were

Table E.1: Planning results for different experiments. $|P|$ is the number of plot points, $|L|$ is the number of locations, $|K|$ is the number of paths or edges in the spatial configuration E, G is the number of goals and $|\mathbb{S}_E|$ is the number of stories found.

	$\|P\|$	$\|L\|$	$\|K\|$	$\|G\|$	$\|\mathbb{S}_E\|$
A	8	7	6	1	10
A	8	7	0	1	0
A	8	7	21	1	100
B	6	5	4	1	2
B	8	6	6	1	21
C	19	4	3	14	8
C	19	4	3	4	192
C	26	11	55	4	384

Table E.2: Number of stories found ($|\mathbb{S}_E|$) in a series of spatial configurations, and their evaluation using both a *average* and a *minimum* aggregation operators.

	$\|\mathbb{S}_E\|$	Avg.	Min.
A_6	10	0.862	0.771
A_8	64	0.851	0.709
A_3	1	0.746	0.746
K_7	100	0.858	0.709
R_{Avg}	64.2	0.701	0.596
R_4	0	0	0
R_5	3	0.919	0.915
R_{11}	85	0.865	0.737
R_{17}	96	0.871	0.747
R_{19}	99	0.855	0.711
G_7^{Avg}	3	0.92	0.915
G_7^{Min}	3	0.916	0.914

manually crafted by ourselves using our subjective aesthetic preferences. We can observe how the number of paths affects positively the number of stories found $|\mathbb{S}_E|$ but has a negative impact on the *average* and a *minimum* evaluation aggregates. We also tested the input A with 89 randomly generated spatial configuration graphs with different number of edges. The fifth row shows the averages for the 89 random graphs followed by 5 samples, R_i, where the subindex i indicates the number of edges in the randomly generated graph. Finally, rows 11 and 12 show results from our ϵ-greedy search after running for 500 iterations. G_7^{Avg} was obtained trying to optimize the *average* evaluation aggregation operator and G_7^{Min} trying to optimize *minimum* instead. The subindex 7 indicates that the spatial configuration found contains seven edges.

As Table E.2 shows, randomly generated maps have a low evaluation, and in some instances, do not even support any story (e.g., R_4). The two spatial configurations generated by our algorithm, however obtain very high evaluation (always above 0.9), while supporting 3 stories. Comparing the authored spatial configurations against the algorithm results we can observe how the later obtained a significantly higher evaluation. However, the human authored spatial configurations take into account some interesting features, like allowing for a larger variety of stories or a more coherent distribution of the space. Those features are not taken into account by our evaluation function but might be desirable in some situations. As part of our future work, we would like to incorporate some of those features in our evaluation function.

The graph embedding algorithm processes a graph and yields a layout $\mathbb{Y}_E$ to be used by the graph realization process. We tested the algorithm by hand crafting a set of 25 planar spatial configuration graphs to test different features. The graphs range from 6 to 16 nodes and from 9 to 27 edges. Our algorithm yields a satisfying layout for 18 out of the 25 graphs.

We also tested the algorithm against 82 of the randomly generated graphs defined in the previous experiment. The graphs are not guaranteed to be planar and feature between 2 to 15 nodes and between 1 to 34 vertices. Our algorithm yields a satisfying layout for 62 out of the 82 graphs.

E.2 Conclusions and Future Work

In this appendix we presented a novel approach to procedurally generating game maps using storytelling techniques. We have described a system that can generate stories and then design a map supporting those stories. The system evaluates the stories in order to maximize the quality of the output. We use an intermediate graph structure to describe the spatial relationships between locations in the map, which we then use to graphically realize the map. Our experimental evaluation showed promising results, although there is a lot of room for improvement.

As our experiments show, our system can effectively explore the space of spatial configurations, searching for those that support high quality stories from the story space. Especially interesting is the capability of our system to generate game maps that support multiple stories or that contain cycles.

As part of our future work, we would like to explore the advantages of using more advanced story planners. The authored spatial configurations exhibit some properties that have not been encoded in the numerical interpretation of our evaluation function and are subjectively of higher quality. We plan on improving our evaluation function to account for additional features. We also would like to incorporate drama-management concepts and use a player model along with annotations in our input game world specification to generate maps tailored to specific player preferences or player archetypes[96]. We would like to tighten the relationship between the planning and the graph embedding processes and incorporate elements from partial order planning into the environment creation process in order create maps more relevant to the story and improve the scalability of the system. Additionally, the graph embedding process has shown to be one of the most challenging parts of the system. In order to reduce the running time to a reasonable bound for our experiments we implemented several heuristics and limited backtracking which may prevent us from finding a planar embedding for graphs that actually are planar. We would like to improve our preprocessing steps for better handling subgraphs homeomorphic to K_5 and $K_{3,3}$. Finally, we would like to connect our map generation system with a game engine in order to have a fully playable game to perform user studies and validate our results. We would also like to look into on the fly map generation incorporating player modeling into our evaluation function and drama management techniques for further improving the player experience.

E.3 Visualizing Story Spatial Information Using Voz

In some exploratory work, we use *Voz* to automatically extract spatial information from the story graph and render it a two-dimensional environment. This is a visual representation of and environment suitable for that particular story to happen. Given a story, we use *Voz* to generate a graph where the nodes are characters and locations extracted from a story. The nodes are connected with edges that indicate a character used a locomotion verb (walk, go, run, etc.) towards a location therefore indicating that the location is accessible. We then filter the locations from the graph and connect them with edges where there was a character traversing a pair of locations. Figure E.5 shows the extracted graph and a 2D realization of the graph embedded in a grid.

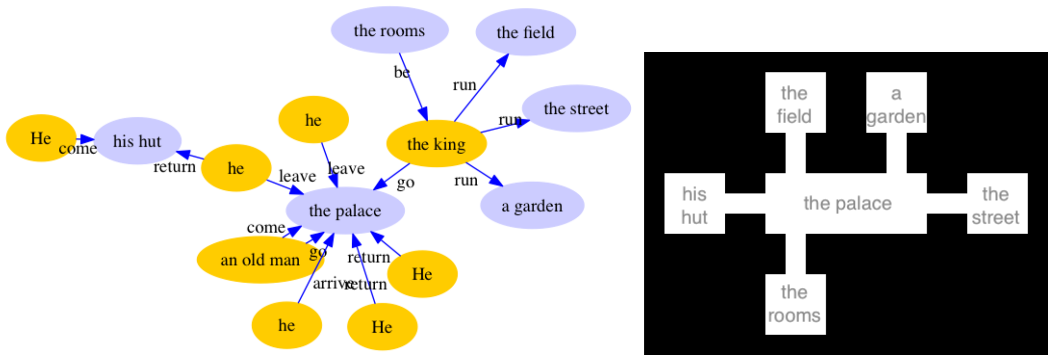


Figure E.5: Graph with character and location information automatically extracted from a story and two-dimensional embedding and realization of the locations in the graph.