PATHWAYS TO THE NATIVE STORYTELLER: A METHOD TO ENABLE COMPUTATIONAL STORYTELLING

BY

ARAMIDE O. KEHINDE

A DISSERTATION SUBMITTED TO THE SCHOOL OF COMPUTING, COLLEGE OF COMPUTING AND DIGITAL MEDIA OF DEPAUL UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN DATA SCIENCE

DEPAUL UNIVERSITY

CHICAGO, ILLINOIS

2020

DePaul University
College of Computing and Digital Media

Thesis Defense Report

I have read the thesis written by:

Name __Aramide Kehinde_____

(***To the advisor)***: The following thesis title is identical to the one on the title page of the draft returned to the student. This title is approved by me and it is to be used when the final copies of the dissertation are prepared.

Title of thesis:

Pathways to the Native Storyteller: A Method to Enable Computational Story Understanding

Advisor's Initials ____RS_____

☒ Acceptable. Candidate may proceed to prepare final copy

☐ Pass, with revisions stated below:

☐ Not Acceptable. Please explain:

Raffaella Settimi, PhD 6/8/2020

Advisor (Print Name) Date

John Shanahan, PhD

1st Reader (Print Name)

Jon Gemmell, PhD

2nd Reader (Print Name)

Tokunbo Hiamang

3rd Reader (Print Name)

4th Reader (Print Name)

ABSTRACT

The primary objective of this thesis is to develop a method that uses machine learning algorithms to enable computational story understanding. This research is conducted with the aim of establishing a system called the Native Storyteller that plans and creates storytelling experiences for human users. The paper first establishes the desired capabilities of the system and then deep dives into how to enable story understanding, which is the core ability the system needs to function. As such, the research places emphasis on natural language processing and its application to solving key problems in this context. Namely, machine representation of story data in a way that adequately respects the contextual information in the story; and, identification of relationships between multiple stories based on this contextual knowledge. To do this, the BookNLP pipeline is used as a backbone for extracting structured data from textual stories sourced from My Book of Bible Stories. The core contribution of this work is the application of extensions beyond the BookNLP pipeline through NLP algorithms and ELMo neural language embeddings to create features that represent the system's computational understanding of its stories, both at a plot and character-level.

1. INTRODUCTION

1.1 Introducing the Native Storyteller:

We live in the days of media streaming platforms. The advantage these platforms bring is the democratization of media and storytelling. They have decreased barriers for many more individuals to participate in both the production and consumption of story content. Such platforms find their competitive edge in their ability to hyper-personalize the user experience in such a way that optimizes users' search for content.

Case in point is Netflix. Netflix has the objective of maximizing member satisfaction and monthly subscription retention, which correlates with maximizing video consumption [1]. As such, Netflix has historically employed Cinematch, which is a proprietary set of recommendation algorithms that drives its hyper-personalization for users. To determine what movies to recommend to a user in each session, their system of algorithms depend on similarity as described in [1]. For example, similarity between movies in the Netflix catalogue, or between users across multiple dimensions. This stays true to the prototypical way that recommendation systems work [7].

Netflix uses a rich set of data features to train its recommendation algorithms and to refresh them after every user session. For example, they use over a billion user-movie ratings and historical views, demographics, temporal data, social data, and metadata about the titles (actors, genres, directors, etc.) to first estimate similarity across multiple dimensions, and then calculate the probability that a given user will watch and enjoy a set of titles [2]. Netflix uses a complex ensemble of algorithms to determine this

probability, which are a mixture of supervised and unsupervised approaches, including for example logistic regression, restricted Boltzman machines, and multiple clustering algorithms. The titles with the highest probabilities are then displayed on the Netflix user interface. Likewise, the position of each recommended title (whether first in the top row of the display screen or in the center of the bottom row) is further optimized with a ranking algorithm. The result is a personalization experience that maximizes consumption of video content. If we take Netflix's underlying system as the prototype for storytelling systems, a closer look at the features it uses for recommendation shows that they do not capture detailed content within the stories in its catalogue [2] . Rather, the features Netflix uses to train its algorithms are primarily external to the story information - they are either metadata about the story, or data about past user or system actions.

In a different vein, this thesis conceptualizes and introduces preliminary results for the Native Storyteller, a storytelling system that uses semantic expressions within the stories in its catalogue to determine what content to display to its users. The system's objective is to curate a meaningful storytelling experience for the user. In this context, a storytelling experience is defined as a prioritized compilation of short-form stories that is designed to create a specific "effect" for the user. The effect can be one of multiple things based on available data for the user (e.g.: past viewing history, demographics, etc.) or the goals the user expresses within the system. In its robust form, the Native Storyteller applies a thorough understanding of both its users and its stories to achieve this effect. It is able to associate the stories in its catalogue with its users, and identify what compilation of stories optimally achieves the said effect on this user. Likewise, the system hosts multiple modes of stories in its catalogue - visual, audio, or text - and can compile stories of different modes in a single storytelling experience.

The focus of this research, however, is to complete the steps needed for the Native Storyteller to "understand" stories and use its understanding to curate a storytelling experience. This phase of research focuses strictly on textual stories sourced from the Bible, as a prototype of short-form stories. Two steps are completed in this research. The first step is the identification of how to create a computational model of textual story content that enables the system's story understanding. This involves applying the BookNLP pipeline and data enrichments (e.g. features from a recurrent neural network architecture) to represent story data in a way that is semantically aware of the details in each story. This also involves determining the feature space for all the story representations in order to achieve the second step, which is association.

For the Native Storyteller to select a compilation of short-form stories in each storytelling experience, it must be able to make associations across the stories in the feature space, and identify the set of stories to display in a storytelling experience. Ideally, this selection is based on the user's data or expressed goals, as stated earlier. However, this phase of research strictly uses story details to make associations across multiple stories in the system.

1.2 Building Blocks for the Native Storyteller:

Roger Schank, a renowned artificial intelligence theorist, maintains in his work that artificial intelligence must mirror human intelligence. Therefore, to create a storytelling system like the Native Storyteller, we must first discover the building blocks of human storytelling which will help to create a model of this type

of intelligence in a machine. Schank's work on artificial and narrative intelligence provides insights into these building blocks. They help to define the capabilities of a system like the Native Storyteller, and highlight the nature of the challenge at hand. The building blocks captured from Schank's work [18] are are summarized below:

A. Human memory is story based

Human memory is a cluster of different experiences accumulated through time. Much of these memories are story based, and are relevant to daily life. A behavioral job interview, for example, requires such story based memories with prompts like 'Tell me about a time when…" The interviewer is practically asking to hear a story. The interviewee uses the terms in the question to retrieve the appropriate story from memory, as well as details about the story that allow him or her to provide an appropriate response to the prompt. In the same way that the interviewer uses the story provided to make a decision about the interviewee, humans also internally parse through and extract lessons from theirs and others' stored stories in memory to interact and make decisions daily. Developing the Native Storyteller then, can be considered as equivalent to building a model of a human story-based memory. This memory has its own cluster of stories, and each story is stored in a way that allows smart interaction with the system's users.

B. Conversation is reminding

The next building block from Schank's work is the process of reminding, which is crucial in conversation. In every conversation, each individual brings his or her unique cluster of stories that have been saved to memory. What makes conversation intelligent is the innate ability humans have to call something to mind once given input, and to express thought based on that reminding. Conversation then, is essentially responsive storytelling that depends on the ability to select the right story at the right time, given the input one has heard from those he converses with. In the context of creating a storytelling experience, the Native Storyteller must perform its own process of reminding. The input it gets in this process is the user's data or expressed goals, and that drives its reminding of the relevant set of stories to display to the user.

C. Storytelling is understanding

Humans tell stories in conversation to indicate understanding of what was heard. For example, a speaker in conversation instinctively assesses how well her listener understands her comments based on the relevance of the story the listener tells in response to her statement. Storytelling and understanding then, are functionally the same thing [18]. A statement heard in conversation requires a relevant response. The hearer must be able to understand both the speaker's statement and the stories she has stored in memory to be able to decide what story to tell in response. That understanding involves associations the hearer makes between candidate stories in memory and the speaker's input to determine the best response. Also, it is a granular understanding of story content involving, for example, awareness of the order in which events in the story take place, the emotions that certain actions trigger, the traits of individuals in the story, etc. For humans, this level of granular understanding goes in a plethora of directions because of the richness of the senses and mental processes.

Given this building block, the Native Storyteller likewise needs to have a degree of story understanding at its core, in such a way that it is aware of both general story plots as well as story details. These details

become the features that are used to train the algorithms the Native Storyteller uses to select stories for each user session. In this research, multiple NLP methods are applied to accomplish this. There will be more discussion on these topics to define how the methods work and the specific features they help generate for each story.

D. Each story in memory is indexed

Story understanding as described in the preceding building block relies on indices humans use to store stories in memory, and efficiently retrieve them as needed. Indexing is subjective from person to person, as no two people will identically understand and store an experience in memory. The ways in which the human brain indexes stories are also very complex. Nonetheless, what consistently happens in each indexing task is that the brain extracts specific elements in the story and stores them in memory as labels for the story. In conversation, understanding involves leveraging the elements as indices for the input story and matching them up with elements used to index stories that are already in memory.

What matters for the Native Storyteller is the manner in which the human brain indexes its memories. For the human brain, there is no supervision of the story indexing process. It happens unconsciously as one experiences an event, and the indexing is based on the detailed data encountered within the event. Humans also do not have an awareness of what indices the brain has assigned to each story in memory - not until something specifically calls for a recollection of the story. In the same way, there is no supervised process applied to the Native Storyteller's understanding and recommendation of stories. Details provided on the methods of analysis in Section 4 will illustrate this further. While this research does generate features that influence how stories are represented in a feature space, it also uses a neural network to embed story details in the space based on semantics in the story, which enriches the features in an unsupervised manner.

In conceptualizing the Native Storyteller, this section has discussed how the system differs from known story recommendation systems. Likewise, it posits key building blocks that inform the Native Storyteller's target capabilities, and highlights the nature of the challenge to develop such a system. The following section presents a review of foundational work in the arena of machine enabled story understanding. They serve as examples that substantiate Schank's theories on artificial narrative understanding, and likewise provide tangible outputs that are extended upon in research and development for the Native Storyteller.

2. FOUNDATIONAL WORK

They two projects described herein have formed a baseline for machine story understanding that multiple researchers have incorporated in their work. These systems were reviewed to understand how they work and identify opportunities for extension as it applies to the Native Storyteller.

2.1 Genesis

Focussing on the challenge of implementing story understanding in a machine, Patrick Winston and his artificial intelligence group at MIT have introduced Genesis [22]. Genesis is a program written primarily in Lisp that is designed to reason about stories. As described by team papers, Genesis reads stories written in English and applies word representations and common sense rules to reason about its stories.

For word representations, Genesis uses a text parser and George Miller's WordNet (2011) to create hierarchical representations for words that it reads in each story. For example, for the word *frog,* Genesis creates the following representation: Frog is an amphibian; amphibian is a subtype of vertebrate; vertebrate is a subtype of chordate; chordate is a subtype of animal, and so on. The words represented (or objects, as the team calls them) can be concrete items (like frog), or more abstract, like words that express relationships (e.g the word 'has'). Then, for a sentence like '*The boy took a frog to school',* Genesis combines the representations of each word in the sentence and categorizes the types of objects (classified as things, derivatives, sequences, or relations) in the composite representation [13].

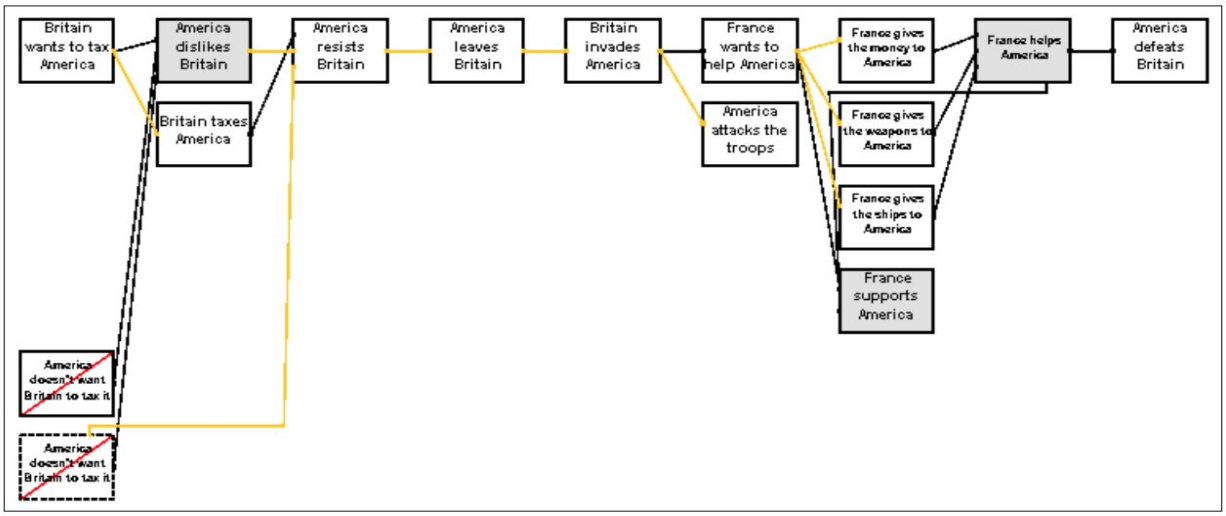
A known challenge in research of this nature is that machines lack knowledge of the world and common sense implications of real situations. Therefore, analysis of text to derive meaning is encumbered by inherent limitations. For example, a machine in a sentiment analysis exercise might be able to associate the word 'war' with negative sentiment, but it cannot independently determine that war implies death and the loss of loved ones, or the other domino effects that war creates. Winston attempted to address this lack of world knowledge by building inference and explanation rules that help Genesis 'reason' about the situations it encounters in its stories. Such rules codify the knowledge needed to identify causal connections in story events. Traditionally, Genesis has needed rule authors to manually encode these rules in Lisp so that it can attribute actions and goals to the story's characters. A sample case is a rule defined in Genesis for 'unfulfilled desire' in Figure 1. Once this rule is codified in the Genesis knowledge base as a specific pattern, Genesis uses that pattern to search all of the stories in its catalogue and identify instances of unfulfilled desire.

Figure 1: Manually encoded Genesis depiction for 'unfulfilled desire'

```
Start description of "unfulfilled desire".

xx is a person.

yy is anything.

xx's wanting yy does not lead to yy.

The end.
```

Upon receiving a story input, Genesis applies its object representations and rules to create an elaboration graph, which is a visual representation of the story. It is important to note here that the team is yet to test Genesis with the entirety of each story. Rather, an author creates a story summary with a few sentences, which he or she then feeds to Genesis for reading. The elaboration graph displays a mapping of events in the story as well as relationships between the events. A study of the graph will reveal key entities (like characters), their actions and desires, as well as causal effects in the story. The image below displays a sample elaboration graph for a summary of the American Revolution.

Figure 2: Genesis elaboration graph



While the Genesis system makes strong headways in story understanding, a deeper look at its mechanics reveals potential areas for improvement as it relates to the goals for this thesis. One area has to do with its recognition of entities in a story. For example, Genesis is able recognize Macbeth as an entity in the play's summary, and it is also able to create an elaboration graph of Macbeth's actions in the play. However, it does not innately identify Macbeth as a noun, or better yet, as a person, unless it is hard coded as a rule into the system when the story is entered. Based on the programming framework Genesis is built on, Macbeth is tagged as a 'thing' (Nackoul 2009). An item like money is also tagged as a 'thing.' As such, there is a high likelihood for a logical breakdown when comparing entities in a story if no custom rules for each story exist, let alone comparing entities across multiple stories.

Genesis' elaboration graph is essentially a visual representation of the inner language it develops for its stories. The underlying mechanism that enables this inner language is Boris Katz's START system (SynTactic Analysis using Reversible Transformations). START (Katz 1997) is a natural language system that allows a machine to translate English input into an internal knowledge base that includes information found in the text. START also has a generating module that produces English sentences in response to user queries.

Nonetheless, the way the START system achieves its question-answering functionality also warrants caution when considering the goals of this thesis. To answer user queries about its internal knowledge base, START reorganizes the query so that it is phrased in a way that is identical to its existing knowledge base. This allows the system to 'fill in the gap' when answering user queries. Consider this input sentence in its knowledge base: 'Bill surprised Hillary with his answer.' START internally represents the statement in two expressions: <Bill surprise Hillary> with answer> and <answer related to Bill>. Now consider the user query: 'Whom did Bill surprise with his answer?' To answer this question, START first reorganizes it into a structure that is similar to the original text input: 'Bill surprised ***whom*** with his answer.' The resulting internal expression in START for this reorganized question then becomes <Bill surprise ***whom***> with answer>. START then uses 'whom' as the matching variable. It recognizes that Hillary takes the place of whom in the initial input sentence as well the user the query. Then, it fills in the 'whom' gap with

Hillary and provides the initial statement back as its answer to the user: 'Bill surprised ***Hillary*** with his answer.'

The approach appears to be smart, but in the event that the user asks a question in a way that is syntactically different from the way the input sentence is structured, START breaks down. This is also true of user queries that are semantically the same. For example, START would not be able to answer the question 'Who experienced surprise?' because its internal representation of the question would not be similar to that of the initial input sentence. Again, we see another loophole for a breakdown in the proposed storytelling system for this thesis.

2.2 The Book NLP Pipeline

Bamman et al. [3] take the concept of entity-centric modeling of stories further with the Book NLP pipeline. Their primary goal in developing this pipeline was to train a model that can infer character archetypes from story text. For example, the model was to have the ability to recognize a 'villain' in the story by processing all story data that the story's author attributes to the character who plays such a role. While Genesis requires summarization of large stories to a few sentences, the Book NLP pipeline is able to scale to book-length documents (e.g.: Oliver Twist).

Prior to creating the pipeline, Bamman et al. first identified a story character as a narrative function. With that mindset characters in a story are considered as collections of psychological or moral attributes. From a data perspective, this allows us to think about characters as 'word masses,' with each word representing a character's attributes, from physical to psychological. With its underlying architecture built upon Stanford NLP's POS tagger, entity recognizer, and external tools for parsing, Book NLP creates clusters from a story's vocabulary and associates them with specific characters. The pipeline also achieves pronominal coreference resolution and dimensionality reduction. Coreference resolution allowed the authors to correctly associate all different mentions of the same character (e.g. Tom Sawyer vs. Tom vs. Sawyer, including references to the same character using the pronouns he, him, etc.) with 82% accuracy. The latter allowed them to address degrees of freedom in their trained model.

A few minor rules do apply to Book NLP and these are handled statistically instead of manually. For example, a character must be mentioned with enough silence in the previous 500 words before Book NLP associates a new mention of her name and related attributes of the mention to her existing cluster. With reputable tests on substantial stories like *Oliver Twist* and *Wuthering Heights,* Bamman et al. show that the Book NLP pipeline can scale to novel-length stories with a high degree of textual detail. Processing a narrative with the Book NLP pipeline yields a structured data file that parses all words (tokens) used in the story. Each token is tagged with information represented by 15 features as follows:

1. Paragraph id - The number / position of a paragraph in the entire story in which the token appears (the first paragraph in the story is given an id of 0)
2. Sentence id - The number / position of a sentence in the entire story in which the token appears (the first sentence in the story is given an id of 0)
3. Token id - The number / position of the token in the entire story (the first word or token in the story is given an id of 0)
4. Byte start - the byte start position for the token

5. Whitespace following the token - Identification of whether or not the token is followed by a whitespace
6. Head token id (-1 for sentence root) - the position of the preceding token upon which a given token has a dependency relationship
7. Original token - Representation of the token as it appears in the story
8. Normalized token (for quotes, etc.) - Normalized version of the token
9. Lemma - Lemmatized version of the token which represents the token in its root form (e.g.: made → make)
10. Penn Treebank POS tag - Part of speech tag for the token
11. NER tag - Named entity recognition tag for the token. NER tag can be one of the following options: PERSON, NUMBER, DATE, DURATION, MISC, TIME, LOCATION, MONEY, ORGANIZATION, SET, O (for tokens for which there is no applicable NER tag)
12. Stanford basic dependency label - The label or type of dependency detected between tokens
13. Within-quotation flag - Identification of whether or not the token appears after an opening quotation mark, signifying that the token is part of a character's dialogue
14. Character id - If the token is a character name or a pronoun, it is tagged with a character id number (all coreferent tokens share the same character id)
15. Supersense - Identification of the noun or verb type for each token (e.g. the word 'come' is tag with a supersence of verb.action)

Considering this outcome, a few questions arise as it relates to the goals of this thesis. While Bamman et al.'s work rightly focuses on character clustering, is there a way to extend the work to entity clustering in general? For example, can we identify both instrumental characters and places in the story? In the history of storytelling, we have seen that not only do characters drive the plot forward, but so do physical locations, items, etc. Additionally, can we in some way model how these entities evolve in the story, or perhaps the general trajectory? This is also instrumental to understanding a story. These sample questions reveal the opportunity to extend provisions of the Book NLP pipeline for the Native Storyteller.

3. RELATED WORK

In addition to the work by Winston and Bamman et al. that serve as foundational work in the area of computational storytelling, some additional works serve as reference points for the approaches taken to analyze data for this thesis. These works either address the challenge of storytelling from different angles, or provide rudiments that are directly used in analysis for this research.

3.1 Finding Narrative Similarity

In their research on story understanding, Srivastava et al. [19] use text analysis to identify correspondences between stories. The team defined story understanding as the ability for a machine to recognize similarities in multiple story plots. As such, their goal was to identify movies within a corpus of plot summaries that are instances, or more specifically, remakes of existing stories. To do this, they formed a hypothesis based on an observed pattern in storytelling and film: while superficial details of remakes are often different from their originals (e.g.: character names), their prominent themes, and even sequence of events to a high degree, remain the same. Then, using a corpus of 577 Wikipedia movie plot summaries, they complete the required analysis.

Srivastava et al.'s core contribution to this type of research is their introduction of a story kernel that assesses narrative similarity using a character centric approach. The story kernel is bifurcated into a plot kernel and a character kernel. The former models surface similarity between two plots at a time in terms of main events and entities, while the latter considers associations at the character-level in terms of their attributes and social relationships.

In the plot kernel, they create a model for each story plot that is represented by all the verbs and non-character entities (as well as adjectives that modify these entities) in the plot. The plot model is a bag-of-words, or BoW for short, composed of these elements. In natural language processing, BoW is a simplified representation of text data that measures the occurrence of words in a corpus of text data. Since a computer only understands ones and zeroes, the goal is to create a representation of text data into a format that a computer can use to make computations. Take for example the following hypothetical corpus of two documents:

1. *"Peters Skating Rink was popular in town."*
2. *"Tom and Jesse met at the rink and married."*

BoW involves creating a vocabulary that comprises all the 14 unique words (or tokens) in this data corpus as follows: ["Peters", "Skating", "rink", "was", "popular", "in", "town", "Tom", "and", "Jesse", "met", "at", "the", "married"]. Then, it scores the words in each document by marking whether each of the words in the corpus vocabulary listed above is present in each document. We use '0' when a word from the vocabulary is absent in a given document, and '1' if it is present. The result is a fixed-length binary vector for all texts in the corpus. Using the vocabulary, we create a binary vector for each document in this example as on Figure 3:

Figure 3: Bag of Words vector for documents in the sample corpus

	Peters	Skating	rink	was	popular	in	town	Tom	and	Jesse	met	at	the	married
Doc 1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Doc 2	0	0	1	0	0	0	0	1	1	1	1	1	1	1

With each item in the movie summary corpus represented as a binary vector, the team calculates S_{plot} (s_i, s_j), which is the cosine similarity between vectors for stories s_i and s_j. Cosine similarity is a measure of the cosine of the angle between two vectors in a feature space. The more similar two vectors are in direction, the closer the angle between them is to 0 degrees, and the closer their cosine similarity is to 1.

In the character kernel, the research team makes slightly more granular comparisons between two stories at the character-level. For this comparison, a constraint is required to make sure that each character in one story is paired with one and only one character in the other story. The objective is to maximize the average alignment score of characters in the narrative pair: S_{char} (s_i,s_j). This metric comprises similarity scores across four dimensions that are calculated for character pairs between the two narratives: 1) Character name S_{name} (c_i,c_j), which is an indicator function that checks if the names of characters c_i and c_j in the two stories are matching strings. 2) Character gender S_{gender} (c_i,c_j), a boolean value that is 1 if their genders are the same and 0 if otherwise. 3) Character prominence S_{prom} (c_i,c_j), which is a comparison of

the character pair based on whether their fraction of mentions in their respective stories are similar. $S_{prom}(c_i,c_j) = 1 - |prom(c_i) - prom(c_j)|$. 4) Character social relationships $S_{reln}(c_i,c_j)$, which is a measure of the similarity between the two characters' social relationships and attributes associated with them in their respective stories. Finally, the similarity between two characters in the two stories is identified using a convex combination of the four dimensions:

$$S_{char}(c_i, c_j) = S_{name}(c_i, c_j) + S_{gender}(c_i,c_j) + S_{prom}(c_i,c_j) + S_{reln}(c_i,c_j)$$

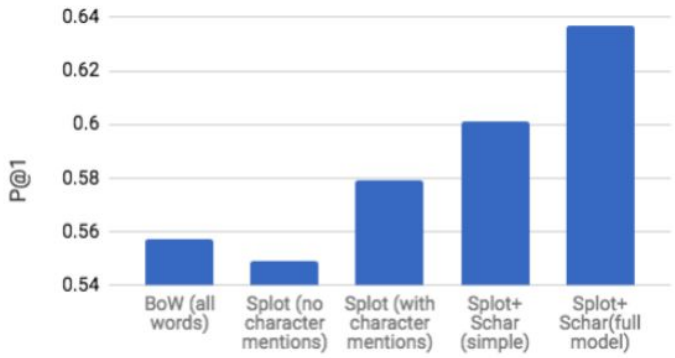
With $S_{plot}(s_i,s_j)$ and $S_{char}(s_i,s_j)$, the story kernel is then defined as:

$$S(s_i,s_j) = a(S_{plot}(s_i, s_j)) + (1 - a)(S_{char}(s_i, s_j))$$

It is notable that Srivastava et al. initially processed their movie plot summaries with the Book NLP pipeline to get dependency parses and to identify main characters in their story corpus. The features provided by BookNLP pipeline allow the team to isolate the verbs and adjectives used to create the story vectors for the plot-level kernel. Even more, they allow the team to isolate personal attributes associated with individual characters. For example, the *supersense* feature which is generated by the BookNLP pipeline will tag tokens it identifies as character attributes with the values 'noun.attribute'.

Srivastava et al. applied a layered approach to their research by gradually testing different methods at their disposal and recording the accuracy scores for each. In each iteration they added more complexity to their model to get gradually increasing accuracy scores in predicting narrative similarity. In one iteration, they test results of bag-of-words across their texts (without introduction of their story kernel) to get an accuracy score of approximately 56%. In another iteration, they pivot from BoW to the plot kernel approach that includes mentions of character names and get an accuracy score of 60%. Lastly, they create a composite model of the plot and character kernels and get an accuracy score of approximately 64%. A full set of results are displayed in Figure 4.

Figure 4: Accuracy scores for multiple methods applied to detecting narrative similarity. Image as displayed by the research team.



3.2 Question-Answering on Story Data

Tapaswi et al. [20] approached story understanding with the perspective that understanding of a scene is better portrayed by the ability to answer any question about the scene. Their hypothesis is that a machine should be able to answer questions about characters' actions as well as the motivations behind them in order to prove its understanding of a story. This implies that they also take a character-centric approach to

story understanding, similar to Srivastava et al. [19]. To achieve their goal, the team created MovieQA, which is a dataset of approximately 15,000 questions sourced from over 400 movies with large semantic diversity. The questions are a set of "Who", "What", "Where, "Why", and "How" quizzes of varying complexity. Each movie in the dataset has a quiz of questions, each with five multiple choice answers and only one of such options is the correct answer.

MovieQA is unique in that it comprises a variety of unstructured data for each movie, namely video clips, subtitles, film scripts, and plots. As such, the resulting data for a story is much richer than other examples encountered in this research. Nonetheless, this research group found that for an algorithm to be able to answer more complex "Why" and "How" questions about a given story, it must be trained on more detailed textual data. Given that, the team primarily used Wikipedia plot summaries and hired annotators to manually gather their quiz questions.

The analysis approach Tapaswi et al. took is informative for the Native Storyteller for multiple reasons. Their work shows the need for a tangible way to prove out a machine's ability to understand its stories. A series of correctly answered questions with varying complexity does this quite well. Reasoning based answers are likely reflective of how a machine can relate multiple ideas in one story. In the context of the Native Storyteller, we can extend this to how the system performs when making relationships between ideas or plot circumstances across multiple stories in order to present a specific story to a user.

The work for MovieQA also provides a good reference point for potential results in story understanding research when considering multiple criteria. [20] presents a detailed view of their approach and results while a summarized version is discussed here. The first criteria has to do with the dataset for analysis. As mentioned earlier, MovieQA is a composition of different types of unstructured data for each movie - plot, video clips, subtitles, and the film's written script. Tapaswi et al. tested and compared their algorithm's question-answering performance on these four types of data sets separately. Secondly, the team experimented with different types of text representations used in NLP when representing the question-answer data in a feature space that a computer can understand. These representations are namely, TF-IDF, Word2Vec embeddings, SkipThought, and finally, a fusion of all three representations using concatenation of the vectors derived from each method. Lastly, the team also experiments with two different methods to predict answers for each quiz question using their text representation(s) of choice. They experiment with a standard cosine similarity calculation between a question and its candidate answers, as well as a convolutional neural network architecture (CNN) that learns cosine similarity with a sliding window across each layer of the network.

The results of these experiments informed choices made for research on the Native Storyteller (Table 1). First, the team consistently gets higher accuracy scores for QA using the Wikipedia plot summaries, which shows that a text with targeted information about a story, and not necessarily all story details, suffices for the task of enabling story understanding in a machine. Regarding the text representations, TFIDF appears to perform better when used by itself in both the cosine calculation and the neural architecture (called SSCB). However, when the team uses a fusion of all three text representations in a neural architecture, they get the highest accuracy score of 56.7%. Granted, a top accuracy score of 56.7%

begs the need for more enhancements, however this speaks to the challenging nature of the QA task in NLP.

Table 1: MovieQA accuracy scores by algorithm and dataset type. Image as displayed by the research team.

Method	Plot	DVS	Subtitle	Script
Cosine TFIDF	47.6	24.5	24.5	24.6
Cosine SkipThought	31.0	19.9	21.3	21.2
Cosine Word2Vec	46.4	26.6	24.5	23.4
SSCB TFIDF	48.5	24.5	27.6	26.1
SSCB SkipThought	28.3	24.5	20.8	21.0
SSCB Word2Vec	45.1	24.8	24.8	25.0
SSCB Fusion	**56.7**	24.8	27.7	28.7

3.3 Story Understanding with External Knowledge Based Attention

Deviating from a character-centric view of story understanding, Li et al. [9] take on the challenge with a 'fill-in-the-gap' approach. They train an algorithm using the ROC story cloze task dataset [12] to achieve the task of predicting the concluding sentence of a story given its preceding sentences. Each story is a simple document of four sentences, and the model must select the reasonable option between two candidate story endings. The correct ending for a story should align with the ideas and sentiments established by the story's preceding sentences, and thus represents a logical ending for the story. On the other hand, the incorrect ending may describe a scenario that diverges from the story's established plot, or present an illogical conclusion for the story.

A sample story reads: *"Sally liked the seltzer waters her mom bought. She grabbed one from the fridge. She was excited to open it and taste it. Sally did not know her brother shook all of them beforehand."*' For candidate endings, the algorithm must select between two options such as: *"The water was very still"* or *"The water exploded in her face when Sally opened it."* For a human reader, the logical ending to the story is very clear, but for a machine that lacks world knowledge to know the import of the brother's trickery, one has to apply creative approaches with existing tools to make the right selection.

Admittedly, a "fill-in-the-gap" approach does not show an algorithm's ability to reason about story data. Nonetheless, Li et al. use methods that notably inform research for the Native Storyteller. They introduce an external knowledge base that guides the training of the model. This knowledge base comprises two things: a TF-IDF representation of each sentence in the story that helps to extract keywords from the corpus of stories, and a calculation of sentiment polarity for each sentence in a story. The knowledge base is then used in the model to calculate the relationship between the story plots and the two potential endings. Note that this relationship is calculated using cosine similarity. Therefore, by extracting keywords across the entire corpus with TF-IDF, the team introduces a type of 'attention' mechanism that removes noise from the story data in preparation for the similarity calculation. As such, at each point the algorithm must compare a story's preceding sentences with its candidate endings in the selection process, the calculation is driven by the more relevant words in the stories and candidate endings.

Additionally, the use of sentiment polarity adds value to the task of adding or finding a logical conclusion to the story. Li et al. reasonably found that the last sentence in the story documents holds much more precedence in selecting the right candidate story ending. As an example of extremes, it would not make sense for the last sentence in the story document to have a highly positive sentiment score, and the selected story ending to have a highly negative sentiment polarity. The external knowledge base thus provides guidance for the prediction process.

The last point is the team's algorithm of choice. Once they encode the textual story data (at a word-level and sentence-level) they combine it with the external knowledge base and feed the components to a recurrent neural network, more specifically, a bi-directional LSTM. They achieve the strongest accuracy scores with this combination of components and algorithms.

3.4 Remarks on Related Works

3.4.1 The Value of a Plot and Character Kernel

Srivastava et al.'s approach to identifying narrative similarity shows the value of analyzing stories at multiple levels of granularity. With the addition of both a high level plot kernel and granular character kernel, their algorithm was able to identify movie remakes with increased accuracy. This provides a key learning for the Native Storyteller. Given the goal of this thesis to enable contextual story understanding in a machine, the initial hypothesis was to approach the task primarily from a granular level inorder to provide the machine with as much data that is needed to represent each story in a way that is respectful of their nuances.

However, the results that Srivastava et al. got with their layered analysis approach shows the value of a high level approach that first characterizes each story's plot as a whole. While a plot-level analysis loses some story information and does not perform strongly in and of itself, the team's results show that it is still able to capture information that a granular story kernel would lack. That said, analysis for the Native Storyteller is also bifurcated into a plot-level and character-level approach. Discussion on applied methods in Section 4 will provide more detail on this approach and its results.

3.4.2 A Fusion of Story Representations

In their research, both Tapaswi et al. and Li et al. applied a fusion of multiple types of word representations, or embeddings, to their analysis. Tapaswi et al. combine TF-IDF, Word2Vec, and SkipThought embeddings to represent each of the stories in their dataset. Li et al. apply TF-IDF for their external knowledge base and combine it with representations they derive from a neural network (LSTM) for their stories. As Tapaswi et al.'s results show in Table 1, a fusion of story representations allowed its algorithms to answer questions about the MovieQA data with increased accuracy.
The key learning for the Native Storyteller is that application of multiple methods to represent story data in a feature space provides algorithms with more information to better accomplish their tasks. Results from both teams show that an ensemble of algorithms applied to the same NLP task combines the strengths of each individual algorithm and accounts for their weaknesses. For the Native Storyteller,

results of the BookNLP pipeline, which represent one method of story representation, are enriched and combined with other methods for story representations that this paper will continue to describe.

A unique inclusion in Li et al.'s work is sentiment polarity. They reasonably hypothesize that the sentiment score of the sentence that appropriately closes a story should not vary much from that of the preceding sentence in the story. For example, in a story with four sentences, they expect that the sentiment score of the third and fourth sentences should be most similar. In that sense, they use sentiment score as a means of representing the second to last and concluding sentences in each of their stories, and pair each story with its concluding sentence based on the polarity between both. This provides an opportunity to explore how sentiment scores, or sentiment polarity between stories can be leveraged in the Native Storyteller. This paper will first describe what language sentiment represents for the Native Storyteller's tasks, and then discuss how it is applied in enabling story understanding.

3.4.3 Considering Word Embedding for Text Representations

Since a computer can only truly understand digits, work in NLP requires us to first translate textual data to a numerical format that computers can understand. This is where word representations come into play, as seen in the previous examples. Word representations are natural language data transformations that allow us to place textual data in a numerical feature space for an algorithm to perform computations on the data. Given the Native Storyteller's task to understand contextual information in its stories, it needs computational word representations that best retain and model the meaning behind each story. The word representations discussed so far do not provide much in that direction.

Srivastava et al. use bag-of-words representations for their plot-level analyses, which comprised a tally of all the verbs in their stories. While this gives a computer something suitable to work with (a numerical vector), the vectors are stripped of all contextual knowledge in the text. As a result, this approach falls short for story understanding. Tapaswi et al. use both term-frequency-inverse document frequency (TF-IDF) and Word2Vec for MovieQA. While TF-IDF has the added advantage of emphasizing more relevant words in the text, one can reason that TF-IDF is naturally more suited to the question-answering task that Srivastava et al. use it for. After all, a question about a story, and its respective answer, would naturally have similar word frequencies or distributions. In a different vein, the Native Storyteller's task is to find relationships between stories of disparate topics, and therefore vastly different word frequencies, in order to determine what stories to display in a storytelling experience.

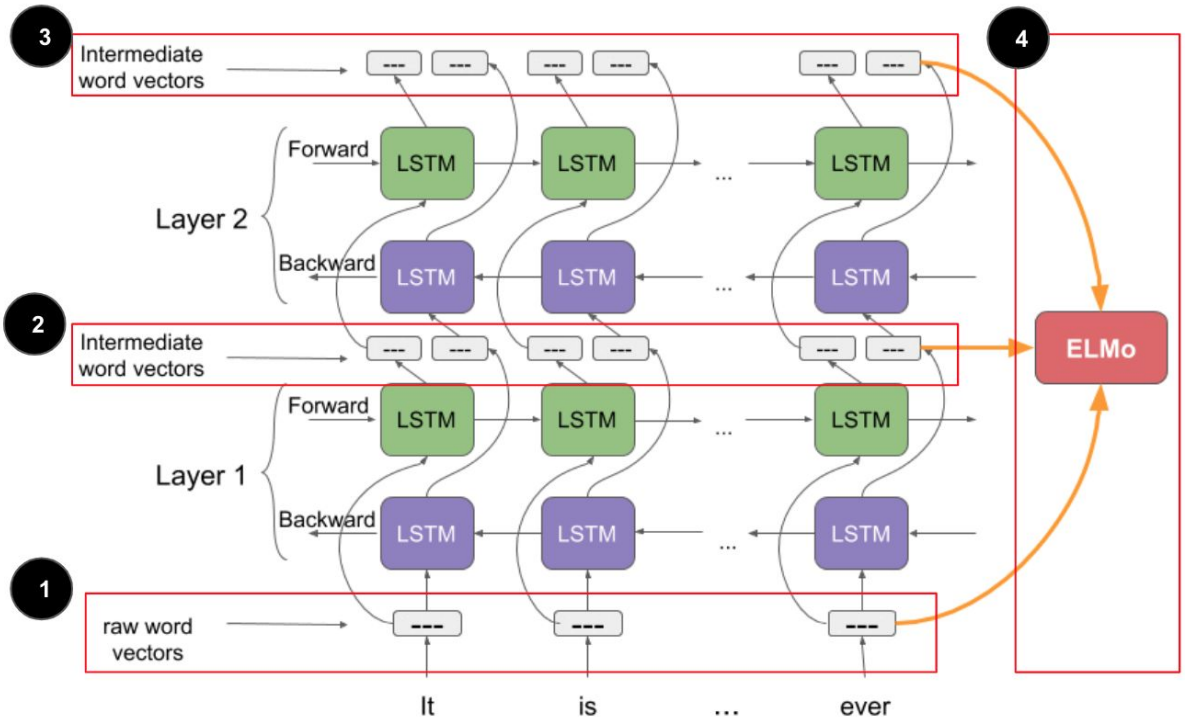
This brings us to the topic of neural word embeddings. Word embeddings are text representations in which words of similar meaning are given similar numerical representations. Word2Vec has traditionally been the neural embedding of choice in NLP, however, research into how it works shows where it also falls short for story understanding. With Word2Vec, each word or token in the text is assigned a numerical vector based on a standard lookup vocabulary. For example, the word 'hit' in a text will always be assigned the same word representation when it is encountered by Word2Vec; but take into consideration the following three sentences:

*"He **hit** John in the face"*
*"He released a new **hit** song"*
*"His career **hit** rock bottom"*

The word ‘hit’ is used in all three sentences, but in each sentence it conveys a different meaning. We as humans instinctively use the words surrounding ‘hit’ to understand it’s meaning each time it appears. Therefore, an algorithm that assigns the same vector to each word regardless of how it is used in the text completely misses their nuanced meanings. This is the problem of polysemy that characterizes natural language.

As an advancement over Word2Vec, Peters et al. introduced a deep contextual word embedding with ELMo (Embeddings from Language Models). ELMo [17] is based on a recurrent neural network architecture (RNN) with two layers, more specifically called a bidirectional LSTM. By nature, an RNN architecture allows for memory retention as input data is processed by each layer of the neural network. This lends itself well to natural language analysis because understanding the meaning of a word in a sentence requires memory of the words that come before and after it.

Figure 5: ELMO’s bi-directional LSTM architecture. Underlying image (without annotations) from Analytics Vidhya



Description of the ELMo deep neural architecture:

1. *Upon input of a natural language sentence, raw word vectors for each word in the input are created using a character-level convolutional neural network*
2. *The first layer of bidirectional LSTM (bi-LSTM) neurons accepts the raw word vectors and generates intermediate word vectors. The forward LSTMs (in green) take the vector for each input word and calculate the probability for the next word given the previous word. The backwards LSTMs (in purple) take a word input and calculate the probability of the previous word. These probabilities feed the intermediate word vectors*

3. *The word vectors from the first layer of bi-LSTMs are treated as input for the second layer of bi-LSTM and the same process as in step 2 happens in the second layer. The result is a second set of intermediate word vectors*
4. *ELMo calculates a linear combination of all vectors introduced or generated at each layer of the bi-LSTM architecture and output those results as the contextual word vectors for the input text*

Instead of using a standard dictionary of words and their corresponding vectors, ELMo uses a bidirectional neural architecture to create word representations in context of surrounding words. To create a representation for a word, for example, 'hit' in the first sentence, ELMo trains one left-context neural network that goes back to the beginning of the sentence the word appears in, and one right-context neural network that goes from that word to the end of the same sentence. ELMo would create three distinct representations for the word 'hit' from the three sample sentences above, thereby accounting for the nuanced meanings of the word in each context. Given the task of the Native Storyteller, this research uses ELMo for the task of story understanding.

4. APPLIED METHODS

This section describes the methods applied in research and analysis for the Native Storyteller. First, it presents a set of tasks to complete in order to enable story understanding for the Native Storyteller. Story understanding is a crucial capability for the Native Storyteller because of the system's objective to independently create a storytelling experience as described in Section 1.1. These target tasks influence the decisions made during the analysis. This section also introduces the dataset used for analysis and discusses novel approaches that extend the previous work done in this area of research.

4.1 Tasks for Story 'Understanding'

The Native Storyteller's goal of creating storytelling experiences for human users is highly dependent on the system's ability to understand and reason about stories in a way that emphasizes semantics within each story. Practically, this means the Native Storyteller should have the following functionalities that necessitate completion of discrete tasks using NLP methods:

- Ability to identify the overarching themes in each story
- Ability to correctly characterize the stories and distinguish between them in a way that accounts for deeper nuances beyond overarching themes
- Ability to understand and represent the sequence of events in each story
- Ability to identify and represent the emotional impact of each story
- Ability to recognize individual characters, and their specific traits or personas

These functionalities are enabled using the following NLP methods which help to derive specific data features for analysis:

Preprocessing using the Book NLP pipeline: As a data preparation step, the BookNLP pipeline tags all words, or tokens, in each story with the 15 features described in Section 2.2. A crucial task the BookNLP pipeline completes is detection of key characters in each story and assignment of a consistent character ID for every mention of the character in the story. Secondly, the BookNLP pipeline tags each token with an individual verb type or noun type, or *supersense,* where applicable. The supersense field is used to derive

a feature that characterizes the stories at a plot and character-level in a way that accurately represents their nuances.

Sentiment analysis: Sentiment analysis is the first step applied to enrich the data provided by the BookNLP pipeline. For this task, a sentiment score is calculated for each sentence in every story using the Google Natural Language API [14]. Therefore, each story gets a sequence of sentiment scores that is the size of the number of sentences in the story. These sequences are aggregated further to represent the story's emotion trajectory as a data feature.

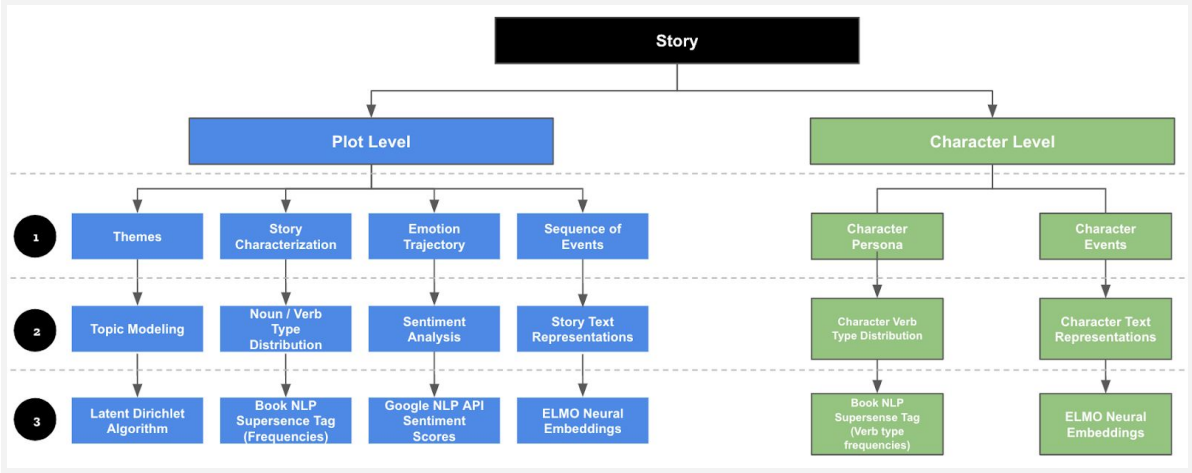
Topic modeling: In NLP, topic modelling is used to identify latent topics in a corpus of documents. These topics are represented by a group of keywords within the texts that are highly coherent in meaning. This method is useful for abstracting themes from the corpus of stories in the dataset selected for analysis. While topic modelling is used across multiple fields like genetics and bioinformatics, the value it adds in literature is the ability to identify latent semantic structures in an extensive body of text that are not readily recognizable by the human eye. For this analysis, a Latent Dirichlet Allocation (LDA) [4] is used to derive topics that map to a set of 16 themes, that are ultimately assigned to each story in the corpus as a data feature (one theme for each story).

Text representation using the ELMo Algorithm: ELMo, as explained in Section 3.4.3, is used to create a computational representation of the sequence of events in each story. This a much denser feature that is used to enable story understanding. By virtue of how ELMo works, the result is a representation of story text that retains the contextual information in the story.

With data from the BookNLP pipeline further enriched by the methods described above, the Native Storyteller is given a set of features that enable its 'understanding' of each story in its catalogue. The system is then able to apply a clustering algorithm to make associations between its stories and characters, and identify candidate stories for a user's experience.

Research for the Native Storyteller is bifurcated into a plot-level and character-level analysis, similar to Srivastava et al. These provide data at differing levels of granularity that help the Native Storyteller function. The plot-level representation accounts for each story's sequence of events, underlying themes, emotional impact, and its characterization at a high level. The character-level representation focuses on all characters introduced in the story corpus and represents the events surrounding each character (i.e.: what they do and what is done to them) and their personas. Figure 6 visualizes the features engineered for the plot and character-levels and maps them to the methods and tools used to create them.

Figure 6: Native Storyteller features and the NLP methods used to derive them



Description of features created for plot-level and character-level analyses in levels a follows:

1. *Plot and character-level data features that the Native Storyteller needs for each story in its catalogue*
2. *The NLP or text analysis methods needed to derive the features in point 1*
3. *The tools or software used for each NLP or analysis method listed in point 2*

4.2 Dataset of Choice

Initially, the target dataset for analysis was Wikipedia movie plot summaries. However, an initial test using the BookNLP pipeline with sample movie summaries revealed the need to pivot. As with any story, a movie plot can involve complicated story elements for entertainment value. Initial tests with the BookNLP pipeline showed limitations of the software in handling these elements. One primary example is that BookNLP is not built to distinguish flashbacks in a story from the present moment. Therefore, it falls short for the task of dependably identifying the sequence of events in a non-linear story. Another limit observed is the software's ability to identify characters that are not tangible or named persons. Some sample stories that were initially tested with the BookNLP pipeline had prominent characters that were nouns; for example, the wind and the sun for a children's story. Results from the BookNLP pipeline showed that the software simply tagged such characters as nouns, and as a result the parsed data for the pipeline made the story appear to have no characters. (For the record, a character in this analysis is defined as a being with identifiable actions or mentions in a story.)

A dataset that abstracts out such complexities was required for an initial proof of concept of story understanding and association in the Native Storyteller. For this reason, the story corpus of choice is Bible stories written for children. Bible stories present the benefit of being short form in nature. Likewise, they are reliably linear in structure, and their characters are, more often than not, tangible characters that can easily be identified with known tools for named entity recognition, which the BookNLP pipeline is built upon. There is also the added advantage of existing relationships between characters and elements in multiple Bible stories. For example, one story in the corpus can speak primarily about the birth of Jesus, while another story in the corpus can be about an instance from Jesus's daily life. This allows a test of how adequate the computational representation for each story is, as well as how well the NLP methods described earlier capture patterns and similarities across multiple stories

The Bible stories for this research are sourced from *My Book of Bible Stories,* an online repository of 116 Bible stories that have been re-written in plain English for children. Minimal adjustments were applied to the data during the web scraping exercise to retrieve the data. For example, the name 'God' was changed to a character name, 'Emmanuel', because the BookNLP pipeline treats the name as a noun instead of a person. Bible stories ascribe tangible actions to God; therefore, his person is a legitimate character to analyze. Additionally, each story and character was given a unique ID number to allow for analysis of both entities across the entire corpus. 98 stories from *My Book of Bible Stories* were used in this analysis for the Native Storyteller. The stories ranged from a length of 10 sentences to 65 sentences.

4.3 Plot-Level Feature Engineering

As displayed in Figure 7, the goal for feature engineering at the plot-level is to generate features that represent a story's *theme*, a granular *characterization*, *emotion trajectory*, and its *sequence of events* in a way that stays true to the semantics within the story.

4.3.1 Identifying Story Themes with Topic Modeling

To identify story themes, the raw tokens from BookNLP pipeline were first combined to form one paragraph for each story, representing the story as one would read it on paper. Additionally, steps to remove stop words (e.g.: a, and, the), bible character names, bible book names, and words that typically occur in abundance in the Bible were applied. For example, words like 'lord' or 'thing' occur frequently and are taken out to ensure that the themes derived from the corpus are meaningful. Likewise, character names were removed before topic modeling so that the algorithm does not identify characters with high mentions as a theme. Case is point is Jesus, a man whom more than half of the Bible stories are explicitly about.

A topic modelling algorithm seeks to automatically find the underlying, or latent, structure in a corpus of documents using a number of iterations through the documents. This allows for assigning or clustering the documents based on the distribution of words that occur in them. LDA does this in a probabilistic manner by first creating a bag-of-word representation for each document (as described in Section 3.1). Then, it uses these bag-of-word vectors to identify the document and topic vectors that best explain the data [4]. The topics derived from LDA display as a collection of keywords that the algorithm assigns to the same cluster. Coherence score is used to evaluate the accuracy of the model by measuring the degree of semantic similarity between the keywords that are assigned to the same topic. The goal when training the topic model for the Native Storyteller is to select LDA parameters that maximize the model's coherence score, including the number of topics or themes identified by the algorithm, the parameter α representing document-topic density, and β representing topic-word density.

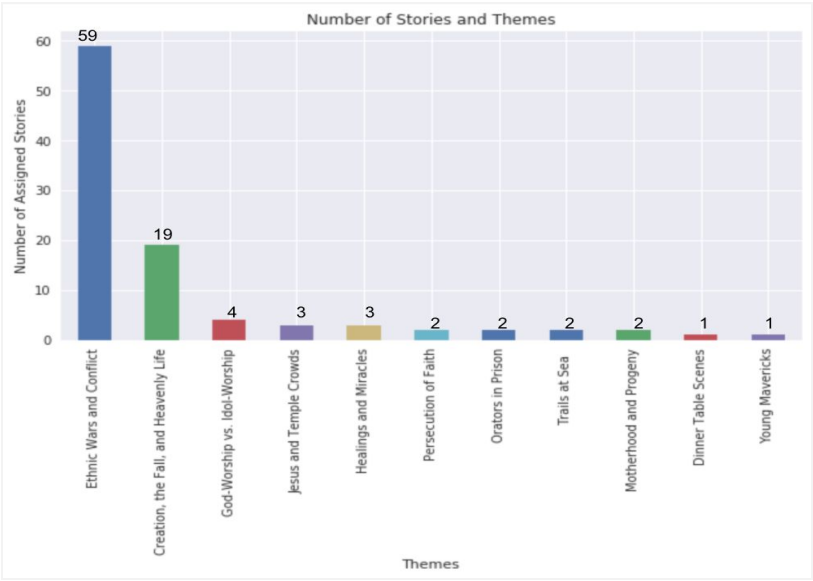
The strangest performing LDA algorithm was trained using the Bible corpus itself. A series of tests were first applied to select optimized parameters that would yield the highest coherence score (16 topics, alpha of 0.91, beta of 0.61). Then, using those parameters, the LDA is trained to derive topics. Figure 7 below shows a sample of those 16 topics and the underlying keywords within each topic. The LDA calculates a weight for each keyword in the topics to represent the keywords' prevalence in the documents to which its respective topic applies.

Figure 7: Snippet of topics derived for the LDA algorithm

```
[(0,
  '0.020*"temple" + 0.013*"wall" + 0.011*"build" + 0.009*"worship" + '
  '0.007*"room" + 0.007*"building" + 0.007*"tent" + 0.007*"burn" + '
  '0.007*"tabernacle" + 0.006*"piece"'),
 (1,
  '0.012*"temple" + 0.012*"animal" + 0.011*"heal" + 0.010*"sick" + '
  '0.008*"bird" + 0.008*"blind" + 0.008*"sabbath" + 0.007*"angry" + '
  '0.007*"money" + 0.006*"shout"'),
 (2,
  '0.027*"boat" + 0.011*"star" + 0.009*"fish" + 0.008*"storm" + 0.006*"piece" '
  '+ 0.006*"night" + 0.005*"onto" + 0.005*"water" + 0.005*"jump" + '
  '0.005*"break"'),
 (3,
  '0.030*"earth" + 0.014*"heaven" + 0.012*"like" + 0.011*"story" + '
  '0.011*"water" + 0.010*"priest" + 0.009*"learn" + 0.009*"live" + '
  '0.009*"happen" + 0.009*"paradise"'),
```

The LDA is then used to classify the Bible stories with one of its 16 topics. Note that LDA results as displayed above generally require a human in the loop to make subjective judgments about what themes to call each topic. For this analysis, a manual review was done to assess each story and the topic that the LDA algorithm classified the story with. This helped to subjectively provide meaningful themes and to assess how intuitive the algorithm's classifications were. A review of each Bible story and its topic classification yielded the distribution of themes displayed in Figure 8. Note that even though the LDA detected 16 topics across the story corpus, it used only 11 of its topics to classify the Bible stories. The common pattern found in the 5 unused topics is that their keyword weights were insubstantial. A detailed mapping of the used LDA topic keywords to the themes defined using subjective judgement is found in Appendix I.

Figure 8: Distribution of themes across the Bible story corpus



A primary purpose of the manual review is also to assess how intuitive the topic assignment for each story is. Understanding how intuitive the topic assignments are provides a measure of the LDA algorithm's predictive power. As such, each theme displayed in Figure 8 is individually assessed to determine if it correctly represents the stories it is assigned to. For the record, a topic or theme assignment to a story is determined intuitive based on how much the story details match the assigned theme, as well as on contextual knowledge about the Bible stories. Table 2 displays the analysis of topic and theme assignments to determine how intuitive they are.

Table 2: Analysis of topic assignments to the Bible story corpus

Topic Id	Theme	Total Num of Story Assignments	Num Intuitive Assignments	Num Non-Intuitive Assignments	Percent Intuitive Assignments
14	Ethnic Wars and Conflict	59	35	24	59%
3	Creation and Heavenly Life	19	16	3	84%
0	God-Worship vs. Idol Worship	4	4	0	100%
1	Jesus at the Temple	3	3	0	100%
15	Healing and Miracles	3	3	0	100%
9	Motherhood and Progeny	2	2	0	100%
6	Persecution of Faith	2	2	0	100%
2	Trials at Sea	2	2	0	100%
13	Orators in Prison	2	2	0	100%
4	Young Mavericks	1	1	0	100%
11	Dinner Table Scenes	1	1	0	100%

The view from Table 1 shows high performance in topics that are assigned to a small number of stories in the corpus. That performance seems to decline as the number of stories that are assigned a topic increases; however, deeper analysis shows interesting patterns. The topic id 14 with the theme titled 'Ethnic Wars and Conflict' is assigned to 59 stories, 34 (58%) of which appear to be intuitive assignments. Looking deeper into the 24 stories that are non-intuitive assignments, however, some common patterns exist. 12 of the stories are old testament Bible stories while 12 of them are new testament stories.

While all of the 12 old testament stories are not about ethnic wars in and of themselves, 8 of them show to be part of a broader story arc of ethnic wars and conflict. An example is a story called 'Isaac Gets a Good Wife,' which the LDA algorithm classifies with the topic that maps to 'Ethnic Wars and Conflict.' The story is about the events surrounding Abraham's desire to find a wife for his son, Isaac. Abraham's servant travels to the land where Abraham's family is from and there, he stops by a well. At the well, a young woman offers to help the servant water his donkeys and she is eventually presented as Isaac's wife. These story details in and of themselves are not about an ethnic war. However, the story states that even though Abraham lived in a land called Canaan, he does not want his son to marry from the tribe of Canaanites because they worship false gods. The arc of an ethnic conflict exists in this story to some

degree even though there is no explicit war between tribes. Other stories in the corpus cover the explicit battles between Canaanites and Israelites. This story exhibits events that are shaped by the ethnic conflict. In Appendix II, is a list of the stories that are assigned this theme that appear to be non-intuitive, yet are part of the broader story arc of ethnic conflict.

Regarding the new testament stories that are assigned this theme, a different pattern exists. 10 of the 12 new testament stories are about moments in Jesus's life that one who is familiar with the Bible canon would categorize as the defining moments of Jesus's life. These stories are also displayed in Appendix II. These stories are notable because the corpus has multiple other stories about Jesus's life, for example, stories about him healing others or teaching others to do good. The LDA algorithm does not assign this theme to those stories, as if to distinguish the defining moments of Jesus's story arc from the others. To summarize the findings for the theme of 'Ethnic Wars and Conflict,' if one considers the intuitive topic assignments and the assignments that are part of broader story arc as part of the same group (titled 'Assignments Part of a Broader Story Arc'), we get the following results across both testaments:

	Total Num of Story Assignments	Num Assignments Part of Broader Story Arc	Percent Part of a Broader Story Arc
Old Testament - Ethnic Wars and Conflict	47	42	89%
New Testament - Defining Moments of Jesus' Life	12	9	75%

The question remains as to why the LDA classified the defining moments of Jesus's life with the same topic as the ethnic war stories. This is clearly a mismatch. Likewise, the keywords for this topic do not provide a reasonable explanation about why Jesus's defining moments are assigned to this theme. This is where the challenge of the explainability of a predictive algorithm occurs: A clear pattern in the classification of the new testament stories exists for this topic, however there is no logical explanation for it. A review of the mathematics behind the LDA algorithm is crucial, especially as one considers that its assignments are based on a bag of word representations that it creates for all of the documents used to train the algorithm.

4.3.2 Story Characterization with the BookNLP Supersense Variable

The goal for story characterization is to create a computational representation of deeper nuances in the stories beyond their themes. While a common theme groups multiple stories together based on topical similarity, story characterization helps to distinguish stories classified with the same theme based on patterns in the types of activities and entities encountered within their plots.

The BookNLP pipeline's *supersense* variable is used to do this. Appendix III displays all the noun and verb types that are possible values of the *supersense* variable. For example, the token 'run' when represented by the *supersense* variable is always tagged as a motion verb (verb.motion), while the token 'think' is tagged as a cognition verb (verb.cognition). The same distinctions are available for nouns. The token 'car', for example, is tagged as an artifact (noun.artifact). Given a story and its set of verb and noun types, a frequency score is calculated to reflect the proportion of each individual verb type to the total

number of verbs in the text, and likewise, the proportion of each noun type to the total number of nouns in the text. These frequency scores represent a set of 41 story characterization features that were generated for plot-level analysis.

Initial analysis of results of the BookNLP pipeline and the *supersense* variable showed that each story's verb and noun type frequencies differ enough to distinguish each story. For example, the stories 'Jonah and the Big Fish' and 'Shipwrecked on an Island' are classified by the LDA algorithm with the same topic, which maps to the theme, 'Trials at Sea.' The first story is about Jonah, whose ship is caught in a storm, and is eventually thrown into the sea where he finds himself in the belly of a big fish. The second story is about Paul, who is captured as a prisoner by Romans and trafficked by ship to Rome and on the journey, they are shipwrecked on an island due to a terrible storm.

While the classification of these two stories with the same LDA topic is intuitive, there are still nuances in the story details that largely differentiate both plots. Both stories involve dangers at sea, but Jonah's story has the particular distinction of being found in the belly of a large sea animal - an unimaginable event. The supersense frequencies calculated for both stories help to represent these distinctions computationally. For example, the top three verb supersense frequencies for Jonah's story are verb.communication (0.18), verb.motion(0.12), and verb.social (0.10) while the top three noun supersenses are noun.person (0.33), noun.animal and noun.phenomenon (0.05), and noun.artifact and noun.substance (0.05). The top three verb supersenses for Paul's story are verb.motion (0.17), verb.contact and verb.change (0.16), and verb.possession (0.08) while the top three noun supersense are noun.artifact (0.20), noun.person and noun.object (0.15), and noun.time (0.12). Appendix IV displays a comparison of both plots. As these and other reviewed examples show, the story characterization features represented by supersense frequencies allow for a more detailed computational representation of the stories at their plot-level.

4.3.3 Representing a Story's Emotion Trajectory

To create a feature that reflects each story's emotion trajectory, each sentence in an individual story is processed using the Google Natural Language API [14]. The API's sentiment analysis method is used to identify the prevailing emotional opinion in the sentence.

The sentiment analysis method provides two measures that are generally interpreted jointly to gauge emotional content: sentiment score and sentiment magnitude. The sentiment score ranges from -1 to 1, and reflects the emotional lean of the text, whether it is positive (closer to 1), negative (closer to -1), or neutral (closer to 0). The sentiment magnitude is a score between 1 and +inf and indicates the strength of the emotion detected in the text. Both scores were assessed for the Bible story sentences, with the aim of using them both to provide a rich representation of story emotion at the plot-level. However, in the case of this corpus, sentiment magnitude was found to consistently be the absolute value of the sentiment score. As such, the maximum possible value for a measure that can scale to infinity is 1 for this dataset, which does not provide significant information. The reason for this is the language used in *My Book of Bible Stories*. Since the stories are targeted towards children, the language is decidedly simple. Therefore, while a sentence in a story might have a positive sentiment score that the API can detect, the sentence will lack the use of words that help the API compute the degree of that positive emotion in a way that allows

for rich emotion analysis. A test of the API using more complex, adult language with emotionally charged words confirms this conclusion. Given these findings, the sentiment magnitude was removed from consideration.

The goal for the Native Storyteller is to have the ability to distinguish between the emotional trajectory of its stories, which represents the story's emotional impact on its audience. This requires knowledge of the change in emotion as the story progresses. For each story, an array of sentiment scores from the Google Natural Language API represents this appropriately since each sentence in a story is mapped to its corresponding sentiment score. The resulting challenge, however, is that each story varies in length, and therefore the array of sentiment scores for each story is of varying sizes. Downstream work to process each story with a machine learning algorithm would require that each instance of data has the same number of features. To achieve that, each story's set of sentiment scores is compared to a vector of the same size which is filled with ones - a one-vector. This vector represents a story without any variance in emotions: essentially, a dull story. Note that a zero-vector which represents the origin in a feature space was not used as the calculation required would result in null values. The story's emotion trajectory is then represented by the degree between its vector of sentiments scores and that of the dull story as follows:

$$\arccos = v_{story} \cdot v_{null} / |v_{story}| * |v_{null}|$$

Where acos a is the angle derived when the dot product of the sentiment score vector (v_{story}) and the dull story vector (v_{null}) is divided by the magnitude of both vectors. The values derived for the emotion trajectory have the range [0 - 2.3] in radian for this corpus or 0 - approximately 130 degrees.

4.3.4 Representing a Story's Sequence of Events

The last feature derived at the plot-level is the sequence of events in each story. Note that none of the features discussed at the plot-level so far represent the actual events in the story and the order in which they occur in a way that is contextually consistent. The ELMo algorithm described in Section 3.4.3 is used to provide neural embeddings at the document level for this feature representation.

For the ELMo algorithm to work, each story is tokenized and the series of tokens is fed into the algorithm for processing. By default, the resulting vector created by ELMo has 1024 features. Therefore, the feature representation for sequence of events is a dense (98 x 1024) matrix.

4.3.5 Summary: Plot-Level Feature Engineering

The four plot-level features generated and hitherto described are concatenated to represent one vector for each plot. The result is a (98 x 1077) matrix that computationally represents all 98 stories in the original data corpus. Concatenating the four features allows for the placement of each story in a feature space where the Native Storyteller can make associations between all stories in its catalogue to determine the best set of stories to display to a user at a given point in time.

In typical data science work, efforts are made to optimize and select the most predictive features for the task at hand (i.e.: making associations between all the stories), especially given the data set size and the number of features available. The task of association in this case is strictly to find similarities between stories in the corpus using an unsupervised clustering algorithm. There are limited options for feature selection in unsupervised classification - most feature selection algorithms require joint processing of the

independent variables with the desired outputs, or labels to determine themost predictive features. Nevertheless, some steps to determine if feature reduction is possible. The variance threshold algorithm is used to assess features for unsupervised learning by detecting and removing all features with low variance. By default, the algorithm works with a threshold of 0 as a standard (i.e.: it removes all features that have the same values in all data samples). A test using the variance threshold algorithm with its default threshold of 0 to analyze the theme, story characterization, and emotion trajectory features yielded no features without variance.

Additionally, correlation tests were performed to detect significant degrees of correlation amongst the non-ELMO features. A test for correlation between the theme and emotion trajectory features showed no correlation. The same finding stands in a test for correlation between the story characterization features and emotion trajectory features. These tests were not only performed for the purpose of feature selection, but also to identify any meaningful relationships between a story's emotional impact and the theme assigned to it, perhaps to find additional meaningful patterns across the themes.

4.4 Character-level Feature Engineering and Analysis

As displayed in Figure 6, the goal for feature engineering at the character-level is to generate features that represent each character's *persona* as well as the *sequence of events* in a story that specifically apply to that character (i.e.: things the character does, or things are done to or in association with the character). The same tools and methods that were used to create plot-level features are applied in this context, however they are translated differently for the character-level analysis.

4.4.1 Representing Character Persona with the BookNLP Supersense Variable

While a story's characterization feature is generated by calculating frequency scores for all noun and verb types that exist in the story (via the BookNLP *supersense* variable), a character's persona is generated by calculating verb-type frequencies for verbs that are strictly associated to that specific character. Take the following short story as an example:

> Jessica **ran** down the street while Shirley **followed** behind. Jessica **stopped** at the scene of the accident and **slammed** the window of the car with her bat to **save** the baby in it. Shirley **cried** as she **saw** the accident from a distance. Shirley **felt** the horror of the scene before her eyes.

These sentences clearly describe two people with very different personas, Jessica being more action oriented and Shirley being more perception or feeling oriented. To computationally represent Jessica's persona, the *supersense* verb-type values for each of the verbs that are associated only to her (emboldened in black) are summed up and aggregated as a proportion of the total count of verbs that are associated to Jessica in the story. A persona for Shirley on the other hand will use only the verbs emboldened in blue, which are associated only with her.

Table 3: Differentiating Character Personas Computationally

Jessica	
Verb	Supersense Verb Type

Shirley	
Verb	Supersense Verb Type

ran	verb.motion
stopped	verb.motion
slammed	verb.contact
save	verb.social

followed	verb.motion
cried	verb.emotion
saw	verb.perception
felt	verb.perception

Each character has four verbs associated with them uniquely. Jessica would score higher on motion verbs (2 out of 4, which corresponds to 0.50). Shirley on the other hand would score higher on perception verbs (also 0.50)

4.4.2 Representing the Character's Sequence of Events

To represent events in a story that are unique to a character, results from the BookNLP pipeline are used to isolate story sentences that explicitly mention the character's name or their associated pronoun used in reference to the character. The result is a dataset in which each row corresponds to a character, their name, and textual data on the events or dialogue in a story that specifically apply to the user. In other words, these are sentences that describe what the specific character is doing or saying.

Once each character's sentences are collated, they are processed using the ELMo algorithm for context aware embeddings of the story details. The result is a dense (43 x 1024) matrix representing each character's event details. Note that while *My Book of Bible Stories* has more than 43 characters, the 43 selected were the ones the BookNLP pipeline detected and tagged with character Ids which were used for the analysis.

4.4.3 Summary: Character-Level Feature Engineering

The two sets of character-level features are concatenated to represent one vector for each plot. The result is a (43 x 1067) matrix that computationally represents 43 key characters in the entire data corpus. These concatenated vectors are placed in the same character-level feature space, where the Native Storyteller can make associations between all characters in its stories to determine the best set of stories to display to a user at a given point based on characters of interest. Note that the idea of identifying a 'character of interest' will typically be derived by data available for the user, which is out of scope for this phase of analysis.

5. EVALUATION

The ultimate aim for placing all plot-level and character-level vectors in their own individual feature spaces is to enable the Native Storyteller to make associations across stories and across characters to determine what stories to display to a user at each point in time. Since this is highly dependent on the system's understanding of its stories, there needs to be an evaluation of how well the system computationally represents its stories and characters in relationship to one another. Therefore, success in this phase of research is determined by how intuitive the placement of the story or character vectors are in relationship to each other when they are in their respective feature space. In other words, if one is to assess the placement of vectors in relationship to each other, there should be some recognizable patterns and similarities. Additionally, after using a clustering algorithm to group stories or characters in the

feature space based on similarities in their features, there should be a recognizable rationale for the stories or characters that are placed in the same cluster.

5.1 Visualizing the Plot and Character-Level Feature Spaces

To assess the placement of vectors in their feature space, the vectors are visualized in a 2D plot and reviewed manually to identify meaningful patterns. Figure 9 displays the plot-level visualization which has some noticeable characteristics. Firstly, the stories are split spatially, with one group of stories positioned in the left zone of the plot, and the others to the right zone. In the right zone of the plot, there appear to be outliers (circled in red).

Figure 9: Plot-Level vectors visualized in a 2D plot

Note: Stories are labeled with their story IDs in the 2D visual plot

Looking deeper into these observations, a few questions are asked. Namely, if we assess the meanings of the stories, is there a clear difference between the stories on the left and right side of the plot that substantiates the large distance between their physical placement on the plot? Likewise, is there something that uniquely distinguishes the outliers on the right side of the plot from all other stories in the corpus?

5.1.1 Analyzing the Plot-Level Feature Space

Regarding the first question, a distinguishing observation of the ‘left zone’ stories from those on the right is that they are largely characterized by narration of an otherworldly being or occurrence in the affairs of daily life. Out of 49 stories that are placed on the left side of the plot, 16 of them (33%) exhibit this characteristic. See Appendix V for a zoomed in view of the stories on the left side of the plot. One might

argue that the Bible is filled with stories in which the earthly and otherworldly coexist, which is true. In fact, there are a small number of stories on the right side of the plot that exhibit coexistence between the earthly and the otherworldly. However, the language used for narration in the stories on the left more often explicitly emphasizes the presence of that otherworldly being or occurrence. The ‘left zone’ stories are narrated in a manner that expresses the peculiar nature of these occurrences. On the other hand, the narration in the few stories on the right side of the plot that show the coexistence does not have this emphasis. In fact, these stories are written in such a way that a reader or agent who lacks semantic understanding of the world can simplistically classify them as regular day to day occurrences. As such, when an NLP algorithm (like ELMo) processes the story, it cannot distinguish those occurrences as out of the ordinary.

Two stories are presented as an example - one from the left zone of the plot in Figure 9 and one from the right. To summarize, the ‘left zone’ story (Story A) is about a man Daniel, and his two friends who are thrown into a fiery furnace by the Babylonian king because they refuse to worship his god. In the furnace, they are accompanied by an angel that saves them from the flames. The ‘right zone’ story (Story B) is about a prophet, Balaam, whose donkey begins to trouble him when he attempts to fight against the Israelites - so much so that the two have a passionate exchange of words. A short snippet from each story is provided below:

Story A Snippet

The king looks into the furnace, and becomes very much afraid.'Didn't we tie up three men and throw them into the burning hot furnace?' he asks . 'Yes, we did,' his servants answer. 'But I see four men walking around in the fire,' he says. 'They are not tied up, and the fire is not hurting them. And the fourth one looks like a god.”

Story B Snippet

Balaam is very angry, and beats his donkey with a stick. Then Emmanuel causes Balaam to hear his donkey speak to him . 'What have I done to you so that you should beat me?' asks the donkey . 'You have made me look like a fool,' Balaam says . 'If i had a sword I would kill you!’ ‘Have I ever treated you like this before?' the donkey asks. 'No,' Balaam answers.

For a human reader, both of these stories are examples of extraordinary occurrences and should be categorized in much the same manner. In both stories however, the treatment in narration is notably different. In Story A from the left zone, there is first a description of fear at the sight of the otherworldly occurrence, followed by an expression of subverted expectations due to that occurence - three men are tied in the furnace and are expected to be burned, but instead the king sees four men unharmed in the fire and the fourth has the appearance of a god. In Story B, the narration simply focuses on the exchange of words between man and donkey. There is no explicit expression of shock at the sight of a speaking animal, and likewise no language used in the narration to emphasize this as peculiar. For an NLP algorithm processing these texts, the linguistic patterns and words used to describe the otherworldly occurrences in both stories, in addition to the different subject matter in both texts, would lead to plot vectors with highly different feature values, hence the distant placements of the two stories on the visual plot.

The second question regarding the visual plot has to do with the identified outliers. In reading these stories, it is clear that they are stories that are unique in the Bible. The occurrences in the stories have no precedents, and likewise have no similar events that follow throughout the text. Two stories, for example, are 'The Great Flood' and 'Crossing the Red Sea.' In the first story, all of earth is flooded with drowning waters that destroy humanity, and in the second, the Israelites cross through the Red Sea on dry land after God pulls back the waters. Stories like these are one-of-a-kind in the corpus, and as such, appear as outliers. Appendix VI lists the outlier stories and a summary of each.

5.1.2 Analyzing the Character-Level Feature Space

The character-level vectors discussed in Section 4.4 are also placed in their own feature space and visualized for assessment. As shown on Figure 10, a similar pattern occurs in that the characters are spatially split to the left and right of the plot. The distinguishing factor of the characters on the left side of the plot is that they are characters in active dialogue. In other words, their unique stories are told using dialogue between the character and others, as opposed to descriptive narration. Of 15 characters of interest on the left side of the plot, 9 of them (60%) are characters in active dialogue.

The characters on the right side of the plot on the other hand, are portrayed using descriptive narration. In these character snippets, the author describes the things the character does and even the things the character says or hears, but does not give a substantial speaking voice to the character. The snippets below provide examples to illustrate - Character A from the left side of the plot and Character B from the right side of the plot:

Character A: Solomon

Solomon is a teen-ager when he becomes king. One night Emmanuel says to him in a dream, 'Solomon what would you like me to give you?' At this Solomon answers 'Emmanuel my Emmanuel I am very young and I don't know how to rule.' Emmanuel is pleased with what Solomon asks. So he says 'Because you have asked for wisdom and not for long life or riches I will give you more wisdom than anyone who has ever lived.' A short time later two women come to Solomon with a hard problem. What will Solomon do? He sends for a sword and when it is brought he says, 'Cut the living baby in two and give each woman half of it.' Finally Solomon speaks, 'Don't kill the child.'

Character B: Eve

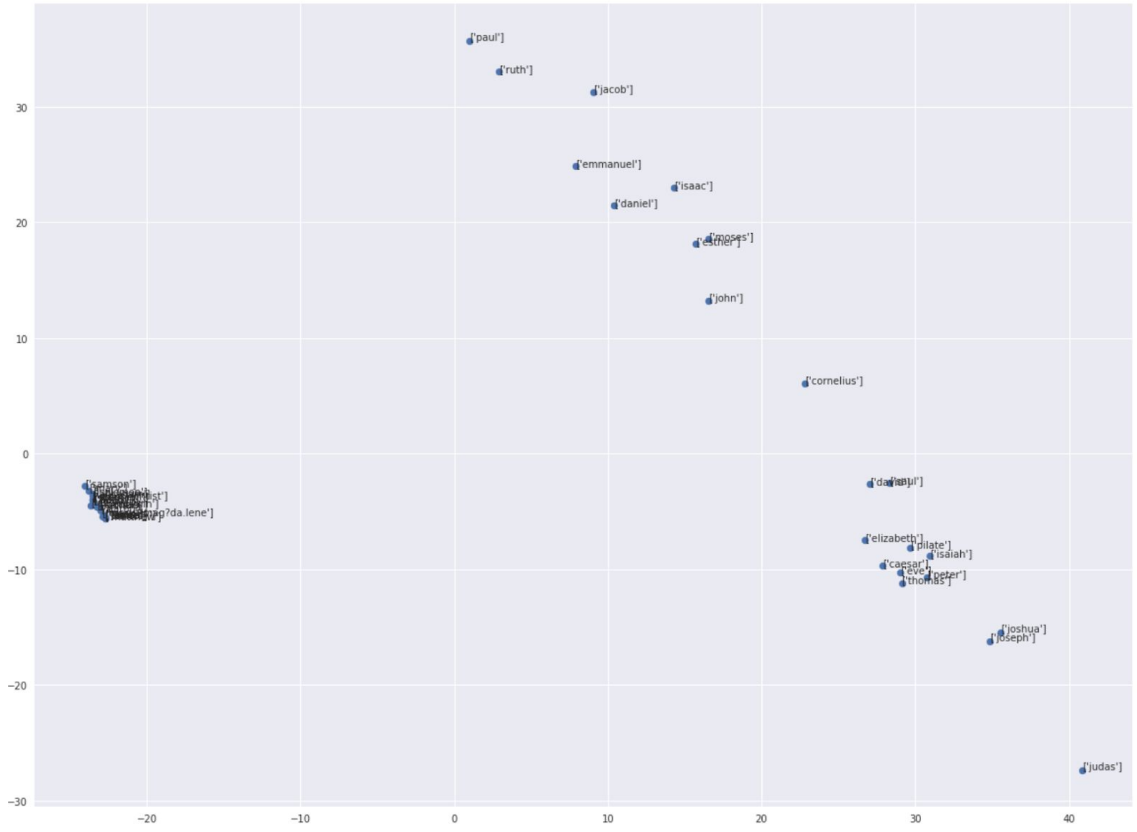
One day when Eve was alone in the garden, a snake spoke to her. It told Eve to eat fruit from the tree from which Emmanuel told them not to eat. Adam and Eve disobeyed Emmanuel and that is why they lost their beautiful garden home.

Of the 22 characters on the right side of a plot, only three of them (13%) have any speech attributed to them, and all three cases are very short snippets of speech embedded within heavy descriptive content. (Appendix VII displays a zoomed in view of characters on the left side of the plot).

The issue of spread is also of interest in understanding the placement of the character-level vectors on the 2D plot. The characters to the left of the plot are clustered closely together while those on the right are spread apart. An additional observation about the characters on the left side of the plot gives a probable reason for this. In each of the character snippets they are either having dialogue with God (Emmanuel in

this case) or an angel. In fact across the board, the name ‘Emmanuel’ is explicitly mentioned in the characters’ text snippets as a driving force for the characters’ actions or dialogue.

Figure 10: Character-Level vectors visualized in a 2D plot



5.2 Making Associations Between Plots and Between Characters

Now that the Native Storyteller has a computational representation of its stories and characters, it is able to achieve the task of identifying a set of stories for a user by making associations across the items in its catalogue. To test how this might be possible, a clustering algorithm is used to predict what stories or characters are closely associated with each other.

It is important that the algorithm used does not require a preset number of clusters to be defined ahead of training much like a K-means clustering algorithm would, for example. Rather, the primary task is for the system to discover the innate structures of the plot and character data, and identify clusters based on those structures. Given that the focus data set in this case is story data, there is no expectation that all stories must be related to each other. In fact, as seen in the visuals displayed for the plot-level and character-level vectors there are outliers. It is vital that the Native Storyteller is able to identify these outliers and treat them as such. Therefore, the DBSCAN algorithm (Density-Based Spatial Clustering of Applications with Noise) is used to identify clusters of similar story plots or characters.

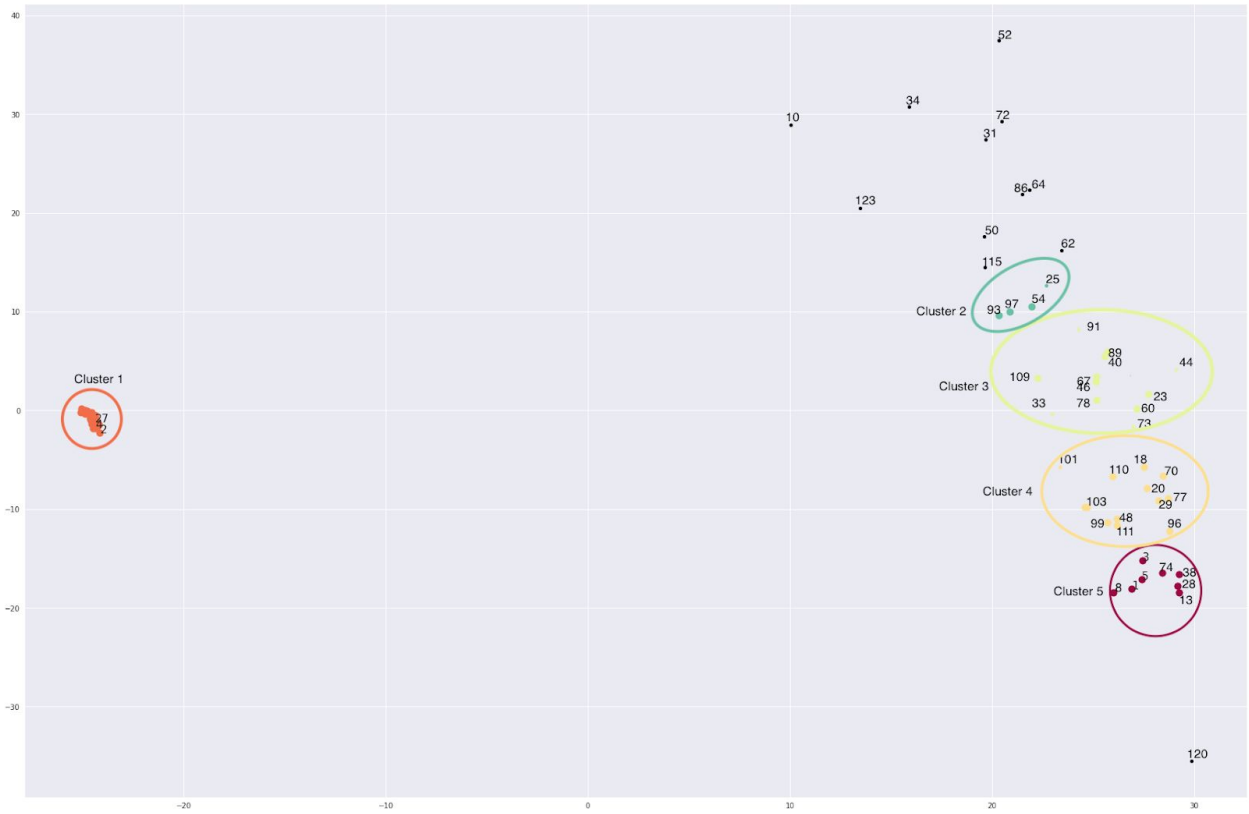
The DBSCAN algorithm thrives in clustering use cases where the spread of the data in question is not uniform. As alluded to in its name, the algorithm uses distance and density of the data points in a spatial

context to determine what points should be clustered together. The driving parameters for the algorithm are the epsilon neighborhood(ε) and min_samples (minimum number of data samples required to form a cluster). These two parameters work together to inform the algorithm of the size of the space it should be looking at at each point in time to determine what points to place in a cluster. If the epsilon neighborhood is too large, the result could be too few clusters. If epsilon is too small, the results could be too many clusters in the data.

By nature of how the DBSCAN algorithm works, the result is a plot that will show core points, border points, and outliers (noise). A core point in a specific cluster is the foundation of the cluster and is contained within the size of the algorithm's epsilon neighborhood. A border point is a point that is part of a cluster but not in the dense area of that cluster, and therefore not a core point. An outlier, which the algorithm designates as noise, is a point that is neither a core point nor a border point. In the visual plots using the DBSCAN algorithms in this context, core points and border points in the same cluster are given the same color, yet the core data points are physically larger than the border points. Outliers are left as black dots with no associated cluster.

For the plot-level vectors, an epsilon of 3 and min_samples size of 3 is used to train the algorithm. The results are shown in Figure 11.

Figure 11: DBSCAN Clustering for Plot-Level Vectors. Clusters encircled for emphasis



There are 5 clusters identified by the algorithm, and a 6th group of outliers that are not associated with any clusters. These are stories that the DBSCAN algorithm would identify as noise. In analyzing the clusters closely, a measure is adopted to reflect the level of alignment within the clusters; or in other words, how well each story in a cluster aligns with an established pattern identified in the cluster. This pattern can be based on similarity in theme, subject matter, characters or entities present etc. Table 4 displays the alignment score for each of the plot-level clusters displayed in Figure 11

Table 4: Cluster Alignment for Plot-Level Vectors

Plot-Level Cluster Alignment				
Cluster	Total Num Stories	Similar Pattern Observed	Num Stories Aligned	Alignment Score
1	48	Presence of supernatural beings or occurrences	16	33%
2	4	Narrations of Jesus's Life	2	50%
3	12	Stories centered around animals	4	33%
4	12	Narrations of Jesus's Life	6	50%
5	8	Building and development of people and cities	7	88%

For the character-level vectors, an epsilon of 3 and min_samples size of 2 is used to train the DBSCAN algorithm to accommodate for the smaller size and sparse nature of the data. The results are shown in Figure 12 and alignment between character-level vectors is shown in Table 4.

Figure 12: DBSCAN Clustering for Character-Level Vectors. Clusters encircled for emphasis

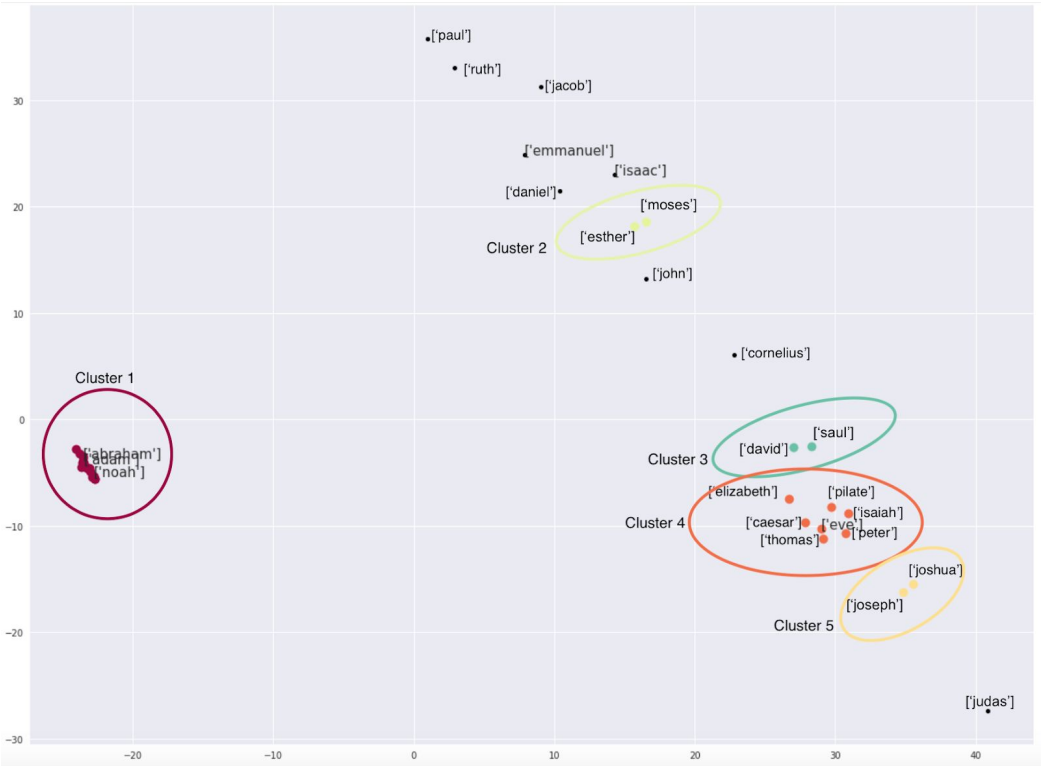


Table 4: Cluster Alignment for Character-Level Vectors

Character-Level Cluster Alignment				
Cluster	Total Num Characters	Similar Pattern Observed	Num Characters Aligned	Alignment Score
1	15	Characters in active dialogue	9	60%
2	2	Characters whose lives are threatened	2	100%
3	2	Characters in a duel	2	100%
4	7	Characters with drastic change in physical, mental, or emotional state	5	71%
5	2	Minor characters (mentioned in support of major characters)	2	100%

5.3 Discussion of Results

Looking at cluster alignment at the plot and character levels, it is clear that alignment at the character-level is much higher. In fact, there is only one cluster at the plot-level with strong cluster alignment (Cluster 5). A couple of questions arise from looking at the results. For one, what explains the large difference in alignment at the plot and character levels? Additionally what distinguishes the clusters with high alignment from those with low alignment?

A notable contributor in the case of the plot and character level difference is specificity. At the plot-level, the entirety of each story is used to generate the features discussed in Section 4.3. On the hand, the character-level features are generated only after all story details are paired down to extract character-specific information. The result is that the candidate data for each character has a more focussed subject matter while the candidate data for each plot covers a wider variety of information. In other words, when we read a sample plot in the corpus, we are more likely to detect a broader range of topics, while a sample character snippet would more likely express one strong element about a character's storyline. This difference translates into the feature generation process and manifests in the resulting vectors and clusters derived at the plot and character levels. There is much more generality or spread in the plot-level clusters in comparison to the character level clusters. So much so, that it becomes challenging to find a common thread that links all the stories that are assigned to the same cluster.

While there are continued improvements in text analysis and natural language methods, there are still limitations in developing a computational representation of a document in a way that appropriately reflects all the topics covered within the document. Perhaps, the challenge may not be to represent all information in a story, but rather to identify the optimal approach to pair down a story to its most relevant elements, and then build a computational representation of such elements. This still requires a human in the loop and a good amount of subjective judgement.

6. CONCLUSION

In reviewing the results of this phase of research, there are key areas where improvements can be applied. The first of them is with the data used for analysis. This corpus of Bible stories comprised only 98 samples of data, which is restrictive for finding a rich set of patterns in the placement of plots and characters in the same feature space, and likewise for finding meaningful patterns when clustering stories together based on similarities. Additionally, the content of the stories are decidedly simplistic. Each story involves language in the second person to engage the young reader, and then covers a very minimal amount of detail. This does not provide much to analyze and extract from each individual story. Given these qualities of the data used, there is an opportunity to use a larger and richer data set and apply the same methods described in Section 4 to get more interesting results.

There is also a need for well-defined methods to evaluate the features generated for the problem at hand. With storytelling being a newer area of research in machine learning, there is a lack of benchmarks to compare methods used to create a computational representation of story content. Some of the NLP methods used as rudiments to build the features have such benchmarks. For example, there are existing benchmarks one can use to compare ELMo against other language embedding models. However, there is a need for a method to evaluate the amalgamation of features generated at the plot and character levels in storytelling to determine what works best.

Also, as implied by earlier statements in this paper, this research was done with the perspective that the storytelling system functions in the context of an interaction with a human user. The currency that the storytelling system seeks is human attention, so that it may inform and guide it in a way that is conducive to achieving a specific goal. The target goal can be one of many options based on the specific human user at a given point in time or over a period of time. As a result, the storytelling system must have access to a dynamic model of the user to determine what storytelling experience to create. Such a model can be composed of attributes such as the user's goals, demographics, geographical location, usage patterns, preferences, etc. This is a dynamic challenge to solve, and one that can also present a rich dataset by which to evaluate the strength of the computational model for stories.

REFERENCES

1. Amatriain, X., & Basilico, J. (2012, April 6). Netflix Recommendations: Beyond the 5 stars (Part 1). Retrieved from https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429
2. Amatriain, X., & Basilico, J. (2012, June 20). Netflix Recommendations: Beyond the 5 stars (Part 2). Retrieved from https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5
3. Bamman, D., Underwood, T., & Smith, N. A. (2014). A Bayesian Mixed Effects Model of Literary Character. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* , *1*, 370–379. doi: 10.3115/v1/P14-1035
4. Blein, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*. Retrieved from http://www.jmlr.org
5. Brownlee, J. (2017, October 9). A Gentle Introduction to the Bag-of-Words Model. Retrieved from https://machinelearningmastery.com/gentle-introduction-bag-words-model/
6. How Netflix's Recommendations System Works. (n.d.). Retrieved from https://help.netflix.com/en/node/100639
7. Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. doi: 10.1016/j.eij.2015.06.005
8. Katz, B. (1997). *Proceedings of the 5th Riao Conference on Computer Assisted Information Searching on the Internet*. *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet*. Retrieved from https://groups.csail.mit.edu/infolab/publications/Katz-RIAO97.pdf
9. Li, Q., Li, Z., Wei, J.-M., Gu, Y., Jatowt, A., & Yang, Z. (n.d.). A Multi-Attention based Neural Network with External Knowledge for Story Ending Predicting Task. In *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 1754–1762). Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/C18-1149
10. Wikipedia. (n.d.). In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/List_of_biblical_names
11. Liu, X., He, P., Chen, W., & Gao, J. (2019). *Multi-Task Deep Neural Networks for Natural Language Understanding*. Retrieved from https://arxiv.org/pdf/1901.11504v2.pdf
12. Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., … Allen, J. (n.d.). A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories. In Proceedings of the 2016 North American Chapter of the ACL (NAACL HLT). Retrieved from https://arxiv.org/abs/1604.01696v1
13. Nackoul, D. D. (2010). Retrieved from http://groups.csail.mit.edu/genesis/papers/Nackoul 2010.pdf
14. Natural Language API Basics. (n.d.). Retrieved from https://cloud.google.com/natural-language/docs/basics
15. Nicholson, C. (n.d.). A Beginner's Guide to Attention Mechanisms and Memory Networks. Retrieved September 10, 2019, from https://pathmind.com/wiki/attention-mechanism-memory-network

16. Nicholson, C. (n.d.). A Beginner's Guide to Word2Vec and Neural Word Embeddings. Retrieved from https://pathmind.com/wiki/word2vec
17. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextual Word Embedding. *Proc. of NAACL*. doi: arXiv:1802.05365v2
18. Schank, R. (1990). *Tell Me a Story: A New Look at Real and Artificial Memory*. Chicago, IL: Northwestern University Press.
19. Srivastava, S., Roth, D., & Chaturvedi, S. (n.d.). Where Have I Heard This Story Before? Identifying Narrative Similarity in Movie Remakes. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Vol. 2, pp. 673–678). Association for Computational Linguistics. doi: 10.18653/v1/N18-2106
20. Tapaswi, M., Zhu, Y., Rainer, S., Torralba, A., Urtasun, R., & Fidler, S. (n.d.). Proceeds, Computer Vision and Pattern Recognition (CVPR). In *Proceeds, Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/CVPR.2016.501
21. Watch Tower Bible and Tract Society of Pennsylvania. (1978). *My Book of Bible Stories*. doi: https://www.jw.org/en/library/books/bible-stories/
22. Winston, P. H. (2016). *The Genesis Story Understanding and Storytelling System* (dissertation). Retrieved from http://groups.csail.mit.edu/genesis/papers/StoryWhitePaper.pdf
23. Winston, P. H. (2015). *International Workshop on Computational Models of Narrative* . *International Workshop on Computational Models of Narrative* . Retrieved from http://hdl.handle.net/1721.1/99102
24. Winston, P. H. (2011, December 15). The Strong Story Hypothesis and the Directed Perception Hypothesis (dissertation). AAAI Fall Symposium Series

APPENDIX

Appendix I - Mapping of LDA Topics to Themes

Dominant_Topic Id	Keywords	Theme
0	temple, wall, build, worship, room, building, tent, burn, tabernacle, piece	God-Worship vs. Idol-Worship
1	temple, animal, heal, sick, bird, blind, sabbath, angry, money, shout	Jesus and Temple Crowds
2	boat, star, fish, storm, piece, night, onto, water, jump, break	Trials at Sea
3	earth, heaven, like, story, water, priest, learn, live, happen, paradise	Creation, the Fall, and Heavenly Life
4	young, smartest, guardian, ashpenaz, train, healthy, problem, palace, abednego, vegetable	Young Mavericks
6	plague, furnace, abednego, throw, young, image, blood, river, worship, stick	Persecution of Faith
9	baby, give, mother, woman, birth, strength, princess, basket, hair, daughter	Motherhood and Progeny
11	meal, passover, lady, evening, leprosy, syrian, gift, catch, wash, flesh	Dinner Table Scenes
13	write, prison, letter, preach, prisoner, become, visit, send, kingdom, book	Orators in Prison
14	israelite, give, kill, time, look, year, happen, name, child, away	Ethnic Wars and Conflict
15	disciple, body, look, tomb, angel, appear, raise, woman, room, dead	Healings and Miracles

Appendix II - Analysis of Non-Intuitive Stories Assigned to the 'Ethnic Wars and Conflict' Theme

Bible Book	Story Id	Notes
Old Testament (Ethnic Wars)	17	Part of a broader story arc of ethnic conflict
	19	Non-match
	20	Non-match
	31	Part of a broader story arc of ethnic conflict
	37	Part of a broader story arc of ethnic conflict
	42	Part of a broader story arc of ethnic conflict
	45	Part of a broader story arc of ethnic conflict
	47	Part of a broader story arc of ethnic conflict
	53	Non-match
	60	Part of a broader story arc of ethnic conflict
	55	Non-match
	72	Non-match
New Testament (Defining Moments in Jesus's Life)	89	Jesus' birth predicted
	90	Jesus born in manger
	91	Wise men led by a north star to meet baby Jesus
	92	Jesus as a 12 year old stands out before old religious leaders
	93	Jesus is baptized and the voice of God is heard by all
	98	Jesus's cousin beheaded
	99	Non-match

	105	Jesus' betrayed by judas
	111	Pentecost: Jesus' spirit given to men
	114	Jesus' appearance from Heaven blinds Saul
	115	Non-match
	117	Non-match

Appendix III - List of noun and verb types from Book NLP pipeline

Supersense Verb and Noun Type	Description
noun.Tops	unique beginner for nouns
noun.act	nouns denoting acts or actions
noun.animal	nouns denoting animals
noun.artifact	nouns denoting man-made objects
noun.attribute	nouns denoting attributes of people and objects
noun.body	nouns denoting body parts
noun.cognition	nouns denoting cognitive processes and contents
noun.communication	nouns denoting communicative processes and contents
noun.event	nouns denoting natural events
noun.feeling	nouns denoting feelings and emotions
noun.food	nouns denoting foods and drinks
noun.group	nouns denoting groupings of people or objects
noun.location	nouns denoting spatial position
noun.motive	nouns denoting goals
noun.object	nouns denoting natural objects (not man-made)
noun.person	nouns denoting people
noun.phenomenon	nouns denoting natural phenomena
noun.plant	nouns denoting plants
noun.possession	nouns denoting possession and transfer of possession
noun.process	nouns denoting natural processes
noun.quantity	nouns denoting quantities and units of measure
noun.relation	nouns denoting relations between people or things or ideas
noun.shape	nouns denoting two and three dimensional shapes
noun.state	nouns denoting stable states of affairs

noun.substance	nouns denoting substances
noun.time	nouns denoting time and temporal relations
verb.body	verbs of grooming, dressing and bodily care
verb.change	verbs of size, temperature change, intensifying, etc.
verb.cognition	verbs of thinking, judging, analyzing, doubting
verb.communication	verbs of telling, asking, ordering, singing
verb.competition	verbs of fighting, athletic activities
verb.consumption	verbs of eating and drinking
verb.contact	verbs of touching, hitting, tying, digging
verb.creation	verbs of sewing, baking, painting, performing
verb.emotion	verbs of feeling
verb.motion	verbs of walking, flying, swimming
verb.perception	verbs of seeing, hearing, feeling
verb.possession	verbs of buying, selling, owning
verb.social	verbs of political and social activities and events
verb.stative	verbs of being, having, spatial relations
verb.weather	verbs of raining, snowing, thawing, thundering

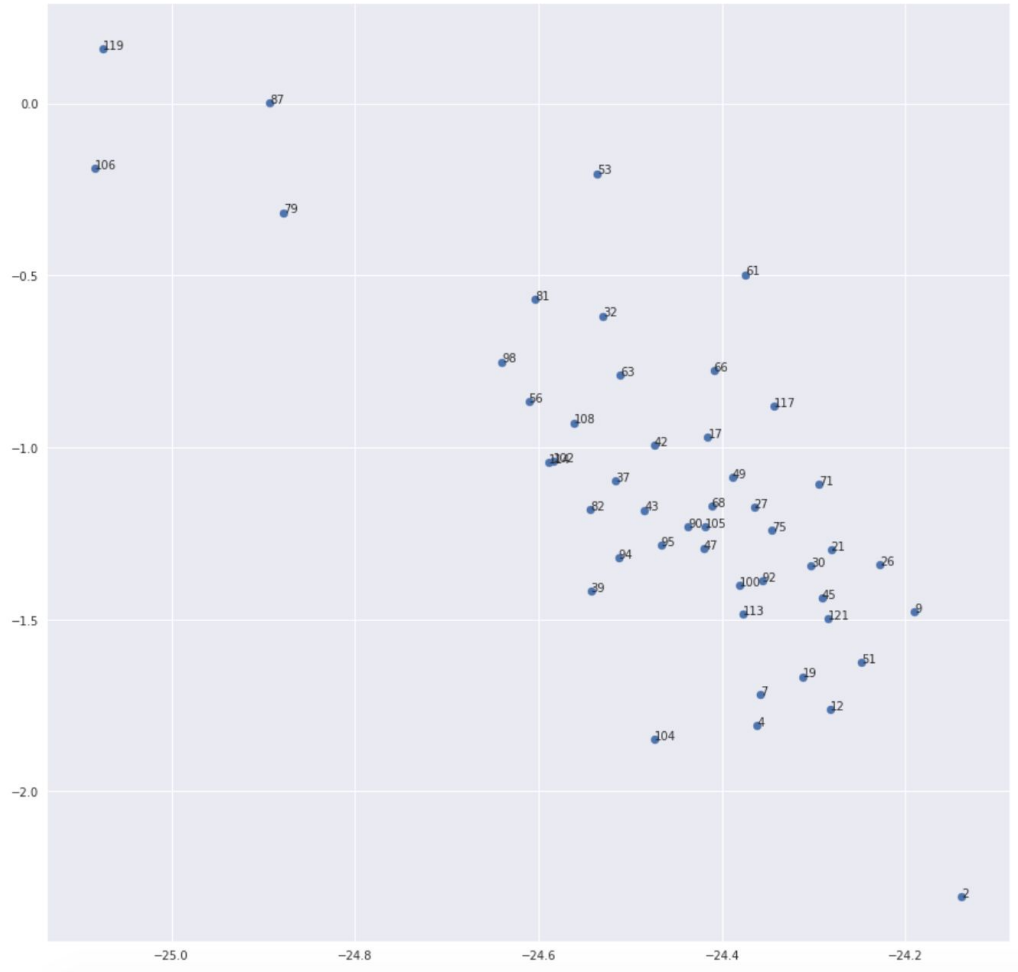
Appendix IV -Comparing *supersense* Frequencies of Two Stories Classified With the Same Topic

- Theme: Trials at Sea
    - Story 1 - Jonah and the Big Fish
    - Story 2 - Shipwrecked on an Island

Jonah and the Big Fish	**Shipwrecked on an Island**
Look at the man in the water. He is in a lot of trouble, isn't he? That fish is about to swallow him! Do you know who this man is? His name is Jonah. Let's see how he got into so much trouble. Jonah is a prophet of Emmanuel. It is not long after the death of the prophet Elisha that Emmanuel tells Jonah: 'Go to the great city of Nine.veh. The badness of the people there is very great, and I want you to speak to them about it.' But Jonah does not want to go. So he gets on a boat that is going in the opposite direction from Nine.veh. Emmanuel is not pleased with Jonah for running away. So He causes a big	Look! The boat is in trouble! It is breaking to pieces! Do you see the people who have jumped into the water? Some are already making it to shore. Is that Paul there? Let's find out what's been happening to him. Remember, for two years Paul is held prisoner in Caesarea. Then he and some other prisoners are put on a boat, and they start for Rome. When they pass near the island of Crete, a terrible storm hits them. The wind blows so hard the men can't steer the boat. And they can't see the sun during the day or the stars at night. Finally, after many days, those on board give up all hope of being saved.

storm. It is so bad that the boat is in danger of sinking. The sailors are very much afraid, and they cry out to their gods for help. Finally Jonah tells them: 'I worship Emmanuel, the Emmanuel who made the heaven and the earth. And I am running away from doing what Emmanuel told me to do.' So the sailors ask: 'What should we do to you to stop the storm?' 'Throw me into the sea, and the sea will become calm again,' Jo?nah says. The sailors don't want to do it, but as the storm gets worse they finally throw Jonah overboard. Right away the storm stops, and the sea is calm again. As Jonah sinks down into the water, a big fish swallows him. But he doesn't die. For three days and three nights he is in the belly of that fish. Jonah is very sorry that he did not obey Emmanuel and go to Nine.veh. So do you know what he does? Jonah prays to Emmanuel for help. Then Emmanuel makes the fish vomit Jonah out onto dry land. After that Jonah goes to Nineveh. Doesn't this teach us how important it is that we do whatever Emmanuel says? Bible book of Jonah.	Then Paul stands up and says: 'Not one of you will lose his life; only the boat will be lost. For last night an angel of Emmanuel came to me and said, "Don't be afraid, Paul! You must stand before the Roman ruler Caesar. And Emmanuel will save all those who are sailing with you."' About midnight on the 14th day after the storm began, the sailors notice that the water is becoming less deep! Because of fear of smashing into some rocks in the dark, they drop their anchors. The next morning they see a bay. They decide to try to sail the boat right up onto the beach there. Well, when they get closer to shore, the boat hits a sandbank and gets stuck. Then the waves begin to smash it, and the boat starts to break in pieces. The army officer in charge says: 'All of you who can swim jump into the sea first and swim ashore. The rest of you jump in after them, and grab some pieces from the boat to hold onto.' And that's what they do. In this way all 276 persons who were on the boat get to shore safely, just as the angel promised. The island is called Malta. The people are very kind, and they take care of those from the boat. When the weather gets better, Paul is put on another boat and taken to Rome. Acts 27:1-44; 28:1-14.
Verb Type Frequencies (Top 3): verb.communication - 0.18 verb.motion - 0.12 verb.social - 0.10 **Noun Type Frequencies (Top 3)** noun.person - 0.33 noun.animal - 0.06 noun.phenomenon - 0.06 noun.artifact - 0.05 noun.substance - 0.05	**Verb Type Frequencies (Top 3):** verb.motion - 0.17 verb.contact - 0.16 verb.change - 0.16 **Noun Type Frequencies (Top 3)** noun.artifact - 0.21 noun.person - 0.15 noun.object - 0.15 noun.time - 0.12

Appendix V - Left Side of the Plot-Level 2D Plot (Zoomed in)



Appendix VI - Outliers at the Plot-Level

Outliers - Stories with one-of-a-kind miracles or human interactions	
Story 10: 'The Great Flood'	A flood of waters suddenly began to fall from the sky for 40 days and 40 nights. All people on earth perish, save Noah and his family who are in the ark the God tells them build.
Story 31: 'The Burning Bush'	Moses is caught off guard when he sees a bush engulfed in flames of fire, yet it does not burn. There, he hears the voice of God speak to him
Story 34: 'Crossing the Red Sea'	Moses leads the Israelites out of Egypt and they find themselves trapped when they reach the Red Sea and the Egyptians are behind them in pursuit. God tells Moses to stretch his stick out over the Red Sea and the waters of the sea were parted and held up on both sides. Then the Israelites began to march through the sea on dry ground.
Story 52: 'Two Brave	A prophetess, Deb′o·rah and a young woman, Sis′e·ra are key women who help to destroy the Israelites

Women'	enemies in the days of judges.

Appendix VII - Left Side of the Character-Level 2D Plot (Zoomed in)