6-2016

Event Parsing In Narrative: Trials And Tribulations Of Archaic English Fairy Tales

Rebecca Lovering
Graduate Center, City University of New York

EVENT PARSING IN NARRATIVE:

TRIALS AND TRIBULATIONS OF ARCHAIC ENGLISH FAIRY TALES

by

REBECCA LOVERING

A master's thesis submitted to the Graduate Faculty in Linguistics in partial fulfillment

of the requirements for the degree of Master of Arts,

the City University of New York

2016

Event Parsing In Narrative:

Trials And Tribulations Of Archaic English Fairy Tales

by

Rebecca Lovering

This manuscript has been read and accepted by the Graduate Faculty in satisfaction of the requirement for the degree of Master of Arts.

Approved:

_____ _____

William Sakas Date

Thesis Advisor

_____ _____

Gita Martohardjono Date

Executive Officer of Linguistics Department

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

Event Parsing In Narrative:

Trials And Tribulations Of Archaic English Fairy Tales

by

Rebecca Lovering

Advisor: William Sakas

While event extraction and automatic summarization have taken great strides in the realm of news stories, fictional narratives like fairy tales have not been so fortunate. A number of challenges arise from the literary elements present in fairy tales that are not found in more straightforward corpora of natural language, such as archaic expressions and sentence structures. To aid in summarization of fictional texts, I created an event class, a Python script, that captures elements predicted to help classify events as important for inclusion. I wrote a processor to run over a new corpus of fairy tales and parse them into events, then to be manually annotated for importance and clustered to create a classifier that could be used on novel texts. These two latter steps could not be completed before publication of this paper, but my proposed approach is covered extensively, as is future work along the same lines. Along with this paper, supplementary materials encompassing the event class, processor code, corpus in full text, and corpus as unannotated event data are included.

Acknowledgments

This work and paper would not have been possible without the guidance and support of the following people:

Andrew Rosenberg
Rachel Rakov
Steven Butler
Max Schwartz
Pablo Gonzalez Martinez
Max Lobdell
Sarah & Tim Johnson

I am forever in your debt.

Table of Contents

Table of Tables

1 Introduction

For my project, I chose to pursue event extraction and automatic summarization, particularly in the realm of fairy tales. Computers have long been excellent at giving the surface metrics of natural language: how many words, how many letters, how many nouns, etc. Getting any deeper analysis of text, even a predictable structural one reliant on word order, like a syntactic tree, is non-trivial, though it has been well done.[1] Getting a computer to tell us what a text means, or even just what happened over the course of a story, is an enduring challenge, especially in realms beyond news.

News stories benefit from a predictable structure and narrower range of novel vocabulary. These structural qualities, combined with the acute interest in scanning and summarizing news content have led to great strides in the field, with beautifully human-like encapsulations of texts. Top of the line systems (e.g., Chen et al. 2014) can find events and disambiguate such tricky words as "fire" in both its artillery and employment senses, constructing relationships between entities and predicates with tremendous success. Narrative in fiction often has sentences that contribute atmospheric, rather than effective, content. Stories, particularly fantasy stories, suffer (from an analytical perspective) from a lot of relative unpredictability, up to and including violation of selectional constraints (pasta pots do a lot of unexpected things for inanimate objects). Subjects and their verbs are wildly different, not just from story to story, but from these stories and other kinds of writing. Accordingly, although some truly marvelous work has been done on news stories in terms of event extraction and automatic summarization, it is not readily transferrable to the realm of fantasy narratives, like fairy tales.

Pursuing computational analysis of natural language requires us to choose a school of thought and approach for the relationship between the two. Linguists traditionally favor a worldview in which human language is special: uniquely structured to compress data actionably and informatively for optimal transmission between speakers. That remarkable abstraction and

[1] It bears saying that all these assertions about accomplishments in computational linguistics refer to English, although some other languages are quite high resource and have shown great strides as well.

compression remains an unsolved mystery, although we have some satisfying models that seem to account for many phenomena, even cross-linguistically. Linguistics, after all, is still a very active field, with experimental and theoretical advances made every year.

To be clear, no linguists claim that the accepted canon of linguistic knowledge reflects, one to one, what human brains are actually *doing* when they do language things, just that we can model many of the rules we've observed in action. So we are already at a remove from the causes and effects of language in a way scientists who study, say, physics, are not. We are already working with abstraction. When we talk about doing computational linguistics, we are talking about modeling a model - trying to get machines to do something we don't even fully understand how we do, ourselves.

Another approach is purely statistical, without relying on the painstaking heuristics gleaned from human study of humans, particularly in the realm of machine translation, where this approach has shown terrific results. All it requires is annotated equivalence, with none of the human conceptual mapping visited on it, just word-to-word and even character-to-character correlations (Bahdanau et al. 2015). With enough data to chunk through, translation at least approaches the top of the line phrase-level translation. This may mean that the human-centric ways of organizing information are not necessary to perform natural language tasks. Why model what a human does by forcing the computer to go through a laborious, complex process to parse texts as humans believe we parse their meaning ourselves, when a simple neural net with enough data can perform equally well?

At present, however, these approaches are computationally taxing, taking many days of processing time to learn and a few even to apply the learning. While advances in hardware may eventually make this viable, at least for now, re-parceling information in as close to human-like manner as possible is still a reasonable, and in some respects, higher-performing way of doing business.

I have chosen to organize this paper's broad strokes in cycles, not unlike the way the experiment itself was run. First are the predictions that motivated the angle of approach to the

project, including what elements I thought might be indicative of storyline importance. Next, I cover the parts of the original proposal I was able to complete, then the parts that remain to be done. With the full scope of the project (as intended) laid out, I go into a discussion of some of the specific problems that precluded completion before moving into ideas for future work. The first part of the first section, on the class I created to capture events, is oriented around the features I elected to include in my event processor, some traditionally associated with semantic events, and some I added to capture story-wide phenomena. The sections on project parts that were not completed is naturally shorter, as I do not have the detailed in-the-trenches knowledge about how they would turn out that I have for the first sections.

In addition to this text, a number of supplementary files are included in the digital profile of this project. The event class itself, as articulated in primEvent.py, the processor, simple_event_tagger.py, the README file (.txt), the folder of original text files, and the event data in both spreadsheet and tab-delimited text form, are all in the supplementary material upload. These files are also available on github via https://github.com/lovering810/StoryTime/.

2 Predictions

2.1 Character Importance

To determine character importance and its influence on event importance, I focused first and foremost on a relatively simple metric: frequency of appearance by name. In fact, "importance" may be a misnomer, as that is not a directly measurable quality, and the metric I actually relied on is the closest I got. I predict that character frequency will end up being a telling feature, but I remain uncertain exactly how. It is my hope that a more accurate representation of importance, or sufficient knowledge of its symptoms enough to create a reliable hidden Markov model, is in the future.

Main characters' actions might more frequently be considered germane to the story, so a high percentage of mentions would predispose the system to consider an event of likely importance, and infrequently-mentioned characters might correlate with less important events. It could be that in a story such as Aladdin, events in which Aladdin appears may prove more important to the story, so the system will pick them up. More tangential characters may not contribute that much to the essential events, acting more as set dressing. If so, those with lower mention scores would be indicators of unimportance for the events in which they appear.

Alternatively, characters who appear only briefly might have a very special role to perform in the story, and those events might be so often important that low-frequency mentions might be more closely linked to important story events. Maybe a merchant appears only once, but it is from him that a character buys the mysterious ring, a core event in the story, or a fairy who only gets two sentences spends them putting a curse on everyone. (As happens an alarming amount in fairy tales.) The very fact that a character appears in almost every event of a story may mean that more of the set dressing is forced to involve them, so their popularity is significantly less helpful in determining whether or not an event is important to the story.

A third option is a combination of these. Perhaps there is a middle zone in which the character frequency is not indicative of the event's importance, but a character importance on either extreme is telling, for the reasons proposed above: a frequently-appearing character

means an event is important to the story by virtue of his character's importance, while an event with a one-off fairy is important for exactly the opposite reason.

It also occurred to me that the frequency with which a character appeared in a certain event role, say, agent, might be informative, distinct from the total number of times one is mentioned. Accordingly, I tracked all characters by name, as well as how frequently each appeared as a subject, object, or indirect object. Sentences describing a character will often have him as the subject, "Aladdin was tall," e.g. which is not an essential event for the timeline of the story. Alternatively, Aladdin may strongly indicate event importance when he appears as the indirect object or object of something happening in the story.

To get these figures, the processor builds several dictionaries per story. Every time a sentence returns a subject, object, or indirect object, a count is added to the dictionary for that role with the character as the key. There is also a master dictionary in which all counts of characters are kept. Once all sentences have been processed, the count dictionaries are put through a small function that turns them into dictionaries of proportion. When the list of events is finalized, these proportions are added to the record for each subject, object, and direct object that is represented there.

2.2 Story Arc

Another metric I foresee having predictive power is how far through a story an event happens. Most stories have an arc: events build to a crescendo and then resolve. I believe that the proportion of important events to a timeline or summary is likely to be higher toward the end of the story. Descriptive, backgrounding events are likely to be in the beginning of a story and many of them, while good for making characters more relatable for a human reader, are not necessary for discriminating between storylines and representing the flow of events. While they might, in the annotation scheme I used for this project, merit a 1, they would not be the essential events that rate a 2. The exact turning point in terms of percentage through the story is not something for which I have a precise prediction, any further than to say I believe it will be generally past the

halfway point. This metric, expanded and tried on other corpora, particularly in other genres, may provide valuable insight for analyzing literature with the goal of matching readers with preferred material. While many prefer a more tightly-plotted story, some relish a slowly-unfolding meander through the world of the author. If the identified peak of a story should come early, winding down very slowly, the second camp would be likelier targets for its consumption. Identifying the narrative apex is a ways in the future, but there are ways we might approach it, to which I hope this research contributes.

3 Intent[2]: Realized Components

The proposal on which I acted for this project included three major components: Event class creation, processing and tagging of data, and machine learning what constitutes an important event from the results. Unfortunately, I was not able to complete all parts of the proposal by the day it came time to start writing, so much of this paper will be forward-looking, at future steps to take in this same pursuit.

3.1 Event Class

The first thing I did was to create an event class, with the eventual goal of employing event-based semantics for narrative analysis. The actual class conveys both more and less than a semantic event, because for it to have predictive power, it needs to be contextualized in an orderly way, not a bag-of-events model. The class contains features that map the event onto the context of its parent story as well as surface grammatical features that inform the derivation of underlying semantic roles. These features are not placeholders, to be cast aside when the improvements I discuss come online, but I would like to augment them with deeper semantic relationship annotations, to better approach the theory.

What follows is a brief illustration of event-based semantics, modeled on the concise description found in Lasersohn's encyclopedia entry on the subject. I have paraphrased and provided more topically relevant examples, given that I worked with fairy tales.

---

[2] The very first version of this project was assessed as wildly ambitious, requiring something like four to six years to complete, as well as the help of several people more educated than I. I appreciate, now especially, the good advice I received on trimming down my plan. The original involved approximating work by Chen, et al. on event extraction, which they have performed with remarkable success on news corpora, and following that, structural analysis of the type practiced by the Genesis group. Given that those are much higher resourced groups with a wealth of experience among them, it was probably a little more than foolhardy to think I could recreate their work myself - even with their generosity, should they have decided to share any of it with me - and build on it in the time given for a Master's thesis. The pie-in-the-sky elements have been moved to section 6.

Event-based semantics are a way of adding to the descriptive power of predicates by mapping them over a new argument: the event. It allows us to quantify over events, usually existentially, as in:

a. Prince Philip slew the dragon.
b. ∃*e* slay(Prince Philip, the dragon, *e*)
c. slay(Prince Philip, the dragon)

This structure lets us analyze more information, like where or when an event took place, or by means of what instrument. This way, we have a nice organization of all the information given by the sentence below:

a. Prince Philip slew the dragon with his sword in the castle at midnight.
b. ∃*e*[slay(Prince Philip, the dragon, *e*) & with(*e,* sword) & in(*e,* castle) &
at(*e*,midnight)

This uncouples the burden of additional adjuncts from the verb itself (which could lead to undesirable technical ambiguities between verbs modified differently, although no such ambiguity exists in grammatical judgment), putting them with the event object instead. Events are useful objects to classify natural language happenings, especially in a project such as mine, where the goal is toward good story summarization, which I envision as a string of events on a timeline. As such, my first effort was toward an event class into which to section the data into objects for later manipulation.

The class I ended up using is defined in primEvent.py. This is not what I envision the most comprehensive such item to be. In fact, I abandoned a more complex version of the class for expedience's sake, while still leaving twenty-six features (several of which are functionally identical, but distributed across subject, object, and indirect object). Some of them I already have ideas on improving, which I will discuss, and others may eventually need the same treatment, but this is a start.

The first argument in the class is the proportion through the story where the event takes place. Initially this was just an integer count of what sentence the event was made from, but a ratio is more robust across stories. As articulated in section 2, it occurred to me that later sentences are more likely to involve vital events. At that point in a story, there is not a lot of reason to have set-up or extraneous description that might be found in earlier sentences. Generally, the end is where the climax is, which usually requires the tying up of loose threads in short order, again without frills. That is, however, a prediction I make as a human reader. What would the data show?

Next in the alphabet of arguments comes the root. This term was borrowed from the first returned item in the list from the dependency parse and is supposed to correspond to the central verb of the sentence, as the dependency parser identifies it. It can, however pick out other words, like "indeed," which presents problems of its own, covered more extensively below. It is structured as a list because the processor picks out other words dependent on the root that are connected to it with a "conj:and" relationship, and adds them. Originally, it was just a string meant to hold one item, but I realized that was losing a lot of information, as nearly every sentence contained more than one verb (as is common in natural language). Initially, only the first verb was picked out and could conclusively be considered important. Once that was confirmed, I found other returns from the parser that could be connected, like "conj:and," a label that proved helpful in populating several arguments of the class. Although this captured more of the detail, it ended up being problematic. Both a description of the problem and how to address it more accurately can be found in later sections.

After root comes comp, the clausal complement of the root. For all the sentences in which a communication verb is identified as the root, this captures the content of the root better than its own label might. This might be quite useful in a story in which the communication of certain facts might prove essential to moving the plot forward, because more of sentences such as "He told me that you are the chosen one," are captured with ccomp.

The subject is the next field in the class, and it is formatted as a list. Originally, it too was a string, but sentences with multiple named subjects just those who's subjects are connected with “and” were not getting documented as they needed to be, so I expanded to the list. For each sentence, the processor adds any word labeled as “nsubj” and adds it to this list.

The person designation for the subject comes next (and works the same way for objects and indirect objects). This cannot be populated by the labeled output of the parser, but must be deduced by a short function in the processor. The default value is 3, for third person, because all the texts are stories told from the third person omniscient perspective. If a first- or second-person pronoun appears, the value changes to reflect that number instead. I am not expecting particularly revealing information from this feature, but it was an easy base to cover in case I am wrong.

Number, which comes next, is trickier because it is not as accurate. Because there are no list of entity characteristics in the current system, the function that assigns number is only run on pronouns and common nouns found in the subject, object, or indirect object position. With coreference resolution and NER data available, this feature will be significantly more accurate, though whether it will have bearing on the end-goal, summarization, is not immediately obvious to me.

The next set of features are the importance of a character in a particular role and his or her importance over all. Originally, the processor only tracked how frequently a character appeared in a particular event role: as subject, object, or indirect object, but the current version also tracks their frequency of appearance in the whole story, regardless of role. Upon the first pass, the processor keeps count of the characters’ appearances in the various roles, and after all events from a story have been preliminarily extracted, translates those counts into percentages of dictionaries by character, which are consulted to assign what I have termed an “importance” score to each event in which the character appears.

Preposition is included as a feature because directionality might play a role in determining plot points’ importance. Two of the seven plot types of Jungian philosophy involve travel:

Quest and *Voyage and Return*, so tracking travel-related prepositions may well help identify events worthy of inclusion in a summary, and those are only a subset of the information captured by this feature.

Passivity and negation are both features as well. Passivity was intended as a precursor to later processing that would correctly assign event roles (agent/patient) rather than grammatical roles (subject/object) to events, while still retaining the stylistic information that the author put in the passive - prioritizing the patient by making it the subject of the sentence, a better indicator of importance than its event role. Negation feeds into emotional content and I hope that later iterations of this class and processor will be able to do more with it as an indicator of Aktionsart and character state-of-mind than the current version can. At present, both features are Boolean, triggered by dependency parser labels in whole ("neg") or in part ("nsubjpass").

Time and manner are optional fields for words returned with a "tmod" or "advmod" label. Another feature, emotional content (shorthanded as "vibe"), is partly calculated from manner with method calcVibe. To give vibe a value, calcVibe lemmatizes the root, treating it as a verb, the manner as an adverb, and the direct object as a noun, then queries the en package for emotional content in each. If it finds that one is an emotional word, it assigns a hyponym for that emotion word to the event. Otherwise, it assigns it a vibe of "neutral." If the event is negated and non-neutral, it assigns a vibe of the antonym of the vibe so far calculated. While I am primarily expecting that emotional events of any stripe will be more frequently important, I included the hyponyms to see if there were trends in type of emotion, too.

Story importance and motive for inclusion in summary are two fields left blank by the processor, holding places for later analysis.

One more method is included in the event class: checkNumbers. This is a back-up check for plural subjects, objects, and indirect objects, in case the author should happen to have made them multiple named and unnamed entities. It catches lists of characters in these fields and if there are more than one, it assigns plural to the appropriate field.

3.2 Corpus

I chose fairy tales because they are short, non-news stories in the public domain. Specifically, I used the classic color fairy books: Blue (1889), Red (1890), Yellow (1894), Violet (1901), and Crimson (1903). Altogether, 198 stories and - for now - 25835 events comprise the corpus.[3]

This choice of corpus presented unique problems, mostly having to do with the age of the content. The oldest of the books I drew on was published in 1889, and the most recently published came out in 1903. This is not, of itself, a problem, but the writing style of the time involved extremely long sentences. Additionally, a number of stories in each volume were translated from German, Japanese, French, or Hungarian. German tends, even now, toward quite long sentences, and the translations follow suit.[4] A far less common, but still strange, problem is that archaically, exclamation and question marks can be deployed mid-sentence, which led to some oddly abbreviated sentences and fragments that produced problems.

By long, I mean not only by number of words (recursion in natural language means that a relatively simple sentence might have a lot of words without being clausally complex) that a given sentence incorporates several clauses, often ones that could be stand-alone sentences themselves. Because events are currently coupled with sentences, this involved a lot of informational loss in the first iteration of the processor and class, in which many fields accepted only one candidate. Expanding those fields to allow lists retained more of the seed sentence, but conflated events within the same sentence.

3.3 Process

---

[3] I say "for now" because at present, events are constructed directly from sentences. While I believe this would not be as problematic and loss-ridden on another corpus, this one needs a higher ratio of events to sentences to yield accurate or meaningful data. In the big picture, the same way that semantic event roles are uncoupled from grammatical roles, events should be completely uncoupled from sentences. The final version of this processor should and will be driven by events, rather than sentences.

[4] A few stories originally included in the books were not used because they were *not* translated from Scottish English, which the parsing components of the processor are not equipped to handle.

Dependency parsing is vital to populating event objects, because it returns grammatical relations like the subject and object, which, with other information the processor captures, can be used to assign event roles like agent and patient. The best tool for the job I found was the Stanford CoreNLP dependency parser, and the best way to use it was to start a server on which Java was always running. (Originally, I used an approach that initiated a Java process for each sentence, which took far too much time to be reasonable for the 198 stories of a few hundred sentences each.) So the first step in the whole process (after importing the relevant tools) is to start up the Java server, ready to be called upon.

With all texts assembled into one folder, the program goes through works on each file one at a time. First, I split each story into its component sentences. Then, each sentence is run through the Stanford CoreNLP dependency parser. This returns a list of tuples that include the label it received from the dependency parser, the word to which it is related in the sentence and the index of that word in the sentence, and the original word and its index. For example, if the word is "Aladdin," from the sentence "Aladdin ran to the cave," the entry in the list for "Aladdin" would look like this:

('nsubj', 'ran', 1, 'Aladdin', 0)[5]

This information is repackaged and passed to the checkRole function, which populates some fields in the event based purely on one-to-one relationships to labels, and others by referring label information to other functions that determine person and number, passivity, and importance (see Section 3.1 for details). Throughout this run, count dictionaries are being assembled of characters appearing in the events. Once all sentences for a story have been run, the count dictionaries are converted to proportion dictionaries. Every event in the timeline is then re-run to assign the proportional importance scores to the actors in events.

---

[5] The only exception to this format is what the dependency parser identifies as the root of the sentence, because it will have a number of words that point to it as their related words, so it points only to itself: ('root', 'ROOT', 1, 'ran', 1).

All together, the full corpus was processed in 22.77 hours. Runtime could be reduced by changing the data storage format with pickle, perhaps, or another, faster-written type.

4 Intent: Unrealized Elements

4.1 Annotation

Upon completion of all data processing, the next step would be to manually fill in the fields for importance to story and reason for importance to story (the last two fields listed for an event object).

Per good advice, the importance was simply indicated by a 0, 1, or 2, offering a little nuance which, as the annotator, I found very helpful. While annotating, I was assessing importance to the story by the rough metric of whether or not an event would need to be included in a summary of that story to make it recognizable as itself, well-formed as a fairy tale, and distinguishable from other stories. These are relatively subjective concepts, and some events do wobble on the edge of vitality to the story's identifying arc, so the three-way distinction between unnecessary to summary (0), marginally or potentially important to summary (1), and vital (2) was more expressive and useful than a simpler binary categorization.

Initially, the encoding for why an event was important arose as a way for me to check myself as annotator. When I reviewed what I had labeled from a previous session, I wanted to have some idea of why my past self had assigned importance to an event. However, I believe there is greater potential for its use as a feature. It would be a good first step to enable automating assembly of the narrative event maps by David K. Elson, which are linked by motivation and causation, among other factors. I am also interested to see what patterns arise from assigning these qualities to events. That is, what kinds of events are most frequently important? Those that motivate a character to later do something? Those that indicate what kind of person a character is? Those that introduce a character or remove her from the story?[6]

The codes I settled on for why an event received an importance value of 1 or 2 are as follows:

[6] Of course, the very fact of my having chosen the encoding will be subject to my own biases. It is my hope that with broader collaboration and more annotators, a better-balanced corpus can be created and worked from.

Reason	Example	Letter Code	Number
Backgrounding	This story happened in a faraway land called Arabia, many years ago.	B	1
Establishing	There lived a poor tailor with a son named Aladdin.	E	2
Storyline	He rubbed the ring and wished.	S	3
Motivation	The magician jealously desired the ring.	M	4
Resolution	His treachery discovered, the magician was put to death.	R	5
Character Development/ Description	Aladdin was an idle boy.	C	6
Not important (default)	The magician fell upon his neck and kissed him.	N	0

TABLE 1: CODES FOR IMPORTANCE LABELS

Character development and motivation have some potential overlap, but with improved discretion for events (see Section 5), I think they will appear more distinct and serve different functions. It should be noted that this is a limited set of possibilities because I recognized problems in the quality of the data and stopped annotating before getting very far into the process. While I have confidence in these categories, it is possible the set will expand to allow for more nuance or cover unforeseen configurations.

4.2 Machine Learning

Once annotation is complete, I intend to train a classifier for event importance. Specifically, I plan to use DBSCAN clustering as enacted in scikitlearn as my initial candidate. It allows for clusters of any shape, a flexibility to be appreciated with a dataset whose qualities are not yet well-known, but which is expected to have a difference of scale between unimportant and important events (the latter being considerably smaller).

The goal of the classifier is to create a summarizer by importance, which returns all the events above some threshold of importance, in chronological order, as a very blunt summarization instrument. At the moment, it is possible to ask the set of events to return all events in which one character is the named subject, object, or indirect object (or all three), but the real

judgment task has not been accomplished. This goal and a further, more involved summarization task are described more in section 6.

While I would like to avoid the pitfalls of the curse of high dimensionality, I want at least the option of incorporating any of the twenty-six feature dimensions captured by the event class. This will require the vectorization of some presently nominal features of the event objects, some of which will be easier than others. Closed classes, like prepositions, can be relatively easily vectorized without loss of information. I would like to explore ways to use word2vec or GloVe and dimensionality reduction to see if that is a reasonable way to handle more of the nuance of the words used in a story in some of the nominal categories. Additional qualities not yet part of the annotation or event class are covered in section 6.

5 Problems and Proposed Solutions

5.1 Incomplete Representation

One of the first problems, outlined above, was with the data structure of a few fields. Initially, the root, subject, object, and indirect object were all single-string containers. Upon reading the preliminary results of the processor next to the original texts, I noticed a number of sentences whose event objects didn't include enough of the original information. For example, take the final sentence of "The Witch" from the Yellow Fairy Book:

> Then they told their father all that they had suffered, and he was so angry with their stepmother that he drove her out of the house, and never let her return; but he and the children lived happily together; and he took care of them himself, and never let a stranger come near them.

Originally, the parser captured only "they told" as the event. As this sentence fundamentally resolves the story, it has narrative significance which I would have coded as very important, but the telling is not the big deal. My solution, expanding the fields to lists and having the parser populate root, subject, and object with candidates that are connected to an accepted item in any of those with "conj:and," succeeded in capturing more per event object, but led to the next problem, too-greedy representation.

5.2 Overzealous Representation

Now I had the problem of capturing a lot of the information found in the sentence, but not successfully separating it into discrete semantic events. The same sentence with the revised list-based storage finds more roots: "told," "all," "angry," "lived," "took," and "let," and "suffered," "drove," "return," and "come," as ccomp members, but parses "they" as the subject for all of them, and "father" as the object.

It is probably possible to compose a suitable set of complex if-statements that encompass every one of the dependency parsing labels and their order by word index to guide the event parser to make better division of all the data, but that would be something of a wheel reinvention. Instead, per helpful suggestion, I tried to set up a shallow parsing pre-processing step. The

goal of such an implementation would be to get separate events out of each clause, so that the father's actions in response to his children's telling would be separate from the telling event. This would occasion an overhaul to the event parser itself to retain connections between events and sentences, but the accuracy added would be entirely worth addressing the ripples.

Sadly, after looking at the options for shallow parsers and settling on MBSP as the one with the highest reported accuracy that I had the skills to implement, I encountered maddening installation problems that prevented me from getting it done in timely enough a manner to report it here.[7] Together with the solution I propose to the problem of inaccurate character importance (below), I believe this will greatly improve the semantic validity and accuracy of the event parser.

5.3 Inaccurate Character Importance Scoring

This is not quite as problematic as the over- and under-zealous capture of verbs and subjects, but it would make a significant difference to the success of the project to solve, and the solution exists, if is not readily available. The way the dictionaries of characters work is to create keys from proper and regular nouns. I suppose I could have added pronouns for extremely broad strokes of information about gender and number in a story, but that runs the risk of muddying the waters, and number is captured elsewhere. The trouble with this approach is that all sentences with pronouns lose some information. What is captured is an accurate portrayal of how frequently people and things appear in their event roles *by name*, but not how frequently they participate in events total. While I believe it is an accurate portrayal in terms of rank of frequency, it is no more specific than that, which seems a shame.

[7] As able as I feel within the scope of my programming, I am still in the process of learning about system architecture and how my machine builds and compiles code. The problems I encountered were in the build process, and though I read through the installation file and feel I understand it pretty well, I do not know how to resolve the problem my computer encountered while running it without compromising the integrity of the parts of it that work. I am hopeful that with more time and some help from someone with the knowledge I lack, we can get this up and running.

The solution I propose is a pre-processing step: coreference resolution. Although this is still very much under construction, there are some implementations that do decently enough. In fact, the documentation on the Stanford Parser and corenlp.py led me to believe it would perform that step in the process of its parsing along with the dependency parse, but it has never done so in my implementation. The level to which I can critically examine the inside of the Stanford Parser is extremely limited, and though I scoured the corenlp.py wrapper to see if the problem lay there, I did not find it. Accordingly, I went on the hunt for alternative coreference resolvers. It is a long story better suited to a social gathering than this paper (it has a tragicomic ending), but the end result was the same as it was for shallow parsing: I could not enact it in time. While I feel that this program of study has admirably prepared me for addressing questions of natural language processing in Python, I find myself very lacking in other frequently-used languages (like Java) and some of the actionable knowledge of computer architecture.

A related problem is the lack of representation of adjectives. While in other genres, this does not probably pose such difficulty, in fairy tales, it is possible to have a character whose most proper reference is a full noun phrase. The story of the Three Billy Goats Gruff, for example, identifies its characters all as goats, differentiated by size. Right now, there would be no way to tell which goat is doing which action, because modifiers are not preserved in event objects. Differentiations based on size and color are relatively common within fairy tales, in fact. Goldilocks comes to mind, as do several stories about the youngest of a set of princesses or sons. Expanding the fields for character identification to allow for whole noun phrases and adding phrasal parsing to the way sentences are broken down are fairly vital steps for further work with this genre, as well as adding a useful flexibility to other genres. This step could be considered further work, but is dwarfed by the scope of what I suggest in section 6, so has been placed here.

5.4 Dependency Parser Problems

Another problem, whose core is not available to me to address in the same way as others, is that of what constitutes “root” words, as determined by the Stanford parser. Of the events generated

from the corpus, an average of 28.6% across the books have no subjects, at least not by the labels enacted. Some of this may be attributed to the presence of dummy subjects like "there,"[8] but some of it is mis-parsing of the root and its attendant relationships. For example, the sentence "At last he clasped his hands in prayer, and in so doing rubbed the ring, which the magician had forgotten to take from him," returns "At" as the root. Naturally, this occasions no subject relationship, and the rest of the event generation process is defective.

	Blue	Violet	Crimson	Red	Yellow	Average
Empty subject	1430	1459	1248	1755	1511	1480.6
Total events	5160	5089	4447	5920	5219	5167
Ratio	0.27713	0.28670	0.28064	0.29645	0.28952	0.28609

TABLE 2: EMPTY-SUBJECT EVENTS BY BOOK

Of those events with no subject, further analysis revealed that 43 events had no root at all, a distinct surprise, though always a remote possibility. The root label is defined by Stanford's manual as "pointing to the root of the sentence," and a fake one is used as a governor, around which all other relationships are oriented. Because of its potential as a placeholder, as I understand it, if the parser never finds a good candidate, it returns empty. It did not occur to me that in the corpus I selected, such a thing might happen, as it is all standard - if archaic - English, known to have generally well-formed sentences. At the time of writing, the 43 sentences remain in hiding within the corpus, not identified for in-depth investigation of how they produced these results. Time permitting in future, I will find and examine them.

Of those events with a root but no subject (6232 in total), 3541 had nonverbal roots, with a total of 4324 nonverbal members. The make-up of the parts of speech contributing to this total are below:

[8] Although with an abbreviated sample sentence, "There was an old tailor," the parser correctly returns "tailor" as the subject, and "was" as the root, so dummy "there" does not exceed the parser's capacity in all cases. Further experimentation, with longer and more elaborate sentences as the samples, should shed some light on when this creates problems.

Tag	Part of Speech	Example	Count
NN	Common Noun	“dog"	2422
IN	Preposition	“in"	790
CC	Coordinating Conjunction	“and"	259
RB	Adverb (“swiftly”)	“swiftly"	258
NNS	Plural Common Noun	“dogs"	226
JJ	Adjective	“idle"	159
DT	Determiner	“the"	60
PRP	Pronoun	“he"	41
WP	WH-pronoun	“what”	30
PRP$	Possessive Pronoun	“their"	28
CD	Cardinal Numeral	“1"	14
WRB	WH-adverb	“however”	12
JJR	Comparative Adjective	“redder"	10
TO	to, in any role	“to"	6
EX	Existential There	“there"	6
MD	Modal Auxiliary	“could"	3

TABLE 3: BREAKDOWN OF NON-VERBAL ROOTS FOR EVENTS WITHOUT SUBJECTS

Because of the way the dependency parser deals with copular declarations of being, nouns are plausible as roots. “Aladdin is an idle boy,” would return (ROOT, boy). Accordingly, the actual number of roots that are nominal is not problematic on its own. Unfortunately, the subset from which these figures are taken are those in which no subject was found. Even for copular declarations of being, there is a subject. The example sentence about Aladdin returns “Aladdin” as the subject of the root “boy,” as one might expect. While these are a minority of cases, their presence is frustrating, and I need to understand their origin better to address them meaningfully.

In looking for the source of this problem, I added a field to all events that captures the original text in full and re-ran the texts. It is possible such a field can later be removed, but it is a lot simpler to check that rather than try to backtrack through the data to find the source of the problem. Unfortunately, there are some problems reading in the out-written data, so my program for error analysis cannot yet operate on these more completely rendered events. Although it cannot be reported here, I hope to get an answer soon. With a better understanding of where these come from and what causes the parser to mis-identify their roots (or correctly identify them, but there is something wrong in the way the sentences are constructed or split from the text that causes the problem), either an additional preprocessing step to screen out problematic members or a specialized function to address the biggest causes might be implemented.

6 Future Work

This section is divided in two parts: improvements I would make within the scope of this project so far, and further expansions to what is done here to a higher level of narrative analysis.

6.1 Improvements

The primary improvement to be made is to fully decouple events from sentences. This will require several changes and additions to realize, most importantly the pre-processing steps of shallow syntactic parsing (see section 5.2) and coreference resolution (5.1). Especially with that decoupling in place, there will be a need to relate events to one another, or at least capture what role between events is at work, to detect story-wide patterns. Several new fields and metrics will need to be implemented.

6.1.1 Reduplication in Reconstruction

Not only should the pronouns be re-rendered as their original referents in pre-processing, but those verbs that are riding the coattails of their neighbors for subjects should get their implied subject overtly represented. That is, to get a correct read on the number of events, a sentence like “Aladdin x’ed and y’ed,” should become two events:

$$\exists e[\mathrm{x}(\mathrm{Aladdin}, e)]$$
$$\exists e[\mathrm{y}(\mathrm{Aladdin}, e)]$$

Some of this can be done with exactly what is here presently, elaborated in my own code, backtracking from a second verb to the subject of the first. This and other, more complex ways of showing the same character’s participation in multiple events will require multiple runs through the data, each time parsing the material to closer closer approach the right level of detail for a semantically accurate event, with the additional story-wide trends captured as well.

6.1.2 Automatic Semantic Role Labeling (ASRL)

A truer semantic event representation requires more nuance than the event class currently possesses. While it captures information that can easily yield the actor/agent of an event, that is not the full picture of possibility within event-based semantics, and that does not even begin to cover the difficulty of representing other roles well. The same textual clues (namely, being the subject of an active verb) that lead us to position a character as the agent of an event should, in some cases, have us putting him in the role of the experiencer, or even force. Similarly, there are subtle differences between the roles patient, theme, and stimulus, though all arrive in the same grammatical guise (usually the object of a verb). Which roles should be used is a question of the verb at work.

FrameNet is an extant resource for determining such roles at a highly granular level, getting as specific as "heat source" for frames around certain verbs for cooking. That is, to my mind, a bit more granular that what I was looking to capture, but it would be fascinating to see what arose from a FrameNet analysis of event verbs and their sentential fellows. SEMAFOR by Das et al. (2014) is the automatic semantic role labeler (ASRL) powered by FrameNet that I would choose to integrate into the event processor as it exists currently, partly for the availability of a Python wrapper (like CoreNLP, it is written in Java). An alternative to FrameNet is VerbNet, which maintains thematic roles more at the level of abstraction I favor. FrameNet gets as granular as identifying a heat source role for certain verbs of cooking, which, while impressively detailed, exceeds the level of nit and grit I envision for this particular project. Some experimentation is needed to see if this step would benefit most from implementation after or before shallow syntactic parsing, but at its most useful, I propose to store its output as a dictionary with role keys and values of the words that fulfill them relative to the root.

At this point, it would not be surprising to wonder why anyone should bother with the surface, syntactical dependencies, and much else of the event class when we could jump straight to quite decent automatic semantic parsing with SEMAFOR and/or VerbNet. Foremost among the reasons I have is that of registering narrative importance. Authors make choices about how to express events for a reason, and interpreting that reason is helpful to guide the determination

of what is and is not important in a story. While we can get the essential events with only semantic roles and accurately create a timeline, we cannot capture their relative importance as clearly. This requires further revision to the method by which character importance is captured. I suggest adding a layer of weighting to mentions in the text, taking into account when focus is brought to them by use of the passive voice and other stylistic deviations from what can be called the standard. Furthermore, while semantic role labeling is tremendously helpful in populating something that is closer to a true semantic event than I have gotten so far, for patterns across events but within stories, the system must still tie the events together, as it does by including proportional distance through the story along with character frequency/importance. Those, coupled with the surface cues that can tell us about style and narrative choices of the author, I believe to be worth retaining in future iterations.[9]

6.1.3 TimeML Augmentation of Time Marking

TimeML (Pustejovsky et al. 2002) is a mark-up language for temporal relationships in text. While the current parser identifies and captures overt temporal markings, it is constrained to the order in which the information is presented. In the case of this corpus, the stories are very straightforward, and the timeline of actual events parallels the timeline of mentions. However, in order to expand the viability of this parser, more complex relationships with time need to be able to be represented.

TimeBank 1.2, the corpus for training automatic time taggers, is, along with many excellent resources, composed of many news reports. While there is some overlap in the vocabulary, both the preponderance of multi-word events and the presence of completely novel words make it an imperfect fit right now. It is possible that some bootstrapping between the lexicons could allow fairytale vocabulary to ride the coattails of terms already annotated by hand, though the exact process is still unclear to me.

---

[9] While it may not seem extremely relevant to this particular corpus to track stylometric features, I hope and believe that these same tools may prove useful to someone in another area of narrative. For that to be true, I am taking metrics seriously that may affect that rather than this corpus.

6.1.4 Aktionsarten (Lexical Aspect/Vendlerian Categories)

One of the axes on which verbs or events differ from one another is their relationship to time. Aktionsarten are the labels we give verbs to distinguish the ways they occur in time, and come in four major flavors: activities, accomplishments, achievements, and statives. Activities do not have a built-in end or beginning, but are homogenous all the way through, like "sleep." Accomplishments are long term activities that have a built-in goal, like "build," after which the action cannot continue. Achievements are instantaneous, like "summit" or "win." Stative verbs are just what they sound like, denoting states, like "know."[10]

The aktionsart of verbs over the course of a story is likely to experience shifts, from a preponderance of backgrounding statives in the beginning to accomplishments and achievements toward which the story moves. For each event, I would like to add a feature that captures the aktionsart of the root verb, to investigate this prediction and use it in classification if at all possible. Some promising work has been done in aktionsart tagging (Im and Pustejovsky, 2009; Zarcone and Lenci 2008), but they are built on corpora of news articles and Italian, which will likely suffer serious inaccuracies on the corpus used here. Unfortunately, retraining requires rigorous supervision, and is unlikely to be viable for a single researcher.

6.1.5 Hyponymous Events and Emotions

Building on some of the work done in the NodeBox package, I propose the extrapolation of broader verb categories than currently register. These can be acquired by a couple of means, one of which is via VerbNet, another WordNet, and another of which involves Word2Vec. Lemmatization is a place to start, but I see that as an intermediary, rather than final, abstraction step.

NodeBox itself works on the principle of finding the greatest overlap among entries that match the search term, which could be done with verbs for a higher classification of verb type.

[10] At least in English, these examples are correct, though they vary cross-linguistically. Additionally, there is a proposed fifth category, called semelfactive, for instantaneous verbs with no end, such as "knock." The debate around the true nature of sleep and of activities versus statives in general is like a compact fission reactor in that it is very small and extremely heated. It is unlike a fission reactor in that it does not have a broad effect zone, and no one has ever died from a meltdown.

How far that will take interpretive measures remains to be seen and would be a project on its own, I imagine. Luckily, for events, VerbNet is already out there, a resource that organizes verbs into classes, according to their shared roles and restrictions, as well as, naturally, their synonymous relations. This is a stronger candidate solution, though it may resist tweaking. If it can be made to work on the archaic and specialized vocabulary of fairytales, its addition to the features of a given event would be highly desirable.

Alternatively and for emotional content, taking the vocabulary of the corpus and using word2vec to position all the verbs and emotion words relative to one another, and then clustering them and using their centroids to determine their hyponymous relations might work better, and would return a result not limited or informed by WordNet's or VerbNet's lexicon, but relative to the corpus at hand. Nearest neighbors in the search space created by word2vec might be a viable back-off option, only invoked if a too-low confidence comes back from the computationally-cheaper options.

Having this information would serve the analysis I want to perform on classifying and predicting important events, but would also contribute meaningfully to further steps outlined in section 6.2.

6.1.6 Summary of Improved Process

All together, the process I envision, based on what I have already done, can be represented with the very high-level pseudo-code below:

```
1. Read in text of story
2. Preprocessing:
      For each pronoun:
            Replace pronoun with referent
      For each VP:
            if subject not overt:
```

```
                insert subject
        Shallow syntactic parse revised text, generating list of
        clauses
3. For each clause:
        Semantic parse of clauses (SEMAFOR)
        Generate events from semantic parse + syntactic/surface
        info
4. Classify events by importance
5. Output ordered events above threshold
```

Alternative goals for output also include prediction of next event or event type, rather than a summary. With sufficient data from which to learn, (isn't that always the sticking point?), a system might learn to predict what is in store for a given character. This could be used to create quantifiers of triteness for a given narrative, a fairytale generator, or a wild-ride choose-your-own-adventure activity.

6.2 Expansions

It is in this section that the author's imagination runs most wild, and though all these proposals are natural extensions of what has been done, they may prove quite distant ones.

6.2.1 Structural Mapping

A big goal of this project was to render a human-like model of a story in computational form. The form I most favor is that devised by David K. Elson in his quite nifty story mapping annotation software, Scheherazade. While Scheherazade is very clear and user-friendly for humans wanting to create story maps by hand, it is not automated. It cap-

tures multiple timelines, allowing for the putative as well as the perfective, the counterfactual and the actual, along with chains of causality through which to track a character's motivation from idea to completion (with good or bad results, the source of the "Resolution" label in my own annotation system). These causality chains and timelines seem most plausibly automatable through a genre translation of the work of Im and Pustejovsky on SUBEVITA (2009).

Im and Pustejovsky created a resource for events' and subevents' annotation called SUBEVITA, which creates sub-events based on events that are overtly articulated. Upon inputting a verb like "receive," for example, SUBEVITA generates the before-state of not-having and the after-state of having whatever was received by the recipient. This seems pedantic to the human reader, but computers need remarkable pedantry to make anything of natural language. Often, these transitions between states of having and having not are causal to something central in the story. Indeed, in the example story in the tutorial for use of Scheherazade, the consequence of the raven dropping the cheese is that the raven no longer has it, and it is available for the fox to pick it up (his goal all along). While only the former of these consequences would be found by SUBEVITA, the not-having of the fox ending when the not-having of the raven begins is a narrative pattern a well-annotated corpus and tagger could be taught to recognize, and output.

Not every event's sub-events are germane to the story or deserve representation in the model of the story. SUBEVITA was created to improve question answering, not for summarization, which benefits from many possible representations being available to correspond to any given question. Summarization, on the other hand, does not need all the same representations, just one that captures what is relevant. Adding the useful

but copious information from SUBEVITA will require rigorous training to sift through for optimal summarization. That said, capturing the implied states behind verbs of transition is highly valuable to determining what and why events are important, so the slog would be worthwhile. With its contribution, links of causality and motivation would be significantly easier to annotate, which could then be mapped to a Scheherazade representation of the whole story.

6.2.2 Summarizing Literary Architecture

At a higher organizational level, stories follow certain themes, like revenge and questing. When summarizing a story, it is useful to indicate such arcs. The Genesis lab at MIT ave done some wonderful work training a classifier to identify classic storylines of this type, but it relies on massively simplified language. While the program can accurately say that Macbeth involves revenge, the language of the summary used is close to *un*natural, and certainly not what Shakespeare wrote.[11]

The Genesis team's work is conducted within the framework of artificial intelligence research, the group's raison d'être, so it is not immediately in line with NLP goals or terminology,[12] but it is a great end-point for narrative analysis. At present, the engine operates on summaries, with a low ratio of events to sentences. The language is very close to being unnatural. As an example text, the group's version of Macbeth from their 2011 paper:

> Macbeth, Macduff, Lady Macbeth, and Duncan are persons. Macbeth is a thane and Macduff is a thane. Lady Macbeth, who is Macbeth's

[11] I do not plan to cross that particular bridge, as I would have to create a digital David Bevington, which exceeds the scope of this project.

[12] A number of terms the group uses are something of reserved words in NLP, so the reader is warned not to visit NLP expectations on the paper when reading it.

> wife, is greedy. Duncan, who is Macduff's friend, is the king, and Macbeth is Duncan's successor. Macbeth defeated a rebel. Witches had visions and talked with Macbeth. The witches made predictions. Duncan became happy because Macbeth defeated the rebel. Duncan rewarded Macbeth because Duncan became happy. Lady Macbeth, who is Macbeth's wife, wants to become the queen. Lady Macbeth persuades Macbeth to want to become the king. Macbeth murders Duncan. Then, Lady Macbeth kills herself. Dunsinane is a castle and Burnham Wood is a forest. Burnham Wood came to Dunsinane. Macduff had unusual birth. Macduff fights with Macbeth and kills him. The predictions came true.

There was a bit of wheel reinvention in the group's work, in that Genesis (the name of the system as well as the group) represented what it called *commonsense if-then rules* with a customized internal system, rather than using a resource like SUBEVITA to take care of consequences like, *if X kills Y, Y is dead*.[13] Bringing these two resources into intentional cooperation, along with the others included and proposed above, could offer a much more robust system, capable of dealing with far more natural language than the current version.

[13] Sometimes the pedantry of spelling these phenomena out is used for comedic effect, another stylometric quality that requires preservation of the surface qualities along with the deeper semantic relationships to register. An example of this can be found in Terry Pratchett's work, a paraphrastic example of which is, "[character] immediately stopped whining and immediately started lying senseless on the floor."

7 Conclusions

There is so far to go before we can reliably avail ourselves of the tremendous speed and power computers can bring to bear on dealing with the staggeringly vast amount of written material out there. Every step on the road to comprehensive analysis has been hard-won. Bringing them together is a big challenge, but a worthy one. It is an unfortunate consequence of the proliferation of literacy that we have produced more to read than anyone ever can in a lifetime. Computers can help us find what we need in this huge ocean of words, as long as we can teach them how. Where the writing is concerned with transferring information as directly as possible (news, e.g.), this is easier. When it requires some leaps of intuition to analyze and summarize, it creates a greater challenge.

More than ever, I am painfully aware of the disjunct between human thought and the machine representation of it. How right Joseph Campbell was when he said, "Computers are like Old Testament gods; lots of rules and no mercy." Codifying the process by which we, human-like, understand narratives is an exercise in alien translation - we add additional symbolism and literary tools for storytelling that get us even further from the straightforward data transfer than natural language already does. At the end of this project,[14] I am simultaneously more daunted and more fascinated than ever.

7.1 On Work Completed

In general, I stand by the approach (being an object-oriented programmer) and the first, wobbly steps toward the class definition covered in this project. There are ways I would add on and improve, but I would not reduce it from what it is now. While it may capture too little at present, what it does capture is part of a foundation on which I look forward to building. I believe that the processor itself has good bones, but the involvement of other modules from such a variety of sources as described in section 6 may render it bloated. As these extra functions are added, experimentation will be necessary to find the most seamless combination that best balances the

[14] As dictated by the submission of this paper, not necessarily the conclusion of the project as I intend it.

features captured with the practical concern of processing time. Although time is not so urgent a concern while processing learning material for the classifier, anything with an end user needs to concern itself with reasonable speed in returning answers after the learning has taken place.

If I had it to do again, perhaps I would select only one of my specific predictors, like character importance, and pursue that single-mindedly, in order to have a more complete research arc to present. I would follow the same approximate steps: event definition and processing texts, but with fewer features and an easier read-out for annotation, I think it might get further into the annotation and machine learning processes, even under the same time constraints, perhaps far enough to validate the smaller-scoped hypothesis. I think, however, that I would end up in the same place: seeing it as a first step with many to follow.

7.2 On Work Unrealized

Having described what was within the scope of my proposal but did not see completion, it is not so much the process that I am sorry not to be able to detail more extensively, but its resultant data. Although it would have suffered the inaccuracies of having had a single annotator, a picture of event patterns in even impressionistic strokes would have been desirable. A corpus complete with importance annotation would also be a more helpful stepping stone to leave the rest of the computational linguistic community, though perhaps throwing the gates open at this stage will ensure a more robust annotation procedure than I could alone.

From a purely selfish perspective, I regret not getting more experience with machine learning as it applies to this project. With the knowledge of the options available to me that comes from reading and theory, I feel confident in my choice of clustering algorithm. That said, there are things that can only be learned by doing, whether the act is instructive in itself, or teaches what questions to ask. It is possible that in the act of actually processing my results and creating my classifier, I would find a better representation for the data and a better process for dealing with it than I have from my armchair. I hope to perform this work myself, but if someone beats me to it, I look forward to reading about their findings.

7.3 On Work Imagined

It was through looking for specific solutions to problems encountered in the process of this project that I found some of the resources I discussed as improvements and expansions. This kind of discovery is iterative, as one finds more and more exactly what is and is not captured, correctly or incorrectly, and learns the terms and qualities others have found, and sometimes addressed. As optimistically as I look at what I have proposed as future work and try to impose a clean finish line on it, it is incredibly unlikely that the goalposts will not move as more discovery is made and greater elegance visited upon the solutions available to the sub-problems of this whole challenge. That said, I believe the suggestions I have made are decent starting points. While readers may disagree on which is most likely to be first in yielding results, either by ease of implementation or quality of underlying engine, all have demonstrated value.

The ongoing pursuit of human-like interpretation of natural language by a computer is a fascinating, multi-faceted adventure. While the work here completed, proposed, and suggested is a small step, I hope it serves future researchers on the way.

References

Bahdanau, Dzmitry, KyungHyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." ICLR 2015 Proceedings (2015). ArXiv. Web.

Chen, Yubo, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. "Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks." Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (2015).

Das, Dipanjan, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. "Frame-Semantic Parsing." Computational Linguistics 40.1 (2014): 9-56. Web.

De Marneffe, Marie-Catherine, Marta Recasens, and Christopher Potts. "Modeling the Lifespan of Discourse Entities with Application to Coreference Resolution." Journal of Artificial Intelligence Research 52 (2015): 445-75. Print.

Elson, David K. Modeling Narrative Discourse. Thesis. Columbia University, 2012. New York, 2012. Print.

Fitzgerald, Nicholas, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. "Semantic Role Labeling with Neural Network Factors." Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015). Web.

Hiong, Siaw Nyuk, Narayanan Kulathuramaiyer, and Jane Labadin. "Natural Language Semantic Event Extraction Pipeline." Proceedings of the 4th International Conference on Computing and Informatics (2013).

Im, Seohyun, and James Pustejovsky. "Annotating Event Implicatures for Textual Inference Tasks." GL2009 5th International Conference on Generative Approaches to the Lexicon Proceedings (2009): 125-31. Print.

Lasersohn, P. "Event-Based Semantics." Encyclopedia of Language & Linguistics (2006): 316-20. Web.

Li, Qi, Heng Ji, and Liang Huang. "Joint Event Extraction via Structured Prediction with Global Features." Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (2013): 73-82.

Mani, Inderjeet, and James Pustejovsky. "Temporal Discourse Models for Narrative Structure." Proceedings of the 2004 ACL Workshop on Discourse Annotation - DiscAnnotation '04 (2004). Web.

Mikolov, Tomas, Greg Corrado, Kai Chen, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." (2013). ArXiv. <http://arxiv.org/pdf/1301.3781.pdf>.

Mikolov, Tomas, Ilya Sutskever, Greg Corrando, Jeffrey Dean, and Kai Chen. "Distributed Representations of Words and Phrases and Their Compositionality." Advances in Neural Information Processing Systems (2013).

Palmer, Martha, Claire Bonial, and Diana Mccarthy. "SemLink : FrameNet, VerbNet and Event Ontologies." Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (1929-2014) (2014). Web.

Täckström, Oscar, Kuzman Ganchev, and Dipanjan Das. "Efficient Inference and Structured Learning for Semantic Role Labeling." Transactions of the Association for Computational Linguistics 3 (2015): 29-41. Natural Language Processing. Google, Inc. Web.

Winston, Patrick. "The Strong Story Hypothesis and the Directed Perception Hypothesis" AAAI Fall Symposium Series (2011). 15 Dec. 2011 © 2011 Association for the Advancement of Artificial Intelligence

Zarcone, A. & Lenci, A. "Computational Models of Event Type Classification in Context" Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 08), (2008).