Abstract

In this thesis, we conduct a literature review on the application of two natural language processing techniques, topic modeling and named-entity recognition (character identification), on collections of literary fiction. These techniques allow us to efficiently identify the dominant themes in a text as well as the placement of named entities in relation to those themes. This process can be extended to the corpus as a whole to gauge the presence of themes across multiple works. We also investigate the use of this data in networks, which allow researchers to create human-readable maps of themes and entities across the corpus.

Contents

1 Introduction 2

2 Background and Motivation 3

3 Topic modeling 3
   3.1 Topics ................................................. 4
   3.2 Latent Dirichlet Allocation ............................... 5
   3.3 Topic analysis ........................................... 7

4 Named-entity recognition 8
   4.1 Part-of-speech tagging .................................. 8
   4.2 Hidden Markov Model .................................. 9
   4.3 Conditional Random Field .............................. 13

5 Network analysis 15
   5.1 Mathematical background .............................. 15
   5.2 Visualizations ......................................... 15

6 Summary and conclusion 18
1 Introduction

In traditional literary analysis, researchers manually scour a text or corpus for terms and phrases that they associate with topics or themes. While this can produce robust analyses, it may be tedious or impractical for large corpora. Additionally, human researchers may overlook some keyword pairings in their readings, leaving their analysis incomplete.

Several computational techniques have been developed since the mid-20th century to allow researchers to conduct broader investigations of literary texts. One of these techniques is a form of text analysis called topic modeling, which algorithmically identifies the recurring themes or topics in a corpus as well as the probability of each of these themes appearing in individual texts in the corpus. The researcher does not choose the topics in advance, only how many they want the topic modeling algorithms to return; instead, the algorithms generate topics according to word groupings throughout the text [Ble12]. John W. Mohr and Petko Bogdanov describe a topic informally as “the constellation of words that tend to come up in a discussion.” The researcher may then create an intuitive label for each of the generated topics based on the terms they contain [MB13]. In other words, topic modeling can be used to generate a “framework” for the researcher to conduct their subjective analysis [Ble12].

Another computational technique useful to literary analysis is named-entity recognition (NER), also called character identification, which identifies and labels entities—mainly proper nouns—in a text. To do this, the researcher must tag each word in the text with its corresponding part of speech, such as “adverb,” “particle,” “plural noun,” and so on. However, because many proper nouns contain more than one word, the researcher must first determine where the boundaries of named entities are through processes like BIO tagging, which “allows us to treat NER like a word-by-word sequence labeling task” with special tags to mark if words begin (B), are included in (I), or are outside (O) a named entity span. Once these entities are identified, the researcher can search through external reference materials to assign more meaningful tags as well, such as “location,” “organization,” and “person” [JM21]. When entities are fully tagged, the researcher can use their now-structured data to efficiently carry out literary analysis.

A network is a data structure made of two simple components: some number of vertices (nodes) and some number of edges each connecting one vertex to another. We can represent nearly anything as a network, including the relationships between objects of literary analysis. We can, for example, represent the texts in a corpus as nodes and topics shared between them as edges; or the characters in a single text as nodes and the interactions they have as edges; or any number of other applications. A character network describes the relationships between characters within a literary text and can be a useful tool in analyzing...
how closely linked various characters are to one another.

Researchers can implement topic models and named-entity recognition analyses using online tools like *spaCy*, an open-source Python library, or code they write themselves [spa17].

2 Background and Motivation

Studying literature through computational methods can expose the researcher to new ways of understanding the material. Topic modeling analysis can quickly identify thematic connections between separate texts in a corpus, such as Herman Melville’s *Typee* (1846) and *Moby-Dick* (1851). This information can be useful for “classification, novelty detection, summarization, and similarity and relevance judgments” in a variety of contexts [BNJ03]. Network analysis can also make it easier to visualize trends and discover new ones.

For example, the researcher may be interested in using a network to represent the interactions between characters in a literary text, such as *Moby-Dick*. They may have a sense from reading the material that certain characters interact more frequently with one another than others do, but cannot easily quantify these relationships. Using named-entity recognition, the researcher can distinguish characters in the text from surrounding material. They can then specify the parameters of an “interaction” between two characters: if they appear within the same paragraph, within a certain number of words or pages, within the same chapter, etc. Additional text parsing can identify when characters are speaking to each other in dialogue. The researcher can then represent each character as a node in a graph, and each interaction as an edge between two vertices. They can then “weight” the graph by assigning a numeric value to each edge, representing the number of interactions between each character throughout the text.

This “map” of interactions can be a useful visualization and may reveal that some characters are more closely linked than researchers doing manual analysis had realized.

3 Topic modeling

*Topic modeling* “describes a method of extracting clusters of words from sets of documents.” Elijah Meeks and Scott B. Weingart refer to it as “distant reading in the most pure sense” because of its large scope and “seductive but obscure results.” Indeed, topic models consider grammar and syntax very differently than human readers. To a topic modeling algorithm, a corpus contains only “buckets of words,” not sentences and paragraphs. Similarly, the *topic* that such an algorithm outputs is generally “a list of words, all apparently related yet in no discernible order” [MW12].

The semantic theory behind topic modeling is that words have meaning through their relations to other words, and that, in the context of computationally analyzing large corpora for themes, “relationality trumps syntax” [MB13].
This phrase invokes the relational model of linguistics, which was conceived of by Swiss linguist Ferdinand de Saussure (1857–1913). In his lecture *Course on general linguistics*, Saussure writes:

“... in language there are only differences. Even more important: a difference generally implies positive terms between which the difference is set up; but in language there are only differences without positive terms” [Sau16].

Vincent B. Leitch et al. paraphrase this quote as “neither ideas nor sounds exist prior to their combination” (2018) in *The Norton Anthology of Theory and Criticism*. To Saussure, repeatedly grouping a set of words together in speech or writing reflexively creates the individual meaning of each word; that meaning exists only in relation to every other word and not on its own. By extension, relational clusters of words are the most important characteristics to analyze when we seek to determine the meaning of a conversation [MB13].

As with words in human language, the individual words in a text have no inherent meaning to a topic modeling algorithm. Topic models can produce meaningful results because they quantitatively group words into topics—the researcher may then subjectively derive meaning from these groupings and explicitly label them with a name that reflects the most abundant words in each topic.

### 3.1 Topics

A topic can be defined as “a set of substantively meaningful coding categories” for words in a text [MB13]. Topics are literally composed of words with a high frequency of co-occurrence in the text; words that regularly appear near each other are considered related. For example, in *Moby-Dick*, topic $T_1$ may contain “whale,” “boat,” and “harpoon,” and topic $T_2$ may contain “fluke,” “spermaceti,” and “cetology.” We can assume that these collections of related words represent themes within a text.

However, it is important to note that these topics are not predefined; in topic modeling, the researcher only specifies how many topics the algorithm should return, not what the topics should be about. Indeed, topic modeling algorithms do not even label topics with a meaningful name; this task is left to the researcher, who reviews the most frequent words in the topic in order to subjectively determine how those terms are semantically linked. The topic can therefore be considered an “unobserved and latent” construct within a text [MB13]. An unlabeled topic is simply a collection of words. The topic modeling algorithm has no inherent understanding of the context behind the words it groups together, so it cannot produce meaningful labels for these “topics.”

In our example from *Moby-Dick*, we can subjectively label $T_1$ as “whaling” and $T_2$ as “whale biology” using our knowledge of the text’s contents and our familiarity with word definitions in English. A realistic topic model may contain substantially more topics, and is likely to analyze an entire corpus, not just one text.
3.2 Latent Dirichlet Allocation

Topic modeling uses a probabilistic technique called *Latent Dirichlet Allocation (LDA)* to identify topics within a corpus. While several such techniques exist, LDA is the most prominent. Introduced in 2003 by David M. Blei, Andrew Y. Ng, and Michael I. Jordan, LDA computationally models the way an author writes a set of texts within a corpus. LDA assumes that the author intends to convey certain topics (themes) in texts, each of which are defined by the words they choose to use in their writing. To an LDA model, a text in a corpus is merely an unordered “bag of words,” and therefore “theme (or topic) is a distribution over all observed words in the corpus, such that words that are strongly associated with the document’s dominant topics have a higher chance of being selected and placed in the document bag” [MB13]. The more common the word in a corpus, the more closely it is associated with some theme of that corpus. LDA assumes that the author writes their text by repeatedly picking some topic, then some word associated with that topic, over and over until they have written a complete text. Thus “The objective of topic modeling is to find
the parameters of the LDA process that has likely generated the corpus. […] in essence it is the task of reverse-engineering the intents of the author(s) in producing the corpus” [MB13].

Of course, human authors do not typically write literature by randomly selecting words from a dictionary, but the output between LDA-generated literature and real-world literature is similar: a set of texts each made of words that collectively convey certain themes. The LDA model can therefore be useful in quantitatively identifying what these themes are, and may reveal previously unknown or unidentified themes to human researchers.

Blei, Ng, and Jordan define LDA as “a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.” The texts within a corpus are therefore represented entirely statistically. They further describe it as “a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities” which collectively represent a text in a corpus [BNJ03].

LDA models the generation of a particular document \( w \) within a corpus \( D \) using the following parameters [BNJ03]:

1. Choose \( N \sim \text{Poi}(\xi) \), where \( N \) is the number of words in the document and \( \text{Poi}(\xi) \) is a Poisson distribution taking a positive real number \( \xi \) as a parameter. The symbol \( \sim \) denotes that \( N \) and \( \text{Poi}(\xi) \) are asymptotically equivalent: \( \lim_{n \to \infty} N / \text{Poi}(\xi) = 1 \). i.e. the LDA assumes that the length of the documents in the corpus follow the Poisson distribution.

2. Choose \( \theta \sim \text{Dir}(\alpha) \), where \( \theta \) is “a \( k \)-dimensional Dirichlet random variable,” \( \alpha \) is a “\( k \)-vector with components \( \alpha_i > 0 \)” and \( \text{Dir}(\alpha) \) is a Dirichlet distribution. \( \theta \) refers to the probability that a given word \( w \) in a text is classified under a given topic \( z \).

3. For each of the \( N \) words \( w_n \):
   (a) Choose a topic \( z_n \sim \text{Multinomial}(\theta) \), where \( \text{Multinomial}(\theta) \) is a multinomial distribution.
   (b) Choose a word \( w_n \) from \( p(w_n|z_n, \beta) \), a multinomial probability conditioned on the topic \( z_n \), where \( \beta \) is a \( k \times V \) matrix in which \( \beta_{ij} = P(w_j = 1|z_i = 1) \).

We can again use the conditional notation \( P(X|Y) \) to mean “the probability that \( X \) will occur, given that \( Y \) has already occurred.”

Blei et al. refer to LDA as “three-level” because the variables in this formula apply at different stages of LDA’s analysis. LDA refers to \( \alpha \) and \( \beta \) only once, on the corpus level; \( \theta_d \) once per document; and \( z_{dn} \) and \( w_{dn} \) once per word. In essence, to construct a text, the LDA repeatedly chooses a topic \( z_n \) and a word \( w_n \) from the probability \( P(w_n|z_n, \beta) \) until it has a document of length \( N \).
The LDA organizes these word probabilities in the matrix $\beta$. A particular cell in this matrix, $\beta_{ij}$, equals the probability $P(w_j = 1 | z_i = 1)$. $\theta$ ultimately represents a topic mixture such that:

$$P(\theta, z, w | \alpha, \beta) = P(\theta | \alpha) \prod_{n=1}^{N} P(z_n | \theta) * P(w_n | z_n, \beta)$$

Blei et al. represent the “probability of a corpus” $P(D | \alpha, \beta)$ with the following formula [BNJ03]:

$$P(D | \alpha, \beta) = \prod_{d=1}^{M} \int P(\theta_d | \alpha) \prod_{n=1}^{N_d} P(z_{dn} | \theta_d) * P(w_{dn} | z_{dn}, \beta) * d\theta_d$$

The three-level structure of the LDA means it can assign multiple topics to a single text. Because literary texts generally contain multiple themes, the design of the LDA model allows for more accurate results than techniques that can only assign a single topic to a text, such as the Dirichlet-multinomial clustering model [BNJ03]. This flexibility in topic assignment lends itself to analysis of large corpora because it allows the researcher to identify a variety of distinct connections between texts in a given corpus.

### 3.3 Topic analysis

Topic modeling does not automate literary analysis, it just makes it easier to do on large corpora. The role of the researcher in the topic modeling process is still critical. David Blei writes:

“the statistical models are meant to help interpret and understand texts; it is still the scholar’s job to do the actual interpreting and understanding. A model of texts, built with a particular theory in mind, cannot provide evidence for the theory” [Ble12].

If the researcher asks their algorithm to generate too few or too many topics, the word collections they receive may be incomprehensible or misleading, and subsequent literary analysis on these topics is likely to be uninteresting or difficult to apply. For the researcher’s analysis to be meaningful, they must be familiar enough with the corpus under investigation to determine whether the topics generated by the model are intelligible—if not, they must be able to experiment with different numbers of topics until they find a good balance between recognizability and novelty [MB13].

In our example from *Moby-Dick*, the hypothetical topics $T_1$, “whaling,” and $T_2$, “whale biology,” while representative of the text, are obvious to any reader. A somewhat less obvious topic $T_3$ may contain the words “bedfellow,” “cannibal,” and “hugging.” Without context, these may appear unrelated and therefore worthless. However, we may notice that these words describe the relationship between two characters, Ishmael and Queequeg, and provide a similar label.
For observations like this to have practical utility, the researcher must be able to use their data as the evidence for a specific analytic theory relevant to the topics generated by the model [MB13]. For example, a researcher could use a topic like $T_3$ to discuss the implications of amorous and matrimonial language between two male characters in a 19th-century novel.

Identifying connections like this is the natural next step of topic modeling. Meeks and Weingart write that “computational techniques like topic modeling...are not an upgrade from simplistic human-driven research, but merely more tools in the ever-growing shed” [MW12]. Other computational analytical methods, such as named-entity recognition, are utilized in a similar way.

4 Named-entity recognition

*Named-entity recognition (NER)* is a process by which the researcher may “find spans of text that constitute proper names and tag the type of the entity” [JM21] accordingly. Daniel Jurafsky and James H. Martin state that the four most common entity tags in most corpora are PER (person), LOC (location), ORG (organization), or GPE (geo-political entity), such as “Herman Melville,” “Nantucket,” “the Whalers' Association,” and “the United States Copyright Office,” respectively. To algorithmically determine which spans of words constitute named entities, the researcher must first tag each word with its corresponding part of speech, implementing a *sequence labeling* mechanism like a *Hidden Markov Model (HMM)* or *Conditional Random Field (CRF)* “to label each token in a text with tags that indicate the presence (or absence) of particular kinds of named entities” [JM21]. This ensures that all entities, even proper nouns spanning multiple words, are tagged appropriately.

4.1 Part-of-speech tagging

*Part-of-speech (POS)* tagging differentiates between adjectives, verbs, nouns, and other parts of speech in a text and is the first step in named-entity recognition. The tags used in part-of-speech tagging are defined in a *tagset*, which may have anywhere from 40 to 200 different tags for each part of speech. Researchers may use very large tagsets in order to “provide distinct codings for all classes of words having distinct grammatical behaviour.” However, simpler tagsets can reduce lexical redundancy by combining similar tags, such as different conjugations or tenses of the same verb. Depending on its complexity, a tagset may differentiate between singular and plural nouns, proper and improper nouns, and so on [MSM93].

For example, in 1991 Mitchell P. Marcus et al. analyzed the Penn Treebank, a corpus of over 4.5 million words, using a tagset containing 36 distinct part-of-speech tags. The tagset included the following tags to describe nouns: NN (singular improper or mass noun), NNS (plural improper noun), NNP (singular proper noun), and NNPS (plural proper noun). The tagging process is often
split into two phases, with tags first assigned algorithmically and then reviewed by researchers for accuracy [MSM93].

Words with only one definition are considered unambiguous, and words with more than one are considered ambiguous. For example, using the Penn Treebank tagset, the word “novel” can either be a singular improper noun NN (as in “fictional book”), or an adjective JJ (as in “new or interesting”). Part-of-speech tagging algorithms are designed to disambiguate words with multiple definitions based on the word’s context. The details of this process depend on the type of model used, but most mature part-of-speech tagging algorithms have an accuracy rate around 97%, which is comparable to that of humans. Part-of-speech tagging algorithms can also rely on basic statistical principles to simplify analysis. For example, an algorithm could identify the part of speech of an ambiguous word by always choosing the most common of the possible tags. Even this “baseline” model has an accuracy of about 92% [JM21].

Named-entity recognition relies on part-of-speech tagging to identify entities like characters. However, identifying an entity is complicated by the fact that many entities are not a single word: a rudimentary tagging algorithm may not recognize that the whale “Moby Dick” is a single entity, not two (“Moby” and “Dick”). Named-entity recognition is therefore a “span-recognition” problem: the researcher must identify the boundaries of each entity, not just which individual words are entities [JM21].

Techniques like BIO tagging are able to tag entities according to their type (person, location, organization, etc.) and also their boundaries. In BIO tagging, the algorithm labels words beginning an entity span with the tag “B,” words inside the span with “I,” and words outside any span with “O.” For example, combined with type identification, the name “Moby Dick” can be tagged as “B-PER” (“Moby”) and “I-PER” (“Dick”). An algorithm that can identify entities in this way is called a sequence labeler. Several such models exist, including the Hidden Markov Model and Conditional Random Fields [JM21].

4.2 Hidden Markov Model

The Hidden Markov Model (HMM) is a sequence labeling algorithm that can be used for named-entity recognition. We give the HMM an input sequence of words, and it outputs the most probable sequence of tags to associate with those words, disregarding all other other possible but less likely tag sequences.

The HMM analyzes sentences using a model called a Markov chain, which describes the probabilities of possible state transitions within a system. For example, given a random word in an unordered set of words, the Markov chain assigns to each other word a probability that it is the next term in the sentence. This system can be represented using a finite state machine, with words represented by states (nodes) and probabilities represented by transitions (edges) between states. The HMM assumes that only the current state is necessary to make predictions about a future state [JM21]. Depending on the use-case, this assumption can weaken the accuracy of the model considerably, although its predictions are generally still useful, especially when reviewed for errors by
human researchers.

The HMM uses the same probabilistic logic as a basic Markov chain, but it already knows the order of the words in a sentence it is given (the observed data). Instead, the HMM computes the probability that each potential part-of-speech tag of a word (the hidden data) is the correct one by making inferences from the previous word in the sequence. For example, suppose there are two words in an HMM, the current state \( X \) and the future state \( Y \). If \( Y \) can be unambiguously given just one part-of-speech tag, the transition from \( X \) to \( Y \) will be 1 (100%). However, if \( Y \) is ambiguous, there could be any number of transitions from \( X \) to \( Y \); their probability must add up to 1, such as 0.5 (PER), 0.4 (LOC), and 0.1 (ORG). Note that the model ignores all other states when calculating the probability of transition from \( X \) to \( Y \) [JM21].

An HMM uses two probabilities to make its assessment:

1. The transition probability \( P(t_i|t_{i-1}) \), which is the probability that a particular tag \( t_i \) will occur given the previous tag, \( t_{i-1} \).

2. The observation likelihood \( P(w_i|t_i) \), which is the probability that a particular word \( w_i \) is the word with which the given tag \( t_i \) is associated.

The transition probability \( P(t_i|t_{i-1}) \) is calculable because languages have definable grammatical structures that sentences written in them tend to follow. For example, many adjectives are immediately followed by nouns in English, while the opposite is rare. The HMM can determine this probability by looking at all the number of times a particular tag is followed by another. We can use the formula \( P(t_i|t_{i-1}) = C(t_{i-1}, t_i)/C(t_{i-1}) \), where \( C(t_{i-1}) \) is the number of times \( t_{i-1} \) appears in the corpus and \( C(t_{i-1}, t_i) \) is the number of times \( t_{i-1} \) is followed by \( t_i \) [JM21].

The next task of the HMM is to determine the observation likelihood \( P(w_i|t_i) \): note that this function returns the likelihood that an observed word \( w_i \) is the word associated with a hidden tag \( t_i \); it does not return the probability of the most likely tag \( t_i \) for the word \( w_i \). We can use the formula \( P(w_i|t_i) = C(t_i, w_i)/C(t_i) \), where \( C(t_i) \) is the number of times \( t_i \) appears in the corpus and \( C(t_i, w_i) \) is the number of times the tag \( t_i \) is associated with the word \( w_i \) [JM21].

Jurafsky and Martin give an example of the Wall Street Journal corpus in which \( t_i = VB \) (verb), \( t_{i-1} = MD \) (modal verb), and \( w_i = will \) (the word “will”; a specific modal verb). Therefore:

1. \( P(VB|MD) = C(MD, VB)/C(MD) = 10471/13124 = 0.80 \): there is an 80% chance that a modal verb is followed by a regular verb.

2. \( P(will|MD) = C(MD, will)/C(MD) = 4046/13124 = 0.31 \): there is a 31% chance that a randomly selected modal verb in the corpus is the word “will” [JM21].
The process of an HMM determining the most probable sequence of hidden variables (tags) for a dataset (sentence) is called decoding. It uses Bayes' theorem to calculate the most probable sequence of hidden tags for a corresponding sequence of observed words: $P(A|B) = P(B|A) * P(A) / P(B)$ for events $A$ and $B$, and where the probability $P(B) \neq 0$. We will use equations provided by Jurafsky and Martin to explain the way we generate this value [JM21].

We first assign the tag sequence $t_1 \ldots t_n$ to $A$, and the observation sequence $w_1 \ldots w_n$ to $B$. If we use the $\text{argmax}$ function on $P(t_1 \ldots t_n|w_1 \ldots w_n)$, which we expand using Bayes' theorem, we return the most likely tag sequence for a given observation sequence. We notate this value with the estimator $\hat{t}_{1:n}$ in our decoding equation:

$$\hat{t}_{1:n} = \text{argmax}_{t_1 \ldots t_n} \frac{P(w_1 \ldots w_n|t_1 \ldots t_n) * P(t_1 \ldots t_n)}{P(w_1 \ldots w_n)}$$

The probability of the observation data $P(w_1 \ldots w_n)$ by itself is a constant value, so we can remove it from the decoding equation:

$$\hat{t}_{1:n} = \text{argmax}_{t_1 \ldots t_n} P(w_1 \ldots w_n|t_1 \ldots t_n) * P(t_1 \ldots t_n)$$

We can make two additional observations about the probabilities on the right side of the decoding equation [JM21]:

1. $P(w_1 \ldots w_n|t_1 \ldots t_n) \approx \prod_{i=1}^{n} P(w_i|t_i)$. The probability of $w_1 \ldots w_n$ given $t_1 \ldots t_n$ is the product of every word’s observation likelihood $P(w_i|t_i)$ from $i = 1$ to $n$. Recall that to an HMM, the probability of some word $w_i$ being in the observation sequence $w_1 \ldots w_n$ depends only on the given tag $t_i$; it does not look at any other words or their tags.

2. $P(t_1 \ldots t_n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$. The probability of a particular tag sequence $t_1 \ldots t_n$ by itself is the product of the transition probability $P(t_i|t_{i-1})$ from $i = 1$ to $n$. Recall that to an HMM, the probability of some tag $t_i$ being in the tag sequence depends only on the previous tag $t_{i-1}$; it does not look at any earlier tags.

If we plug these products into the decoding equation, we get its most straightforward form:

$$\hat{t}_{1:n} = \text{argmax}_{t_1 \ldots t_n} \prod_{i=1}^{n} P(w_i|t_i) * P(t_i|t_{i-1})$$

We already know how to calculate a given transition probability $P(t_i|t_{i-1})$ and observation likelihood $P(w_i|t_i)$, and this equation simply requires us to combine the product of both and take the maximum probability in the resulting distribution. The decoding equation generated from Bayes’ theorem offers a convenient way to mathematically represent the sequence of hidden variables in the HMM [JM21].

Most HMMs perform tag decoding with a dynamic programming method called the Viterbi algorithm, “a probabilistic nonsequential decoding algorithm”
named for engineer Andrew Viterbi in 1967 [Vit67]. As input, the algorithm represents the probability distribution as a matrix, with rows representing each possible state $s_i$ and columns representing each observed variable (word) $o_t$. Thus each cell in the matrix, $v_t(j)$, represents “the probability that the HMM is in state $j$ after seeing the first $t$ observations and passing through the most probable state sequence $s_1, \ldots, s_{t-1}$.” The values in the matrix are “computed by recursively taking the most probable path that could lead us to this cell” [JM21]. The algorithm “performs $q_{k-1}$ comparisons each among $q$ path likelihood functions,” where $q$ is a cell in the matrix and $k$ is a possible path (or branch) from a given state [Vit67]. On an input length (number of observed words) $w$, the algorithm therefore runs in $O(wq^2)$ time.

Figure 2: A directed graph showing the possible tags an HMM can generate for an input string “Janet will back the bill.” Tags with a non-zero probability of applying to a particular word are shown in bright blue. Possible transitions from $t_i$ to $t_{i+1}$ are shown with arrows. The most probable transition is shown with a bold arrow [JM21].

Many transition and observation probabilities are substantially lower than the examples of 80% and 31% respectively from Jurafsky and Martin shown above. The chance of a randomly selected singular noun in the text *Moby-Dick* being the word “whale” may be high relative to some other nouns in the text, but there are so many nouns in English that the absolute value may be very low. The HMM cannot accurately predict what individual word will apply to a given tag in this way. However, there is a substantially higher probability that the word “whale” (noun) follows the word “white” (adjective) or the word “the” (determiner). When the HMM comes across the phrase “White Whale,” another name for the whale Moby Dick, it must be able to identify this as an entity.
If it cannot, the model has failed to produce accurate results. As with topic modeling, the researcher still plays a vital role in interpreting data produced by an HMM. They must be familiar enough with their corpus to be able to identify any absurd or incomplete results and adjust their model accordingly for future analyses.

Hidden Markov Models also often have difficulty labeling unfamiliar words in part-of-speech tagging, especially proper nouns, acronyms, slang, and “loan words” adopted from other languages, all of which come into use rapidly and may not exist in older corpora. Although we can modify an HMM to better recognize characteristics of certain types of words, such as adding case-sensitivity to better recognize proper nouns, this process can be difficult. An alternative algorithm called a **Conditional Random Field (CRF)** may implement this functionality more easily instead [JM21].

### 4.3 Conditional Random Field

A **Conditional Random Field (CRF)** is another sequence labeling model that we can use in named-entity recognition. Most natural language processing applications with CRFs use a variety called a *linear chain CRF*, which calculates the most probable tag sequence of an input string with a different approach than an HMM. While an HMM seeking to identify a tag $P(Y|X)$ must use Bayes’ theorem to first calculate $P(X|Y) \ast P(Y)$, a CRF directly computes $P(Y|X)$ by identifying predefined *features* of words and state transitions and computing the probability of these features representing the output sequence. The CRF calculates this probability using a *feature function* $F$, which “maps an entire input sequence $X$ and an entire output sequence $Y$ to a feature vector” [JM21]. A *feature vector* describes the characteristics of these sequences and the relationships between them. The CRF does not calculate the probability of each possible tag $y_i$ for each word $x_i$ in the chain; instead, it uses the *local* values of the feature vector to calculate a *global* probability of the tag sequence [JM21].

A *local feature* $f_k$ at position $i$ in tag sequence $Y$ contains information about a particular word, tag, or word-tag association, such as whether the word contains a particular prefix, which tag it is given, etc. An example feature is $1\{x_i = \text{whale}, y_i = \text{NN}\}$ for the word “whale” and its tag “NN” (noun). The $1\{x\}$ notation represents the truth of the feature as a binary value: “If $x$ is true, return 1; else, return 0” (this numeric value is what allows features to be easily represented in a feature vector). We can represent a local feature with the feature function $f_k(y_{i-1}, y_i, X, i)$, which, “in a regular multinomial logistic regression, can be viewed as a function of a tuple $[x$ and $y]$.” In this function, $k$ signifies which feature in a list of features is being applied to the function, $y_i$ is the current tag, $y_{i-1}$ is the previous tag, $X$ is the current word, and $i$ is the index [JM21].

A *global feature* is the aggregate of each local feature $f_k$ in a sequence. We can represent a global feature with the function $F_k(X, Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$ [JM21].

The CRF mapping function is a log-linear model, meaning it expresses some
elements in the form $\exp(c + \sum kw_k F_k(X, Y))$, where $\exp$ refers to the exponential function $\exp(x) = e^x$, $w_k$ is the weight of a feature $F_k$, and $F_k(X, Y)$ is the global feature equation shown above, with $X$ as the input sequence and $Y$ as the tagged output sequence (out of all possible output sequences $Y_p$). The equation to calculate $P(Y|X)$ is as follows [JM21]:

$$P(Y|X) = \frac{\exp(\sum_{k=1}^{K} w_k F_k(X, Y))}{\sum_{Y' \in Y_p} \exp(\sum_{k=1}^{K} w_k F_k(X, Y'))}$$

In a linear chain CRF, the algorithm can only use $y_i$ and $y_{i-1}$ as reference data; a general-purpose CRF does not have this constraint and can use any computed tag in the sequence to make its calculation. The probabilities returned by the CRF can be more accurate with this additional data. However, implementing a general CRF can be very complex, so most named-entity recognition algorithms use linear chain CRFs [JM21].

The researcher defines which individual features to use in their analysis in a feature template, which may include information like a word’s length, shape, morphology, and tag. For example, the feature template $\langle y_i, x_i \rangle$ always produces the values associated with $y_i$ and $x_i$ for some feature $f_k$, such as $f_{123}: y_i = NNS$ and $x_i = harpoons$. Other features useful in part-of-speech tagging include those identifying capital letters, prefixes, suffixes, and more information that can provide grammatical or semantic context to a word in a sequence. The features used in named-entity recognition tasks may include those for a POS tagger, as well as external references defining various types of proper nouns. For example, a feature template may specify that a word must appear in a gazetteer (listing names of locations), and another may use census data (listing names of people). This information can help the CRF algorithm determine if the entity it finds is a person or a place [JM21]. The researcher can combine multiple tagging techniques when analyzing their corpus to ensure the most accurate results. For example, BIO labels can be used alongside census data and capitalization features to identify proper names in a corpus.

In order to decode sequences, many linear chain CRFs also use the Viterbi algorithm, explained in Section 4.2. § Hidden Markov Model. However, while in an HMM the cell $v_t(j)$ is defined by the maximum of probabilities like $P(s_j | s_i)$ and $P(\alpha_t | s_j)$, the CRF takes:

$$\max_{i=1}^{N} v_{t-1}(i) \sum_{k=1}^{K} w_k f_k(y_{t-1}, y_t, X, t)$$

In other words, it sums local feature functions into a single global probability for the entire sequence [JM21].
5 Network analysis

5.1 Mathematical background

A network (graph) is a set of vertices (nodes), denoted \( V \), connected by a set of edges (transitions), denoted \( E \). Vertices can be connected to zero, one, or multiple edges. Networks can be directed, meaning we can traverse from vertex \( A \) to vertex \( B \) but not from vertex \( B \) to vertex \( A \), unless there is a third node \( C \) such that there exists edges \( (B, C) \in E \) and \( (C, A) \in E \). We use the format \((V_1, V_2)\) to refer to a transition from vertex \( V_1 \) to vertex \( V_2 \). Networks can also be weighted, meaning we can assign a numerical value to an edge which can be useful in mathematical computations [New18]. In the context of literary analysis, for example, we can create an undirected weighted graph of character interactions, where the weight of each edge refers to the number of interactions between two characters, represented by nodes.

5.2 Visualizations

![Network visualization](image)

Figure 3: A visualization of character interactions in Shakespeare’s *Hamlet* via an unweighted, undirected graph. Each character is a vertex, and each interaction is an edge [Mor21].

Networks allow us to visualize the information extracted from a literary corpus. Franco Moretti writes that this process of visualization, or “the possibility
of extracting characters and interactions from a dramatic structure, and turning them into a set of signs that I could see at a glance, in a two-dimensional space” is his greatest takeaway from implementing network theory in his analysis of literature [Mor21]. These networks can reveal implicit characteristics of a text’s structure and plot by laying out such character interactions across the story as a whole.

In his publication “Network Theory, Plot Analysis,” Moretti analyzes the play Hamlet (c. 1601) by William Shakespeare as well as the novels The Story of the Stone (c. 1791) by Cao Xueqin and Our Mutual Friend (c. 1865) by Charles Dickens. He begins with a network of interactions between characters in Hamlet, where an interaction is defined as any instance of dialogue. Moretti does not apply a weight to these edges, a numerical quantity that signifies the importance of a particular interaction. This means that a discrete conversation containing one line of dialogue is considered equivalent to a conversation between those same characters containing 100 lines. He also does not add direction to the edges, meaning his network does not distinguish between character A speaking to character B and the opposite. However, he notes that using more robust modeling techniques is feasible for language processing networks in general. Other approaches may even link characters who are physically present at the same time in the text, such as during the same scene [Mor21].

Networks have no inherent variable representing time, so an interaction at the beginning of a text is generally represented the same way as one at the end (unless an analysis is designed to simulate interactions temporally). Moretti states that this process “mak[es] the past just as visible as the present.” It abstracts the text, reducing it into a model; it therefore deconstructs the linear progression of the literary text altogether. But Moretti states that “a model allows you to see the underlying structures of a complex object. It’s like an X-ray: suddenly, you see the region of death [within Hamlet], which is otherwise hidden by the very richness of the play.” In particular, the way we understand centrality in a text can be deeply influenced by the character interactions shown in a network [Mor21]. Networks can reveal that supposedly minor characters are more crucial than otherwise thought, especially when a character interaction network is paired with additional networks showing plot elements such as deaths, in the case of Hamlet, or marriages, in the case of Jane Austen’s Pride and Prejudice (1813).

Moretti further observes that once a network is generated, it becomes possible to easily experiment with hypothetical situations: he uses the example of removing central characters from a network outright, just “to see what happens.” He refers to the effect of removing certain nodes from a network as a function of its “stability”; the centrality of certain characters can unbalance a text by removing possible paths from characters more than one degree of separation away, although the removal of some nodes can produce more unstable results than others despite each having similarly “central” qualities. To explain this phenomenon, Moretti discusses the concept of network clustering, or of certain characters being situated in a “denser” part of the network. According to Mark Newman, “If vertex A is connected to vertex B and vertex B to vertex
C, then there is a heightened probability that vertex A will also be connected to vertex C. In the language of social networks, the friend of your friend is likely also to be your friend.” In a dense part of a network, node removals are less likely to affect the stability of the network as a whole because other nearby nodes are already interconnected with one another. The opposite behavior is also true: the sparser an area of a network, the more a particular removal destabilizes it [Mor21]. Literary analysts may use this quantitative information to make more advanced analyses about the meaning of a particular text (including the significance of certain characters to its major themes), the structural preferences of an author, or broader historical trends in literature.

Moretti also investigates what he calls the symmetry of character networks in literary texts: if we can visualize character interactions at the beginning, middle, and end of a text in a symmetrical map. To do this, he compares the networks of *The Story of the Stone* and *Our Mutual Friend*. He concludes that while traditional analysis of Western novels has paid little attention to the symmetry of character interactions throughout a text, Dickens’ *Our Mutual Friend* shows a surprising amount of “dualism” in its structure, which he posits is an effect of characters often appearing in pairs, and further that this dualism is magnified by the small number of characters in the text: “with few characters, symmetry seems to emerge by itself, even in the absence of an aesthetics of symmetry.” In contrast, Moretti also quotes Andrew Plaks on the structure of traditional Chinese literature: “the overall sequence of chapters (...) will add up to a round and symmetrical number, typically 100 or 120. The pronounced sense of symmetry (...) provides the ground for a variety of exercises in structural patterning. Most noticeable among these is the practice of contriving to divide an overall narrative sequence precisely at its arithmetic midpoint, yielding two great hemispheric structural movements.” But Moretti’s network analysis of *The Story of the Stone* shows far less symmetry in character interactions than such a reading as Plaks’ might suggest. Moretti also attributes this to the number of characters in the text: “if with few characters symmetry emerges almost by itself, with many characters it becomes implausible.” However, he also notes that some texts exhibit symmetric qualities on a local scale, in a particular chapter, but not on a global scale, or vice versa, due at least in part to specific cultural practices carried out by characters in the texts [Mor21].

A network visualizing the results of a named-entity recognition analysis has a similar structure to that of a topic modeling analysis. For example, we can represent texts in the corpus as nodes in our graph and topics shared between those texts as edges between nodes. We can also weight the edges to signify the significance of the connection. The resulting visualization can provide immediate insight into thematic connections between texts in the corpus; some nodes may have several outgoing edges; while others may have few or none.

The researcher can represent or manipulate the quantitative data from topic modeling and named-entity recognition in an effectively infinite number of ways, especially with appropriate weighting. Networks offer a “map” with a level of accessibility unmatched by formulas and tables of obtuse statistics. Simplified network visualizations can be useful in representing analytical data both to gen-
eral audiences as well as to academics familiar with complex mathematical and statistical modeling representations. However, overly complicated networks can be similarly difficult to understand, and poor variable selection in a network can produce meaningless results. As with topic modeling and named-entity recognition, proper network analysis requires specific, informed analytical decisions from the researcher. When used wisely, it is a highly useful tool for literary analysis.

6 Summary and conclusion

This literature review has investigated the natural language processing techniques of topic modeling and named-entity recognition on corpora of literary texts, as well as the application of data derived from these models in network analysis. We have explored the probabilistic models used in both computational areas, Latent Dirichlet Allocation, Hidden Markov Models, and Conditional Random Fields, and have summarized the utility and pitfalls of these various approaches. We conclude that these techniques can be extremely useful in analyzing literary corpora on a large scale with the caveat that, like all work in the digital humanities, computational linguistic analyses rely heavily on the researcher to structure their analysis in useful ways and interpret data accurately.

7 Future directions

Researchers can analyze literary texts using the methods described in this literature review using a natural language processing library like spaCy. This module includes support for part-of-speech tagging, named-entity recognition, and other techniques, and supports analysis of a variety of real-world languages. The researcher could also use a similar library or create their own models from scratch.

An example of a literary corpus is that of Herman Melville (1819–1891), an American author best known for his 1851 epic novel Moby-Dick. He published nine novels during his lifetime, and one more was published posthumously. In addition, Melville wrote numerous short stories that he published individually or in collections. The researcher could easily analyze his 10 published novels and two short story collections with spaCy, shown in the following table:
<table>
<thead>
<tr>
<th>Title</th>
<th>Published</th>
<th>Type</th>
<th>Length (words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typee: A Peep at Polynesian Life</td>
<td>1846</td>
<td>Novel</td>
<td>106,675</td>
</tr>
<tr>
<td>Omoo: A Narrative of Adventures in the South Seas</td>
<td>1847</td>
<td>Novel</td>
<td>101,186</td>
</tr>
<tr>
<td>Mardi, and a Voyage Thither</td>
<td>1849, March</td>
<td>Novel</td>
<td>196,804</td>
</tr>
<tr>
<td>Redburn: His First Voyage</td>
<td>1849, November</td>
<td>Novel</td>
<td>117,915</td>
</tr>
<tr>
<td>White-Jacket; or, The World in a Man-of-War</td>
<td>1850</td>
<td>Novel</td>
<td>138,613</td>
</tr>
<tr>
<td>Moby-Dick; or, The Whale</td>
<td>1851</td>
<td>Novel</td>
<td>212,131</td>
</tr>
<tr>
<td>Pierre; or, The Ambiguities</td>
<td>1852</td>
<td>Novel</td>
<td>151,617</td>
</tr>
<tr>
<td>Israel Potter: His Fifty Years of Exile</td>
<td>1854–55</td>
<td>Novel</td>
<td>64,485</td>
</tr>
<tr>
<td>The Confidence-Man; His Masquerade</td>
<td>1857</td>
<td>Novel</td>
<td>92,849</td>
</tr>
<tr>
<td>Billy Budd, Sailor (An Inside Narrative)</td>
<td>1924</td>
<td>Collection</td>
<td>30,166</td>
</tr>
<tr>
<td>The Apple-Tree Table, and Other Sketches by Herman Melville(^a)</td>
<td>1850–56 (texts), 1922 (collection)</td>
<td>Collection</td>
<td>53,139</td>
</tr>
<tr>
<td>The Piazza Tales by Herman Melville</td>
<td>1853–56 (texts), 1856 (collection)</td>
<td>Novel</td>
<td>79,361</td>
</tr>
</tbody>
</table>

\(^a\)In addition to fictional short stories, *The Apple-Tree Table, and Other Sketches by Herman Melville* includes a review of several works by author Nathaniel Hawthorne. This item can be omitted from the corpus as it is not strictly literary.

### 7.1 spaCy

To do named-entity recognition, the researcher can use spaCy’s *tokenization* feature to split an input text into words and punctuation, which makes it easy for algorithms to parse. Once the text is properly formatted, they can tag each word using spaCy’s part-of-speech tagger. The researcher can then apply spaCy’s statistical named-entity recognition models to determine which nouns are proper. They can then use functionality provided by spaCy to implement a “knowledge base” of real-world entity names in order to categorize entities according to their nature as people, locations, objects, and so on [spa17].

To do topic modeling, the researcher can use spaCy’s named-entity recognition pipeline to format the text in a useful way. They can then use custom code or an additional library such as Gensim to run the topic models themselves using Latent Dirichlet Allocation [Kel19].

spaCy’s online documentation contains more robust explanations of each of the features the software offers. With such a tool available for free, carrying out natural language processing tasks on large literary corpora is substantially easier than manual methods [spa17].
References


