Detecting High Level Story Elements from Low Level Data

by

Erek R. Speed

B.S., Massachusetts Institute of Technology (2009)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author . . . .
Department of Electrical Engineering and Computer Science
May 21, 2012

Certified by . . . .
Patrick H. Winston
Ford Professor of Engineering
Thesis Supervisor

Accepted by . . . .
Prof. Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

Detecting High Level Story Elements from Low Level Data

by

Erek R. Speed

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2012, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The problem addressed here is yet another artificial intelligence problem that is easily solved by young children yet challenges even sophisticated computer programs. This thesis's canonical example is a scene featuring two entities drinking. In one scene, a cat drinks from a faucet. In the other, a human drinks from a glass. Even young humans can identify that the two images are similar in that they both involve drinking. However, low-level analysis of the scene will find many more differences than similarities in the case cited above. In my research examines ways to detect high-level story elements such as drinking from low-level data such as that which might be produced from analyzing pictures and videos directly.

I present a system that accepts as input a collection of high-level events represented in transition space. I analyze, then select the affinity propagation clustering algorithm to group the events using only their low-level representations. To this end, I present a novel algorithm for determining how similar any two points in transition space are.

Due to the lack of vision systems capable of providing a varied dataset, I create a system which translates English language descriptions of high-level events and produces a specially formatted transition space file.

To support my hypotheses, I presents the results of two experiments using the system described in this thesis. The first experiment uses English language files and the second uses data produced from a set of experimental videos. Using the English language files the system was able to detect groups corresponding to flying and walking among others out of a total set of 16 events.

Thesis Supervisor: Patrick H. Winston
Title: Ford Professor of Engineering

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People experience the world as an ever expanding story. More specifically, it is an ever expanding high-level story. Low-level events are combined into high-level events, which allow us to communicate effectively and efficiently. The difference between high and low-level stories can be seen in a simple example. For instance, except in very special circumstances, I would not describe a person eating cake with a play-by-play commentary:

The fork lowers. The fork pierces the cake. The fork separates a chunk of cake. The fork ascends toward the mouth. The fork, along with cake, is inserted to the mouth. etc.

In reality, all of the detail, most of it boring and useless, is abstracted away into a single high-level element: "he ate cake." As it turns out, we exist in this high-level world while many of the machines and devices with perception capabilities operate mainly in the low-level world.

1.1 The Problem

Just as we experience the world with high-level story elements, we also form connections using those same elements. For instance, imagine a human drinking from a glass of water. Due to a lifetime of shared experience, it is easy to assume you are imagining something similar to figure 1-1.

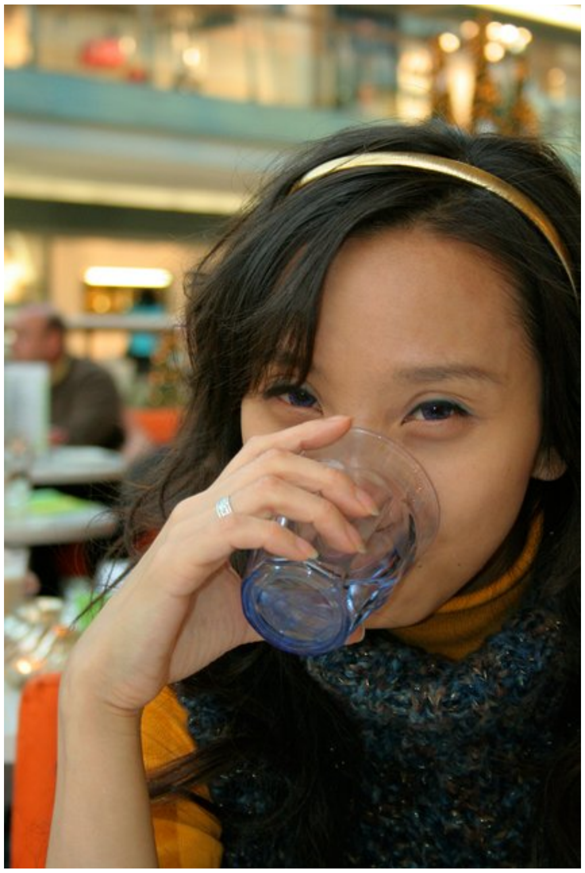


Figure 1-1: A person drinks from a glass of water.[Kristiaan()]

A modern low-level image detector could be primed with a library of humans drinking water from glasses and asked, "Is this a person drinking water?" and would most likely reply with the correct answer.

Or perhaps, if it operated at a slightly higher (yet still low) level, it could produce a set of rules that matched drink 'patterns.'

- Glass of water moves toward mouth.

- Glass of water contacts mouth.
- Water transfers from glass to mouth.

Assuming an average lead-up to the frame in figure 1-1, it is easy to imagine a correct answer using this system as well.

The first method uses low-level image data, which humans cannot parse with efficacy. The second method has human readable rules but they are very similar to the cake example from before: tedious and verbose. One might assume that it is acceptable for a computer to use such verbose language as long as it performs well. Unfortunately, the limitations of the system are made clear when we look at figure 1-2.

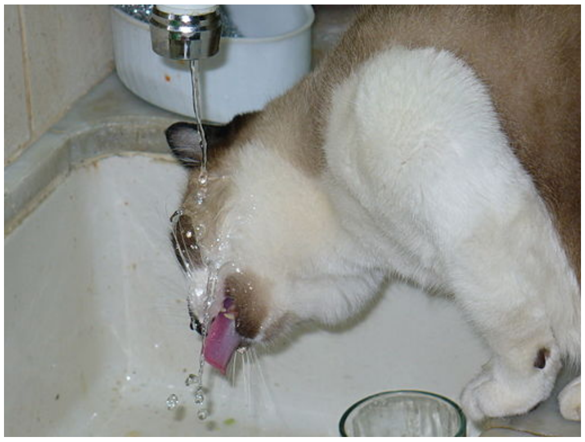


Figure 1-2: A cat drinks from a faucet. [Y()]

This picture is a bit different than the one in figure 1-1 but most people would agree that they tell the same story [1]. Certainly, if one were asked "Is this a picture of drinking?" they would reply in the affirmative with a reasonably high certainty. (Of course, our world is too ambiguous to know anything with full certainty.)

On the other hand, a computer primed with pictures of humans drinking will never match that of a cat drinking. We can fix this specific problem by priming the system with pictures of cats. Even so, it will fare no better when trying to match a horse, dog, or bird drinking. Most likely, it will fail to match another human drinking using a different method (such as with her hands).

[1]This 'cat problem' was introduced by vision researcher Shimon Ullman.

The rules of the second system are more general than the first so one might expect it to perform better. In general, it might have if the cat was drinking from a glass, but unfortunately, it has no rules that match drinking from a faucet. Once again, we can add the rules that will make this work. Encouragingly, unlike before, a rule based system as described here will be more robust. Assuming I add rules covering water traveling from faucets to mouths, the system should be able to match anything that drinks via glasses, faucets and mouths. Of course, there are still many more ways to drink and we have to make rules for many of them if we want performance approaching a human's.

The problem with both systems when faced with a question similar to "Is this drinking?" is that their world view is a simple composition of many low-level elements. Because the question posed is asking about a high-level event, the example systems are overwhelmed by problems that are easily solved by human children.

While humans tire more quickly than computers, it still may help to imagine how tedious interacting with others would be if we only had access to the verbose description of cake eating mentioned previously. Parsing such a description requires strict attention, and even then, one must strain to separate the important statements from the fluff.

I have described the toy systems here for demonstration, but most competent vision systems existing today rely heavily on massive amounts of data to achieve good results [Spitkovsky and Norvig(2012)]. These systems are engineering marvels but have avoided the core of the problem rather than addressing it.

The goal of this thesis is to describe a system, reasonably grounded, that attempts to build high-level story elements out of low-level data.

1.2 The Solution

In general, a system that solves the problem at hand would accept as an input a representation of low-level data. As with any naive system perceiving the world, objects may be identifiable but would not be associated with any rich semantic information. It is not a very interesting experiment if it is known a priori that water is something a living entity drinks. Given this representation, I then want to find the high-level story elements that are represented in the low-level details. I propose this must be done in two steps:

1. Less important details must be filtered out.
2. Important details must be grouped together and named.

These steps encode the essential abstraction that humans perform when identifying the eating of cake or the drinking of water. We ignore what doesn't matter and give names to what we consider important.

In order to achieve the first step, one must decide on which details are important and which ones can be discarded. However, this way of thinking is a trap. A similar argument was made for the rule-based robot described previously. A clever researcher could define a clever rubric for what stays and what goes, but inevitably the rubric would eventually exclude what should stay and keep what should be left out. Moreover, to manage such a feat for all but the smallest domains would be a task of unimaginable complexity.

Instead, I propose that an unsupervised method be used to detect regularity and group low-level descriptions of events. Such systems avoid much of the bias associated with hand picked grouping criteria. Moreover, as the system captures an increasing amount of data, the groupings it produces will expand and contract based on the new information it receives.

Note that the clustering step combines filtering and grouping. All that remains is to label the groups using regular supervised methods. These labels represent the high level abstractions that make up the stories we are constantly experiencing and conveying to others. When new low-level events are processed and placed into a labeled group, the system should detect the new item as the same high-level element the whole group has and describe it accordingly.

Even without reviewing the results of my implementation, it is possible to develop an intuition for why an algorithm like this might work. As it stands, one might argue that an unsupervised clustering algorithm would surely put all cats in one group and all humans in another, ignoring incidental details like water and its proximity to any mouths. Indeed, in such a scenario, the improved system would perform no better than the toy systems analyzed in the last section.

To quell such fears, I present in table 1.1 a tabular version of a low-level description of a cat drinking such as seen in figure 1-2.

Faucet	drips water	
Cat	tilts head	
Cat	extends tongue	
Tongue	scoops water	
Tongue	brings water	into mouth

Table 1.1: A grid of low-level events loosely transcribed from figure 1-2. Lighter colored squares appear less often in descriptions while darker colored squares appear more often. The darker a square becomes the less impact it has on overall clustering.

This table represents a reasonable guess as to what a low-level representation of figure 1-2 might look like. For clarity, the cells of elements that I expect to appear often across various categories of events are darker than ones that I expect to appear in fewer, more specific events. We can view dark cells as less useful for identification, while lighter colored cells are the more useful. Producing an accurate table like this is what an expert would have to do absent an unsupervised step. For unsupervised clustering however, no one need worry about picking the right cells or shadings; the algorithm finds these properties naturally.

Why is this? Consider a situation in which all events have cats involved. Each of the events gain a bit of similarity with each other one. Note that the widespread presence of this single low-level element creates a new baseline for comparing events. If a few events also had an ‘into mouth’ they would emerge as a separate group due to similarities that differentiated them from the crowd.

This means that as a side effect of finding the best groups, such unsupervised clustering algorithms necessarily find meaningful distinctions between important information and noise.

Given the discussion so far, we can produce a system block diagram such as the one in figure 1-3. The abstract design shown here provides a basic idea of how the system corresponds to the general requirements specified throughout this section. Implementation details manifest as specifications for each of the blocks in the figure. What follows is a brief outline of what to expect in the remaining document.

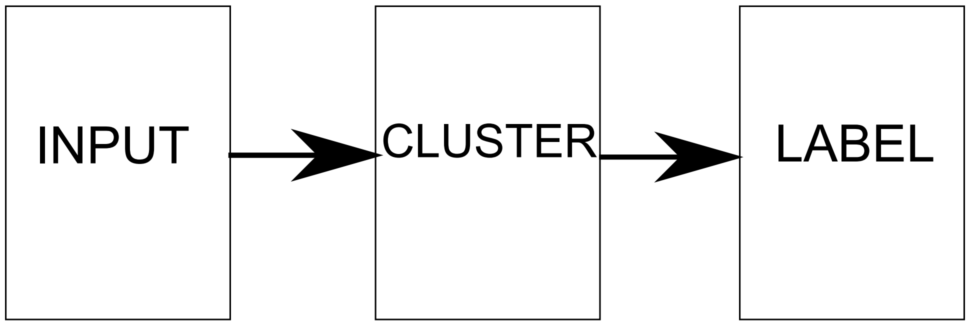


Figure 1-3: A figure showing the block diagram of a general system conforming to the requirements set out in this section.

Chapter 2: System Grounding provides non-technical discussion on the reasoning behind the algorithm. It explores similarities to early childhood language acquisition as well as limitations of low-level data.

Chapter 3: System Overview gives an overview of my specific implementation of the general system shown in figure 1-3. It will describe the sub-systems and design choices that are detailed in subsequent chapters.

Chapter 4: Input and Representation describes how input for my implementation was gathered and processed. It also describes the low-level representation that was chosen for this project.

Chapter 5: Clustering describes the unsupervised clustering algorithm used and its sub-systems.

Chapter 6: Results examines the initial results of the system. Specifically, the clustering performance is examined for the input described in chapter 4. In addition, a data set generated via a separate method will be processed analyzed.

Chapter 7: Contributions lists the contributions I have made in this thesis.

Chapter 2

System Grounding

In general, I believe that serious attempts in creating artificial human intelligence should try to ground such efforts in actual human intelligence. In the short term, it is often possible to achieve impressive results with little thought to the species computers are supposed to be mimicking. However, I believe that such efforts do little to advance the core purpose of artificial intelligence, a computational model of human intelligence. As such, I believe it is important to examine the present system in an effort to establish what, if any, grounding it has in human intelligence. Especially because, at a glance, I have proposed a system that contains at its heart a data centric model.

Sensitive to this fact, I have included this section as a medium through which to discuss how such the system described in this thesis might be grounded in human intelligence. Additionally, I explore the adequacy of low level data as it pertains to categorizing certain intangible high-level elements.

2.1 Grounding in Human Intelligence

A common criticism of many artificial intelligence systems is the massive amounts of data that is required for a computer to learn something non-trivial. It is certainly surprising that some systems might take thousands upon thousands of examples before they can reliably identify a tree while a child can learn a tree from only a few examples. On face, it seems that algorithms driven by massive data have forgone a grounding in human intelligence for a grounding in statistics.

Of course, humans are also ‘acquiring’ data as we experience the world. There is evidence that from a very early age children are making connections and storing information in some kind of internal representation. By the time a child must learn the word ‘tree’ and what it corresponds to, he/she has seen many many trees. Using our special internal representation, by early childhood, we already have a grouping assigned to trees, we simply do not know what they are called.

Previous work has shown that infants and young children rely heavily on statistical learning as they integrate into the world [Lany and Gmez(2008)]. At as early as 2 months old, they are detecting regularities in vision. Detection of regularity in aurally presented stimuli comes as early as seven months. Moreover, prior experience in statistical learning benefit infants when they try to learn more complicated patterns.

The detection of regularity, as well as the correlation of success to experience map directly to the techniques proposed in this thesis. Again, the main goal is to detect regularity via clustering and use that information to inform future classifications.

Furthermore, the idea that children form groups internally and then quickly apply names to them is supported by research in language-acquisition. Specifically, first language acquisition corresponds to the initial ‘labeling’ phase of the human clustering algorithm. The text, *Contemporary Linguistic Analysis*, describes 3 different strategies that children around 18 months of age appear to use to label objects [O’Grady and Archibald(2008)].

- *The Whole Object Assumption:* Label only applies to this object. (dog $\rightarrow$ the family pet)
- *The Type Assumption:* Label refers to a type of thing. (dog $\rightarrow$ a type of animal)

- *The Basic Level Assumption:* Refers to types of things that are alike in basic ways. (dog $\rightarrow$ medium sized, four legged creature)

A possible explanation for some the differences in labeling is that they correspond to the state of the child's internal grouping for the object being classified. While the child is still acquiring examples, the groups vary from being too broad to too narrow. Of course, eventually enough data and experience is gathered such that most children develop into fully functioning adults. It is this path, the system described herein tries to follow.

Contemporary Linguistic Analysis also adds some interesting insight via a discussion on the errors children make during the language-acquisition process [O'Grady and Archibald(2008)]. The chapter describes the two main classes of errors children make when learning language. The first is called overextension. Simply, this is when a child uses a single label to incorrectly refer to multiple different objects. A ball may be any round object. A dog may be any medium sized four legged animal. However, there is evidence that these errors are merely compensation for a limited vocabulary. Indeed, often when a child learns the right word for an object that was previously mislabeled they immediately switch to the new word.

This supports the idea that children have an internal representation that groups objects and labels them as labels become available.

The second class of error is the opposite. Underextension happens when a child uses a general word to specifically designate an example of the more general class. To use the example from [O'Grady and Archibald(2008)], *kitty* may be used exclusively for the family pet but for no other cats. This error again maps easily to the clustering domain. If there are few examples of a class or a disproportionate number of varying types of a class, a clustering algorithm can incorrectly split a group in a similar way.

It is hard to ascertain for certain whether any given algorithm properly models human intelligence. I have provided some evidence that a unsupervised clustering based algorithm has several telling similarities to studies involving children tackling a similar problem. Though this grounding does not affect the final results of my system, it provides an additional lens with which to view them.

2.2 The Limitations of Low-Level Data in Describing High-Level Story Elements

Another potential theoretical criticism of this work is that many types of low level data may be unsuitable for describing high-level events that do not stem from the observable tell-tale signs that low-level systems analyze. One example of this are events that are characterized by achieving a goal. The real uniting factor is not the proximity of a liquid and a mouth but the desire to sate one's thirst. In general, if we can identify this sentiment then we have identified drinking.

Unfortunately, the examples of low-level data given so far have no capacity to determine the sentiment of an actor with a perfect confidence level. Moreover, this is not simply an artifact of a poor choice of example representations. Consider whether any possible difference could be detected between someone drinking with the sentiment described in the previous paragraph or accidentally swallowing with the original intent on spitting the liquid out. (Though the dictionary defines drink to include both, for this example we use the sentiment criterion.)

This is a definite limitation of the any system that models the one described in the paper as far as plain low-level representations are involved. However, the question remains whether such a limitation detracts from the goal and usefulness of the present work.

The first thing to realize is that in this situation, humans would fare no better given the same set of physical cues. However, verbal cues and confirmations could likely remove the ambiguity that exists otherwise. Even so, when providing evidence for any observation the most weight is given to actual observable physical cues and not to invisible sentiment. In fact, more often than not, humans use physical cues to infer sentiment. The detection of the various low-level physical aspects of drinking are enough to make a human reasonably sure that the person drinking is also thirsty.

That being said, the systems presented in this paper have much less granularity than that of an average (or below average) human. This is accounted for by avoiding events that account heavily on sentiment for disambiguation.

Finally, the research presented in this thesis could easily act as a module in a larger system that had advanced features for detecting sentiment via verbal input etc. This leads to the conclusion that claims of such limitations are valid, but not reason enough to doubt or discount the importance or validity of the results of the current work.

Chapter 3

System Overview

Previously, I have described a general system that solves the problem presented in the introduction (see figure 1-3). The first block accepts and processes input. The second block does the heavy lifting in clustering the data. The final piece is labeling. This last point is the only point of human interaction and is a lightweight addition to the rest of the system. Indeed, the system can operate perfectly well without labeling until such a time as a human is available.

In this section, these will be described in terms of the actual implementation used to test the previous assumptions. To this end, first refer to figure 3-1 to find the blocks expanded into their respective sub blocks corresponding with the design and implementation choices that were actually made while building the system.

The remaining sections of this chapter give an overview of each main block in turn, introducing the various sub-blocks and preparing the reader for the in depth implementation chapters that follow.

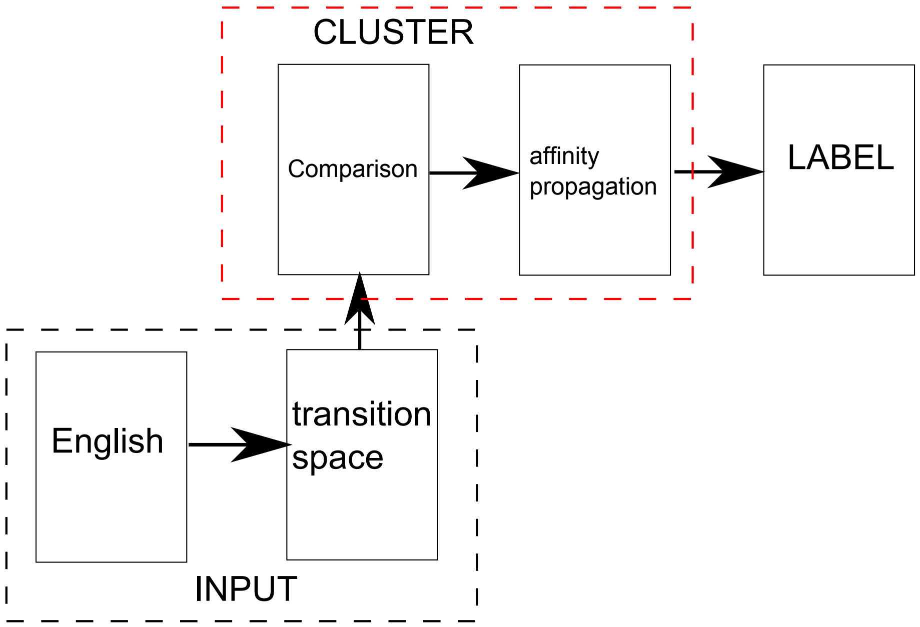


Figure 3-1: A system block diagram corresponding to the specific implementation of this thesis. The correspondences are indicated with a dashed line block around the new system pieces.

3.1 Input

This system block covers the source data and final representation of the low-level data that our system will operate on.

In figure 3-1, it has been expanded into an `English` sub-block and a `transition space` sub-block. From there, it heads into the next main block, `cluster`. The details of the `input` block are covered in chapter 4.

The primary representation of the system is transition space. Thus, the first task of this chapter is to define and give an overview of the representation.

The `English` sub-block collects low-level English descriptions that can be translated into the final representation. The detailed description in this chapter will provide insight into the criteria for choosing which events were included in the experiments used to evaluate the system. It will also include a detailed description of the form and structure of the English sentences so as to be ready for the `transition space` sub-block.

The `transition space` block covers details that relate to the representation of the final input. This section will discuss the process of translating English into transition space using Genesis. Genesis is a software system that excels in story understanding and contains convenient tools for analyzing English sentences in general. The necessary parts of Genesis are also described.

The output of the `transition space` sub-block is one or more files corresponding to the transition space representation. These travel to the `cluster` block.

3.2 Cluster

The `cluster` block is responsible for the heavy lifting of the system. It is fully described in chapter 5. It features a `comparison` sub-block leading into an `affinity propagation` sub-block.

The `affinity propagation` block is last in the system flow, however, its existence and properties inform all other aspects of this section of the system. Thus, it is covered in detail before the other sub-blocks. First, there is a discussion on how affinity propagation uses messages to find and group items based on their mutual similarity. Second, an analysis of the favorable properties of affinity propagation is provided. Finally under this section, the chapter describes relevant details of the specific implementation of affinity propagation used for this thesis.

Affinity propagation requires a measure of similarity between all the items it tries to cluster. For this reason, before the `affinity propagation` sub-block can be reached, the data must go through the `comparison` block. This block covers the process used to come up with a consistent and useful similarity metric for the incredibly irregular and varied transition space representation. This section covers the overall algorithms used as well as improvements. It also provides discussion of the intermediate results of the comparison methods.

Upon leaving the `cluster` block, the input events have been grouped based on the underlying regularity of the low-level data. They are now ready to be labeled in the next block.

3.3 Label

The implementation of the `label` block is simple and has no sub-blocks. In testing the current implementation, groups are simply automatically labeled with the high-level description known to correspond to the 'center' of the group. No further detailed description is necessary.

Chapter 4

Input and Representation

Ideally, the system proposed in this thesis would accept images and movies as input in much the same way as a human perceives his environment. Unfortunately, as far as I am aware, there currently exist no systems that can process arbitrary image/video streams to produce even a low-level representation of what is seen.

However, there are currently systems which can process simple stylized scenes and produce the type of low-level data the system aims to analyze. Sajit Rao has developed a system which operates on videos and produces attentional traces, which contain information about the people and objects in the video, include their spatial relationships [Rao(1998)]. I have the results of this process as applied to simple events such as 'give' and 'bounce.' The actors' clothes and the objects in these videos were constrained to specific colors and the actions are exaggerated to facilitate the processing. These videos come from the seedling phase of a DARPA project known as Mind's Eye [DARPA(2010)].

To fill in the gap left by current vision research, I decided to use English to produce arbitrary scenes for analysis. English files are easy to create and maintain and have none of the restrictions of the video files mentioned above. After processing, both the Mind's Eye videos and the English files of my making exist in the transition space representation described below.

After describing transition space, this section describes the process of selecting examples for use in the dataset used to test the system. Third, it describes the English language files that must be provided as the input into the system. Finally, the process of translating the English language files into format expected by the `cluster` block is explained.

4.1 Transition Space

The initial input is transformed into Gary Borchardt's transition space [Borchardt(1993)]. This representation emphasizes the changes between states rather than the states themselves. Borchardt originally introduced transition space in order to analyze causality. If a car crashes into a wall, what causes the car to be destroyed? The states involved do not provide enough information to determine the cause of the crash. Before the crash, the car and wall were not in contact so that state could not have been responsible. The state of the car being in contact with the wall after the crash also cannot explain the crash. There are many ways for a car to come in contact with a wall that do not lead to crashes. The true cause of the crash lies in the properties of the transition from no contact to contact. The event can also be viewed as a table (see table 4.1).

attribute	T1	T2	T3
speed(car)	increase	increase	disappear
distance(car, wall)	decrease	decrease	disappear
state(car, destroyed)			appear

Table 4.1: A table view of the transition space representation of a car crashing into a wall. Blank cells represent a lack of information about the attribute at that specific transition. In this case, the cells could reasonably contain 'not-appear', which is how such tables are filled generally. However, my opinion is that declaring the non-existence of attributes does not correspond with reality and thus I have omitted them.

Transition space is useful because it makes the key elements of events explicit. However, it has other benefits as well. First, it is easy to describe transitions using simple stylized English. This allows us to keep the dataset of sample events as simple English language files. Second, by using transition space, I have the capability to test the system with any input that also uses transition space. An example of this will be presented in the chapter 6 results.

Now, that transition space has been defined, we can explore the process of selecting events for inclusion in the dataset and the translation from English into the final transition space format.

4.2 Selecting Events For Trial Classification

The first part of the system must be the selection of interesting samples for grouping and clustering. We have already seen the canonical example in the introduction: 'drink'. Drinking is an especially good sample because there are many low-level ways to represent the high-level story element. A closer look at the 'drink' event introduces a another interesting property. Previously, we compared the person in figure 1-1 to the cat in 1-2. This is an example of a positive example that many low level algorithms miss. Now imagine comparing the figure of a person drinking to the image of Ronald Reagan toasting below (4-1).

A human toasting is an event that has many low-level similarities with a human drinking. A low-level system will match the human drinking and human toasting even though the high-level story is quite different. Examples of this type provide an extra test of robustness to our system. Not all examples added to our initial dataset will exhibit this property but it is considered a positive factor in selecting events for inclusion.

A good example in 'drink' has been with us since the introduction, but I have yet to provide a example that would be better off not in our dataset. There are not many examples that need to be forbid except for those that I consider too hard for the system to analyze. Specifically, these are any events that differ mostly in the sentiments of the actors involved. For example, 'fight' has the potential to be a good

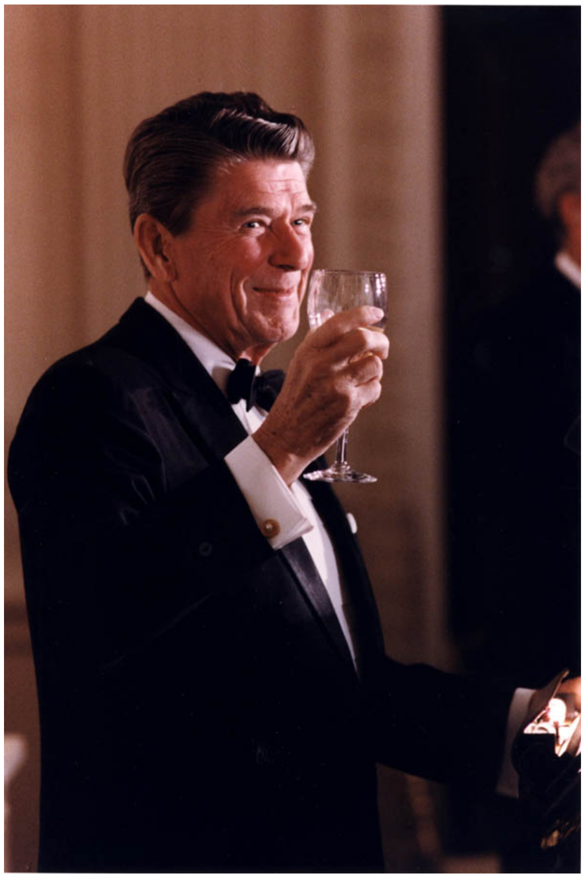


Figure 4-1: An average human making a toast.

event due to the wide differences between how humans and many animals engage in such an activity. On the other hand, humans and many animals can also engage in the similar activities that involve the same actions but without the ill will implicit in ‘fight.’ Realistically, events like these could be added with little harm as long as the tricky examples were avoided. I decided to not include them because there was no shortage of other events that did not come with such questions.

This analysis of ‘drink’ and ‘fight’ suggest several useful rules of thumb for building a dataset of interesting examples.

- The event should have several low-level representations that have many differences and few similarities. The goal is to test the ability of the system to detect the key data points among noise.
- If the event is connected to other events via low-level similarities then that

should be considered as an added bonus. In fact, it immediately implies another event to be added to the dataset.

- Finally, avoid events that have similar analogs with essentially no low level differences. These can be saved for systems that can handle sentiment at a level more in line with human ability.

Using these criteria several categories of events were produced for the inclusion in the dataset used for testing the system. For example, 'drink' of the human, cat, and bird varieties along with 'toast.' For a complete list, of events included see appendix A.

4.3 English Language Representation

The next step after deciding which events will be in the dataset is producing the actual files. As mentioned above in section 4.1, transition space lends itself to English language description. Figure 4-2 shows the English version of the car crash from table 4.1.

```
The speed of the car increases.
The distance between the wall and car decreases.

Then, the speed of the car increases.
The distance between the wall and car decreases.

Then, the speed of the car disappeared.
The distance between the car and the wall disappeared.
The state of the car being destroyed appeared.
```

Figure 4-2: A English language representation of the set of transitions corresponding to a car crashing into a wall. Each section corresponds to a column in the table. 'Then' starts each new transition point.

The connection between the basic transition space as represented in tabular form and the English produced from it is clear. In the files produced for this implementation, there is also a preamble sentence, which allows the filename to be tracked during translation. This can be seen in the English form of `bird-drinks` in figure 4-3.

```
This is bird_drinks.

//Frame 1
Distance between bird and pond of water appears.

//Frame 2
Then, Distance between bird and pond of water decreases.
Vertical position of bird decreases.

//Frame 3
Then, Distance between bird and pond of water disappears.
Contact between bird's mouth and pond of water appears.
Vertical position of bird decreases.
Horizontal position of bird increases.
```

Figure 4-3: An English language representation of `bird-drinks`. The event is imagined as a three frame sequence where each frame is about two seconds apart. Line 1 contains a sentence declaring the filename for tracking purposes.

The form displayed in the figure is repeated across all events in the dataset. Specifically, all events are imagined as three frame 'cartoons' where each frame is two seconds apart. Once these files have been produced they can be read by the translation system, which will produce the transition space files formatted for the clustering algorithm.

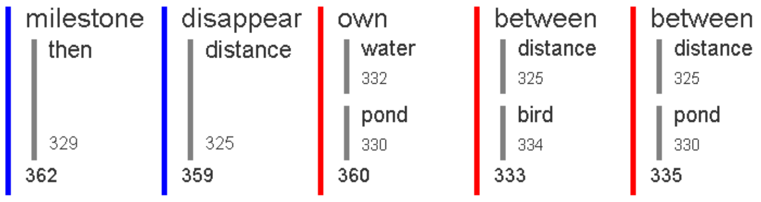
4.4 Translation

Though it is convenient to produce and store English language versions of the transition space events, the `cluster` block uses a less human readable version for convenience reasons. While the format produced by this translation step was convenient for my purposes it is easy to imagine a version of the system that operates on the English language files directly.

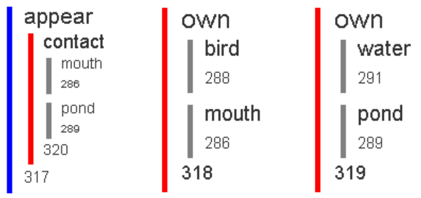
In order to conveniently parse the English files, the system uses a subset of Genesis. Genesis is a large software system that provides support for understanding stories

[Winston(2011)]. For the purposes of this project, Genesis was used as a means to parse English input into its easy to use internal representation. Genesis's ability to parse English text is powered by Boris Katz's Start Natural Language Processor [Katz(1997)]. I wrote a Java program that accepted these internal structures as input and produced an object with `String` output equivalent to the file format expected by the remainder of the system.

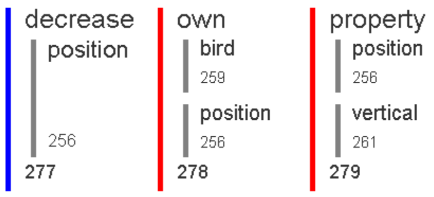
Figure 4-4 shows the Genesis representations for selected lines in the `bird-drinks` example above in figure 4-3.



(a) Genesis representation of the sentence, "Then, Distance between bird and pond of water disappears." Because of the milestone 'then' creates it easy to keep track of transition points.



(b) Genesis representation of the sentence, "Contact between bird's mouth and pond of water appears." Notice the difference in how 'between' is represented as compared to (a).



(c) Geneis representation of the sentence, "Vertical position of bird decreases." The 'position' frame also has the 'vertical' property even though it is not seen here.

Figure 4-4: The internal Genesis representations for three sentences from `bird-drinks`. They correspond to lines 11-13 respectively in figure 4-3

In addition to creating the functionality to parse each individual Genesis frame, I created a simple user interface inside of Genesis for reading a directory of English language files and producing a directory of transition space files.

Chapter 5

Clustering

This chapter goes into detail on how the system implemented takes in a collection of events represented in transition space and finds optimal groupings for them. The clustering system chosen for the system is called affinity propagation. A key part of affinity propagation is knowing how similar each item is to each other item. To that end, in addition to describing affinity propagation this chapter also goes into detail on the comparison methods that allow affinity propagation to work on transition space events.

5.1 Affinity Propagation

Affinity propagation is a clustering algorithm (based on loopy belief propagation) introduced by Frey and Dueck in 2007 [Frey and Dueck(2007)]. Belief propagation simply refers to an algorithm that sends messages along the edges of a network to find some optimal mapping. For instance, in a Bayes net, belief propagation tries to find the correct assignment of probabilities. The loopy moniker refers to the fact that the algorithm is meant to be run on networks that contain cycles.

Affinity propagation has many properties that make it suitable for the clustering of transition space events.

1. It works for any type of data so long as some comparison function exists. It merely needs to know how similar the data points are to each other. Moreover, the similarities do not need to obey the triangle inequality, which is hard to guarantee for non-euclidean data.

2. The number of clusters does not need to be specified. With a goal of naturally identifying clusters of low-level data, in general it cannot be known how many groupings should appear.

3. The algorithm works well even when expected groups are small. A system trying to organize the world in general exemplifies such a use case.

For the purposes of this thesis, affinity propagation serves as a black box clustering algorithm provided by the scikit-learn machine learning software package [Pedregosa et al.(2011)Pedregosa
What follows is a description of the inputs into the algorithm and how varying them affects final performance.

The input is an $n \times n$ similarity matrix, S, generated from a pairwise comparison of n examples. The similarity $S(i, j)$ indicates how well the jth data point is suited to be an exemplar for point i. For the purposes of affinity propagation, exemplar simply refers to the point that is representative of itself and others it may be assigned.

The values inside of S may be any value positive or negative, however, it is important to remember that small values imply less similarity than large values. The comparison methods I implemented treat 0 to mean most similar while 1 was least similar. In order to convert my matrix into one affinity propagation could work with, I applied a Gaussian heat kernel to each value (The kernel is $e^{-\frac{x^2}{2\delta^2}}$ where δ is parameter which controls spacing of values after the transformation. Using the median of S for this value has led to reasonable results.) . This has the effect of turning 0 values into 1 while the largest values before the transform become the smallest afterward.

The second parameter needed by affinity propagation is a size n preference vector p. Affinity propagation does not accept a parameter for an exact number of clusters but one may affect the value in general using this vector. If certain data points in the vector p are bigger than others than they are more likely to become exemplars. If there is no a priori knowledge of which points should be exemplars than all points should be given the same value in p. If the values in p are small compared to the similarities in S than a small number of clusters will be produced. Likewise, if the values are high relative to S than a large number of clusters will be generated. (The limiting cases are 1 cluster and n clusters respectively.) When tested on my dataset, which included similarity values ranging from 0 to 2.5, the best value of p ranged from 110% to 130% of the median of S.

Once a similarity matrix can be generated, interfacing with the affinity propagation system is simple enough to allow ample experimentation. To that end, the rest of the chapter covers the details of the similarity metric I developed for comparing transition space events.

5.2 Comparing Events in Transition Space

Creating a metric for comparing events in transition space is not a simple task. Even though there are many known algorithms for computing the distance between various kinds of sequences, the common ones rely on various forms of regularity, which cannot be guaranteed or even vaguely hoped for with a given sample in transition space.

Recall that an event in transition space is a series of transition points happening at arbitrary times, applying to arbitrary attributes. In comparison, each slot in a regular string is chosen from a small finite set of characters and is only one character long. It is no wonder that the assumptions made for string distance algorithms fail to even approach validity in transition space.

Surprisingly, there does exist a comparable sequence that allows for essentially all of the variation described in the previous paragraph: musical sequences. Each point (note) in a musical sequence can have varying length and surprising variety. Moreover, it is common for such musical sequences to have overlapping notes similar to the overlapping transitions of different attributes in transition space.

Fortunately, significant work has been done in computing the distance between such sequences. The most applicable algorithm is provided by Mongeau and Sankoff in an article from 1990 [Mongeau and Sankoff(1990)]. They treat the music as a sequence comparison problem solvable with the following dynamic programming recurrence:

$$d_{ij} = \min \begin{cases} d_{i-1,j} + w(a_i, \emptyset) & \text{(deletion)} \\ d_{i,j-1} + w(\emptyset, b_j) & \text{(insertion)} \\ \{d_{i-1,j-k} + w(a_i, b_{j-k+1,\ldots,}b_j), 2 \leq k \leq j\} & \text{(fragmentation)} \\ \{d_{i-k,j-1} + w(a_{i-k+1,\ldots,}a_i, b_j), 2 \leq k \leq i\} & \text{(consolidation)} \\ d_{i-1,j-1} + w(a_i, b_i) & \text{(replacement)} \end{cases} \tag{5.1}$$

In the equation, a and b represent two musical sequences of varying lengths. $w()$ represents the cost function for a given a or b. The key contribution made was the inclusion of consolidation and fragmentation. These are two transformations that only apply to sequences where various points in the sequence may have varying lengths. In the music domain, this corresponds to comparing a single whole note to four quarter notes. Without consolidation, such an operation would cost a replacement and three insertions, which is almost certainly not a constructive penalty given the domain.

Assuming a cost function w and a singly enumerable sequence of transition points, the above recurrence can be applied directly to comparing two transition space events. Figure 5-1 shows a matching between two cartoon transition space events using the recurrence from equation 5.1.

Given truly arbitrary events, this construction is necessary to ensure the correct minimum distance. Unfortunately, the algorithm runs in $O(n^3)$, which is quite large for a method called in the second layer of a doubly nested loop. ($O(n^2)$ is possible in

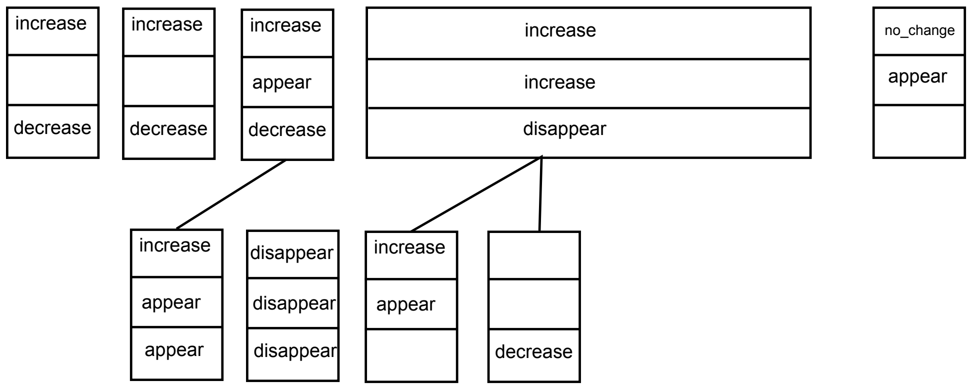


Figure 5-1: Minimum cost matching of two arbitrary transition space events. Notice the presence of consolidation and replacement.

many cases if the fragmentation and consolidation recursions are capped.) However, there is an easy optimization if there are guarantees about the nature of the events to process. In this case, if transitions occur at regular intervals than consolidation and fragmentation will never produce a lower cost than insertion, deletion, or replacement. All the data at hand for this project allowed for this simplification and so the initial version of the algorithm was never implemented.

The improved algorithm runs in $O(n^2)$, which is fine for the small examples shown in the previous chapter, but for larger datasets it still leads to each comparison taking on the order of 5 seconds. Even though performance is not a stated goal of this particular implementation it would be preferable to do as well as possible.

The final optimization is much more risky. It no longer guarantees the minimum distance and requires even more regularity in the data to be compared. Specifically, if the number of transition points in both events are the same then every insertion and deletion has to be paired with another insertion or deletion later in the sequence. Assuming the cost of replacement is on the order of the cost of insertion/deletion then it is reasonable to expect that replacement to almost always be better. By applying this optimization, real time performance is achieved for all datasets. As a caveat, while this optimization happens to work in large and small data sets tested for this thesis, it is unclear how data in general would behave. I believe it mostly depends upon how the data is generated, but for now I can only urge caution.

A similar diagram as shown in figure 5-1 is shown in figure 5-2. Note the regularity that allows the simplification to happen.

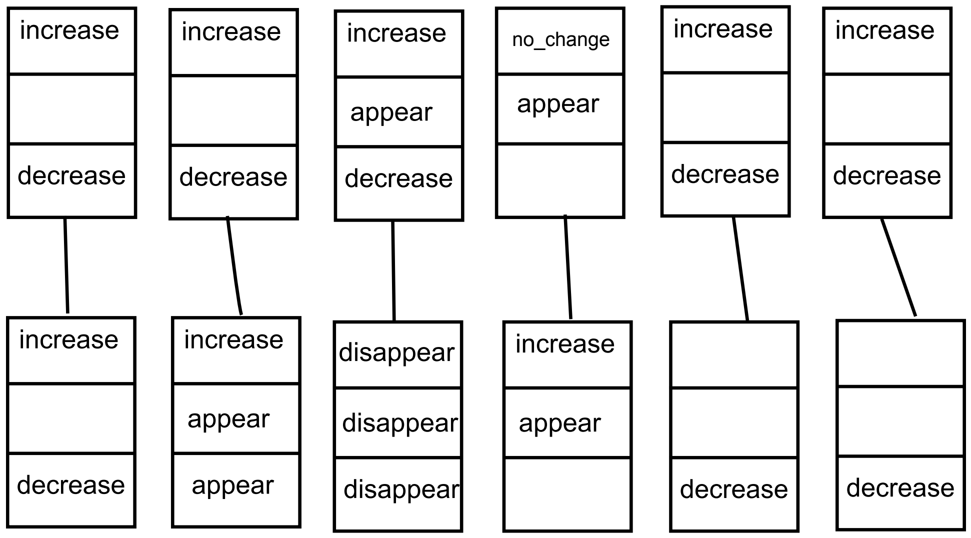


Figure 5-2: An example of how the most optimized algorithm finds the minimum distance between events. The only thing measured is the cost of replacement for corresponding points in the two sequences.

Using this optimized recurrence will work well for our system, but there is still the open question of what the cost function w should be.

5.3 Comparing Individual Transition Points

The cost function w needs to take any two transition points in any two events and return the distance between the two. This is the same as the cost to replace one with the other. Every transition point has a transition value for each measured attribute or is blank. (If a transition point is a single column in figure 4.1, then transition values include the 'increase' and 'decrease' literals.) In other words, they are made up of attribute/transition-value pairs.

A naive algorithm simply compares the attributes in the two transition points and if there is a match returns a value based on how similar their transition values are. The average over all the attributes is the cost for the entire transition point. For this implementation, the pairwise costs for all the transition values are hard coded to what I think are reasonable values. Table 5.1 lists a subset of them.

	increase	disappear	blank
increase	0	.75	.8
disappear	.75	0	.8
blank	.8	.8	0

Table 5.1: A subset of the hand coded distance values for transition values. Blank refers to the situation where either transition point doesn't contain the attribute or has not recorded a value at this specific point. 0 is identical while 1 is completely different.

Though the values I chose seem reasonable to me, I believe using such 'magic' constants is poor design in general. Given a number of numeric constants, you can imagine a number of learning algorithms that could no doubt converge on a better set of values than I provided. Given that it is inside of an unsupervised algorithm, however, it might be tricky to determine whether any set of constants was good or bad without tedious human intervention.

A more pressing problem is that due to requiring an exact matching between attributes when comparing two transition points only very similar events will match at all. Using this algorithm, there would be zero connection made between the mouths of a cat and a human and their interaction with water. Given that this limitation is directly counter to the overarching goal of this thesis, a better algorithm is needed.

5.4 Inexact Comparison Between Attributes

Attributes have varying components, which allows for partial matches. For instance, the attribute, `contact(between, (mouth, water))`, can match on it's primary key, 'contact'; the relation, 'between'; or the two objects, 'mouth' and 'water'.

For the purpose of allowing inexact matches between attributes, I developed a method that would take attributes like those in the above paragraph and return a value corresponding to how close a match there was. The current matching used requires the primary key to be the same. Assuming that, it distributes .5 penalty points maximum for the differences it finds in the remainder of the nested structure. For example, if I compared the attribute in the above paragraph to a similar attribute that had vodka listed instead of water, a penalty of .25 would be applied. This penalty is added to the cost of any comparisons done in the overall algorithm.

While this algorithm allows me to compare any two attributes and returns an analog result as the specification demands, with no further modifications to the broader algorithm, an interesting problem arises. Specifically, due to every attribute with the same primary key providing a partial match there is no simple way to decide which attributes to pair while iterating over a transition point.

In order to solve this problem, I compare all potential matchings between the two sets of attributes before continuing with the rest of the cost function w. This process produces the mapping with least penalty. Now, choosing the appropriate attribute is a simple dictionary look-up.

Given the steps and concerns addressed in this section, a reasonable w has been developed. By using this w, the algorithm in the previous section can compute the similarity between any two transition points. Affinity propagation requires nothing else and thus, the system has been fully developed.

Chapter 6

Results

6.1 Experiments

6.1.1 Custom Event Data

The implementation described in previous chapters was tested using the dataset listed in Appendix A.

With the preference parameter set to 130% of the median of S, clusters were formed according to table 6.1. The clusters are labeled according to their exemplar, though a production system would use a separate labeling system.

The groupings are not perfect but they are reasonable considering the amount of data and the fact that children also make grouping errors early on. Especially good were the performances of flying and ambulate. Human_sinks was somehow grouped with the rest of the fliers but that could be due to consistent motion, even if it is along the wrong axis. Looking at the similarities in B.1, we can confirm that though human_sinks is considered similar to the other water based events, they are outclassed by the airplane and superman evets. Unfortunately, `cat_drink` was left out of the rest of the drinkers but `human_toast` was left in. The key here is probably the fact that the grouped events involve vertical motion while the excluded event does not.

EXEMPLAR	GROUP
superman_flies	superman_flies
	bird_flies
	human_sinks
	airplane_flies
bird_drinks	bird_drinks
	human_drinks
	human_toasts
cat_drink	cat_drink
fish_swims	fish_swims
horse_ambulates	horse_ambulates
	human_ambulates
human_climbs_rope	human_climbs_rope
spider_climbs	spider_climbs
	human_climbs_wall
human_glides	human_glides
human_swims	human_swims

Table 6.1: The groupings produced by running the dataset described in chapter 4 through the clustering system described in chapter 5.

An interesting extension to these types of events would be to simulate a child by generating slightly different events from the ones hand made. In this way, a true corpus would be available for analysis.

6.1.2 Video Data

Because the system uses transition space as its representation, it can also process other data that has been output in the same transition space format used by the clustering system. One such set of data are the Mind's Eye videos that correspond to common actions like bounch, catch, and approach. Table 6.2 shows the results from running this separately produced data through the clustering system.

The key things to notice about these results is that bounce overspecialized for almost every example. Approach2 was left out of the approach group but it is also very different than the other two approaches. Differences have a greater impact in these results due to noise from the video capture. Looking at the data in C.1, we can find

EXEMPLAR	GROUP
approach0	approach0 approach1
approach2	approach2
bounce0	bounce0
bounce1	bounce1
bounce2	bounce2
carry1	carry1 bounce3 carry0 carry2
catch1	catch1 catch0 catch2

Table 6.2: Clustering produced from running simple video data through the clustering system.

some confirmation of the groupings. Approach2 has low similarity marks to everyone except itself, and the bounces seem especially uncoordinated with themselves. Results like this can be the cause of noise, or could warrant changes in the comparison metric.

In general, the algorithm performed well considering once again the small size of the groups and the fact that the data was produced from noisy video.

Both these results and the ones generated using the hand produced events show that the system is capable of extracting regularity from low-level data in order to identify high-level story elements.

Chapter 7

Contributions

Through this thesis I have made the following contributions:

- Provided a clear grounding in human intelligence for the method of unsupervised learning to group objects and events based on their low-level properties. Provided an argument for the robustness of low-level data in categorization, even in the presence of goal and sentiment based actions.

- Developed a distance metric for transition space events. In the process, discovered direct analog between event sequences and musical sequences. Created a means for comparing individual points in event sequences using hand coded similarity values. Allowed for imperfect matching by developing a distance function for attributes. Using these differences, I was able to create the optimal mapping between any two points taken from arbitrary event sequences.

- Provided justification for the selection of affinity propagation clustering algorithm. Developed a wrapper that allowed pre-built affinity propagation software to interoperate with my similarity metric. Provided analysis on the inputs into the third-party software package as well as reasonable default values.

- Created a system for translating English language files into transition space files using Java. Designed it to accept the most common transitions formatted as

English sentences and use a START processor powered Genesis to produce a specially formatted transition space file.

- Created 17 English rendered event files. Each file is based on real scenes and they are all chosen according to a set of rules I created for this purpose.
- Tested my system using English rendered event files. Showed that even with small numbers of example events my system would cluster them according to the high-level story they represented. Provided explanation for aberrations. From a set of only sixteen events, it found nine clusters. This included a cluster corresponding to all events that featured flying and a cluster corresponding to all events with walking.
- Tested my system using processed Mind's Eye video files, showing that my system was robust in terms of input source. From a set of only thirteen events, it found 7 clusters, including perfect groupings for carry and catch events.

Appendix A

Events Suitable for Experiments

Verb	Example	Example	Example	Counter-Example
Flying	airplane flies	bird flies	superman flies	
Drinking	cat drinks	bird drinks	human drinks	human toasts
Swimming	fish swims	human swims		human sinks
Ambulate	horse ambulates	human ambulates		human glides
Climb	human climbs rope	human climbs wall	spider climbs	

Table A.1: A table of events that are good candidates for testing the clustering algorithm presented in this thesis.

Appendix B

Event Similarity Matrix

	af	bd	bf	cd	fs	ha	hua	hcr	hcw	hd	hg	hs	hsw	ht	sc	sf
airplane_flies	1.00	0.71	0.75	0.19	0.61	0.56	0.56	0.67	0.29	0.61	0.44	0.61	0.54	0.71	0.18	0.86
bird_drinks	0.32	1.00	0.47	0.20	0.23	0.40	0.27	0.18	0.26	0.70	0.23	0.38	0.16	0.80	0.18	0.25
bird_flies	0.88	0.61	1.00	0.14	0.61	0.56	0.56	0.75	0.29	0.58	0.29	0.69	0.71	0.65	0.22	0.88
cat_drink	0.54	0.56	0.51	1.00	0.64	0.69	0.69	0.49	0.61	0.73	0.63	0.80	0.61	0.54	0.66	0.54
fish_swims	0.75	0.69	0.69	0.25	1.00	0.58	0.58	0.61	0.24	0.58	0.56	0.73	0.55	0.65	0.24	0.75
horse_ambulates	0.78	0.61	0.71	0.29	0.71	1.00	0.87	0.57	0.21	0.69	0.34	0.76	0.66	0.61	0.21	0.78
human_ambulates	0.78	0.61	0.71	0.42	0.71	0.87	1.00	0.54	0.25	0.69	0.51	0.76	0.68	0.61	0.25	0.78
human_climbs_rope	0.67	0.71	0.61	0.15	0.75	0.54	0.52	1.00	0.44	0.61	0.50	0.66	0.72	0.71	0.46	0.67
human_climbs_wall	0.66	0.69	0.75	0.51	0.51	0.47	0.47	0.52	1.00	0.52	0.54	0.53	0.63	0.57	0.91	0.66
human_drinks	0.27	0.99	0.29	0.41	0.25	0.42	0.37	0.26	0.28	1.00	0.31	0.27	0.29	0.79	0.26	0.27
human_glides	0.54	0.57	0.51	0.63	0.69	0.64	0.72	0.66	0.66	0.61	1.00	0.75	0.77	0.57	0.74	0.54
human_sinks	0.86	0.69	0.75	0.36	0.67	0.61	0.61	0.51	0.34	0.67	0.37	1.00	0.65	0.65	0.22	0.86
human_swims	0.57	0.61	0.76	0.21	0.61	0.61	0.62	0.66	0.41	0.62	0.51	0.65	1.00	0.61	0.22	0.57
human_toasts	0.31	0.93	0.33	0.22	0.24	0.40	0.31	0.38	0.28	0.79	0.28	0.32	0.23	1.00	0.38	0.39
spider_climbs	0.54	0.61	0.51	0.51	0.51	0.47	0.47	0.67	0.74	0.54	0.54	0.61	0.54	0.61	1.00	0.54
superman_flies	0.86	0.71	0.75	0.19	0.61	0.56	0.56	0.67	0.29	0.61	0.44	0.61	0.54	0.71	0.26	1.00

Table B.1: The similarity matrix for the event data used in experiments.

Appendix C

Mind's Eye Video Data Similarity Matrix

	a0	a1	a2	b0	b1	b2	b3	car0	car1	car2	c0	c1	c2
approach0	1.00	0.88	0.61	0.15	0.00	0.26	0.65	0.59	0.63	0.61	0.67	0.60	0.62
approach1	0.88	1.00	0.59	0.22	0.01	0.26	0.66	0.57	0.65	0.45	0.62	0.60	0.65
approach2	0.16	0.15	1.00	0.11	0.04	0.12	0.13	0.12	0.13	0.11	0.14	0.15	0.15
bounce0	0.15	0.22	0.37	1.00	0.18	0.74	0.25	0.22	0.23	0.21	0.28	0.43	0.35
bounce1	0.05	0.05	0.61	0.18	1.00	0.17	0.05	0.04	0.05	0.04	0.06	0.10	0.08
bounce2	0.26	0.26	0.52	0.74	0.04	1.00	0.45	0.40	0.43	0.41	0.49	0.64	0.57
bounce3	0.65	0.66	0.60	0.25	0.00	0.45	1.00	0.85	0.88	0.81	0.88	0.82	0.88
carry0	0.59	0.57	0.53	0.22	0.00	0.40	0.85	1.00	0.86	0.83	0.81	0.72	0.76
carry1	0.63	0.65	0.57	0.23	0.00	0.43	0.88	0.86	1.00	0.86	0.86	0.76	0.80
carry2	0.61	0.45	0.50	0.21	0.00	0.41	0.81	0.83	0.86	1.00	0.78	0.68	0.73
catch0	0.67	0.62	0.61	0.28	0.00	0.49	0.88	0.81	0.86	0.78	1.00	0.86	0.84
catch1	0.60	0.60	0.65	0.43	0.01	0.64	0.82	0.72	0.76	0.68	0.86	1.00	0.91
catch2	0.62	0.65	0.63	0.35	0.01	0.57	0.88	0.76	0.80	0.73	0.84	0.91	1.00

Table C.1: The similarity matrix for the video data used in the experiments.

Bibliography

[Borchardt(1993)] Gary Borchardt. Causal reconstruction. Technical report, MIT Artificial Intelligence Laboratory, 1993.

[DARPA(2010)] DARPA. Mind's eye program. Broad Agency Announcement, 2010.

[Frey and Dueck(2007)] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007. URL `www.psi.toronto.edu/affinitypropagation`.

[Katz(1997)] Boris Katz. Annotating the world wide web using natural language. In *RIAO Conference on Computer Assisted Information Searching on the Internet.*, 1997.

[Kristiaan()] Kristiaan. URL `http://commons.wikimedia.org/wiki/File%3AWoman_drinking_water.jpg`. CC-BY-2.0.

[Lany and Gmez(2008)] Jill Lany and Rebecca L. Gmez. Twelve-month-old infants benefit from prior experience in statistical learning. *Psychological Science*, 19 (12):1247–1252, 2008. doi: 10.1111/j.1467-9280.2008.02233.x. URL `http://pss.sagepub.com/content/19/12/1247.abstract`.

[Mongeau and Sankoff(1990)] Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

[O'Grady and Archibald(2008)] Willian O'Grady and John Archibald. *Contemporary Linguistic Analysis*. Pearson Canada, 2008.

[Pedregosa et al.(2011)Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofe F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[Rao(1998)] Sajit Rao. *Visual Routines and Attention.* PhD thesis, Massachusetts Instittue of Technology, 1998.

[Spitkovsky and Norvig(2012)] Valentin Spitkovsky and Peter Norvig. "from words to concepts and back: Dictionaries for linking text, entities and ideas", May 2012. URL http://googleresearch.blogspot.com/2012/05/from-words-to-concepts-and-back.html.

[Winston(2011)] Patrick H. Winston. The strong story hypothesis and the directed perception hypothesis. MIT Computer Science and Artificial Intelligence Laboratory, 2011.

[Y()] Yuval Y. Charlie the cat drinking tap water.jpg. URL http://upload.wikimedia.org/wikipedia/commons/6/60/Charlie_the_cat_drinking_tap_water.jpg. By Yuval Y (Own work) [CC-BY-SA-3.0 (www.creativecommons.org/licenses/by-sa/3.0).