Enabling Imagination through Story Alignment

by

Matthew Paul Fay

B.S. Chemistry
Purdue University, 2009

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree

of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February, 2012

Signature of Author ……...
Department of Electrical Engineering and Computer Science
January 26, 2012

Certified by ……...
Patrick H. Winston
Ford Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ……..
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, EECS Committee on Graduate Students

Enabling Imagination through Story Alignment

by

Matthew Paul Fay

Submitted to the Department of Electrical Engineering and Computer Science

on

January 20, 2011

in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Electrical Engineering and Computer Science

ABSTRACT

Stories are an essential piece of human intelligence. They exist in countless forms and varieties seamlessly integrated into every facet of our lives. Stories fuel human understanding and our explanations of the world. Narrative acts as a Swiss army knife, simultaneously facilitating the transfer of knowledge, culture and beliefs while also powering our high level mental faculties. If we are to develop artificial intelligence with the cognitive capacities of humans, our systems must not only be able to understand stories but also to incorporate them into the thought process as humans do.

In order to work towards the goal of computational story understanding, I developed a novel story comparison method. The techniques I present in this thesis enable efficient and effective story comparison through story alignment. My algorithms, implemented into the Genesis system, allow the comparison and combination of stories which is a step towards enabling imagination in artificial intelligence. This capability is made possible by reducing the runtime of a previously intractable computational problem to polynomial time.

In the course of this research, these algorithms have been applied to a variety of story analysis problems. By comparing short, 10 sentence summaries of the Tet Offensive and the Yom Kippur War, the system predicts information omitted from both stories. In the analysis of a brief synopsis of Shakespeare's Macbeth, my algorithm is able to correctly match actors and events between two different variations of the tale by cutting down a search space of over 10^{30} nodes to a mere 546 nodes. My techniques also demonstrate promise as a component of a larger video analysis system. The story alignment capabilities are used to fill in missing gaps in descriptions of videos, corresponding to missing video data, by comparing video feeds to an existing video corpus.

Thesis Supervisor: Patrick H. Winston
Title: Ford Professor of Electrical Engineering and Computer Science

Contents

List of Figures

List of Tables

Enabling Imagination through Story Alignment

Introduction

Why Stories?

Stories are an essential piece of human intelligence. They exist in countless forms and varieties seamlessly integrated into every facet of our lives. Stories fuel human understanding and our explanations of the world. Narrative acts as a Swiss army knife, simultaneously facilitating the transfer of knowledge, culture and beliefs while also powering our high level mental faculties. If we are to develop artificial intelligence with the cognitive capacities of humans, our systems must not only be able to understand stories but also to incorporate them into the thought process as humans do.

In order to take story understanding to the next level, I have decided to focus on the problem of story comparison. People intuitively use story comparison to draw on old experiences to construct and understand new ideas. Whether engaging in conversation or trying to understand a unfolding scenario in the world, people engage by connecting their experiences with the current situation at hand. A student studying *Hamlet* for the first time may discover acts of revenge and draw the connection to a previous viewing of Disney's *The Lion King*. A couple enjoying dinner at a local restaurant exchange stories by recalling tales that remind them of what their partner just said. A chemist developing a new drug discovers a novel reaction pathway and must tackle the problem of why the atoms behaved as they did. Prior experiences with other mechanisms allow her to imagine how the pieces may fit together yielding the experiments needed to test her hypothesis. A child hears a story about the tooth fairy and promptly begins

experimenting with methods to yank out teeth by slamming a door. What fuels all of these examples is the human ability to perform analogy and draw strong comparisons between stories in order to generate new ideas (Schanck 1990).

In this thesis, I present a novel computational story comparison method which allows for efficient and effective story comparisons with broad applicability. Doing naïve story comparison is a computationally hard problem. The algorithms I display in thesis are capable of providing a polynomial time solution to an otherwise exponential time problem. When working on a version of the Macbeth story, my algorithms give a decrease in processing required by reducing the search space from over 10^{30} nodes to only 546 nodes, which allows progress on an otherwise infeasible problem.

The organization of my thesis follows.

Chapter 1. Story Understanding in Related Works

The first chapter is dedicated to identifying how language and story understanding are the basis for human intelligence. This includes a brief anthropological overview of the onset of human intelligence. Explanations from reference work indicate the importance of story understanding

Chapter 2. Background

The second chapter covers all the background information related to this thesis. I include a detailed overview of the Genesis System which serves as a backbone for the implementation of the ideas presented in my work. This includes all of the representations used for expressing stories in a computational environment.

Chapter 3. Story Alignment

The third chapter introduces the approach I have taken in working with story comparison. I introduce an alignment algorithm from the bioinformatics literature which is used as a basis for aligning stories.

Chapter 4. Simultaneous Matching and Alignment as a Solution to the Matching Problem

The fourth chapter describes, in depth, the algorithm used for matching the actors between stories. The method presented combines alignment and bounded search. The chapter also outlines a number of optimizations which make the computationally complex problem run in near real time.

Chapter 5. Reflective Alignment

In the fifth chapter, I demonstrate how using higher level reflective knowledge can empower the story comparison algorithm. Reflective knowledge is used to both speed up the processing and focus on interesting comparison results.

Chapter 6. Applications and Continuing Efforts

The sixth chapter outlines a number of examples of applications of the simultaneous matching and alignment algorithm via the analysis of stories from the Genesis corpus. Additionally, I demonstrate how the techniques are robust, capable of working with both textual and video data, in the context of the Mind's Eye project for video analysis. Finally, the chapter includes a short discussion of story clustering work which uses the algorithms of my thesis.

Chapter 8. Contributions

The final chapter summarizes the contributions I have made in this thesis work.

Chapter 1: Story Understanding in Related Works

In order to develop artificial intelligence that is capable of thinking and reasoning on the level of a human, we should understand what it is that makes humans unique from other species. In this chapter, I describe why narrative understanding is an important part of human intelligence. I also outline related works which motivate the work presented in this thesis.

1.1 What makes us human?

The naïve view of evolution is that it is a simple progression over time: older species are less evolved and thus lesser than modern species. This simply is not the case. Similarly, humans are not products of a gradual evolution from Chimp-like primate to modern humans.

Ian Tattersall's research in human evolution has revealed that early humans have little physical distinction separating them from other hominid species evolving contemporaneously (Tattersall 2008). In fact, many different species of hominid flourished across parts of Africa, Asia and Europe as early as 500,000 to 1,000,000 years ago. These groups were quite advanced and were known to have crafted and wielded a variety of simple tools carved from stone and bone. It wasn't until around 50,000-80,000 years ago that *homo sapiens* as we know them today emerged suddenly. Within a relatively short span of time, all other hominid species vanished. What was the spark that allowed modern humans to be so successful compared to the earlier hominids? Tattersall suspects that it was symbolic reasoning. The first explosive growth of the *Homo sapiens* population is marked by the first emergence of artwork and jewelry, as well as rapid development of new tools far beyond the previous designs which had lasted nearly a million years (Tattersall 2008). Tattersall believes this rapid emergence was due to and made possible by

a great evolutionary coincidence in which a number of independent developments in the brain gave rise a symbolic reasoning system.

1.2 Pervasiveness of Language

A symbolic reasoning system seems to lie at the heart of human intelligence and as Tattersall's work shows, it seems to be a key factor in advancing human kind from early primates to modern humans. Today, the power and versatility of language is present in nearly every aspect of our lives. People use language to construct and archive ideas, stories, and knowledge. The domain of these stories seems to be vast containing everything from news, current events, and schedules, beliefs, knowledge, conversations, and novels. Language appears to be intricately connected to how people index and retrieve nearly all the information they process (Schanck 1990). Additionally, language seems to enable the comparison and combination of stories. This drives a powerful component of human intelligence, the ability to imagine. Noam Chomsky's position is that part of the uniqueness of humans relies in our ability Merge: to use language to combine old ideas into new ad infinitum (Chomsky 2005).

1.3 Power of Stories

Observations of how humans use stories have caused many to believe that stories are highly connected to intelligence. Patrick Winston suggests The Strong Story Hypothesis (P. Winston 2011) which is the idea that the human ability to manipulate stories is what separates them from other primates. This is strongly coupled with The Directed Perception Hypothesis**:** the heart of knowledge and understanding is the capability of our symbolic reasoning systems to purposively direct both real and imagined perceptions in order to answer questions about the world (P. Winston 2011).

Kay Young and Jeffrey Saver's work suggests that many dysfunctions from brain damage can accurately be described as dysnarrativia, which is the loss of ability to properly understand stories (Young and Saver 2001). They describe how damages to the brain can actually cause errors in story understanding specific to the region of the brain that was damaged. This suggests that story understanding is an important part of reasoning that uses a wide array of the brain's resources. In Table 1, constructed from Young and Saver's work, a number of specific dysnarrativia are described with examples.

Forms of Dysnarrativia

Clinical Manifestation	Neuroanatomic Substrate	Example Behavior
Arrested Narration	Amygdalohippocampal System	Can construct coherent story of life leading up to injury, but not at all beyond.
Unbounded Narration	Amygdalohippocampal System	Construct fictitious stories to fill memory gaps, often contradictory even within a few minutes. Seem to only be trying to answer queries, not attempting to exaggerate or entertain.
Undernarration	Orbitofrontal cortices	Unable to imagine outcomes or consequences of actions, become impulsive and vacillating.
Denarration	Dorsolateral/Mesial frontal cortices	Apathetic state, lacking in spontaneous activity, unable to provide account of any experiences, wishes or actions, yet fully cognizant in visual, auditory and tactile domains.

Table 1 - Forms of Dysnarrativia
Various forms of Dysnarrativia are described by the work of Young and Saver (Young and Saver 2001). They provide a number of examples of brain injuries that seem to cause specific pieces of person's ability to understand narrative to become impaired. Interestingly, they've identified what parts of a story specific parts of the brain seem to handle based upon the impairments in story understanding caused by damage to those parts.

In psychology, a great deal of research has been done determine how people understand and think about the world. This includes work on how stories can be used for problem solving, (Jonassen and Hernandez-Serrano 2002) and how analogy and similarity influence understanding (Gentner 1997). One discovery is that there seems to be a distinct difference in how people reason about situations dependent on the level of expertise the person has specific to that situation. One commonly noted difference between novices and experts that has been noted through both cognitive studies and computational analysis is that novices focus on similarity whereas experts focus on analogy (Finlayson and Winston 2006). As a simple example, consider the stories of *The Lion King*, *101 Dalmations*, and *Hamlet*. On the level of similarity one may connect *The Lion King*, and *101 Dalmations*. Both contain a variety of personified animals and are animated works. At the level of analogy, however, one notices that the major plot elements and character interactions in *The Lion King* mesh much closer with those in *Hamlet*.

Understanding how to compare stories is a powerful and important facet of human understanding. Because of this, I believe that computational story understanding and comparison is a necessary component of artificial intelligence. Accordingly, my research is dedicated to my goal of taking computational story understanding to the next level. In the next chapter, I describe Genesis story understanding system in development at the MIT Computer Science and Artificial Intelligence Laboratory, which serves as an important back bone for my research in story comparison.

Chapter 2: Background

In this thesis, I describe my approach for tackling problems in computational story understanding. In order to focus on the goals of the thesis, I leverage the power of existing systems. Most importantly, my work focuses on using the power available in the Genesis system to develop a robust story comparison technique. This chapter gives an overview on the current status of Genesis and the important capabilities of the system pertaining to my thesis work.

2.1 Genesis

Genesis is a comprehensive human intelligence system that has been in active development in the Computer Science and Artificial Intelligence Lab at MIT (P. H. Winston 2011). The high level objective for Genesis is to provide a complete computational system for understanding stories with a focus on modeling how the human mind reasons. Because the Genesis research group consists of a number of researchers with interests ranging from story analysis, human intelligence, visual understanding and cognition, the group has made architectural decisions that make it easy to include new collaborators and facilitate the investigation of new ideas.

2.2 Wire-Box Paradigm

An important design aspect of Genesis is the Wire-Box paradigm which has been inspired by the "Propagator Model" (Sussman and Radul 2009), proposed by Gerald Sussman, `and the "Observer pattern" (Gamma, et al. 1994), a common programming style. In the Wire-Box model (P. H. Winston 2011), each component of the Genesis system is divided into its own independent module or box. Each box interacts with other boxes via wires which can provide both input and output channels. The advantage of this interaction style is that boxes need not be

aware of the implementation or even the source of data. Similarly, boxes simply output their data through an output pipe without needing to know the destination of the information. From a development standpoint, the wire-box paradigm allows numerous people to work on different aspects of the project simultaneously without needing to worry about conflicts or broken code. Additionally, boxes can be updated or even replaced without dependent modules needing to be modified so long as the input and output wires are properly maintained.

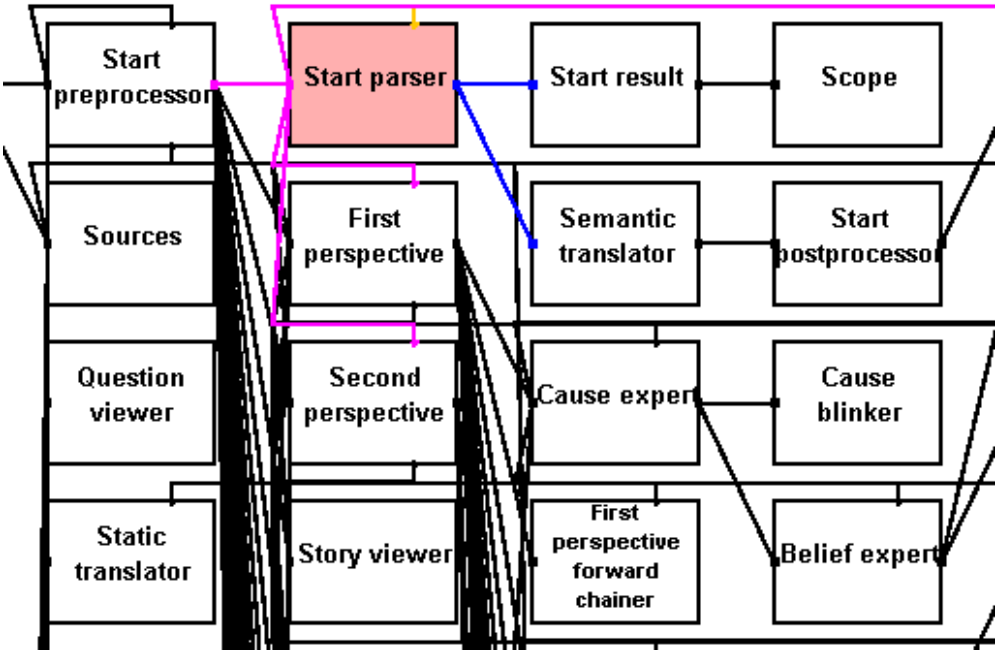


Figure 1 - Wired Boxes in Genesis
Internally, Genesis uses the wired box model to allow many independent components to work together. Each box represents an independent module which can be connected to any number of other modules via input and output wires. Each module has a specialized purpose and is connected to other modules for the input it needs. In this view the colors show the direction of the wires. For the highlighted pink box, pink wires represent input channels and blue wires represent output channels.

The wire-box abstraction has been designed to work seamlessly regardless of the physical machine running a particular box. This means that from a coding standpoint you can create a box without needing to know whether the input wires are over a network connection, between

processes on a machine, or simply running in the same instance of the program (P. H. Winston 2011).

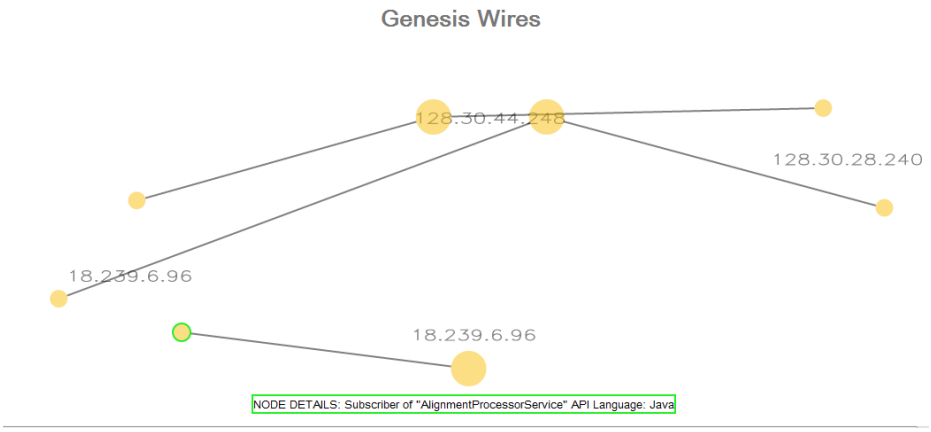


Figure 2 – Genesis Networked Wire Graph
Over the network, the wired box model allows development of modules that need not know whether a box they communicate with is a local or remote module. The connected network boxes can be visualized over the web via a browser interface. Each IP address corresponds to a network connection and each circle is a service or consumer of services running on that network connection. Each node can be queried via a web interface to see details about the type of service provided which is shown in the green box at the bottom.

The seamless network integration of connecting wired boxes has the added advantage that it helps drive active collaboration between groups. In Chapter 6 I discuss in more detail how we've used this methodology to drive collaboration in the Mind's Eye project.

2.3 Language Processing

One aim of Genesis is to make sure all input is in an easily reusable, human readable form. All the stories I discuss and analyze during the course of this thesis are stored as English text in the Genesis story corpus. This means that the same story can be interpreted by both Genesis and a human reader without needing to exist in any preprocessed form. Additionally, this means that our corpus is robust to large-scale system changes – an overhaul of the Genesis system does not require the story corpus to be changed in any way.

In order to facilitate language processing and keep the focus of Genesis on story understanding, the system uses the Start Natural Language Processor (B. Katz, Annotating the World Wide Web using Natural Language 1997) to interpret English language input. Many language processors exist including popular statistical language parsers such as the Stanford Parser (Klein and Manning 2003). However, unlike many language processors available Start constructs not only parse trees of sentences, but it also creates a rich semantic net (B. Katz, Annotating the World Wide Web using Natural Language 1997) (P. Winston 2011) of the sentence being parsed. The semantic net is capable of identifying and tracking nouns, relations, and actions across sentences which is valuable for tracking characters and objects through a story. Start has been set up as a question answering system on the web (B. Katz, Annotating the World Wide Web using Natural Language 1997). Using an internal knowledge base, Start can answer questions with informative English responses. As an example, the question "What is Jupiter's atmosphere made of?" receives the response "The atmosphere of Jupiter contains hydrogen, helium, methane, ammonia, ethane, acetylene, phosphine, water vapor, and carbon monoxide" (B. Katz, Start Natural Language Question Answering System 1993). A version of Start has been made available via a wired box to facilitate interaction with external systems.

Figure 3 shows some of the semantic information available in Start. Given the sentence "Mary went to the store to buy groceries," Start generates an internal semantic net. For external applications such as Genesis, Start makes the data in the semantic net available through the triples notation. Each triple expresses a relation among the constituents of the sentence. For example, Start identifies in the first triple [go+1 has_purpose+2 buy+2] that the purpose of going to the store is to buy something. These elements are all tracked via id tags so Start and in turn

Genesis can easily identify all the information involving any action, entity, or description in a sentence or story.

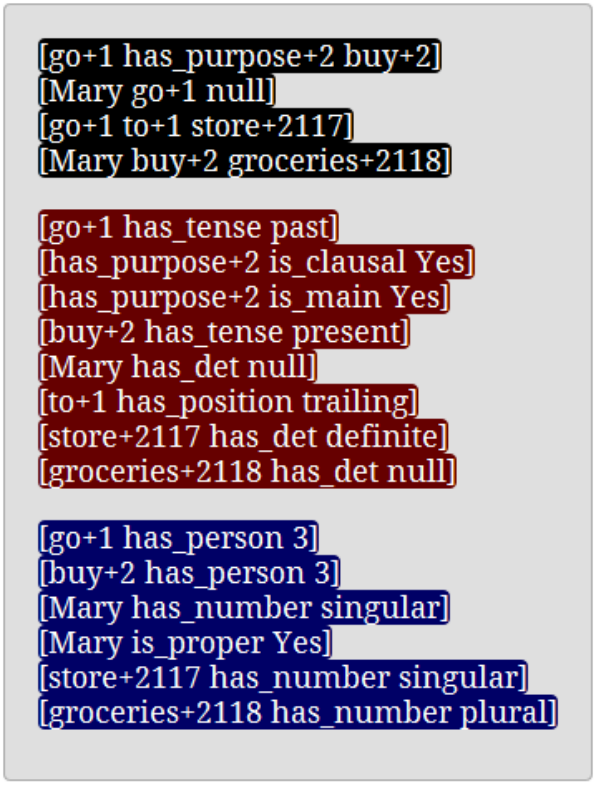


Figure 3 - Start Triples
The Start Natural Language System is capable of parsing English sentences into a semantic net. The triples representation shown in this figure used for exporting Start's internal understanding of the sentence. The sentence which generated the triples shown was "Mary went to the store to buy groceries."

In addition to using language as an input, Genesis uses Start to generate English sentences from Genesis's inner representations. This allows for English to be seamlessly integrated into the overall architecture. All output and input for the user can be in English while beneath the surface all the story processing can take place in the representational language used by Genesis. This is possible because of Genesis's ability to seamlessly convert between its own inner representations and the triple notation used by Start.

2.4 Genesis-ese

Genesis uses an inner representational language colloquially called Genesis-ese. This representation has been tailored to cater specifically to the needs of story understanding. The language is capable of representing and storing many types of knowledge, including low level causal rules, motivations, themes, and higher level reflective knowledge. In doing so, it is capable of handling stories from a wide variety of domains, from simple stories such as "Matt gave Emily the mug," to complex narratives such as historical war stories.

At the implementation level, Genesis-ese is composed of four basic componentents:

- A **thing** represents a single actor, object, place, or other entity taking part in a story. Examples include "Matt", "a ball", "the United States", "a king", and "a castle".
- A **derivative** is a composition of a modifier and a thing, often representing a portion of the contained thing. Examples include "the top of the table", "the back of the room", "the fork in the road".
- A **relation** connects two things via a modifier. It can represent any sort of interaction between the things. Examples include "John kissed Mary", "the ball hit the ground", and "Jane likes the flower."
- A **sequence** is a grouping of any number of things. It can represent any set of things, relations, or derivatives. Examples include, "Matt and Mark", "The top of the table and the back of the room", and "the United States and Great Britain".

We can represent ideas of arbitrary complexity, using only this set of base units. This means Genesis can easily handle multiple levels and types of knowledge.

2.5 Representational Knowledge

A variety of knowledge representations comprise the inner language, Genesis-ese. These include, but are not limited to, *classes*, *transitions*, *goals*, *persuasion*, *social relations*, *property*, and *trajectories***.** Each can be expressed both in common English and in Genesis's inner language (P. Winston 2011). Examples of these types of knowledge are displayed in Table 2 below.

Representation	Example
Cause	Alpha killed Bravo because Bravo angered Alpha.
Class	A keyboard is a type of input device.
Goal	Charlie wanted to be a pilot.
Job	Delta is a pilot.
Mood	Echo became unhappy.
Persuasion	Foxtrot persuaded Golf to commit the crime.
Possession	Hotel had the prize.
Property	India is large.
Role frame	Juliet killed herself with the dagger.
Social relation	Kilo was Lima's brother.
Time	Next, Mike looked around.
Trajectory	Oscar went quickly through the city.
Transition	The current month became November.

Table 2 - Types of Representational Knowledge
Genesis contains a number of types of representational knowledge. Collectively the set of representational knowledge Genesis creates and understands is known as Genesis-ese.

These representational knowledge types allow Genesis to understand a wide array of story level information. All the representational knowledge of Genesis-ese can be stored in the Thing, Derivative, Relation, and Sequence frames of Genesis. This forms the underlying structure for all the story analysis that Genesis performs. Figure 4 shows a couple examples of how the representational knowledge of Genesis-ese can be stored in Genesis.

Figure 4 – Representational Knowledge in Genesis's inner language
Examples of three different types of representational knowledge are shown above. The red bars represent relations, the blue bars represent derivatives, the gray bars represent things, and the black bar represents a sequence. Each of the three examples comes from an English sentence which has been translated by Genesis using the Start processor into the inner language of Genesis. The left side represents the sentence "Delta is a pilot." The middle represents "Echo became unhappy." The right side represents "Juliet killed herself with the dagger."

2.6 Commonsense Knowledge

Commonsense knowledge is an important category of higher level knowledge for story understanding (P. H. Winston 1980). This type of information is the background knowledge that a person brings to the table before even reading the first word of a story. It is the set of causal rules about the world which allow a person to connect events together. In the Genesis system this knowledge comes in a number of forms, all translatable to and from natural English expressions (P. Winston 2011).

Some causal knowledge comes in the form of sentences such as "If XX kills YY, then YY is dead," and "YY harmed XX because YY attacked XX." These rules represent predictions of events and what the effects of events are. When translated into Genesis-ese, the rules take on the following forms:

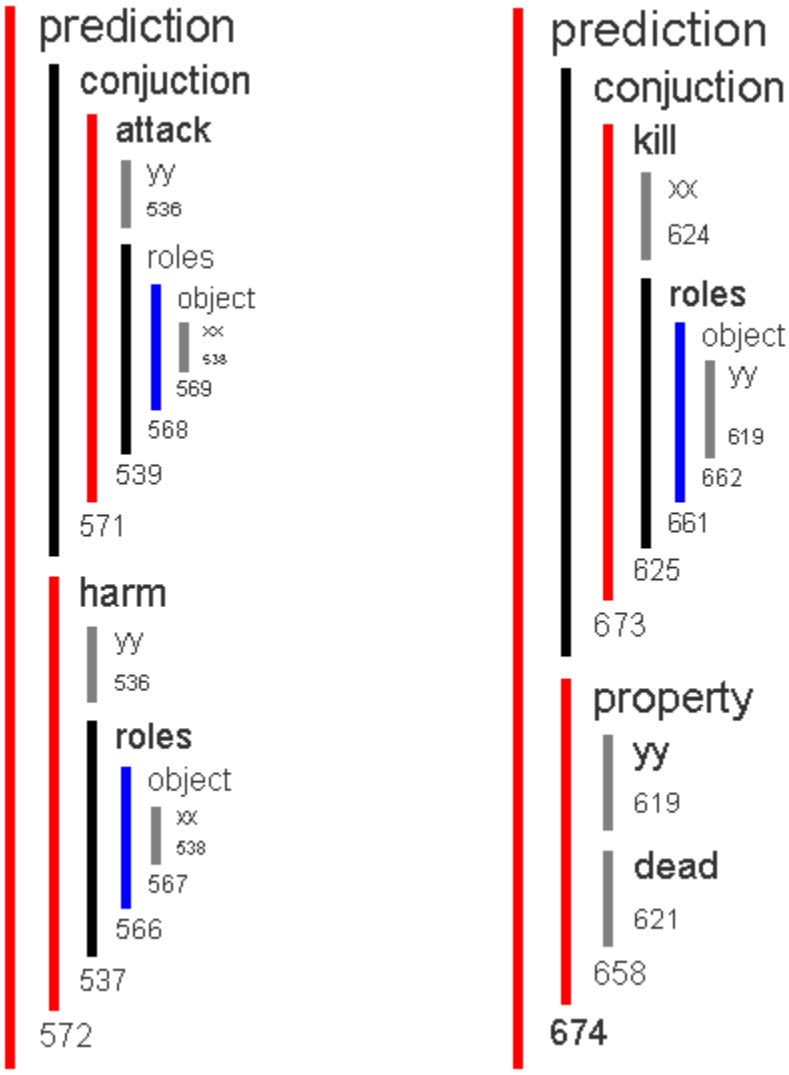


Figure 5 – Predictive Commonsense Knowledge
Genesis represents commonsense rules about predictions using the same underlying implementation as all of its knowledge. The representation on the left was generated from the commonsense rule, "YY harmed XX because YY attacked XX." The representation on the right is from the rule, "If XX kills YY, then YY is dead"

Other causal knowledge takes on a more uncertain form. Examples include "If XX harms YY then YY may want to harm XX" and "If YY is the king and XX wants to be the king, then XX may kill YY." These rules indicate to Genesis that while the events don't necessarily cause each other, if they co-occur then they are likely connected. The rules have the following form in Genesis-ese, which is similar to that of the predictions:

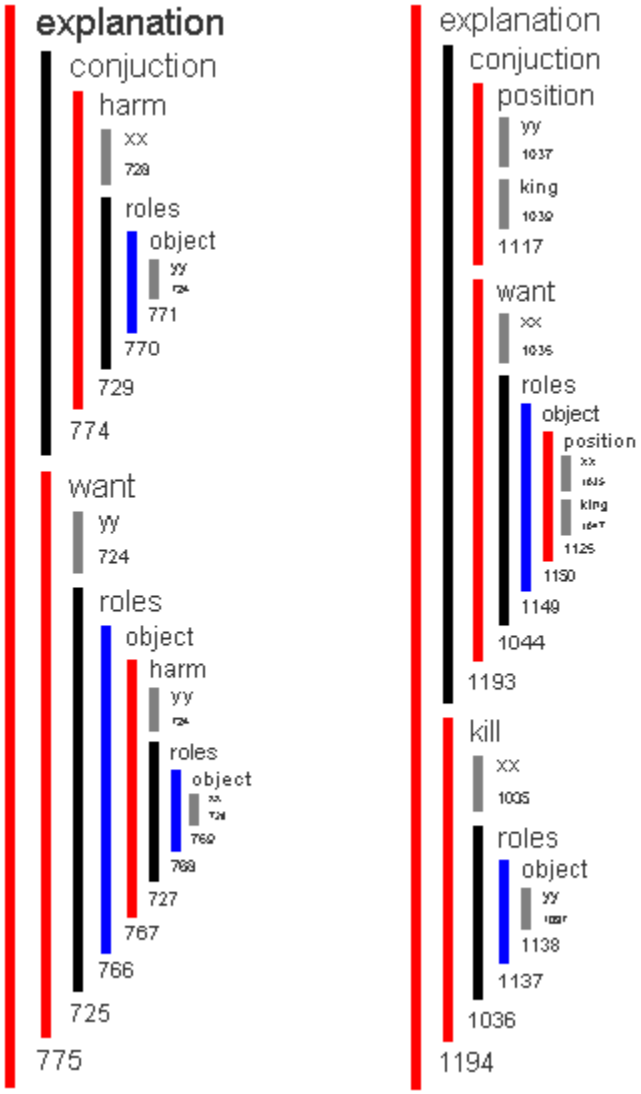


Figure 6 - Commonsense Explanations
Genesis is also capable of understanding commonsense rules dealing with explaining the connections between events. The representation on the left was generated from the commonsense rule, "If XX harms YY then YY may want to harm XX." The representation on the right was generated from the rule, "If YY is the king and XX wants to be the king, then XX may kill YY."

These types of commonsense rules allow Genesis to construct an elaboration graph, which is a visual representation of how all the events in the story are connected to each other and the inferred events from common sense knowledge. As an example, a rendition of Macbeth in the Genesis story corpus has 21 commonsense rules as background knowledge and 20 sentences in the story summary. The resulting elaboration graph is shown in Figure 7.

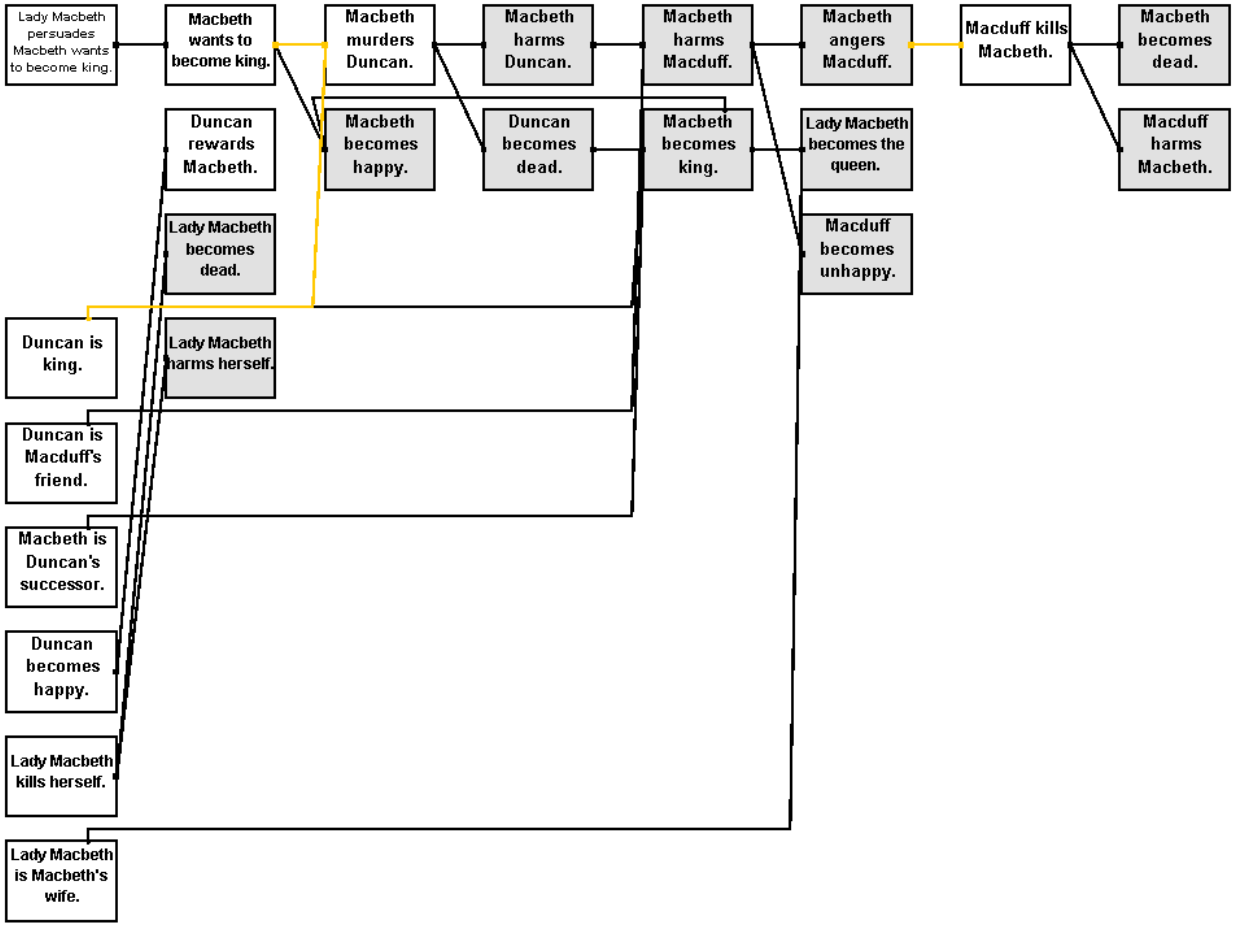


Figure 7 - Macbeth Elaboration Graph
This elaboration graph is constructed from reading a rendition of Macbeth from the Genesis story corpus containing 21 commonsense rules and 20 plot points. In this visualization story events are represented by boxes. White boxes represent events which have been read directly from the story. Gray boxes represent events which were inferred from background knowledge. Events which have been determined to have been connected based on the commonsense background knowledge are connected by a line.

2.7 Reflective Knowledge

Another higher level category of knowledge Genesis understands is reflective level knowledge. Reflective knowledge consists of high level ideas, themes, and motivations that tie together a number of story events. Examples of common reflective knowledge include 'Revenge', 'Pyrrhic Victory', 'Suicide', and 'Mistake'.

In Genesis, the reflective knowledge is represented as a sequence of events that can be stored as a series of English descriptions required to trigger it. For example, consider the concept of "Revenge". One possible explanation of what revenge is, is "XX harms YY. Then, YY harms XX because XX harmed YY." In Genesis, this definition of revenge manifests itself as the frame representation shown in Figure 8 on the following page.

Revenge English Representation

Start description of "Revenge"
XX is an entity.
YY is an entity.
XX's harming YY leads to YY's harming XX.
The end.

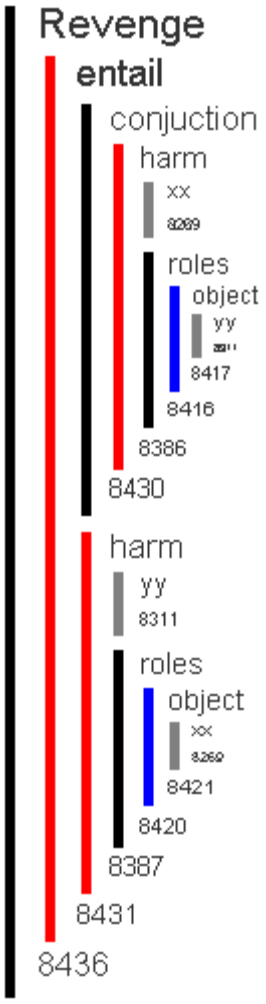


Figure 8 - Reflective Knowledge
Reflective knowledge is a higher level of understanding available in Genesis. The table on the left shows the English definition of 'Revenge' as found in the Genesis story corpus. The Genesis-ese representation on the right shows the instantiation of this definition. Genesis is capable of searching through a story to determine which reflective plot units occur in a story.

While simple stories may contain little to no instantiations of reflective knowledge, more complex stories may contain numerous examples. The same elements of reflective knowledge may occur repeatedly throughout a single story. Consider the previously described story of Macbeth. Based on a small set of 12 reflective knowledge types, the story is found to contain 7 occurrences of the reflective plot units. These plot units include the success of Macbeth seizing the throne, Macduff's revenge on Macbeth, and Lady MacBeth's suicide.

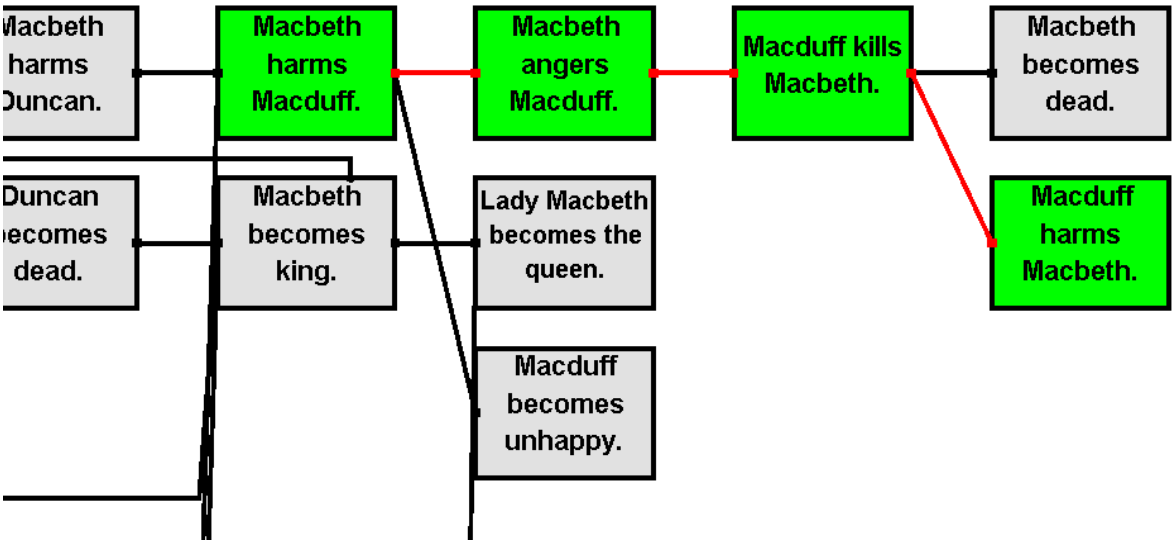


Figure 9 - Instantiation of Revenge
Revenge is a common plot unit occurring in the story of Macbeth. The boxes each represent story elements and the lines connecting them represent common sense knowledge connections. The highlighted chain of events has been recognized by Genesis as being a revenge plot unit. The definition of revenge used in this case is that shown in Figure 8 which defines revenge as a harm leading to a reciprocal harm.

This chapter reviewed the capabilities of Genesis available for language processing and story understanding. My work in story understanding leverages these available technologies to develop novel story comparison techniques. In the next chapter of this thesis, I describe the low level algorithms I use for comparing stories and their components.

Chapter 3: Story Alignment

In this chapter, I discuss the sequence alignment algorithm which acts as a stepping stone for my story comparison algorithm. I begin the chapter by demonstrating the advantages of representing stories as linear sequences of events. Then, I introduce a robust sequence alignment algorithm from the literature which I use as a stepping stone for story alignment. Finally, I demonstrate the use of the classical alignment algorithm to further the goal of computational story understanding.

3.1 Stories as Sequences

As I described in the previous chapter, Genesis comes equipped with a wide array of story analysis tools and representations. The tools currently available in Genesis create a few potential choices for how to best represent a story. Using a library of common sense knowledge and reflective level plot units, Genesis can create a complex elaboration graph of how the story elements relate to each other. Going a step further, another possible representation of the story could be the set of instantiated plot units occurring within the story. These plot units characterize the sets of interesting events in the story. For my research however, I have decided to focus on the story level elements for representing stories and enabling comparisons. In particular, I define a story in terms of being a sequence of time ordered story elements, which fits well with the underlying representation structure used in Genesis.

The simple story, "The dragon kidnapped the princess. The prince slew the dragon. The prince rescued the princess. The prince and the princess lived happily ever after," is shown in Figure 10 as a sequence of Genesis story events.

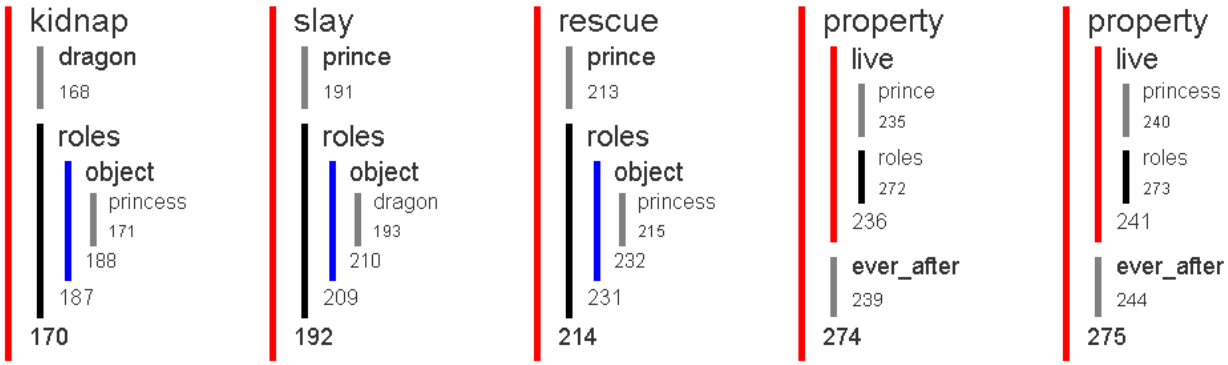


Figure 10 – Story as Sequence
An example of a simple story understood by Genesis is, "The dragon kidnapped the princess. The prince slew the dragon. The prince rescued the princess. The prince and the princess lived happily ever after." The story is represented as the shown sequence of story events in Genesis.

The advantages of representing stories as sequences outweigh the cons for my work in story comparison. An elaboration graph representation of a story offers increased information on how events in a story are connected and fit into the larger puzzle. However, the additional information comes at a high cost; doing story graph comparison is extremely costly in computation time. Because the goal for my thesis is to enable rapid story comparison as a building block towards intelligence, the cost is prohibitive. On the other end of the spectrum, a great deal of worth can come out of comparing stories by the merits of their reflective level plot units. Reflections can be compared easily and contain much of the crux of a story. However, using only plot units for story comparison means you lose information from story elements that don't occur in the set of analyzed plot units.

Choosing to represent stories as sequences of story elements allows for great flexibility in accomplishing the goals of my thesis. While it does not encode the full causality information of an elaboration graph, the time ordered sequence still implicitly encodes a great deal of causal information and allows the construction of a fast and useful algorithm, as I demonstrate. Additionally, while a story as a sequence of story elements does not encode reflective level

knowledge, I return to the discussion of plot units in Chapter 5 and demonstrate how higher level knowledge can be incorporated into the story comparison algorithm.

3.2 Sequence Alignment Algorithm

3.2.1 Needleman-Wunsch

Alignment algorithms have been researched heavily and been proven to be very powerful in the study of bioinformatics due to the enormous lengths and quantities of DNA and protein sequences that must be analyzed. Because I represent stories as sequences, I've decided to turn my attention to previous work from that field in order to give a solid baseline for my story comparison techniques. The canonical algorithm used for the global alignment of pairs DNA sequences is the Needleman-Wunsch algorithm (Morgenstern, et al. 1998). In typical use, the algorithm provides the best alignment between two sequences of maximum length *n* in $O(n^2)$ time while allowing for both insertions and deletions. Because this algorithm is an important base component of my algorithm, it is important to review its mechanism (Needleman 1970).

The inputs to the algorithm are two sequences, *A* and *B*, and a similarity matrix, *S*, comparing all element types that exist in the domain. The goal is to find an alignment between sequences with a maximal alignment score where the score is calculated by summing over the similarity of each pair of elements in the alignment. Gaps are allowed anywhere in the alignment and are given a constant gap penalty *d*.

To find the optimal alignment, the algorithm first constructs a matrix *F* with each row representing one element in order from the first sequence and each column representing one element from the second sequence. Letting *i* be the row index and *j* be the column index, the

values F_{i0} and F_{0j} are initialized to be $F_{i0}=i*d$ and $F_{0j}=j*d$ respectively. The rest of the elements of F are generated via the following recursion:

$$F_{ij}=\max(F_{i-1,j-1}+S_{Ai,Bj},F_{i,j-1}+d,F_{i-1,j}+d)$$

Once all elements of F have been generated, F_{nm}, where n is the length of A and m is the length of B, represents the maximal alignment score. To obtain the alignment that gives this score, the algorithm walks through F starting at F_{nm} by moving either (0,-1), (-1,0), or (-1,-1). The walk ends once the algorithm arrives at F_{00}. Each horizontal or vertical step represents a gap in a sequence for the alignment. Each diagonal step represents a pair of matched elements in the alignment.

	A	**C**	**G**	**T**
A	4	-7	-7	-7
C	-7	4	-7	-7
G	-7	-7	4	-7
T	-7	-7	-7	4

Table 3 - DNA Similarity Matrix
A similarity matrix for aligning the nucleotides of DNA requires only a 4x4 matrix. Higher numbers represent a greater degree of similarity between the element types. The values can be adjusted to fit the goal of a particular alignment strategy. The values shown represent a good set for finding 88% identity alignments. This means this matrix is optimal for aligning sequences of DNA that are expected to 88% homologous (Eddy 2004).

For example, consider the two hypothetical DNA sequences, X, "AGACTAGTTAC", and Y, "TCGATTTAC". A potential similarity matrix is shown in Table 3. Using the Needleman-Wunsch algorithm and the shown similarity matrix, the X and Y sequences can be aligned quickly and effectively. Recall the first step of the algorithm is the computation of the F matrix. Once computed, the matrix can be used to quickly find the optimal alignment.

F matrix	T	C	G	A	T	T	T	A	C
A	-7	-14	-21	-17	-24	-31	-38	-45	-7
G	-14	-14	-10	-17	-24	-31	-38	-45	-14
A	-21	-21	-17	-6	-13	-20	-27	-34	-21
C	-28	-17	-24	-13	-13	-20	-27	-34	-28
T	-24	-24	-24	-20	-9	-9	-16	-23	-30
A	-31	-31	-31	-20	-16	-16	-16	-12	-19
G	-38	-38	-27	-27	-3	-23	-23	-19	-19
T	-45	-45	-34	-34	-23	-19	-19	-26	-26
T	-52	-52	-41	-41	-30	-19	-15	-22	-29
A	-59	-59	-48	-37	-37	-26	-22	-11	-18
C	-7	-14	-21	-28	-28	-33	-29	-18	-7

Table 4 - Sample F Matrix
The first step of the Needleman-Wunsch algorithm is to calculate the *F* matrix. The matrix is computed using the two sequences being compared and the similarity matrix for comparing elements. Once the matrix is built, the optimal alignment can be found by tracing the path of maximal costs from the bottom right of the *F* matrix to the top left. Diagonals represent a matched pair of elements. Orthogonals represent insertions or deletions.

The result of aligning the two sequences is shown in Table 5. The Needleman-Wunsch algorithm guarantees that this alignment is optimal based on the similarity matrix and given sequences. In other words, the total score computed by summing the pairwise similarity scores is maximal.

X:	-	A	G	A	C	T	A	G	T	T	A	C	Total
Y:	T	C	G	A	-	T	-	-	T	T	A	C	Score
Score:	**-7**	**-7**	**4**	**4**	**-7**	**4**	**-7**	**-7**	**4**	**4**	**4**	**4**	**-7**

Table 5 - DNA Sequence Alignment
The alignment of the DNA sequences "AGACTAGTTAC" and "TCGATTTAC" is shown above. The first two rows show the sequences being aligned with dashes inserted corresponding to gaps in the alignment. The bottom row is the similarity between each pair of elements between sequences in the alignment. Because gaps and mismatches both count for a -7 score, it is sometimes more beneficial for elements to be mismatched than to be marked as gaps. As an example, compare the first A and third A in sequence X. The first A is a mismatch because neither the A in X nor the C in Y have corresponding matches. Gapping both would be a total penalty of -14 while grouping them is only a score of -7. The third A in X has no nearby unmatched partner in Y and so is identified to be a gap.

3.2.2 Story Domain

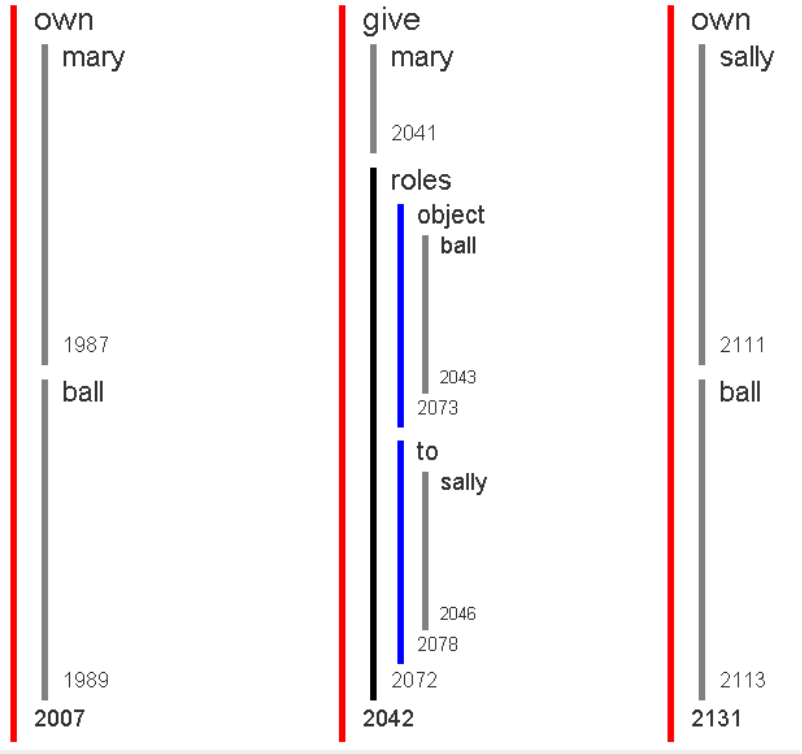
The Needleman-Wunsch algorithm works extraordinarily well for DNA where the number of types of elements is small and static. Moving into the story domain requires much greater complexity. As I explained previously, in the Genesis inner language implementation, story elements are built up from the basic building blocks of **Things, Derivatives**, **Relations**, and **Sequences**. These basic units can be nested within each other, creating story elements of arbitrary complexity and theoretically infinite depth. Therefore, creating a complete similarity matrix over all possible story elements is impossible. In order to accommodate this shift in domain, the original Needleman-Wunsch is insufficient. However, I can leverage it as a simple and fast backbone for aligning sequences with known similarity matrices.

In order to deal with the high complexity of the story domain, the first step is to generate a constrained similarity matrix. Instead of using a hypothetical complete similarity matrix, a partial similarity matrix can be generated on a story by story basis. In order to do this, first I must first demonstrate how to compute the similarity between any given pair of story elements.

Computing the similarity between two story elements can be done by analyzing the underlying structures of the elements within Genesis. I developed an initial quick comparison method which uses recursion to compare the Genesis components, **Things**, **Relations**, **Derivatives** and **Sequences**, that make up the element. Recall that these components contain the nouns, verbs, and other structural and representational information contained within the story element (See examples in Figure 11 below.) If the complete structures formed by the underlying components of the story elements are the same, the similarity returned is 1. If there are any differences in structure or entities of the element, the similarity is 0. This binary similarity metric serves for explaining the basic alignment algorithm. However, it is not powerful enough

to capture the level of detail necessary to truly compare story elements. Therefore, I replace it with a more robust function in Chapter 4.

Now that the similarity between story elements can be calculated, a similarity matrix consisting of just the events occurring in the two stories can be computed. The ability to generate a similarity matrix for the story elements existing in a pair of stories is a powerful tool as it enables us to align the two stories. Table 6 shows a simple example of the story alignments this technique enables.



Story 3.A: "Mary has the ball. Mary gives the ball to Sally. Sally has the ball."

Story 3.B: "Mary gives the ball to Sally. Sally has the ball."

Figure 11 - Visualization of Simple Story for Alignment
The story elements "Mary has a ball.", "Marry gives the ball to Sally.", and "Sally has the ball." are shown above in the Genesis inner language. Story A and Story B are nearly identical stories except in Story B the first story element has been omitted.

Story 3.A	Mary has the ball.	Mary gives the ball to Sally.	Sally has the ball.	
Story 3.B	---	Mary gives the ball to Sally.	Sally has the ball.	Total Score
Score:	**0.0**	**1.0**	**1.0**	**2.0**

Table 6 - Simple Story Alignment
Stories 3.A, "Mary has the ball. Mary gives the ball to Sally. Sally has the ball." and 3.B "Mary gives the ball to Sally. Sally has the ball." can be aligned simply. The bottom row of the table shows the similarity score resulting from comparing the pair of story elements in the alignment above it. Not shown in the table is the fact that the similarity score from comparing "Sally has the ball," with "Mary has the ball," is a 0 because Sally is not the same entity as Mary.

However, as you might expect, this approach alone is not robust enough to handle general story alignment. Consider the following examples in Table 7, which illustrate some of the limitation of this simplistic approach.

Story 3.C: "Mary has the ball. Mary gives the ball to Sally. Sally has the ball."

Story 3.D: "Sally has the ball. Sally gives the ball to Mary. Mary has the ball."

Story 3.C	Mary has the ball.	Mary gives the ball to Sally.	Sally has the ball.	---	---	
Story 3.D	---	---	Sally has the ball.	Sally gives the ball to Mary.	Mary has the ball.	Total Score
Score:	**0.0**	**0.0**	**1.0**	**0.0**	**0.0**	**1.0**

Table 7 - Simple Story Misalignment
Two stories can be easily misaligned by the simple story element similarity function. Because the simple similarity function only recognizes exact matches, the alignment does not match what a person would logically determine to be the proper alignment. A person would likely reverse the roles of Mary and Sally between the two stories, this would give a good alignment in which all the story elements of stories can be aligned with corresponding elements from the other story.

The example in Table 7 illustrates a major problem with the current similarity function. Because the similarity function for comparing story elements looks only for exact matches, it behaves contrary to human logic. A person would likely assume the roles of Mary and Sally between the two stories would be swapped, thus allowing an alignment in which all the story elements are aligned. A possible algorithmic solution to this problem could be to relax the constraint that nouns, verbs and other representations must match exactly. In effect, the similarity function could be changed to only do a structural comparison and ignore the content such as entities and verbs. This would allow the example shown in Table 7 would align properly. However, this change would introduce new problems as illustrated in Table 8.

Story 3.E: "Mary has the ball. Mary gives the ball to Sally. Sally has the ball."

Story 3.F: "Sally has the ball. John has the gift. John gives the gift to Tim. Tim has the gift."

Story 3.E	Mary has the ball.	---	Mary gives the ball to Sally.	Sally has the ball.	
Story 3.F	Sally has the ball.	John has the gift.	John gives the gift to Tim	Tim has the gift.	Total Score
Score:	**1.0**	**0.0**	**1.0**	**1.0**	**3.0**

Table 8 - Alignment Matching Error
The two stories, 3.E and 3.F shown above align poorly as judged by a human interpreter. The alignment produced has a number of logical errors due to the simplicity of the similarity function used.

The alignment of the example stories in Table 8 showcases additional logical problems with the simple similarity function being used. First, the ball from story 3.E is aligned with both the ball and the gift in story 3.F in a way that doesn't make sense to a human reader. Second, the aligner matches the first story element from story 3.E with both of the first two elements in story 3.F equally well. This causes the alignment shown to be accepted, but the logical flow of the give, which takes place over all of story 3.E and all but the first element of story 3.F, is completely missed.

As the previous examples illustrate, a simple pairwise comparison of the elements in the sequence of a story is simply not powerful enough. The alignments produced in this manner often do not match a human's intuition as to how two stories are correlated. A human understanding of a story requires continuity between actors that my approach does not take into account at this point. In the next chapter of my thesis, I look into this problem more closely in order to determine how I can bring my system's capabilities closer to that of human story understanding.

Chapter 4: Simultaneous Matching and Alignment as a Solution to the Matching Problem

In this chapter, I discuss the matching problem, a computational complexity issue which arises whenever two stories need to be compared. My solution, the match tree, is motivated by the need for high performance. Additionally, I demonstrate optimizations I've made to increase both speed and robustness.

4.1 Matching Problem

For a demonstration of the matching problem, consider the following two simple stories in

Table 9. The stories are simple yet quickly highlight the matching problem.

Story 4.A: "Mary has the ball. Mary gives the ball to Sally. Sally has the ball. Tony has the cup."

Story 4.B: "Sally has the ball. John has the gift. John gives the gift to Tim. Tim has the gift."

Story 4.A	Mary has the ball.	---	Mary gives the ball to Sally.	Sally has the ball.	Tim has the cup	
Story 4.B	Sally has the ball.	John has the gift.	John gives the gift to Tim	---	Tim has the gift.	Total Score
Score:	**1.0**	**0.0**	**1.0**	**0.0**	**1.0**	**3.0**

Table 9 - Matching Problem
The simple stories 4.A and 4.B depict the need for entity continuity in story understanding. The alignment of the stories seems odd to a human reader because there is no continuity of the associations of the actors and objects in the stories. All the entities are matched differently each time they occur in the alignment.

The matching problem causes difficulty for many story comparison techniques. In order to understand stories, continuity of actors and objects, or entities, is of high importance. If

you're trying to understand higher level plot units, you can assume that entities exist in a continuous world. If a reader learns something about Mary early in a story the reader assumes that later mentions of Mary are referring to the same actor as before. Similarly, if Mary has the ball early in the story the reader assumes Mary has the ball later in the story until something occurs to disturb this state.

During story comparison, continuity of entities is very important. When a person imagines a give action, they might picture a first person holding the object, then handing the object to another person and finally that person holding the object. Although there are three distinct states in that story, there is a continuity of the entities involved between states. In order for any two stories to be compared successfully, the entities playing similar roles should take part in the same sorts of states and transitions between states.

One way to deal with this continuity is to construct a binding list that associates entities between whatever stories are being aligned. For the stories mentioned in Table 9, proper associations between the actors are shown in Table 10.

Entity Associations	
Story 4.A Entities	Story 4.B Entities
Mary	John
Ball	Gift
Sally	Tim

Table 10 - Example Binding List
When aligning stories, the entities in one story must be paired off with their closest match in the other story. The entities shown correspond to the best matches for the stories shown in Table 9. Each row represents a constraint that associates a pair of entities for the most effective alignment.

This set of associations, or binding list, is essentially a set of constraints on how to perform the alignment of the two stories. The binding list can be incorporated into the described

alignment algorithm by modifying the similarity function. When comparing entities in the Genesis inner language, the algorithm checks the binding list and then returns one if the entities are paired in the binding list and zero otherwise. Using this revision allows the alignment of stories 4.A and 4.B to be greatly improved as shown in Table 11.

Story 4.A	---	Mary has the ball.	Mary gives the ball to Sally.	Sally has the ball.	Tim has the cup	
Story 4.B	Sally has the ball.	John has the gift.	John gives the gift to Tim	Tim has the gift.	---	
Score:	**0.0**	**1.0**	**1.0**	**1.0**	**0.0**	**3.0**

Table 11 - Story Alignment Using a Binding List
Using the binding list in Table 10 stories 4.A and 4.B can be aligned as shown in this table. This alignment keeps the logical flow of the give actions in both stories. The alignment is additionally unaffected by the surface similarities between the two balls in the stories which share an object type but do not share their roles in the stories.

As this table shows, using a binding list greatly improves the capabilities of the story alignment algorithm. Finding the best alignment between stories requires finding the optimal set of pairings between entities in the stories. Therefore, an important step in story alignment is determining how to effectively find these sets. Unfortunately, doing a simple search proves to be ineffective as there are exponentially many possible sets of pairings of entities between stories. In to determine the optimal match set, I created the match tree data structure. The rest of this chapter covers the construction and use of this data structure.

4.2 Match Trees

4.2.1 Match Tree Creation

The match tree data structure is used to search through the match sets and find the pairings of actors which yield the best alignment between two stories. The construction of the match tree is an important part of the algorithm and so this section focuses on how a match tree is created.

A match tree begins with as a single root node. The root node contains two sets of entities, *A* and *B*, one for each complete set of unique entities from each of the two stories being compared. Additionally, the root has a value of an empty set of pairs of entities. To construct the children of a node, an entity, *alpha*, is selected from *A*, and an entity, *beta*, is selected from *B*. A new node's value is the parent's value plus the new pair of *alpha* and *beta*. This node is given the same two sets *A* and *B* as the root node minus the two entities which were chosen, *alpha* and *beta*. In the same manner, a node is created pairing *alpha* with each other entity in *B*. One additional node is created which pairs *alpha* with *null*. This node represents the pairing where *alpha* does not have a corresponding entity in the second story. This process of node creation then continues by creating children for each newly created leaf node until eventually all leaf nodes have empty sets of entities.

Once the match tree is constructed, an arbitrary match set can be examined by simply selecting a leaf node and looking at the value of the node. Each leaf node corresponds to a unique set of pairings between the actors of the stories being compared. In order to construct the full set of all match sets, each leaf node needs to be examined.

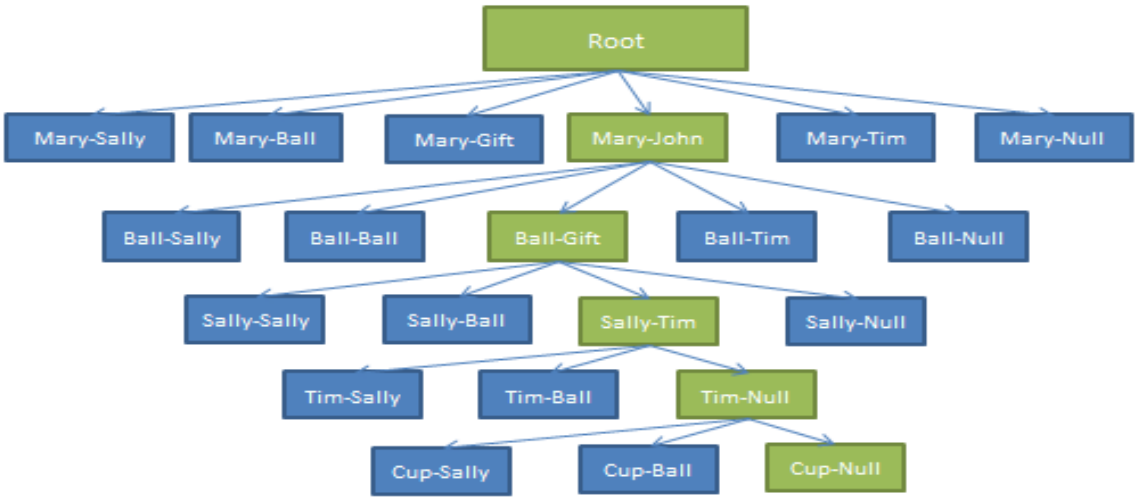


Figure 12 – Match Tree Sample
A partial match tree shown illustrates how the match tree algorithm works. The entities in this match tree are those from stories 4.A and 4.B shown previously in Table 11. Each box is one node from a match tree. The label on the box is the pair of entities added to the match set on the creation of the node. The entity on the left side of the node is an entity from story 4.A and the entity on the right side of the node is the corresponding entity from story 4.B. This tree has been pruned for space to only show the nodes leading to the optimal match set, highlighted in green.

In order to choose the optimal match set from the complete set, the system iterates over the sets and runs the alignment algorithm on the two stories for each possible set from the leaf nodes. The optimal match set is the one which yields the highest alignment score. Some pairs of stories have multiple optimal match sets which each have the high score. In these cases, there may be more than one best alignment between the two stories.

4.2.2 Brute Force Matching

Even using the Match tree described, determining the best pairings for entities across stories is a very difficult task. Given two stories with *n* entities each, the number of possible sets of pairings is O(n!). Thus, a brute force approach to this task is not scalable beyond very small sets of entities. As an example, consider the simple story, “Mary has the ball. Mary gives the

ball to Sally. Sally has the ball." (4.C). This story contains only three entities: "Mary", "Ball", and "Sally". Matching this story with itself results in the match tree shown in Figure 13.

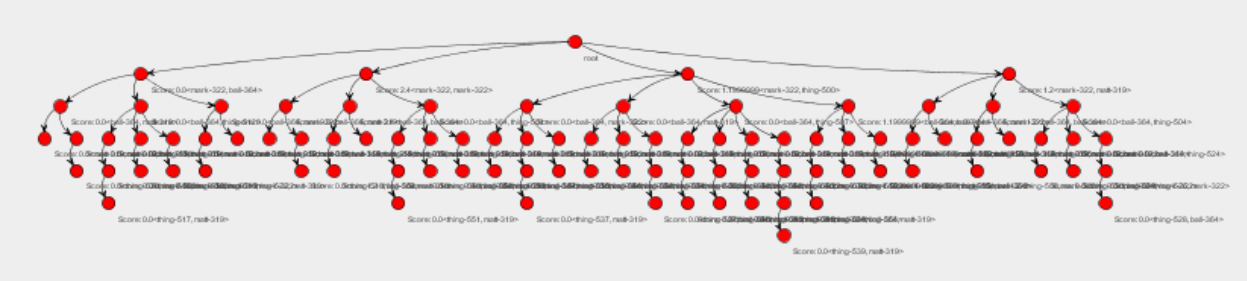


Figure 13 - Complete Match Tree
The entire match tree generated from aligning two stories containing only 3 entities each already results in a large graph. Each node represents one additional pairing relative to its parent. The root node is the node at the top of the tree. The leaf nodes are the nodes without any connected children below them. The graph has 34 leaves to search for the best match set.

This simple case of three actors per story yields a match tree consisting of 91 nodes. The match set search space has 34 leaf nodes. Each leaf corresponds to one complete match set which needs to be scored via the alignment algorithm. The brute force approach shown gets rapidly worse as the number of entities in the stories increases. A story with only 10 entities would need to construct nearly a billion nodes and would have to test a quarter of a billion match sets.

4.3 Search Optimization

In order to make story alignment feasible for large, complex stories, the algorithm needs to be able to find the best match sets as efficiently as possible. The remainder of this chapter is dedicated to optimizations to the match tree algorithm which reduce the expected run time to $O(n^2)$ where n is the larger of the number of entities between the stories being aligned.

4.3.1 Alignment Bounded Search

The first and most critical improvement to the match tree algorithm is the inclusion of an alignment bounded search. In this section, I demonstrate a technique that allows the use of the alignment algorithm to direct a fast search through the match tree in order to find the optimal match set.

Recall that each node in the match tree stores a set of all the entities which have been paired. This match set can be considered a partial match set for all non-leaf nodes because only some subset of entities has been paired. This match set is always the same as the parents match set except for the addition of one additional pair of entities. Adding to this information, I now include a score value with each of the nodes in the tree. After determining a new node's partial match set, the alignment algorithm is run on the two stories using the match set of the node. The score output by the alignment algorithm is used as this node's score.

As the alignment being run does not have a complete match set, some entities between stories are not yet paired off. Because of this, the alignment calculations are allowed to be "loose", unpaired entities are allowed to match with any entity from the other story and these matches are allowed to vary between events.

Partial Match Set	
Story 4.A Entities	Story 4.B Entities
Mary	John
Tim	Tim

Story 4.A	---	Mary has the ball.	Mary gives the ball to Sally.	Sally has the ball.	Tim has the cup	
Story 4.B	Sally has the ball.	John has the gift.	John gives the gift to Tim	---	Tim has the gift.	
Score:	**1.0**	**0.0**	**0.0**	**0.0**	**1.0**	**2.0**

Table 12 – Alignment with Partial Match Set
During alignment bounded search, some of the entities between stories are tightly bound while others are allowed to be loose. Unbound entities are matched loosely meaning they can match with any other unbound entity. The gift in story 4.B is allowed to match with both the ball and the cup from story 4.A, while Mary and Tim are locked into matching with John and Tim respectively.

The looseness of the alignment algorithm allows me to exploit a powerful property of our newly scored match tree. Consider that in the match tree each new level of depth is guaranteed to add stronger constraints to the entity matches. In turn, this means that the possible alignments for a child can only become more constrained, never less so. While adding an additional pair of actors to the match set may cause the algorithm to reject an alignment it previously accepted, the reverse cannot occur. Therefore, the score of any node in the tree acts as a upper bound for the possible scores of all that node's descendants.

Using the upper bound heuristic, the original tree generation algorithm can be modified to search for the optimal leaf without needing to generate the entire tree. The modification takes a greedy approach and includes the possibility for backtracking. Whenever a node is created, that node is added to a priority queue which outputs nodes in order of highest score. The algorithm

starts with the root node that is assigned a maximum score value and placed into the queue. At each step of the algorithm, the node with the highest score is removed from the queue. Next, all of this node's children are generated, alignment scored, and then added to the queue. This process of simultaneous matching and alignment repeats until the algorithm draws a leaf node from the queue. The leaf node is guaranteed to have the best match set for aligning the two stories.

Once a node with a complete match set is generated, it is selected as the optimal match set because it has the highest possible alignment score. At this point, the algorithm can stop and return the best alignment or it can be allowed to continue finding more match sets in ranked order of how well they allow the stories to align.

4.3.2 Example of Simultaneous Matching and Alignment

As an example, consider the stories 4.A and 4.B that that I have used throughout this chapter. The stories share a set of three story elements comprising give actions which need to be aligned: the initial object ownership, the object transfer, and the new object ownership. Each story has five entities. This means that a full match tree would have 4,725 nodes to construct and 1,546 different match sets to search through. Using the simultaneous matching and alignment, however, yields a significant reduction in size of the search space. Figure 14 shows the match tree resulting from using the simultaneous matching and alignment technique on the stories.

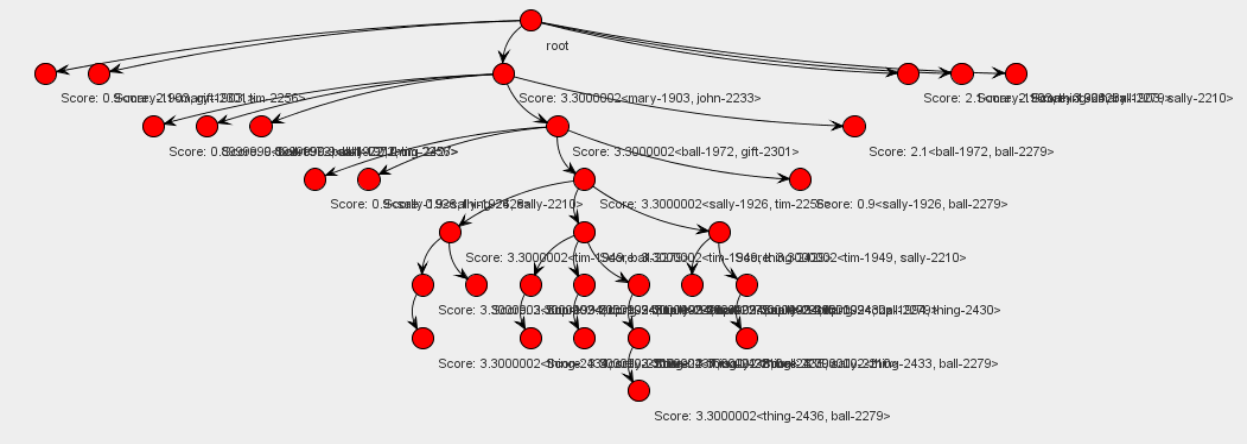


Figure 14 - Simultaneous Matching and Alignment
Simultaneous matching and alignment significantly improves the matching algorithm from the brute force approach. The result shown is from performing simultaneous matching and alignment on the stories 4.A and 4.B. The stories contain 5 entities each. A full match tree would contain 4725 nodes. The simultaneous matching and alignment technique allows the discovery of the best match set and alignment by constructing only 31 nodes.

As seen in the figure, the discovery of the best match set and thus the optimal alignment only requires the construction of 31 nodes. This reduction in the size of the match tree greatly decreases the time needed to align and compare stories. In many cases, such as when the two stories have a particularly good possible alignment, the search only needs to traverse down the tree without ever backtracking which allows the best leaf node to be found in $O(n^2)$ time where n is the number of entities in the stories.

4.3.3 Improved Score Metrics

The simultaneous matching and alignment bounded search I described greatly improves the ability of the matcher to find the optimal match set. However, for many complex stories the differences in alignment score may be very small. This is due to the binary similarity measure used in comparing story elements. Ideally, the alignment and node scores should be as descriptive as possible. Scores from comparing story elements should form a continuous scale in which a low score means story elements match poorly and a high score indicates the elements

match well. In order to do this, I modify the previously described binary scoring metric used to compare story elements. Initially, this function simply returned 1 for matching elements and entities, and 0 otherwise. Now, however, a similarity factor is generated on the scale of 0 to 1, with 0 being entirely different and 1 being exactly the same. In order to accomplish this, I leverage the rich semantic information Genesis attaches to entities.

Within the Thing representation in Genesis, one piece of important classification information is the tread representation. The thread representation is based on the concept of thread memory by Lucia Vaina (Vaina and Greenblatt 1979). This representation is capable of storing classifications, features, and definitions in a thread which is a set of string descriptors related to the entity. Genesis uses WordNet (Stark and Riesenfeld 1998), a large database of definitions and word associations in order create the thread representations. Entities can have multiple threads relating to what role the entity fulfills in different contexts. For example, the word "hawk" might have features such as "animal" and "bird" in one thread related to nature, but have features such as "hunter" and "person" in a thread related to politics. Each Thing has a primed thread which Genesis has marked as most relevant to the story at hand.

Using the threads, I can compute a similarity metric between two entities in a story by counting the number of matched elements in the entities' primed threads and dividing the result by the maximum length of the entities' threads. The binary entity match previously used in the recursive structure matching algorithm is replaced with the similarity score. The scores computed from the recursion are multiplied together to determine a final similarity score comparing any two story elements. Table 13 shows a number of example threads and Table 14 shows a set of computed similarity scores generated using the thread comparison function.

Entity	Primed Thread
Cat	thing entity physical-entity object whole living-thing organism animal chordate vertebrate mammal placental carnivore feline cat
Lion	thing entity physical-entity object whole living-thing organism animal chordate vertebrate mammal placental carnivore feline big-cat lion
Person	thing entity physical-entity object whole natural-object body human-body person
Car	thing entity physical-entity object whole artifact instrumentality conveyance vehicle wheeled-vehicle self-propelled-vehicle motor-vehicle car

Table 13 – Threads
Threads generated are from WordNet in Genesis. The length and quality of the thread information is highly dependent on the amount of detail available in WordNet for that particular word.

Entity	cat	lion	person	bear	airplane	car	truck	bus
cat	1	0.875	0.466667	0.866667	0.333334	0.333334	0.333334	0.333334
lion	0.875	1	0.4375	0.8125	0.3125	0.3125	0.3125	0.3125
person	0.466667	0.4375	1	0.5	0.384615	0.384615	0.384615	0.384615
bear	0.866667	0.8125	0.5	1	0.357143	0.357143	0.357143	0.357143
airplane	0.333334	0.3125	0.384615	0.357143	1	0.692308	0.692308	0.615385
car	0.333334	0.3125	0.384615	0.357143	0.692308	1	0.923077	0.615385
truck	0.333334	0.3125	0.384615	0.357143	0.692308	0.923077	1	0.615385
bus	0.333334	0.3125	0.384615	0.357143	0.615385	0.615385	0.615385	1

Table 14 - Score Matching
The similarity between any two entities can be calculated using the implemented match scorer, which takes advantage of the WordNet threads available in Genesis. A value of 1 implies the objects are as similar as possible and a value of 0 implies the objects are as different as possible when using this metric.

The alignment algorithm and in turn the match tree generation immediately benefit from this addition of a scored story element comparison method. Each alignment score automatically incorporates the richer similarity scoring for comparing actors in the stories. Additionally, the increased variance of the alignment scores allows the match tree to be pruned more quickly and accurately via the simultaneous matching and alignment method.

4.3.4 Minor Tweaks and Optimizations

In order to make the system run as efficiently as possible, a number of smaller additions and tweaks have been implemented into the alignment code.

Calculating the scores between entities in stories is not a trivial computational step. The improved scoring algorithm has to perform an $O(p^2)$ search comparing the primed WordNet properties of each actor. Because this step is performed at least once per story element comparison, this becomes a computational bottleneck. Fortunately, this can be easily accommodated by caching all score calculations made between each pair of actors. This is allowable because once acquired for a particular entity the WordNet definitions is constant for the duration of the story comparison.

The simultaneous matching and alignment algorithm I presented in this chapter performs very well under most conditions. However, it still has an exponential worst case run time. This typically only occurs if the two stories being compared have little or nothing in common. In this case, the simultaneous alignment and matching would be unable to find pairs of entities that match well because there simply would not be enough information to evaluate pairs. Under these circumstances, a polynomial running time is guaranteed through the introduction of a threshold variable. Essentially, the priority queue is restricted to holding only a limited number of nodes. If this threshold size is set to one node this guarantees that the algorithm runs in polynomial time compared to the total number of entities in the stories. With this change, the algorithm becomes a purely greedy variant which always chooses the best child node at each depth and throws away all other nodes which guarantees a $O(n^2)$ runtime. Even with the loss of accuracy from the lack of backtracking up the match tree, this greedy variant performs well. Examples of these improvements are shown in Chapter 6.

The simultaneous matching and alignment algorithm presented in this chapter forms a major part of my thesis work as it allows for stories to be aligned quickly and accurately. In the next chapter, I demonstrate an additional improvement made to my story comparison technique which leverages additional higher level knowledge in order to improve the robustness of the comparison and make the results more human-like.

Chapter 5: Reflective Alignment

Humans use a vast amount of literary tools to compare and analyze stories. As mentioned previously, one important tool for analysis is higher level reflective knowledge. Reflective knowledge is the knowledge of themes and expected overarching plot units that occur during a story. Incorporating reflective knowledge into story comparison should improve both the comparisons and alignments as well as making the comparisons more human-like. This chapter describes how reflective knowledge has been incorporated into the story alignment.

As seen in Chapter 2, Genesis has the capability to identify plot units present in stories. In order to include this information for story comparison, many of the techniques previously described for alignment and matching can be reused.

5.1 Plot Unit Alignment

The first step is to determine the correspondence between the plot units occurring in each of the stories. To do this, my algorithm performs an alignment of the plot units between the stories. Because plot units span multiple story elements and can potentially overlap, the plot units are assumed to occur at the time of the last element of the plot unit for the purpose of alignment. Alignment of the plot units proceeds via Needleman-Wunsch with matches receiving a 1.0 score and gaps receiving a 0.0 score.

Once an alignment is acquired, the algorithm pairs off the plot units that best match between the stories. The next step is to look at the underlying story elements that make up the plot units. Each plot unit can actually be thought of as its own small story which is a subset of the larger story. The story alignment and matching algorithm can be run on each pair of plot units found to match between the stories. These alignments are very fast because each of these

sub-stories has a small number of entities and story elements compared to the full stories being compared. Plot units that were not matched with a plot unit from the other story are simply ignored.

Story 5.A	---	Mistake	Revenge	Leadership Acquired	
Story 5.B	Answered Prayer	---	Revenge	Leadership Acquired	
Score:	**0.0**	**0.0**	**1.0**	**1.0**	**2.0**

Table 15 - Plot Unit Alignment
Reflective level plot units can be aligned using a modified alignment algorithm. The revenge and leadership acquisition plot units occur in both stories and align well. The "Mistake" and the "Answered Prayer" plot units do not align in this case.

5.2 Plot Unit Matches

Once these alignments are complete, the system has a comprehensive comparison of the plot units of the stories available. The final step is to use this reflective knowledge to direct the low level story element alignment. From each of the plot unit comparisons, the algorithm obtains a set of entity matches that works well in aligning the story elements of the plot units. Because the plot units highlight the important aspects of the stories, these entity matches should be emphasized in the global story comparison.

Revenge 5.A	Scar kills Mufasa.	Scar harmed Simba.	Simba kills Scar.	Simba harmed Scar.	
Revenge 5.B	Macbeth kills Duncan.	Macbeth harmed Macduff	Macduff kills Macbeth	Macduff harmed Macbeth	
Score:	**1.0**	**1.0**	**1.0**	**1.0**	**4.0**

Table 16 - Plot Unit Matching and Alignment
Once the plot units themselves are aligned, the system aligns the story elements that compose them. This matching and alignment for the two revenge plot units shown helps to guide the global story alignment. The pairings from this story which include Scar binding to Macbeth and Simba binding to Macduff can be used in aligning the full stories.

Before the matches can be included, they first need to be sanitized. Recall that the story alignment algorithm must explicitly pair entities with a null entity if a match for that entity does not exist in the other story. Because the plot units only contain subsets of the actors in each of the stories, the null pairings may be too strict. Therefore, only the successful matches between entities in the plot units are kept for use in story alignment.

Revenge 5.A	**Revenge 5.B**
Simba	Macduff
Scar	Macbeth
Mufasa	Duncan
Nala	---
---	Lady Macbeth

Table 17 - Plot Unit Bindings
One potential match set created by aligning the revenge plot units between the two stories is shown above. In this case the bindings for Simba, Scar, and Mufasa are kept because they have been explicitly matched to an entity in the other story. Nala and Lady Macbeth did not get matched to corresponding entities in the alignment. Therefore these null bindings formed. These bindings should be thrown out so potentially better matches for Nala and Lady Macbeth can be found during the full story alignment.

5.3 Directing Story Alignment

Next, the good matches from the plot unit alignment are incorporated into the full story alignment. The match tree used in story alignment always starts with a single root node which sets up how the rest of the algorithm proceeds. This makes it a convenient point at which to insert the information from plot units. Each pair of entities matched in the plot unit analysis is added to the initially empty match pair set of the root node. The entities involved in each of these matches are removed from the entity lists for the stories. After this is done, the story alignment is allowed to proceed as normal. This spoon-feeds the important matches from the plot units into the match tree and then uses the low level story alignment to match and align the rest of the entities and story elements.

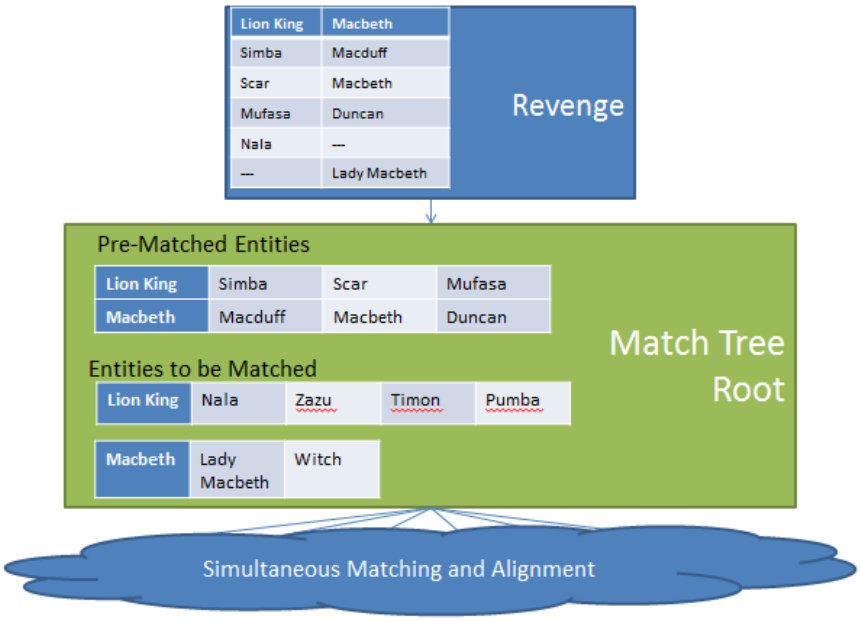


Figure 15 - Guided Match Tree
The general process through which reflective knowledge is used to guide alignment flows from the alignment of the high-level plot units through a process that gives initial bindings for the story alignment algorithm. The top box shows the matches made during the analysis of the revenge plot units occurring in the stories. The non-null matches from this analysis are used as the set of pre-matched entities in the match tree root. The rest of the actors are matched using the simultaneous matching and alignment method.

Using this technique, the simultaneous matching and alignment algorithm now includes Reflective level knowledge as an important criterion for story comparison. The advantages of this inclusion are two-fold. First, this allows the story comparison algorithm to easily focus on story aspects that are deemed to be more interesting to humans. Second, by pruning the entity search space, this technique increases the speed of the story alignment. In the next chapter of this thesis, I demonstrate examples of the reflective alignment technique and the general effectiveness of the simultaneous matching and alignment algorithm.

Chapter 6: Applications and Continuing Efforts

In order to test and demonstrate the capabilities of the story comparison techniques I've developed, I turn to the Genesis story corpus. Within the Genesis group, we have created and accumulated a reasonably sized set of stories that span genres, themes, topics, and complexities. This chapter showcases the capabilities of my techniques on a number of story comparison problems.

6.1 Identifying Cultural Differences

The interpretation of stories in writing, video, and the world around us, depends greatly on the interpreter. Thus, an interest of Genesis group and story understanding as a whole is to develop models for comparing how people of different cultures, politics, backgrounds, and genders might understand a story. Working towards this goal, we have developed varying sets of common sense knowledge that can be used to represent varying points of view.

One canonical example the Genesis group uses is two differing interpretations of Shakespeare's Macbeth. A typical reader likely identifies an instance of revenge when Macduff kills Macbeth after Macbeth kills Macduff's friend, Duncan. However, it's possible that a second person reading the story might not know the concept of revenge and so may interpret the act differently. To that person, Macduff attacks and murders his friend Macbeth. The second person classifies the same murder as an act of insane violence because he knows no other explanation.

Putting two interpretations of the entire story of Macbeth through Genesis and in turn the story alignment algorithm represents a computationally intensive task. In its current version, the story contains 32 entities and uses 21 commonsense rules for causal understanding. Without

using the optimizations and the simultaneous matching and alignment techniques discussed in this thesis, aligning the two versions of the story is a computationally infeasible task. Brute force determination of the complete match set requires searching through over 10^{30} possible match sets. The simultaneous matching and alignment algorithm allows the match tree's construction to be efficiently directed. The match tree that results requires only 546 nodes to be constructed and is shown in Figure 16.

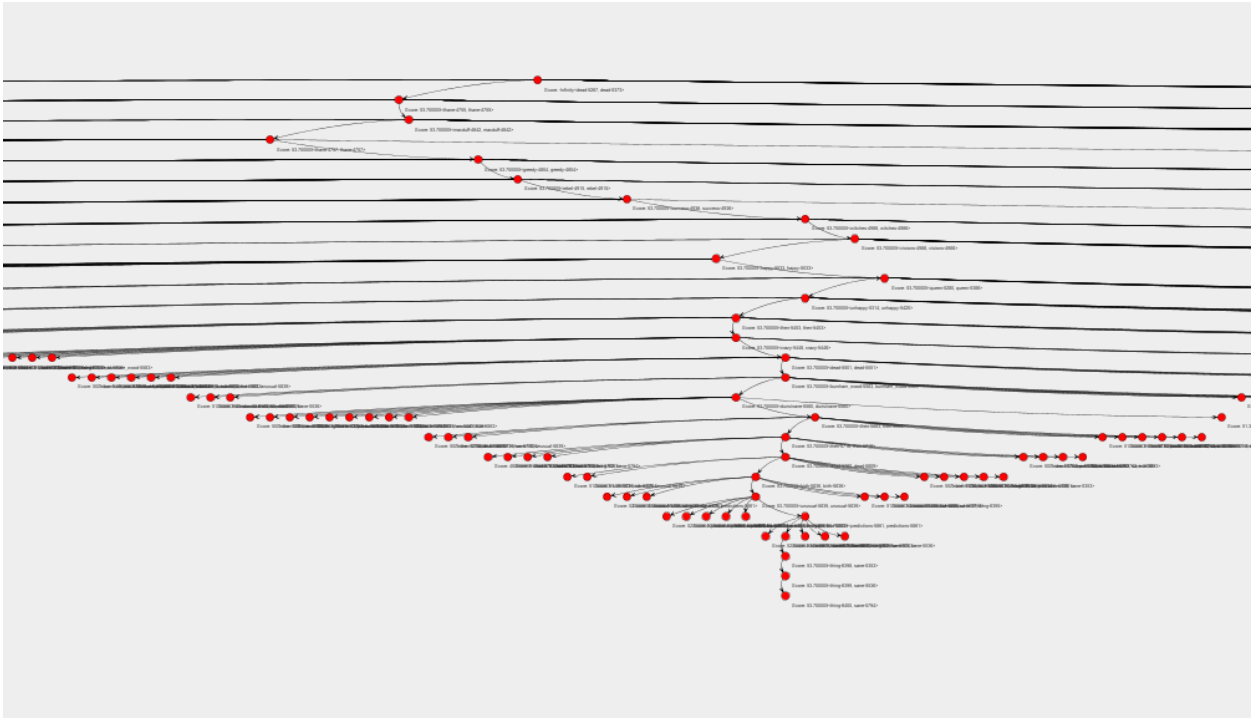


Figure 16 - Macbeth Match Tree
A small portion of the match tree generated by comparing two different interpretations of the Macbeth story appears above. The full tree contains 546 nodes which were searched in order to find the match set yielding the best alignment. This partial view of the tree shows the root node at the top and the path that is followed to get to the leaf node with the best complete match set, visible in the figure as the lowest node on the tree.

Element 51	Element 52	Element 53	Element 54	Element 55	Element 56
Macduff kills Macbeth.	Macduff kills Macbeth. because Macduff isn't sane..	Macduff isn't sane.	---	---	Macbeth becomes dead. because Macduff kills Macbeth..
Macduff kills Macbeth.	---	---	Macduff kills Macbeth. because Macbeth angers Macduff..	Macbeth angers Macduff.	Macbeth becomes dead. because Macduff kills Macbeth..

Figure 17 - Revenge/Insanity Alignment
The alignment algorithm can find differences in understanding between two different readings of a common story. The occurrence of a gap in one story followed immediately by a gap of similar size in the other story can represent a difference in understanding when the stories share a common base but were interpreted using different knowledge sets.

The alignment algorithm also identifies the differences in background knowledge used while interpreting the stories. Figure 17 shows one part of the Macbeth tale that has been found to have been interpreted differently. The revenge and the act of insane violence are aligned to be adjacent with one another with a corresponding gap of information in the aligned story.

Additionally, this comparison is a good example of how including plot level knowledge can both improve the story comparison and the efficiency. Although major plot units in the stories - the revenge and the act of insane violence - do not align, most of the minor plot units still do.

Success	Answered Prayer	Mistake	Leadership Acquired	---	---	Pyrrhic Victory	Suicide	---	Revenge
Success	Answered Prayer	---	Leadership Acquired	Insane Violence	Insane Violence	---	Suicide	Insane Violence	---

Table 18 - Plot Unit Alignment of Macbeth
The plot units between two different interpretations of the Macbeth story are aligned in the above table. The alignment algorithm is able to properly align the reflective knowledge and use the entity matching from these plot units to match the entities in the overall stories.

Using the reflective alignment, Macbeth, Lady Macbeth, Duncan, and a number of smaller entities are matched before the larger stories are compared. This, in turn, reduces the size of the full match tree by about 33% from 546 partial match nodes to 363 nodes.

6.2 Predicting Future Events

Another goal of computational story understanding is using stories to make intelligent predictions about future events. In order to test our systems capabilities in this regard, we have built up a corpus of conflict stories, which are narratives about various wars, attacks, and conflicts that have occurred throughout history. One useful application for story alignment in this domain is to align partial stories of conflicts to see what future events can be predicted from historical occurrences.

Element 1	Element 2	Element 3	Element 4	Element 5
The Israelis know the Egyptians prepare to attack them.	The Israelis know to defeat the Egyptians.	The Israelis know that the Egyptians know that they defeat the Egyptians.	The Israelis believe that the Egyptians don't attack them.	---
The USA knows that the Viet Cong prepares to attack it.	The USA knows to defeat the Viet Cong.	---	The USA believes the Viet Cong not to attack it.	The Viet Cong attacks the USA.

Table 19 - Tet Offensive and Yom Kippur War
The partial summaries of the Yom Kuppur War and the Tet Offensive have been aligned and compared using the alignment algorithm. Additionally, for this experiment, both summaries have had some details omitted. The two conflicts align well which is expected as both begin with unexpected pre-emptive strikes, despite the information that has been removed from the stories.

Table 19 shows the alignment of two conflict stories available in Genesis. The first is some of the events leading up to the Yom Kippur war. The second outlines the lead up to the Tet Offensive. Both stories have missing information, but align pretty closely, which is expected for two conflicts that begin with surprise attacks.

I also use this example to demonstrate the collaborative gap filling feature of the alignment algorithm. Once two stories have had their entities matched and story elements aligned, the algorithm can attempt to fill the differences or gaps between the two stories. In the case of these stories, this technique allows the prediction of future events in a story based on

Genesis's knowledge of other stories. Table 20 shows this collaborative gap filling along with using the English generator to display the results in simple English.

Element 1	Element 2	Element 3	Element 4	Element 5
The Israelis know that the Egyptians prepare to attack them.	The Israelis know to defeat the Egyptians.	The Israelis know the Egyptians know that they defeat the Egyptians.	The Israelis believe the Egyptians not to attack them.	The Egyptians attack the Israelis.
The USA knows the Viet Cong prepares to attack it.	The USA knows to defeat the Viet Cong.	The USA knows the Viet Cong knows that it defeats the Viet Cong.	The USA believes the Viet Cong not to attack it.	The Viet Cong attacks the USA.

Table 20 - Gap Filled Yom Kippur and Tet Offensive
Alignments of two stories can be used to imagine missing events. For example consider the Yom Kippur War the Tet Offensive summaries used previously. The green boxes are events that did not exist in the original stories but were extrapolated based on the knowledge of the other parallel story. Additionally, this table uses the English generator to display the results in simple English.

6.3 Input Robustness

Until now, I have focused the discussion of my thesis work on the understanding and comparing of stories that have been provided in English text. However, human intelligence is multimodal and works across a number of domains including, but not limited to, language, vision, sound, and tactile sensation. In order to advance towards the goal of truly intelligent programs, our systems need to be capable of reasoning across these domains. The rest of chapter shows how I've taken steps to apply my ideas to visual information and combine reasoning across the language and visual domains.

6.3.1 Mind's Eye

Mind's Eye is a large scale research project that aims to develop systems capable of understanding videos in which human actors and objects interact. One of the milestone goals is a system capable of identifying and describing videos which contain examples of verbs from a collection of 48. These verbs have been deemed as being important to visual understanding and feasible within the current state of technology (DARPA 2010). Some of the Mind's eye goals are

rooted in the ideas of Sajit Rao's work in describing videos using attentional routines (Rao, Visual Routines and Attention 1998).

Figure 18 - Mind's Eye Video
Mind's Eye videos have been analyzed by Sajit Rao. (Rao, Yuret and Winston, Vision-Language-Learning Video 2008) This portion of the video shows the blue man giving the green ball to the red man. The goal of the Mind's eye project is to be able to understand such videos on a story level.

One of the more ambitious goals of the Mind's Eye project is to be able to fill gaps in broken or noisy video feeds. For example, imagine you are watching a video in which two people are standing near each other and one of them is holding a coffee mug. Then the video feed cuts out to black for a few seconds. When it resumes the two people are still standing near each other but the other person is holding the mug. What may have happened? A plausible explanation is that the first person gave the mug to the second person. Another explanation could be that the mug was taken by, or thrown to the second person. Most people would agree that these descriptions would be more likely than the people jumped up and down, even though that is certainly possible. In order to do this analysis computationally, my story comparison techniques are valuable.

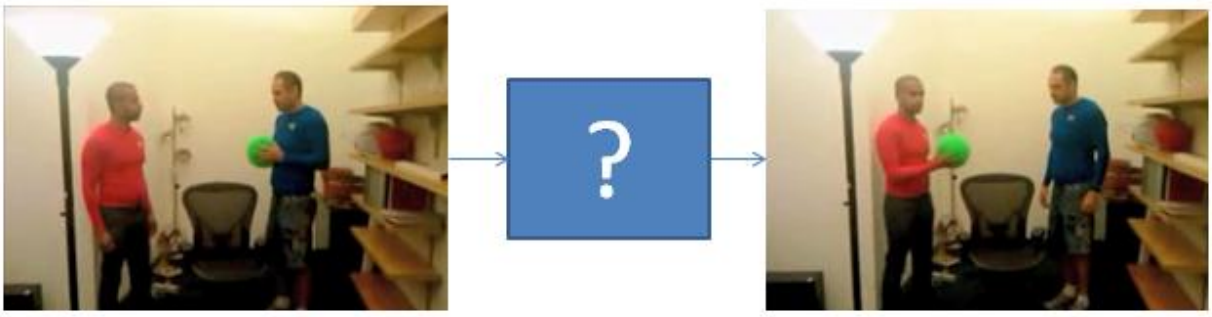


Figure 19 - Mind's Eye Gap
One aspect of Mind's Eye is the gap filling problem. In this example, the video shows the blue man holding the ball and then approaching the red man. Then the video feed is lost. When the video resumes the red man is holding the ball. The goal is to determine what happened during the lost frames.

In order to work with Mind's Eye videos, MIT CSAIL and the company Co57 have been developing a video analysis pipeline. A Co57 system called Beryl, developed by Sajit Rao, works directly on the videos and generates attentional traces. These traces are a representation of the people, their primary body parts, and the objects in the video and how they spatially relate to one another over time (Rao, Visual Routines and Attention 1998). At this time, Beryl also has preliminary capabilities for learning and identifying basic actions, such as "makes contact" and "jumps". This processed video data is next passed into a system developed by Gary Borchardt called Impact (Borchardt 1993). One of the goals of Impact is to use human understandable representations of actions to translate attentional trace data into more complex actions that are taking place. For example, a "give" may consist of Actor A being in contact with Object B, Actor C coming into contact with Object B, and then Actor A losing contact with Object B. Once Impact has created a timeline of events occurring in a video, the events are converted into the Genesis inner-ese.

Once in Genesis, all the videos can be analyzed using the tools readily available for story understanding. I have had success in filling the gaps in a number of videos which had missing

data using my story comparison techniques. The process for gap filling videos proceeds as follows. First, a training set of videos is fed into Genesis through Beryl and Impact. These video timelines are stored in a library of known actions. Next, a video containing a gap is analyzed. The video can be run through the alignment algorithm, finding a best match alignment with each of the videos contained within the library. The scores of the alignments between the gapped video and the library videos are compared and the alignment that yields the highest correlation is selected to fill the gap. The gap is filled with imagined story events created by combining the information in the match set used for alignment and the story events from the video library.

As an example, consider the missing "Give" action from Figure 19. After watching a data set of videos, the alignment gap filler takes in the video with a gap and aligns it with the training videos. The best match is used to fill in the gap.

Gap Story	Blue man has the ball.	Blue man approaches Red man.	A gap appears.	---	Red man has the ball.	
Give Story	Alpha has the object.	Alpha approaches Bravo.	Alpha gives the object to Bravo	---	Bravo has the object	
Give Score	1.0	1.0	1.0	0.0	1.0	4.0
Take Story	Alpha has the object.	---	Bravo approaches Alpha	Bravo takes the object from Alpha	Bravo has the object.	
Take Score	1.0	0.0	1.0	0.0	1.0	3.0

Table 21 - Video Gap Filling Example
A number of good matches can exist for an example gapped video. The best matches were selected by comparing the video in question with a library of videos containing multiple examples of nine different verbs. In this case "Give" was found to be the best match and Take was found to be a reasonable second best match.

This technique was tested on a set of 9 example gapped videos which were passed through developmental versions of Beryl, Impact, and Genesis. These videos were tested against a library containing 51 total videos containing a scattering of examples of the verbs in question. The best alignments for each gapped video were ordered by best match and then checked to see how close actual missing element was from the top of the list of matched videos. The results of this experiment are shown in Table 22.

Verb Obscured From Video	First Exact Verb Match Ranking
Approach	2
Bounce	4
Catch	3
Drop	5
Follow	1
Give	1
Pickup	1
Putdown	1
Take	1

Table 22 - Seedling Gap Filling Experiment
Preliminary gap filling analysis has been done on seedling video experiments. The left column shows the verb which had its frames cut from the video. The right column shows the ranking of the of the proper verb based on alignments from the video library of 51 videos. A 1 indicates that the best match from the library correctly filled the missing verb. A 2 indicates that the second best match from the library was the first to correctly fill the missing verb, and so on.

A lower number indicates that the best match chosen by the algorithm correctly filled the gap in the videos. The results are mixed. Some of the verbs, such as "Follow", "Give", and "Pickup", perform very well. Other verbs, such as "Drop", perform much worse. Upon further investigation, the poor results seem to be due to the quality of the processed video library currently available. The current examples and test videos often contain a great deal of noise. An early pilot study underway in the Genesis group indicates that the videos are not easily categorized by primary verb even by a human observer. This seems to be due to the fact that the

videos often have multiple actions occurring within a short time frame. Continued work on improving the video processing and obtaining higher quality videos is currently underway.

6.3.2 Language and Vision

One of the advantages of working in the story domain is that it gives our system a great deal of power to not only work on multiple domains but to actually use the domains in combination to achieve more robust reasoning. Stories learned from watching videos can be compared to those learned through language. Scenes that have never before been seen can be imagined from textual descriptions Genesis has previously read.

Part of the goal of the systems involved in our video pipeline is for the computational understandings of actions to be closely related to the human understandings. After Beryl processes the video data, it is passed for further analysis to Impact and Genesis. Because the representations of Genesis can be mapped directly to and from English, video data can be compared to any other input regardless of its source. Thus, the video library used for gap filling in Genesis does not even need to be constructed from a single input channel. Actions, video descriptions, and other stories can be constructed in natural language and become automatically available for completing event sequences from any source.

As an example of this utility, consider the video gap filling example from Table 21. Although the video with a missing "Give" was filled using a library of video data, this library could have been populated with additional stories from textual descriptions of events.

6.4 Story Clustering

The ability to construct a library of stories from a variety of different sources is a very powerful tool. However, its effectiveness is limited if it cannot be searched through and used

effectively. As the number of stories in the library grows, the computational time needed to find the best match continually increases. One way to mitigate this negative effect would be to group stories by their similarity so that you can quickly prune the story space by types of story. To that end I've begun preliminary work to allow stories to be clustered based on how well they align.

I've implemented a K-Clustering method which uses the simultaneous matching and alignment algorithm to compute a similarity metric for comparing stories. My implmentation proceeds via the well-known Lloyd's algorithm (Lloyd 1986). Once the stories are clustered by similarity, the story with the highest similarity score to the rest of the stories in its cluster is chosen to be the canonical example of that cluster. In the future this means that in order to perform gap filling using a story library, you could compare the gapped video to each of the canonical videos for the clusters and then against the videos of the best matched cluster. This process could be expanded to a hierarchical library in the future in which there are levels of clusters to search down through.

The current implementation of the clustering algorithm has been tested against a mockup set of stories that resemble stories expected from the Mind's Eye project. The results of the clustering are shown in Figure 20.

Story	Symbol
Exchange	A
Flee	B
Give	C
Mediated Exchange	D
Take	E
Follow	F
Throw-Catch	G

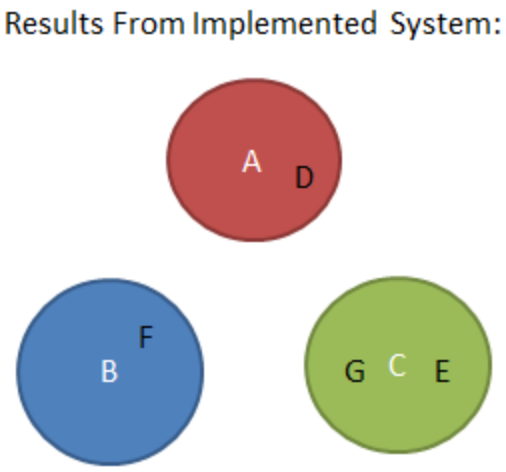


Figure 20 - Clustering Stories
Story alignment can be used to power story clustering. The clusters generated by the implemented system are shown in the colored circles. The list of sample stories clustered is shown in the table on the left along with the identifier for each story.

In this chapter, I discussed a number of applications for my simultaneous matching and alignment algorithm in the domain of story understanding and beyond, including a number of prototype examples that have been made available in the Genesis code base. In the next chapter, I conclude my thesis with the research contributions I have made while undertaking my thesis work.

Chapter 7: Contributions

In my research, I have made the following contributions:

- I **designed** an algorithm for comparing stories through sequence alignment. The algorithm draws from cross-disciplinary domains and succeeds in providing a fast and efficient base-line for doing story comparisons.
- I **developed** a technique for simultaneous matching and alignment to solve an otherwise computationally infeasible problem. My solution to the matching problem allows for the comparison of larger and more complex stories than was previously possible. In an example experiment this reduces the search space of a 32 entity problem from over 10^{30} nodes to only 546 nodes.
- I **demonstrated** the importance of reflective knowledge in story understanding and its utility in comparing stories. This research shows how higher levels of knowledge can be incorporated into a comprehensive story comparison system.
- I **implemented** all the algorithms described in this thesis into the Genesis code base, making it available to all and taking its story comparison capabilities to the next level. This enables imagination by allowing events to be imagined through story comparison.
- I **applied** my work and **analyzed** the results of my work by applying my techniques to the Genesis story corpus and to Mind's Eye videos. These applications demonstrate the broad applicability of my work and highlight the multimodal capabilities of Genesis.
- I **identified** future work in story comparison such as the potential for story clustering.

References

Borchardt, Gary. *Causal Reconstruction.* A.I. Memo, Cambridge, MA: MIT Artificial Intelligence Laboratory, 1993.

Chomsky, Noam. "Some simple evo-devo these: how true might they be for language?" *Unpublished Manuscript.* The Evolution of Language Morris Symposium, 2005.

DARPA. *Mind's Eye Program.* Broad Agency Announcement, Arlington, VA: Defense Advanced Research Projects Agency, 2010.

Eddy, Sean. "Where did the BLOSUM62 alignment score matrix come from?" *Nature Biotechnology*, 2004: 1035-1036.

Finlayson, Mark, and Patrick Winston. *Analogical Retrieval via Intermediate Features: The Goldilocks Hypothesis.* Cambridge, MA: Massachusetts Institue of Technology, 2006.

Gamma, E, R Halm, R Johnson, and J Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Boston, MA: Addison-Wesley, 1994.

Gentner, D. and Markman, A.B. "Structure mapping in analogy and similarity." *American Psychologist*, 1997: 45.

Jonassen, David H, and Julian Hernandez-Serrano. "Case-based reasoning and instructional design: Using stories to supoprt problem solving." *Educational Technology, Research and Development*, 2002: 65-77.

Katz, .B. and Lin, J. and Felshin, S. "The START multimedia information system: Current technology and future directions." *Proceedings of the International Workshopn on Multimedia Information Systems.* 2002.

Katz, Boris. "Annotating the World Wide Web using Natural Language." *RIAO Conference on Computer Assisted Information Searching on the Internet.* 1997.

—. *Start Natural Language Question Answering System.* December 1, 1993. http://start.csail.mit.edu/ (accessed December 14, 2011).

Klein, Dan, and Christopher Manning. "Accurate Unlexicalized Parsing." *Proceedings of the 41st Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, 2003. 423-430.

Lloyd, Stuart. "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, 1986: 129-137.

Morgenstern, Burkhard, Kornelie French, Andreas Dress, and Thomas Werner. "DIALIGN: Finding local similarities by multiple sequence alignment." *Bioinformatics*, 1998: 290-294.

Needleman, S.B. and Wunsch, C.D. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of molecular biology*, 1970: 443-453.

Rao, Sajit. *Visual Routines and Attention.* PhD Thesis, Cambridge, MA: Massachusetts Institute of Technology, 1998.

Rao, Sajit, Deniz Yuret, and Patrcik Winston. *Vision-Language-Learning Video.* Cambridge, MA, August 8, 2008.

Schanck, Robert. *Tell Me a Story.* Evanston, IL: Northwestern Univeristy Press, 1990.

Stark, Michael M., and Richard F. Riesenfeld. "WordNet: An Electronic Lexical Database." *Proceedings of 11th Eurographics Workshop on Rendering.* MIT Press, 1998.

Sussman, Gerald Jay, and Alexey Radul. *The Art of the Propagator.* Cambridge, MA: MIT Computer Science and Artificial Intelligence Laboratory, 2009.

Tattersall, Ian. "An Evolutionary Framework for the Acquistion of Symbolic Cognition by Homo sapiens." *Comparative Cognition & Behavior Reviews*, 2008: 99-114.

Vaina, Lucia M., and Richard D. Greenblatt. *The Use of Thread Memory in Amnesic Aphasia and Concept Learning.* Cambridge, MA: MIT Artificial Intelligence Laboratory, 1979.

Winston, Patrick H. "Learning and Reasoning by Analogy." *Communications of the ACM*, 1980: 689-703.

Winston, Patrick Henry. "S3, Taking Machine Intelligence to the Next, Much Higher Level." *Genesis Group.* February 17, 2011. http://courses.csail.mit.edu/6.803/pdf/whitepaper.pdf (accessed May 1, 2011).

Winston, Patrick. *The Strong Story Hypothesis and the Directed Perception Hypothesis.* Cambridge, MA: MIT Computer Science and Artificial Intelligence Laboratory, 2011.

Young, Kay, and Jeffrey Saver. "The Neurology of Narrative." *SubStance*, 2001: 72-84.