StorySpace: Spatial Semantic Comparison of Narrative

by

Michael L Puncel

B.S. Electrical Engineering and Computer Science, M.I.T. 2012

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
May 2013 [June 2013]

Author: ______________________

Department of Electrical Engineering and Computer Science
May 24, 2013

Certified by: ______________________

Catherine Havasi, Thesis Supervisor
May 24, 2013

Accepted by: ______________________

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

Abstract

It is well known that humans are far more adept than computers at identifying similarities between stories. Humans are able to communicate values and event patterns back and forth through these narratives. Parents communicate through the telling of "The Tortoise and the Hare" that hard work and determination can often trump talent, and that hubris can lead to one's downfall.

It would be quite useful to develop a computational technique to apply this type of analysis to a story to relate to more generic cases. In this paper, I demonstrate the beginnings of a technique called *Spatial Semantic Analysis of Narrative* that identifies a "trajectory" for each story that enables comparison between them. These trajectories take into account the temporal progression of a story, which aims to provide a dimension of information beyond traditional "bag of words" comparisons. I present promising results when this technique is applied to a corpus of "how-to" articles scraped from the Internet as well as a corpus of Islamic texts annotated using Mark Finlayson's Story Workbench application. I also present next steps for improving the algorithm and allowing it to operate on standard untagged datasets.

Table of Contents

Introduction

It is the belief of many scholars at the forefront of the Artificial Intelligence community that the capability to tell and understand stories is critical to human intelligence. Story telling is a very critical part of human interaction in every day life. The desire to share the daily events that happen in ones life to his or her peers is intrinsic to human nature. Roger Schank [7, 8] argues that story telling and understanding is not only central to society, but that it is one of the main indications of intelligence. A person who is able to respond to what he or she is being told with clear, insightful comments that demonstrate an understanding of the subject matter, they are likely to be labeled as an intelligent person by their peers.

Story telling is also one of the main ways that values and lessons are communicated to one another. Children are taught to be helpful and kind to their neighbors with "good Samaritan" stories. People often choose automobiles due to anecdotal evidence acquired from friends that a certain brand or model is very reliable. This type of reasoning is all derived from the ability to draw upon a memory bank of stories, and discover similar patterns between those stories and the situation that one finds themselves in during the present.

The intelligence of computers is often judged by their ability to outperform humans on certain systematic tasks. For example, computers are able to beat the best human chess players with regularity. Schank, among others, argues that this is not true intelligence. Computers are very good at solving problems within a restricted domain, when their solutions can be methodically derived. However, this is not the end goal of Artificial Intelligence. To imagine a truly intelligent computer is to imagine one that can process information given to it in the format of natural human speech: not constrained to any particular grammar or syntax or topic. It would be able to draw upon past experiences to form coherent responses that offer new information to the user interacting with it. The project described in this paper, along with many others in the field represent attempts at taking one step toward this goal.

Background

OMCS

Humans utilize their common sense when they read and understand stories. They can tell what parts of a story are unrealistic based on past experience. They can make guesses about what will happen next by recognizing a common sequence of events. And they can apply lessons learned from a story to add more common sense knowledge. The Digital Intuition Group at the Media Lab has spent a great deal of time developing techniques for enabling computers with the cognitive ability to look past simple diction and examine the underlying concepts. A computer needs assistance to draw the connection between the words "student" and "teacher", for example. Upon seeing those words, a human is innately able to draw upon a bed of past knowledge to identify that they might be likely to find a student and a teacher occupying the same room. They are able to further realize that teaching is an occupation much like accounting or carpentry.

Computers are not able to perform this type of analysis. They are only able to match words based on their strings. Thus, an article with frequent mentions of the word "teacher" will not appear to be similar to one with the word "accountant", even if those articles were both discussing a similar topic such as compensation.

To solve this problem, the Digital Intuition Group began work on the Open Mind Common Sense (OMCS) [9] project. This project collected millions of English "facts" that define relations between concepts. Humans rely on known facts to process any statement they hear or read. For example, if the first sentence of a novel informs the reader that a man put on his crown, we are likely to make a lot of inferences. We know that a crown is a symbol of royalty, and that the character is likely to be a member of royalty, probably a king. We furthermore can surmise that he has political power over the region in which he lives, and that he is wealthy. We might be able to make a decent guess about the plot of the story as well. Perhaps the story is about the king's fall from power, or his conquest of foreign lands, or of a tragic assassination of someone in the royal family. All of this knowledge provides a context in our minds that we will apply when reading the second sentence. The second sentence might mean something altogether different to us now if the contents of the first were changed. Previous experience sets a stage which will influence the way the rest of the story as well as future ones make an impression upon us. We are capable of a much deeper reasoning than the simple word level. This is the problem that OMCS hopes to solve.

ConceptNet

ConceptNet [3, 6] is a component of the OMCS project which organizes the aforementioned common sense facts into a network of "concepts" and "relations". This network is represented as a graph where nodes are made up of concepts and edges are made up of relations. For example, the concept "overachiever" is linked to "student" with the "IsA" relationship. This signifies than an overachiever is a student. There is a similar rule that indicates that an underachiever is also a student. This representation allows projects built on top of ConceptNet to make smarter comparisons between differing concepts. Programs will now find similarities between “student” and “teacher” because they are both people in an academic environment. A more naïve technique might only look at the words and consider “student” to be as related to “frog” as it is to “teacher”.

Related Work

AnalogySpace

There have been many attempts in the AI field to computationally identify similarities between concepts. One application built by the Digital Intuition Group that utilizes ConceptNet is AnalogySpace [10], which attempts to identify similarities between concepts using the database of facts that ConceptNet holds. It can be used to define a concept category, and is then able to identify concepts from a corpus that fit into that category.

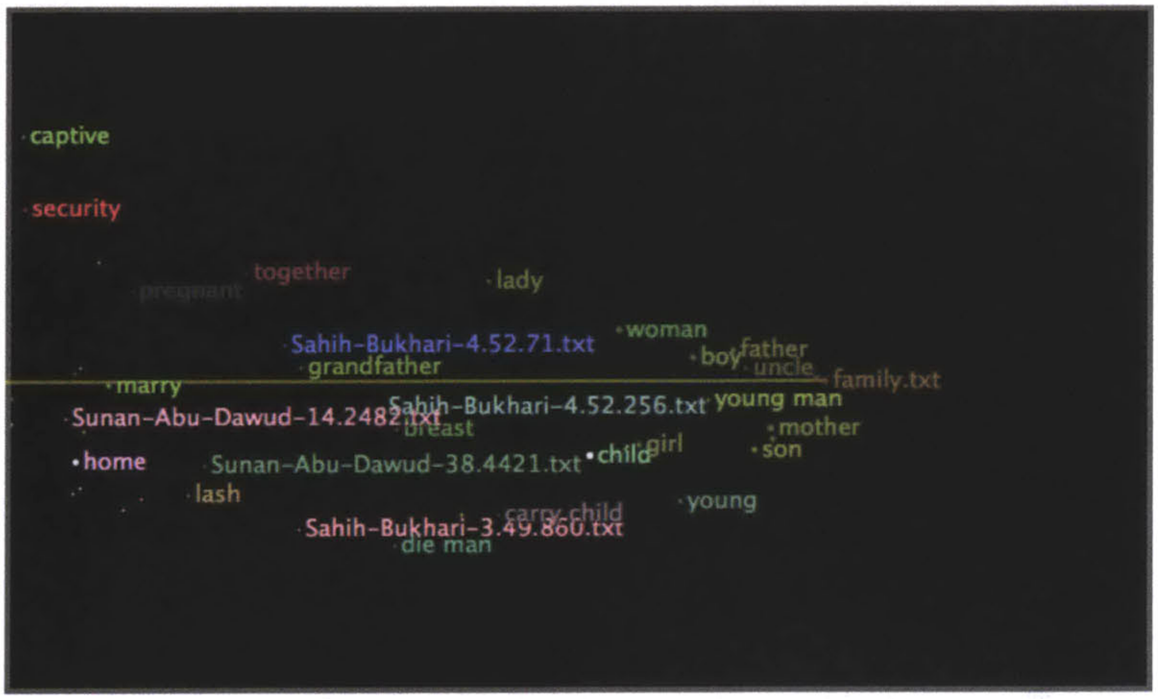


Figure 1: A visualization of AnalogySpace using Luminoso [11]

Figure 1 view shows a spatial representation of concept clusters, with each dimension representing a type of meaning. For example, the "family.txt" node that can be seen on the right is simply a file that contains the word "family". To generate this visual, I defined family.txt to be along the X-axis. All other nodes that are clustered near it were determined to be semantically similar. It is apparent that this works decently well. For example, the words "boy", "father", "mother", and "son" all appear along that axis very close to bad.txt. Progressing further left, the concepts have a lesser component along the "family" axis, and it could be surmised that these concepts are not as related to "family".

AnalogySpace does this sort of blending of concepts by building a matrix of concepts along with feature information that comes from ConceptNet, and then performing dimensionality reduction on it. The matrix is built with one concept in each row, and each column representing a "feature". The value of each entry in the matrix is a "1" if the corresponding concept has the corresponding feature, "0" if unknown, and "-1" if it does not have that feature. For example, the concepts "ivy leaguer" and "art student" would both have a "1" for the (IsA, student) feature, signifying that both ivy leaguers and art students fall under the category of student. In this way, a similarity score between two concepts can be found by taking the dot product of the two corresponding rows. The score will increase if both concepts have the same polarity for a given feature, and will decrease if they have opposite polarities.

The matrix in this form, however, is quite unwieldy given the sheer amount of concepts in a corpus as well as the number of features present in ConceptNet. Thus, dimensionality reduction comes into play as a useful way to reduce the computational load to compute the similarity between concepts. This is done by performing the singular value decomposition of the concept/feature matrix, resulting in a relation of the form $A = U\Sigma V^T$. Ordering the singular values in $\sum$ from largest to smallest and discarding all but the highest *k* components results in an approximation for the full matrix A of the form $A_k = U_K \sum_K V_K{}^T$. Performing dot products in this realm is more computationally feasible, and also has the beneficial result of giving useful similarity scores where there might not have been any in the original space, which would occur when two concepts share no features in common.

AnalogySpace's technique of producing a spatial representation of concepts in order to find similarities is the inspiration for the algorithm underlying StorySpace.

CrossBridge

CrossBridge [4] is another application with the goal of identifying analogies between a "source domain" and a "target domain". It relies on a knowledge base similar to ConceptNet in that it defines relations between concepts in the form of a graph. CrossBridge operates under the assumption that an analogy can be found by identifying isomorphism between subgraphs of the knowledge base (the source domain) and a target domain. An example of such an analogy given in the paper includes likening a bird to a car through the relations PartOf(wing, bird) and PartOf(wheel, car). The graph linking nodes "wing" and "bird" with the "PartOf" relation is isomorphic with the graph linking nodes "wheel" and "car" with the identical "PartOf" relation. The more edges the graphs have in common, the stronger the analogy.

CrossBridge seeks to improve upon other applications that attempt to find isomorphism between subgraphs. One drawback to previous techniques is that they are NP-complete, making them very inefficient for large data sets. CrossBridge also boasts the ability to identify analogies that previous implementations were unable to. This improved technique begins with building a matrix with each row corresponding to a "domain" and each column corresponding to a "relation structure". Here, the term "domain" refers to a subgraph of the knowledge base with a particular vertex ordering. The allowable size of each domain is constrained in order to control the size of the matrix. The example posed in the paper only allows domains with 3 concepts and at least 2 relations. It also places restrictions on the size of the relation structures. The authors mention that typically only relation structures with a minimum of 1 and a maximum of 3 edges are allowed. Analogies are unlikely to be found with domains that consist of more than 3 edges due to the increasing complexity of these domains proportional to the number of edges.

Even with the imposed restrictions on size, this matrix is still very unwieldy and computationally costly. Therefore, dimensionality reduction is used to cut down on the computation cost of finding analogies. Much like in AnalogySpace, a truncated singular value decomposition is used to generate a matrix approximation with a smaller dimensionality.

Once the dimensionality reduction has been taken place, it is possible to retrieve analogies between the source domain (the knowledge base in the reduced matrix) and a target domain. If the target domain has the same number of concepts as the domains in the matrix (3 in this case), then a simple nearest neighbor search is all that is necessary to find the domains in the knowledge base that are most similar. This works by building the same vector that each of the domains in the matrix has, with each entry indicating the presence of the corresponding relation structure. This vector is then projected into the dimensionality reduced domain space using the *V* matrix from the SVD. Then the cosine similarity is determined between all source domains. This list is sorted to rank the analogies by score. In the event that the target domain is larger than 3 concepts, it is broken down into several 3-concept domains and the resulting analogies are merged using a heuristic search.

Deriving Narrative Morphologies via Analogical Story Merging

Mark Finlayson presents a technique for identifying common morphologies between stories within a corpus via "Analogical Story Merging" [2]. He holds that folk tales told by a particular culture often show similarities in their structure, such as "the *Three Brothers* in Slavic cultures, the *Trickster* in North American native cultures, or *Cinderella* in the West." Identifying these structures is often done manually, and they require much time to discover and verify. Finlayson posits that Analogical Story Merging can perform this function computationally on a given corpus in a fraction of the time. The base functionality for Analogical Story Merging is derived from Bayesian model merging. An example use for this given in Finlayson's paper is to generate a single grammar that can explain the two character sequences, "ab" and "abab". Figure 2, which comes from Finlayson's paper shows the step by step process of merging two Hidden Markov Models (one for each character sequence) into one grammar that fits both, and in fact generalizes more to anticipate other possible sequences.

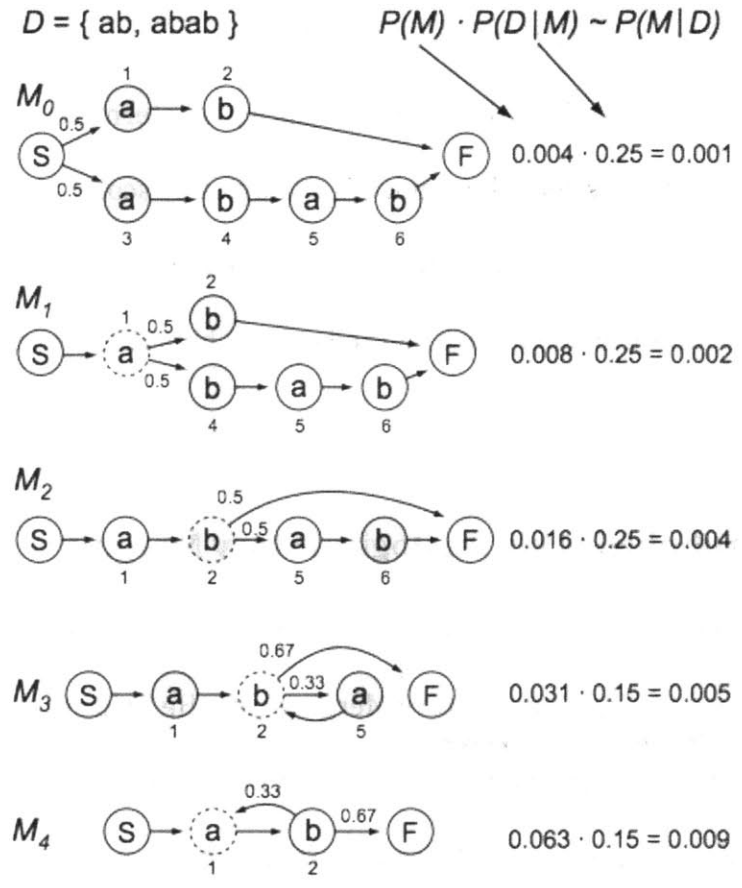


Figure 2: Bayesian model merging.

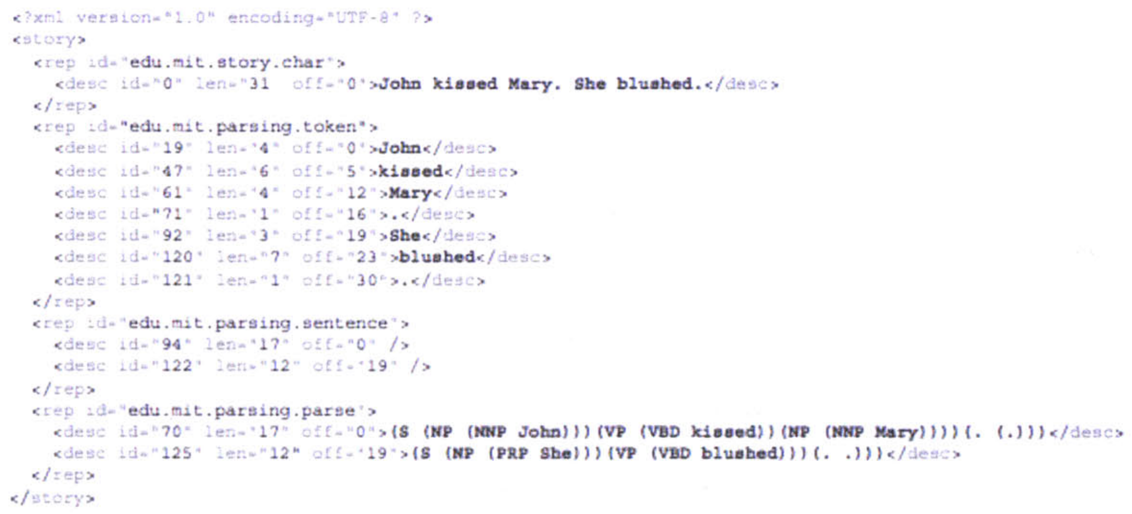
The process begins by constructing an HMM that is the union of works by searching over the space of possible state merges, and testing the resulting probability of the model given the data, which is the product of the prior probability of the model with the probability of the data given the model. The first two merges performed in this example do not change the possible output sequences of the model, while the third results in allowing all sequences with any number of instances of "ab" (written as (ab)+). In the figure, gray states are the ones that will be merged together in the next step, and dashed states were merged in the previous step. On the right side of the figure, the Bayesian probabilities used to make each merge decision are shown.

This technique is used in Analogical Story Merging by expressing each of a set of stories as a sequence of states, and taking their union to generate the initial morphology M_0 as shown in Figure 2 for the general case. At this point, Bayesian model merging can take place to derive a general morphology that best fits the story corpus.

Story Workbench

Finlayson also presents a tool to speed up the process of semantic annotation of large quantities of text called the Story Workbench [1]. This tool was used to annotate one of the corpora StorySpace was tested on. It provides "guesses" at the proper placement and identification of annotation marks, and then allows a human to make corrections or elaborations upon those guesses. In this way, it greatly reduces the amount of work required to annotate a large corpus of text. This is valuable because algorithms, such as StorySpace, often benefit from these tags. A fast way of annotating is a potential boon to the natural language processing field.

Story Workbench works by combining well-accepted and highly accurate automatic tagging techniques with the ease of a graphical user interface. Finlayson claims that previous work in the area of semi-automatic text annotation required



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<story>
  <rep id="edu.mit.story.char">
    <desc id="0" len="31  off="0">John kissed Mary. She blushed.</desc>
  </rep>
  <rep id="edu.mit.parsing.token">
    <desc id="19" len="4" off="0">John</desc>
    <desc id="47" len="6" off="5">kissed</desc>
    <desc id="61" len="4" off="12">Mary</desc>
    <desc id="71" len="1" off="16">.</desc>
    <desc id="92" len="3" off="19">She</desc>
    <desc id="120" len="7" off="23">blushed</desc>
    <desc id="121" len="1" off="30">.</desc>
  </rep>
  <rep id="edu.mit.parsing.sentence">
    <desc id="94" len="17" off="0" />
    <desc id="122" len="12" off="19" />
  </rep>
  <rep id="edu.mit.parsing.parse">
    <desc id="70" len="17" off="0">(S (NP (NNP John)))(VP (VBD kissed))(NP (NNP Mary))))(. (.)))</desc>
    <desc id="125" len="12" off="19">(S (NP (PRP She)))(VP (VBD blushed)))(. .)))</desc>
  </rep>
</story>
```

Figure 3: Example story annotated by Story Workbench in XML format.

the users to be highly trained in the specifics of each program. The Story Workbench presents a GUI that is intended to be familiar to anyone with general computer experience.

A corpus tagged by the Story Workbench is of particular use in StorySpace because it contains information related to the sequence of events in the story. These events are relied upon by StorySpace's underlying algorithm, as will be described in detail later in this paper.

Using Narrative Functions as a Heuristic for Relevance in Story Understanding

Emmet Tomai and Ken Forbus present a technique for automated story understanding aided by a large knowledge base (much like ConceptNet) [12] known as ResearchCyc [5]. The goal of the technique is to understand the relevance of each sentence in the story relative to the others. For example, a sentence might serve to introduce a character that is present in sentences that appear later in the story. The technique involves mapping each term in the knowledge base to a "semantic frame", which is a logical structure that has slots for other terms to fit in. An example frame for "went" is shown in Figure 4. These frames can be nested within each other to develop a number of possible parse trees for a given story. This technique requires very heavy computation, as multiple frames exist for each term, and thus there will be many possible frame structures that can be generated for a given story.

```
(and
 (isa :ACTION Movement-TranslationEvent)
 (primaryObjectMoving :ACTION :SUBJECT))

(thereExists (TheList hel movel)
 (and
  (isa movel Movement-TranslationEvent)
  (primaryObjectMoving movel hel)))
```

Figure 4: Sample frame for "went"

StorySpace

Overview

The algorithm behind StorySpace that is used to compare stories to each other is inspired by the spatial mindset of AnalogySpace. The idea is to map each story in a given corpus to a vector in a many-dimensional space. These vectors can then be compared to each other using various metrics that I will discuss later in this paper. One benefit to a technique such as this one is that large amounts of data can be preprocessed, namely the construction of this space. Once these vectors have bene computed, they can be used repeatedly to make comparisons to any number of test stories.

The success of StorySpace hinges upon the ability to capture each story's relevant semantic information in each vector. One might think of the vector as the story's "trajectory" or a linear approximation of the plot. Stories that are similar will start in a similar point in space, and progress along similar paths to similar endpoints. A theoretical pair of stories that have no similarities to each other would progress along orthogonal vectors.

Each dimension in this space that the story vectors live in corresponds to a concept. Much like in ConceptNet, a concept can be nearly any English word (or in some cases, a duplet of words). The dimensionality of the space is confined only to concepts with nonzero frequencies within any given context in order to preserve memory and improve computational speeds.

Data

Due to the temporal nature of the algorithm and the desire for simplicity in its development process, the two corpuses that have been used both contain event information. This eliminated the need for automatic or manual event extraction from the corpus, and allowed me to develop and test the algorithm with the fewest number of factors that can affect its performance.

The first data set that was used is a corpus of "how-to" articles scraped from various websites such as ehow.com and wikihow.com. These articles each have a title that represents the lesson that the article is attempting to teach to the reader. An example article (with only the first two "steps" included) is shown in Figure 4. This data set is ideal for StorySpace because each article can be thought of as analogous to a natural human story, with multiple events culminating in an outcome. In the case of these articles, the title can be thought of as the outcome (i.e. "cancel a credit card" as in the example in Figure 5) and each step along the way can be thought of as an event. As stated before, this is convenient because events do not need to be automatically detected as the data is naturally segmented along event boundaries.

The second dataset is a collection of texts annotated by Finlayson's Story Workbench [1]. This dataset contains stories told in a natural form, but are annotated with event information. The stories are in the same format as shown in Figure 3 (although that particular example does not have tagged event information). It gives precise start and endpoints for each event in the text. The text is split on each event starting point. This means that every word present in the story is used in

```
Cancel a credit card
1. Sort your credit cards. Decide which cards you
use and which ones you do not. It might also help to
know which cards have the highest interest rates.

2. Keep two or three cards you use frequently. That
way, you have credit cards to make online
reservations, to pay for a rental car and to bail
you out in the event of an emergency. But you
shouldn't need more credit cards than that.
```

Figure 5: Sample how-to article with first two steps shown

the algorithm, but simply segmented along event boundaries. Precise details on how this information is used will be discussed in the next section.

Algorithm

As discussed in the previous section, the algorithm works by mapping each story in a corpus to a vector in a many-dimensional space. This process begins with mapping each event to a point in this space, and then determining the vector that is the best-fit line through those points. In order to do this, the concepts in each string of text representing an event in the story must be extracted. This is done using a function called "extract_concepts()" that discards any words with little semantic value (such as "the" and "and") and leaves only the ones that give a better sense of the content of the source string. A concept can be made up of a single word or a pair of words. The utility of allowing a concept to consist of three words is outweighed by the performance decrease of the algorithm when doing so.

Once each event has been converted to a set of concepts, it can then be converted into a SparseVector, which is an object type that comes from the Divisi package. As its name suggests, it is particularly useful in situations where the dimensionality of the space that the vector inhabits is much higher than the number of nonzero components in the vector. It saves memory by omitting any zero valued components in the vector, and avoids the loss of information by labeling each axis. In this case, each axis is labeled with its corresponding concept. Figure 6 shows an example of a string of text being converted first to a list of concepts, and finally a SparseVector instance that can be used later on.

```
>>> concepts = extract_concepts("Visit a store that
offers cold-pressed carrier oils")
>>> print concepts

[u'visit store', u'visit', u'store offer', u'store',
u'offer cold-press', u'offer', u'cold-press carrier, … ]

>>> Divisi2.category(concepts)

SparseVector (101 of 101 entries): [oil no=1, oil
grapeseed=1, essential oil=1, vary=1, aroma=1, scent
go=1, go=1, subtle=1, find=1, cold-press=1, avocado=1,
almond oil=3, extensive=1, include=1, offer extensive=1,
visit store=1, apothecary health=1, oil vary=1, health
store=1, benefit=1, ...]
```

Figure 6: demonstration of converting event text to a point in space

Once one SparseVector has been obtained for each event, that information can be used to compute the vector that will later represent the story in question. Despite the fact that the concepts are wrapped up in a SparseVector, it can be thought of as a point in space. Thus, each story has a series of points in number equal to the number of events in the story.

The story vector is computed using linear algebra techniques. First, a matrix is constructed with each row consisting of one of the SparseVectors and each column representing a particular concept. In this way, the number of rows is equal to the number of events in the story, and the width of the matrix is equal to the size of the story's vocabulary of concepts. Once the matrix is constructed, a singular value decomposition of it is performed. This gives us an equation of the form $A = U\Sigma V^T$, with Σ being a diagonal matrix of singular values. We find the maximum singular value, and take the right singular vector that corresponds to it. This vector represents the best-fit line through the collection of event points, and will be the story vector.

After each story vector is computed, they are written to a file in JSON format for later use. Another program then reads each of these vectors in, and compares them using one of two methods. Each of these methods outputs a distance metric for each pair of stories it tests. A high distance value indicates that the stories in a particular pair are not very similar, and a low value indicates that they are similar. These values aren't indicative of much on their own, but they enable a relative comparison between stories in a corpus. For example, we can choose a single story and ask which of the remaining are most similar to it. This would obviously be the story that results in the smallest distance value.

The first distance metric I tried to compare story vectors was Euclidean distance. This involves thinking of the vectors as a point in space, and then computing distance. To test the performance of this metric on the corpus of how-to articles, I ran an $O(n^2)$ algorithm that compared each story vector to all of the other story vectors one at a time. This involves selecting a "left vector" and then testing all remaining vectors as the "right vector" one at a time. For a given left vector, the smallest observed distance value is remembered. Any time a new right vector generates a distance value that is smaller than the one recorded, a line is printed to an output file containing information identifying the left vector, right vector, and the calculated distance. The expected output for each left vector is a list of right vectors with decreasing distance values. If the algorithm is working properly, the stories corresponding to the right vectors would be less similar toward the top, and more similar toward the bottom. Sample output is shown in Figure 7. The numbers on the left represent the distance between the two vectors being measured. The first string in the list is the title of the story corresponding to the left vector, and the

```
43.510749864[u'mix essential oils with a carrier oil', u'add scent to
a candle']
42.7946014271[u'mix essential oils with a carrier oil', u'make an
equine drawing salve']
40.9016129383[u'mix essential oils with a carrier oil', u'feed equine
oil as a fat supplement']
40.3264222565[u'mix essential oils with a carrier oil', u'make
natural odor spray for a dog']
35.5135971457[u'mix essential oils with a carrier oil', u'get rid of
ear mites with essential oils']
35.0702762468[u'mix essential oils with a carrier oil', u'make
scented bath oils']
33.9185447192[u'mix essential oils with a carrier oil', u'make
aromatherapy massage oil']
27.7643960511[u'mix essential oils with a carrier oil', u'make a
relaxing massage oil']
```

Figure 7: Similarity progression for story "mix essential oils with a carrier oil". Each line that is printed shows a right story that is more similar to the left one than any previous iteration. Thus, we expect a progression from less similar to most similar from top to bottom. In this case, it seems to work well.

second string is the title of the story corresponding to the right vector. It appears that the stories on the right become more and more similar to the one on the left as we progress from top to bottom. The first right story to appear, "add scent to a candle" is the first story that we test from the corpus. By default, it has the smallest distance observed so far, because it is the only distance observed at this stage. At the end, we are comparing "mix essential oils with a carrier oil" to "make a relaxing massage oil". This is a very good match because mixing essential oils with carrier oils is an important step of making a massage oil.

In many cases, however, Euclidean distance between the story "vectors" seems to miss the mark. An example of this is shown in Figure 8. As we progress

```
54.9290750054[u'write an executive summary', u'mix essential oils
with a carrier oil']
35.6023747013[u'write an executive summary', u'add scent to a
candle']
24.7979827751[u'write an executive summary', u'know if a town is pet
friendly']
20.9772495503[u'write an executive summary', u'boil an egg']
18.2690353424[u'write an executive summary', u'buy things on amazon
without a credit card']
17.8123813676[u'write an executive summary', u'make a graph using
excel']
16.3116884347[u'write an executive summary', u'make money on the
internet']
15.4037688596[u'write an executive summary', u'shop for a wireless
home security alarm system']
13.9918418677[u'write an executive summary', u'put together a child
safety kit']
13.0907773778[u'write an executive summary', u'stop your cat from
urinating and defecating inappropriately']
12.4348839522[u'write an executive summary', u'use a drill']
12.3172031362[u'write an executive summary', u'clean a hamster cage
with urine that is hardened']
12.2159123277[u'write an executive summary', u'calm your horse']
11.8669532939[u'write an executive summary', u'organize a horse
show']
```

Figure 8: In this example, Euclidean distance measurements do not perform well

from top to bottom, we go from "mix essential oils with a carrier oil" and "add scent to a candle" (which appear first because they are the first two stories in the corpus, thus the first two we compare against "write an executive summary") and end up at "organize a horse show". This means that the algorithm believes "organize a horse show" to be the most similar story to "write an executive summary" out of the entire corpus. This is obviously not a desired result. Measuring the Euclidean distance between story vectors is an inferior technique to measuring the angle between them, because two vectors might be "close" to each other in space even if they are orthogonal or nearly orthogonal. Stories that have a small vector magnitude will appear similar to each other even if they do not share similarity in semantic content.

Measuring the angle between vectors instead of the distance proves to be more fruitful. Figure 9 shows the results for the same left story, "write an executive summary", when analyzed using vector angles. The top-down progression makes much more sense in this case. The most similar right story that the algorithm finds now is "write an executive summary for a research report". This is clearly a much better choice than when Euclidean distance was used.

The angle between vectors was the metric used on Finlayson's corpus of annotated texts as well. The algorithm is largely the same, with the only difference being the way events are split up. In the how-to dataset, events are split long the boundaries indicated in the article by step number. In Finlayson's corpus, character indices are given that show the event boundaries. The text is simply split along these boundaries in order to get the text for each event. After this stage, it is all the same. Each event is mapped to a point in space, and the best-fit line through those points is found using the singular value decomposition of the resulting matrix.

```
1.54070611919[u'write an executive summary', u'mix essential oils with a
carrier oil']
1.53621649574[u'write an executive summary', u'add scent to a candle']
1.52899541426[u'write an executive summary', u'cancel a credit card']
1.51324756018[u'write an executive summary', u'buy things on amazon
without a credit card']
1.47390747491[u'write an executive summary', u'make money on the
internet']
1.46602592186[u'write an executive summary', u'cure a dogs upset
stomach']
1.46310446977[u'write an executive summary', u'read a yorkie contract
before buying']
1.46086452826[u'write an executive summary', u'write a name on a plastic
dog id tag']
1.45711759517[u'write an executive summary', u'puppy proof the house']
1.45410906269[u'write an executive summary', u'succesfully raise and
care for a freshwater angelfish']
1.44589630599[u'write an executive summary', u'make a refugium out of a
10 gallon fish tank']
1.43725741628[u'write an executive summary', u'create a habitat for
dwarf hamsters']
1.43568013865[u'write an executive summary', u'ride horses in the
winter']
1.42120544826[u'write an executive summary', u'select a dog breeder']
1.40109142194[u'write an executive summary', u'word church wedding
programs']
1.39807055264[u'write an executive summary', u'use graphic organizers as
a reading/writing strategy']
1.35006014375[u'write an executive summary', u'write a research report
for high school']
1.34543234023[u'write an executive summary', u'develop an expository
writing paper']
1.32397905404[u'write an executive summary', u'write a report on oceans
for grade 4']
1.29102195637[u'write an executive summary', u'write an essay in outline
format']
1.09902910515[u'write an executive summary', u'write an executive
summary for a research report']
```

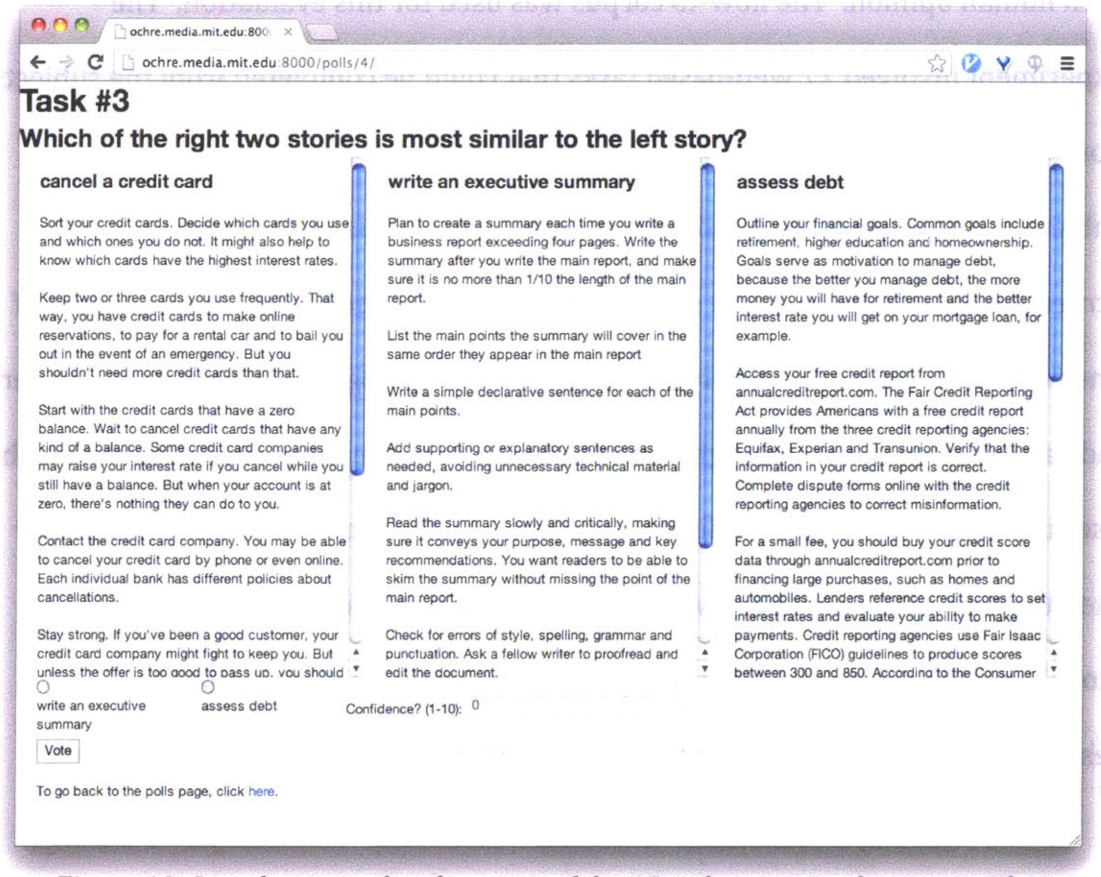
Figure 9:Story comparisons for " write an executive summary" when using angle instead of Euclidean distance to compare vectors. Notice that the similarity values are now angles.

Evaluation

An evaluation of the algorithm was performed using 21 human subjects in order to determine how well the results of running the algorithm on a corpus align with human opinion. The how-to corpus was used for this evaluation. The experiment involved 17 web-based tasks that could be completed from the subjects' homes. Each task involved displaying to the user three separate juxtaposed. The subject was then asked to decide which of the two stories on the right side of the screen was most similar to the story on the left side of the screen. A screenshot of one such task is shown in Figure 10. In addition to giving a multiple choice answer indicating which of the two rightmost stories is most similar to the leftmost one, the user is asked to indicate a confidence level on a scale from 1 to 10 inclusively. For the purposes of this section, I will refer to the story on the left as the "reference story", and the center and right (test) stories as "story one" and "story two", respectively.

For each task, there is a "correct" choice, and an "incorrect" choice, with respect to the output of the algorithm. The correct choice is whichever of story one and story two has the smallest angle to the reference story. This is because it is the choice that the algorithm believes to be closest to the reference. The incorrect choice is the story with the larger distance to the reference. In order to quantify the general opinion expressed by the users, I calculate the "average confidence" of the responses for each task. This value is defined as the average confidence score given in the submissions, with confidences corresponding to an "incorrect" choice treated as negative. For example, if two users select the correct choice with a confidence of

5 and a third user selects the incorrect choice with a confidence of 1, we would calculate an average confidence score of (5 + 5 - 1) / 3 which equals 3. The 1 is negatively weighted because it corresponded to an incorrect choice.



Task #3

Which of the right two stories is most similar to the left story?

cancel a credit card

Sort your credit cards. Decide which cards you use and which ones you do not. It might also help to know which cards have the highest interest rates.

Keep two or three cards you use frequently. That way, you have credit cards to make online reservations, to pay for a rental car and to bail you out in the event of an emergency. But you shouldn't need more credit cards than that.

Start with the credit cards that have a zero balance. Wait to cancel credit cards that have any kind of a balance. Some credit card companies may raise your interest rate if you cancel while you still have a balance. But when your account is at zero, there's nothing they can do to you.

Contact the credit card company. You may be able to cancel your credit card by phone or even online. Each individual bank has different policies about cancellations.

Stay strong. If you've been a good customer, your credit card company might fight to keep you. But unless the offer is too good to pass up, you should

write an executive summary

Plan to create a summary each time you write a business report exceeding four pages. Write the summary after you write the main report, and make sure it is no more than 1/10 the length of the main report.

List the main points the summary will cover in the same order they appear in the main report

Write a simple declarative sentence for each of the main points.

Add supporting or explanatory sentences as needed, avoiding unnecessary technical material and jargon.

Read the summary slowly and critically, making sure it conveys your purpose, message and key recommendations. You want readers to be able to skim the summary without missing the point of the main report.

Check for errors of style, spelling, grammar and punctuation. Ask a fellow writer to proofread and edit the document.

assess debt

Outline your financial goals. Common goals include retirement, higher education and homeownership. Goals serve as motivation to manage debt, because the better you manage debt, the more money you will have for retirement and the better interest rate you will get on your mortgage loan, for example.

Access your free credit report from annualcreditreport.com. The Fair Credit Reporting Act provides Americans with a free credit report annually from the three credit reporting agencies: Equifax, Experian and Transunion. Verify that the information in your credit report is correct. Complete dispute forms online with the credit reporting agencies to correct misinformation.

For a small fee, you should buy your credit score data through annualcreditreport.com prior to financing large purchases, such as homes and automobiles. Lenders reference credit scores to set interest rates and evaluate your ability to make payments. Credit reporting agencies use Fair Isaac Corporation (FICO) guidelines to produce scores between 300 and 850. According to the Consumer

○ write an executive summary ○ assess debt Confidence? (1-10): 0

Vote

To go back to the polls page, click here.

Figure 10: Sample screenshot from one of the 17 tasks presented to users. The radio buttons at the bottom are used to indicate whether the user thinks the story in the center is most similar to the story on the left, or if the story on the right is more similar. They also put in a confidence score of 1 to 10 inclusively.

Attempts were made to vary the "difficulty" of each task. For some tasks, story one and story two have very different angular distances to the reference story. This scenario means that the algorithm is very confident that one of the test stories is more similar to the reference story than the other. If the algorithm works as intended, the users would be expected to lean equally heavily toward the story with the smaller distance relative to the reference. This phenomenon would manifest itself both in the frequency of users that made the correct selection as well as a high average confidence score. For a difficult task, meaning that both of the test stories have similar angle differences with the reference story, we would expect lower average confidence scores. I quantify the difficulty of a task using the "incorrect to correct distance ratio", which I will explain later in this section.

Of the 17 tasks, the average confidence score was positive for 15 of them. This means that the human subjects in aggregate agreed with the binary decision of the algorithm 88% of the time. The minimum average confidence score observed was -2.71, and the maximum was 8.65. This indicates that even when the humans disagreed with the algorithm, the average confidence score was relatively close to zero. On the other side of the spectrum, the subjects strongly agreed with the algorithm in a few cases.

We would expect the average confidence score to vary depending on the difficulty of the task. To quantify the difficulty of a given task, I use the "incorrect-to-correct distance ratio". This is defined as the ratio of the angles of the incorrect choice to the reference and the correct choice to the reference. For example, if the correct choice has an angle of .5 relative to the reference story, and the incorrect choice has an angle of 1 relative to the reference story, the incorrect-to-correct distance ratio is 2. The higher this ratio is, the larger the gap between the two choices, meaning that the task is less difficult. We would expect to see that the average confidence score for a task is positively correlated with the incorrect-to-correct distance ratio. Figure 11 shows a plot of the 17 data points with average confidence on the y-axis and incorrect-to-correct distance ratio on the x-axis. We do see the positive correlation between the two metrics.

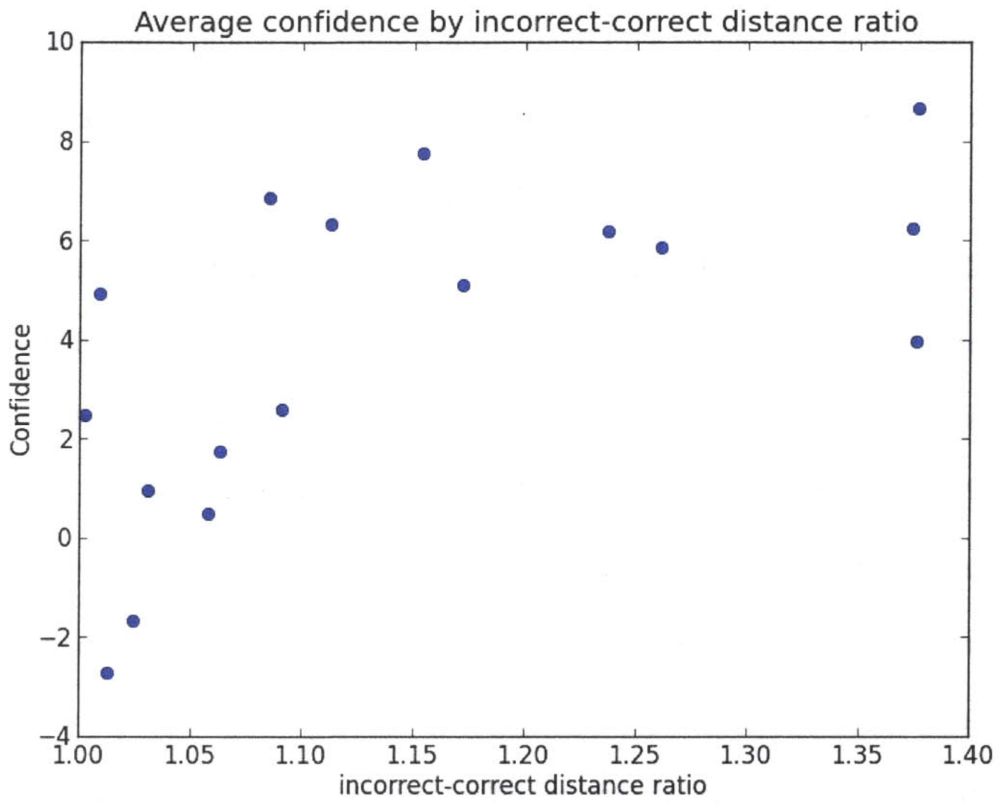


Figure 11: Average confidence score vs. incorrect-to-correct distance ratio

I also hypothesized that decisions would be more difficult when the correct choice is far away from the reference; even if it is far superior to the incorrect choice. After viewing output from StorySpace, I noticed that it is easy to spot that the stories with the lowest distances are very similar to the given reference, however in the mid to high distance ranges it is difficult to determine relative similarity. For example, when using "cancel a credit card" as a reference story, "sell books on Amazon" is the test story that has the greatest angular distance of 1.64. "Give a keepsake Christmas present for Mom" has a distance of 1.45. I hypothesized that a human would not be able to tell which of these two stories is relatively more similar to the reference. As a result, some tasks include a pair of test stories with neither being very close to the reference. To quantify this scenario, I define the "correct-to-best ratio" as the ratio of the angles of the correct choice to the reference

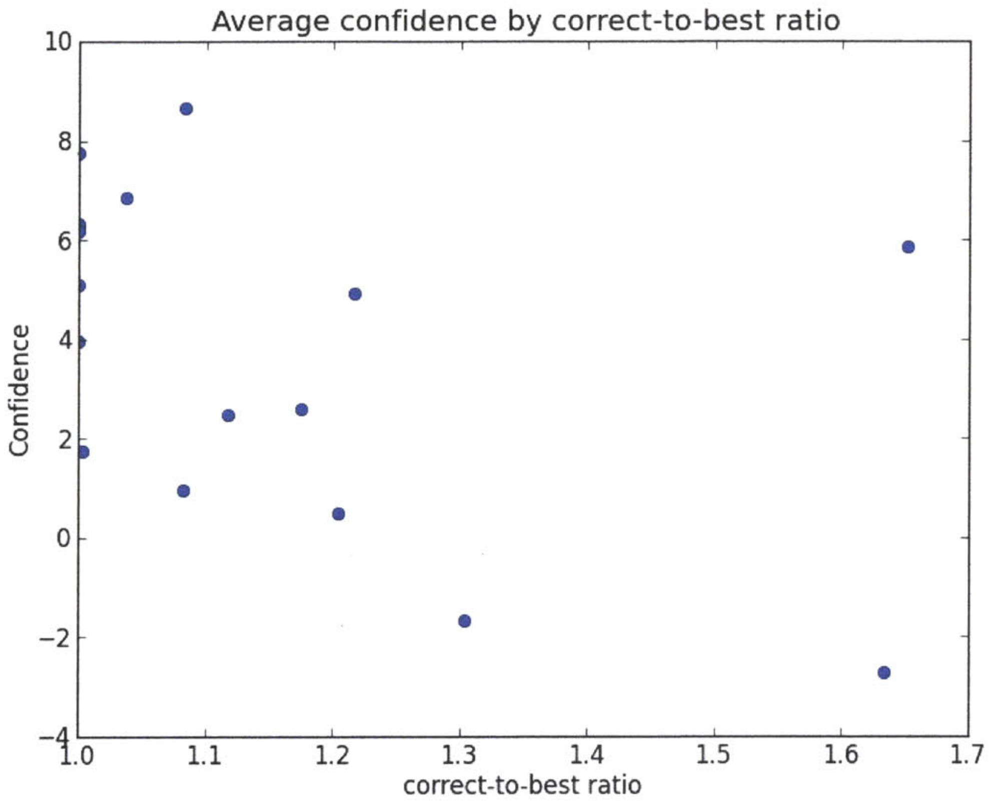


Figure 12: Average confidence vs. correct-to-best ratio.

and the best story to the reference. The best story is the story in the corpus that is closest to the reference, whether or not it is included in the task. A correct-to-best ratio of 1 means that the best story in the corpus is in fact one of the test stories in the task. A high correct-to-best ratio means that neither of the stories presented to the user is very similar to the reference story. Figure 12 shows the average confidence of each task with respect to the correct-to-best ratio. There does appear to be a general negative trend here, but with a large amount of deviation. It is interesting to note that the two data points with negative average confidences (i.e. humans making the incorrect aggregated choice) fall in the top 3 of tasks ranked by correct-to-best ratio.

Conclusion

StorySpace aims to identify the relative similarity between stories in a corpus using criteria that look a little deeper into the content of each story than traditional algorithms. It examines the semantic content at different events in the stories, and constructs a spatial (vector) representation of each story that it then uses to make its decisions. At the time of this writing, it depends upon explicit event tagging in the corpuses it is used on. The how-to corpus contains implicit events in its "step" structure, and the corpus of Islamic texts contains explicit event information tagged by Mark Finlayson's Story Workbench.

StorySpace seems to perform very well based on the evaluation results on the how-to corpus. Human subjects are easily able to make judgment calls about relative similarities that are in agreement with the values computed by the algorithm in a dataset as large as the how-to corpus. No evaluation has yet been performed with the corpus of Islamic texts, but at first glance it is difficult to discern how well the algorithm performed. This is perhaps due to the small size of the dataset (63 stories).

StorySpace yields the results it does with a very simple algorithm. It does not yet take advantage of the knowledge present in ConceptNet; it only uses the concept extraction functionality to convert a string of text into a vector of concepts. There is much room for improvement, but the results so far are encouraging. The human subjects agreed with the algorithm in 88% of the tasks. The confidence levels of the humans also appeared to be correlated with the values computed by StorySpace. When one of the test stories is a clear numerical winner based on the algorithm's outputs, the average confidence score of the humans seems to show a similar skew. We are confident that improvements can be made to the algorithm in addition to generalizing it to operate on more generic and natural datasets of human narrative.

Future Work

The work presented earlier in this paper represents the first few steps taken toward the broader goal of StorySpace. The datasets used for testing purposes were easily suited for developing the techniques involved, but clearly do not represent the type of natural human narrative that we eventually hope to be able to process. The next step toward that goal will be to use a corpus of news stories. This data is not naturally segmented into events like the how-to corpus is, and it will not be tagged using the Story Workbench. Instead, we will have to incorporate a form of automated event extraction into StorySpace to serve this purpose.

StorySpace also would benefit from an increase in computational efficiency. It takes an hour to run the algorithm on the corpus of Islamic texts, which has only 63 stories in it. A speedup could come about using dimensionality reduction similar to CrossBridge and AnalogySpace. We would maintain the dimensions that give us differentiation between stories, and cast out dimensions that do not hold valuable information. This also might have a desirable side effect of increased accuracy in identifying the similarity between stories, because the dimensions that would be lost carry little value in determining similarity, and may skew calculations.

Some value could also be added to the algorithm by incorporating knowledge from ConceptNet. Currently, similarity is only found when word choice is the same. ConceptNet, however, would grant the ability to compare similarity based on word classes. This would improve results in cases where two stories have similar plots, but the topics are different enough that the same words do not appear in each text.

There would additionally be some merit to a visualization for StorySpace. Currently, all output is text based, which makes it difficult to draw any conclusions about the corpus using the results of the algorithm. Currently the best way to do this is pick a particular story as a reference point and sort the rest of the stories by their distance from that point. This makes it difficult to identify story clusters or other phenomena in the results.

Finally, the algorithm currently does not take order of events into account. The best-fit line technique used by computing the singular value decomposition of the event matrix discards ordering of the temporal events. This might be a good thing due to the fact that stories are often told out of order, however in many cases it would change the meaning of the story. Techniques to incorporate temporal information should definitely be looked into.

Acknowledgements

I would like to thank both Defense Advanced Research Projects Agency and Deloitte for sponsoring this project. I would also like to thank my supervisor, Catherine Havasi, for sharing her expertise and guidance throughout this project. Rob Speer was also very helpful with providing the server administration I needed to run StorySpace test runs.

References

[1] Finlayson, Mark Alan. "Collecting semantics in the wild: The story workbench." Naturally Inspired Artificial Intelligence, Technical Report FS-08-06, Papers from the AAAI Fall Symposium. 2008.

[2] Finlayson, Mark Alan. "Deriving narrative morphologies via analogical story merging." Proceedings, 2nd International Conference on Analogy. 2009.

[3] Havasi, Catherine, Robert Speer, and Jason Alonso. "ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge." *Recent Advances in Natural Language Processing*. 2007.

[4] Krishnamurthy, Jayant, and Henry Lieberman. "CrossBridge: Finding Analogies Using Dimensionality Reduction." 2010 AAAI Fall Symposium Series. 2010.

[5] Lenat, Douglas B., and Ramanathan V. Guha. *Building large knowledge-based systems; representation and inference in the Cyc Project.* Addison-Wesley Longman Publishing Co., Inc., 1989.

[6] Liu, Hugo, and Push Singh. "ConceptNet—a practical commonsense reasoning tool-kit." BT technology journal 22.4 (2004): 211-226.

[7] Schank, Roger C. *Tell me a story: Narrative and intelligence.* TriQuarterly Books, 1995.

[8] Schank, Roger C. "Tell me a story: A new look at real and artificial memory." (1991).

[9] Singh, Push, et al. "Open Mind Common Sense: Knowledge acquisition from the general public." On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE. Springer Berlin Heidelberg, 2002. 1223-1237.

[10] Speer, Robert, Catherine Havasi, and Henry Lieberman. "AnalogySpace: Reducing the dimensionality of common sense knowledge." Proceedings of AAAI. 2008.

[11] Speer, Robert H., et al. "Finding your way in a multi-dimensional semantic space with luminoso." Proceedings of the 15th international conference on Intelligent user interfaces. ACM, 2010.

[12] Tomai, Emmett, and Ken Forbus. "Using narrative functions as a heuristic for relevance in story understanding." *Proceedings of the Intelligent Narrative Technologies III Workshop.* ACM, 2010.