Topic Modelling via Community Mining of Term Co-occurrence Networks

by

Eric Austin

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Topic modelling seeks to uncover the conceptual and thematic content of collections of documents. These topics can be used as features for document indexing and classification. However, topic models are increasingly important as tools of applied research. As we seek to develop agents capable of having real conversations with humans, topic models are needed to control topic drift and guide the conversation. Unfortunately, the most popular topic models in use today do not provide a suitable topic structure for these purposes and the state-of-the-art models based on neural networks suffer from many of the same drawbacks while requiring specialized hardware and many hours to train.

We take a fundamentally different approach to topic modelling. Our algorithm, Community Topic, is based on mining communities of terms from term-occurrence networks extracted from the documents. In addition to providing interpretable collections of terms as topics, the network representation provides a natural topic structure. The topics form a network, so topic similarity is inferred from the weights of the edges between them. Super-topics can be found by iteratively applying community detection on the topic network, grouping similar topics together. Sub-topics can be found by iteratively applying community detection on a single topic community. This can be done dynamically, with the user or conversation agent moving up and down the topic hierarchy as desired.

We evaluate Community Topic against two contenders. We find that our algorithm detects topics with the highest coherence as measured by two stan-

dard automated metrics. Our algorithm has the fastest run time and detects topics in few seconds with no specialized hardware required. It is hyperparameter free and can detect topics at multiple scales. It finds coherent sub- and super-topics at multiple levels. This makes Community Topic an ideal topic modelling algorithm for both applied research and practical applications like conversational agents.

Preface

The work done in Chapter 4 has been accepted for publication at the International Conference on Computational Linguistics 2022 under the title "Community Topic: Topic model inference by consecutive word community search."

To my parents

For always supporting and believing in me.

You shall know a word by the company it keeps.

– J.R. Firth, 1957.

Acknowledgements

I would like to thank my supervisor, Osmar Zaïane, for his kind guidance. His feedback and support made this work possible. I would also like to thank Christine Largeron and my colleagues on the Meerkat and ANA teams. No work is possible in isolation and the tremendous amount of learning and development that I have experienced over the course of my program is thanks to my collaboration and interaction with them.

Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

- LDA - Latent Dirichlet Allocation
- BOW - Bag-of-Words
- TF-IDF - Term Frequency-Inverse Document Frequency
- LSA - Latent Semantic Analysis
- LSI - Latent Semantic Indexing
- SVD - Singular Value Decomposition
- NMF - Non-negative Matrix Factorization
- pLSA - Probabilistic Latent Semantic Analysis
- pLSI - Probabilistic Latent Semantic Indexing
- EM - Expectation Maximization
- CTM - Correlated Topic Model
- HLDA - Hierarchical LDA
- PAM - Pachinko Allocation Model
- DAG - Directed Acyclic Graph
- ETM - Embedded Topic Model
- VAE - Variational Autoencoder
- SIWO - Strong Inside, Weak Outside
- NMI - Normalized Mutual Information
- LFR - Lancichinetti, Fortunato, Radicchi
- CPM - Constant Potts Model
- NHL - National Hockey League

Chapter 1

Introduction

1.1 Motivation

Researchers have long sought ways to discover the themes and concepts of large collections of unstructured text documents. These topics can fulfill multiple roles. They can act as features for document classification. They can serve as indices for information retrieval. However, one of the most important functions of these topics is to assist in the exploration and understanding of large corpora. Researchers in history, social science, political science, computer science, and the physical sciences all seek to better understand the main ideas and themes of document collections too large for a human to manually read and summarize. This requires topics that are interpretable and coherent to the human users who study the corpus.

In more recent years, another new area has emerged where topics can provide a great deal of utility: conversational agents or "chat bots". A conversational agent is a computer program that is able to carry on a conversation with a human. The conversation is an end in itself; the purpose of speaking with a conversational agent is to converse, to be entertained, to express emotion and be supported. This goes well beyond asking Siri to set a timer or Alexa to play a song. One key component of an agent that is capable of having an actual conversation with a human is the awareness and use of the topic of conversation. Work has been done on enriching the agent's response using the detected topic [26]. However, there is much more that can be done with topics to improve the functioning of a conversational agent. They can be used to

detect and control topic drift in the conversation so that the agent's responses make sense given the context of a statement. If the user is engaged with the current topic, then the conversational agent should be able to stay on topic or quickly detect sub-topics to focus the conversation. The agent should also be able to quickly detect super-topics to broaden the range of conversation. As the conversation progresses, the agent should be able to move to related topics or, if the user becomes bored or displeased, jump to dissimilar topics. This type of control over the flow of the conversation is crucial to human communication and is needed for human-computer interaction as well.

The features that would make a topic model useful for a conversational agent are the same that make it useful as a tool of exploration for applied researchers. The topics themselves must be coherent and interpretable to be useful to a researcher and for an agent's response to fit into a conversation. A topic is coherent when the terms or documents that share a topic have related semantics. A topic is interpretable when the semantics of the topic are accessible from the representation of the topic itself, e.g. a topic represented by a set of terms is more interpretable than a topic represented by a real number vector of some latent space. A topic model that has a measure of relatedness between topics allows for a natural flow to exploration and conversation. A topic model with a natural hierarchical structure allows both a researcher and a conversational agent to drill down into more specific sub-topics or find broader super-topics on the fly as they explore the corpus/engage in a conversation.

Unfortunately, the most widely used topic modelling algorithm, Latent Dirichlet Allocation (LDA), does not have many of these features. It has other drawbacks as well. The number of topics to be found must be specified. Multiple runs with different numbers of topics are thus required to find the best topics. It performs poorly on short documents. Different runs of the algorithm on the same corpus can produce different topics, especially if the order of the documents is different [66]. Common terms can appear in many different topics, reducing the uniqueness of topics [78].

As in many other fields, neural networks and deep learning have recently pushed forward the state-of-the-art in topic modelling. While neural topic

models have produced topics of greater coherence, they retain many of the weaknesses of LDA, such as the need to specify the number of topics, while having a tendency to find models with many redundant topics [13] and demanding greater computational resources and specialized hardware, i.e. GPUs.

These drawbacks have inspired us to search for an alternative approach to topic modelling, one that can operate quickly on commodity hardware and that provides not only a set of topics but their relationships and a hierarchical structure. Given the growing importance of relational data and graphs in representing complex systems [100], it seems natural to take a network-based approach to topic modelling.

1.2 Problem Definition and Challenges

Given a collection of documents, the problem of topic modelling is to find the topics in the documents, i.e. the prominent concepts, themes, and ideas. A good topic model should provide the following:

- **Interpretable and coherent topics** - the topics found by the model should be interpretable by humans as topic models are used to understand large collections of documents. A collection of terms is interpretable; a 300 dimensional vector is not. The topics must also be coherent and group terms that are semantically related.

- **Discovery of the number of topics** - a user should not have to specify the number of topics as this is often not known.

- **Relationship between topics** - different topics are more or less related. The "movie" topic is more related to the "TV" topic than to the "farming" topic. The topic model should determing and provide these similarities.

- **Topic hierarchy** - Topics do not exist at only one scale. There can be a "computer" topic at one scale, but this contains "hardware" and "software" sub-topics. It is also part of a "technology" super-topic with

topics like "smart phones". The topic model should provide access to such a topic hierarchy and allow users to find sub- and super-topics as desired.

- **Fast run time** - Topic models are often used by researchers working with simple, single processor hardware to explore document collections. The topic model such run quickly on such hardware to make it a practical tool.

It is a challenge for a topic model to meet all of these requirements and currently none do. Traditional topic models require a specification of the number of topics and do not provide topic relationships or a topic hierarchy. Extensions to find relationships and a hierarchy require specifying features of the hierarchy such as the depth and are limited in their expressiveness. Methods to improve topic quality using neural networks drastically increase the training time even when special hardware such as GPUs are used, and still do not discover a topic hierarchy.

1.3 Thesis Statements

This thesis focuses on two main subjects: social network analysis and topic modelling. The thesis hypotheses are:

Thesis Statement 1: Information about relationships between terms is encoded in the pattern of co-occurrence in natural language text.

Thesis Statement 2: The patterns of connections in a term co-occurrence network contains information about the themes and concepts of the text.

Thesis Statement 3: Community detection algorithms are able to extract topics from the term co-occurrence by finding groups of closely related terms.

Thesis Statement 4: Topics extracted via community detection from term co-occurrence networks are more coherent than those found by Latent Dirichlet Allocation.

Thesis Statement 5: The networks structure provides not only a set of topics but also provides the importance of terms, the relationships between

topics, and the topic hierarchy.

Thesis Statement 6: The edge weights of a network can be combined with the presence of common neighbours to get a better measure of the strength of connections between vertices in a network.

Thesis Statement 7: The SIWO algorithm, for community mining in a graph, can be extended and improved by incorporating edge weight information.

1.4 Thesis Contribution

There are three major contributions in this work:

1. We extend the SIWO community search algorithm to handle weighted networks. SIWO has already proven itself to be one of the best performing local community search algorithms. By extending it to handle weighted networks, we have created one of the only community search algorithms able to work on weighted networks. This also extends the SIWO+ global community detection algorithm to handle edge weights, bringing it on par with most other community detection algorithms that can handle edge weights. We demonstrate that SIWO is one of the best performing algorithms on weighted networks.

2. We define and analyze term co-occurrence networks that can be extracted from corpora using not just raw co-occurrence counts as edge weights but also Normalized Pointwise Mutual Information. These networks can be created in a single pass over the corpus yet contain information on the topics of the documents.

3. We develop a novel topic modelling algorithm, Community Topic. This algorithm finds more coherent topics in a shorter period of time than its competitors. As well, it provides a topic structure that identifies the important terms in a topic quantifies the relationships among topics. Sub- and super-topics can be found dynamically from the topic structure. It does not require hyperparameter tuning and can find topics at various

scales. This structure is ideal for downstream applications such as corpus exploration and conversational agents.

1.5 Thesis Organization

Chapter 2 reviews background and related work for both topic modelling and social network analysis. It covers early topic modelling methods, Latent Dirichlet Allocation, the most widely used topic modelling algorithm, improvements and extensions of Latent Dirichlet Allocation, recent topic modelling approaches based on neural networks, and evaluation methods for topic modelling. We also review social network analysis with a focus on community detection and search. We review SIWO, a community mining algorithm previously developed at the University of Alberta which we later extend. We cover evaluation methods for community detection and search.

In Chapter 3, we present our extension of SIWO to handle community search and detection on networks with weighted edges, SIWOw, and empirically evaluate it against a range of modern algorithms.

In Chapter 4, we present our novel Topic Modelling algorithm, Community Topic. We analyze the structure of the term co-occurrence networks constructed from the documents. We conduct a thorough investigation of the possible algorithm configurations and hyperparameters. We compare our algorithm against two contenders on topic coherence, run time, and stability.

Chapter 2

Background and Related Work

In this chapter we present a summary of works related to our research into the application of social network analysis to topic modelling. We first review the field of topic modelling and the main approaches in use today. We then review the field of social network analysis, particularly the sub-field of community detection and search. We pay particular attention to the SIWO algorithm which we later extend in one of this thesis' contributions.

2.1 Topic Modelling

The process of topic modelling seeks to uncover the themes and concepts running through a collection of documents. Topic modelling emerged from the field of information retrieval where researchers have long worked on methods to reduce dimensionality and more effectively represent documents for indexing, query matching, and document classification. In recent years the performance of topic models on these tasks has been surpassed by deep neural models but topic models have become extremely popular tools of applied research both inside and outside of computing science [42]. Topic models enable researchers to better understand large collections of documents and have been used in fields as varied as political science [45] and bioinformatics [63].

In this chapter we review the field of topic modelling. We begin by discussing the origins of the field and some of the first methods developed. Then we cover Latent Dirichlet Allocation (LDA), by far the most widely used algorithm, and some of its many variants. We then discuss some alternative

approaches that have emerged in recent years. Finally, we cover evaluation methods for topic modelling.

The following definitions are necessary:

- A **term** t is the basic unit of data and are typically words but can also be combinations of words or words that have been pre-processed in some way, e.g. stemmed [92].
- A **document** d is a sequence of terms. As with terms, pre-processing of documents is common.
- A **corpus** D is a set of documents. Typically these documents are related in some way.
- The **vocabulary** W is the set of all valid terms extracted from the corpus. This can be every term or a filtered subset.
- The **frequency** of a term t in a document d is the number of occurrences of t in d and is denoted $f_{t,d}$.

2.1.1 Early Methods

Researchers have long sought methods for better representing and reducing the dimensionality of unstructured text documents for the purposes of classification and information retrieval. One commonly used model is the bag-of-words (BOW), where each document in a corpus can be represented as a vector of length $|W|$ where the i^{th} entry of the vector is $f_{t_i,d}$. This vector representation can be used as input to various classifiers and as an index for query matching using various similarity measures. However, this representation using raw counts can give undue weight to common and thus uninformative terms such as "the".

To remedy this issue, the term frequency-inverse document frequency (TF-IDF) model was developed [101]. The term frequency of a term t and a document d is the relative frequency of t in d with higher values indicating that t is a more important feature of d. It is calculated by:

$$\text{tf}(t, d) = \frac{f_{t,d}}{|d|} \tag{2.1}$$

The inverse document frequency gives higher weight to terms that appear in few documents and are thus discriminating. Terms that are very common throughout the corpus have a low inverse document frequency as they do not provide information that would help classify a document or retrieve relevant information for a query. It is calculated by:

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{2.2}$$

These two measures are combined to give one scalar value for each combination of term and document in the corpus that represent how important that term is as a feature of that document:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D) \tag{2.3}$$

As with the BOW scheme, each document d in the corpus D can be represented as a vector of length $|W|$ where the i^{th} entry of the vector is the value tf-idf(t_i, d, D). While TF-IDF improves upon BOW as to the quality of the vector representations, they both suffer from two important drawbacks. Firstly, the degree of dimensionality reduction is limited. The vectors are compressed representations of the raw documents, but with large corpora the vocabulary size can be tens of thousands of terms resulting in vectors with tens of thousands of dimensions. Secondly, these schemes work with exact terms such that two documents "The small baby cat is sleeping" and "The tiny kitten is napping" will not have similar vector representations even though they are conceptually similar since the specific terms they use are different. The search for representations that reflect the conceptual content of a document marked the beginning of the development of what we now call topic modelling.

A major step forward arrived when researchers proposed Latent Semantic Analysis (LSA) and its use as a document index, Latent Semantic Indexing (LSI) [21]. Either the BOW or TF-IDF scheme can be represented as a term-by-document matrix $\mathbf{X} \in \mathbb{R}^{|W| \times |D|}$ where each column is a document vector.

LSA uses singular value decomposition (SVD) to decompose X into three matrices $\mathbf{T} \in \mathbb{R}^{|W| \times h}$, $\mathbf{S} \in \mathbb{R}^{h \times h}$, and $\mathbf{D}^T \in \mathbb{R}^{h \times |D|}$. For $\mathbf{TSD}^T$ to perfectly reconstruct $\mathbf{X}$, h must equal the rank of $\mathbf{X}$. However, h can be chosen to be smaller than the rank of $\mathbf{X}$ by only keeping the largest h singular values that run along the diagonal of $\mathbf{S}$. $\mathbf{T}$ and $\mathbf{D}$ are representations of the terms and documents, respectively, in an h-dimensional space. The value of h should be large enough that the latent semantic structure of the data can be fully captured and small enough that noise is eliminated; the authors use 100 as a value of h. Each dimension in this space can be thought of as a concept or topic, however they are artificial and cannot be interpreted. So while a document can be thought of as a linear combination of the h different concepts, there is no way to examine these topics to understand or explore the corpus.

Another method based on matrix decomposition is Non-negative Matrix Factorization (NMF) [60]. NMF decomposes the matrix $\mathbf{X} \in \mathbb{R}^{|W| \times |D|}$ into two matrices $\mathbf{W} \in \mathbb{R}^{|W| \times h}$ and $\mathbf{H} \in \mathbb{R}^{h \times |D|}$ which are constrained to have all entries be non-negative. The h columns of $\mathbf{W}$ can be viewed as topics with higher values in the entries corresponding to terms that have higher relevance to that topic. Each row of $\mathbf{H}$ corresponds to a document and the entries along the row give the weights of each topic in that document. Since all values are non-negative, it is easier to examine the term composition of topics and the topic composition of documents.

Another attempt to improve upon LSA is Probabilistic Latent Semantic Analysis (pLSA) with its corresponding indexing method pLSI [40]. The developers of this method were unsatisfied with the lack of a solid statistical foundation to LSA and so rather than use matrix decomposition they developed their method with a generative probabilistic model of the data. In their model, documents are mixtures of latent topic variables z and term-document co-occurrences are generated by the following process:

- select a document d with probability $p(d)$
- select a topic z with probability $p(z|d)$
- generate a term t with probability $p(t|z)$

This process is modelled by the following probability distribution:

$$p(d,t) = p(d) \sum_{z} p(t|z)p(z|d) \tag{2.4}$$

and is shown as a probabilistic graphical model in Figure 2.1.

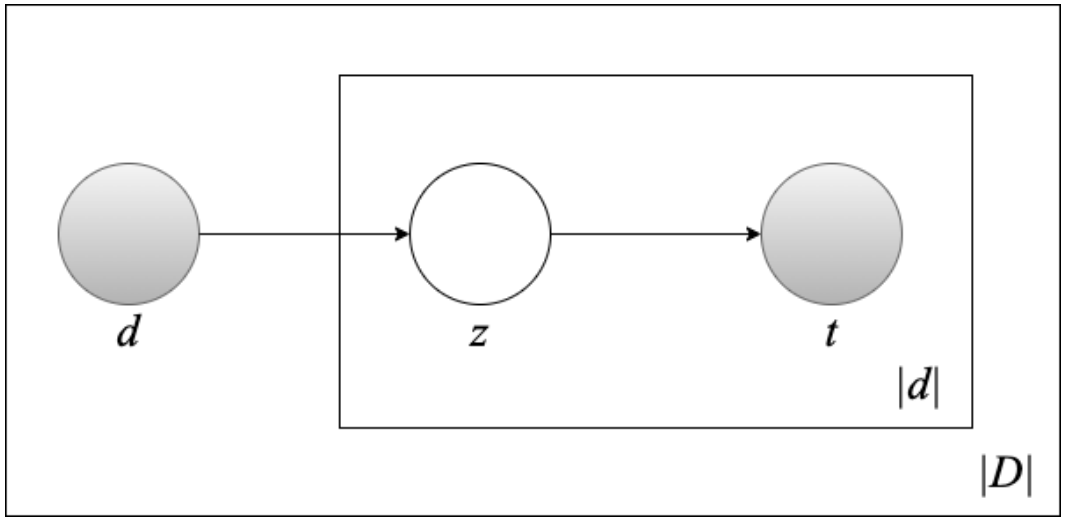


Figure 2.1: Probabilistic graphical model of pLSA. For each document d in the corpus D, a topic variable z is sampled from $P(z|d)$ $|d|$ times. For each sampled z, a term t is sampled from $P(t|w)$. Shaded circles indicate observed variables while white circles are hidden latent variables.

The algorithm takes as input the BOW term-by-document matrix $\mathbf{X}$ and uses a variation of the Expectation Maximization (EM) algorithm [22] to find the maximum likelihood distributions. Topics are probability distributions over terms and are thus interpretable, allowing researchers to gain new insight and knowledge about collections of documents.

Interestingly, even though they approach the problem from different directions, NMF and pLSA have been shown to be equivalent [34]. Even though they can be shown to optimize the same objective function they are distinct algorithms and will not necessarily converge to the same solution, but both methods can be combined to improve performance over either one on its own by alternating optimization steps to avoid getting trapped in local minima [24].

One major drawback of the pLSA algorithm is the lack of a generative process for the topic distributions of each document. The topic mixture $p(z|d)$

is estimated separately for each document d, so the number of parameters to be fit grows with the size of the corpus and these probability functions cannot be applied to unseen documents.

2.1.2 Latent Dirichlet Allocation

To address the shortcomings of pLSA, researchers developed what has become by far the most popular and well-known topic modelling algorithm: Latent Dirichlet Allocation (LDA) [9]. LDA is also a hierarchical probabilistic model, but it is a fully generative model as it places a Dirichlet distribution prior on the latent topic mixture of a document (hence the name Latent Dirichlet Allocation). Rather than estimate a separate $p(z|d)$ for each d, the $p(z|d)$ is a multinomial distribution over the h possible topics parameterized by θ where θ is itself a random variable sampled from the prior Dirichlet distribution parameterized by α. The generative process of a document is thus:

- Sample θ from the Dirichlet distribution $p(\theta; \alpha)$
- For each term position in the document, sample a topic z from the multinomial distribution $p(z; \theta)$. Then sample a term t from the multinomial distribution over the vocabulary $p(t|z; \beta)$ with β estimated from the corpus.

This generative model is illustrated in Figure 2.2.

The number of topics h must be specified by the user. The algorithm then works by finding parameters that maximize the probability of the observed corpus assuming that it was generated by the hidden latent variables of the model. The probability of a corpus D is modelled as:

$$p(D; \alpha, \beta) = \prod_{d \in D} \int p(\theta_d; \alpha) \left(\prod_{i=1}^{|d|} \sum_{z_{d,i}} p(z_{d,i}; \theta_d) p(t_{d,i}|z_{d,i}; \beta) \right) d\theta_d \qquad (2.5)$$

where θ_d parameterizes the multinomial topic distribution of document d and $z_{d,i}$ is the topic that generates the $t_{d,i}$, the term at position i in document d.

However, to find these parameters requires computing the probability of the parameters given the observed terms in the documents taken from the BOW/TF-IDF term-by-document matrix $\mathbf{X}$. This computation is intractable [9] and thus one can only approximate the true solution. There are several methods to do so, including variational inference [46] and Markov chain Monte Carlo [47]. Once this learning is complete, LDA provides h topics, each represented as probability distributions over terms and thus interpretable to human users. The trained model can also be applied to unseen documents to discover that document's topic distribution.

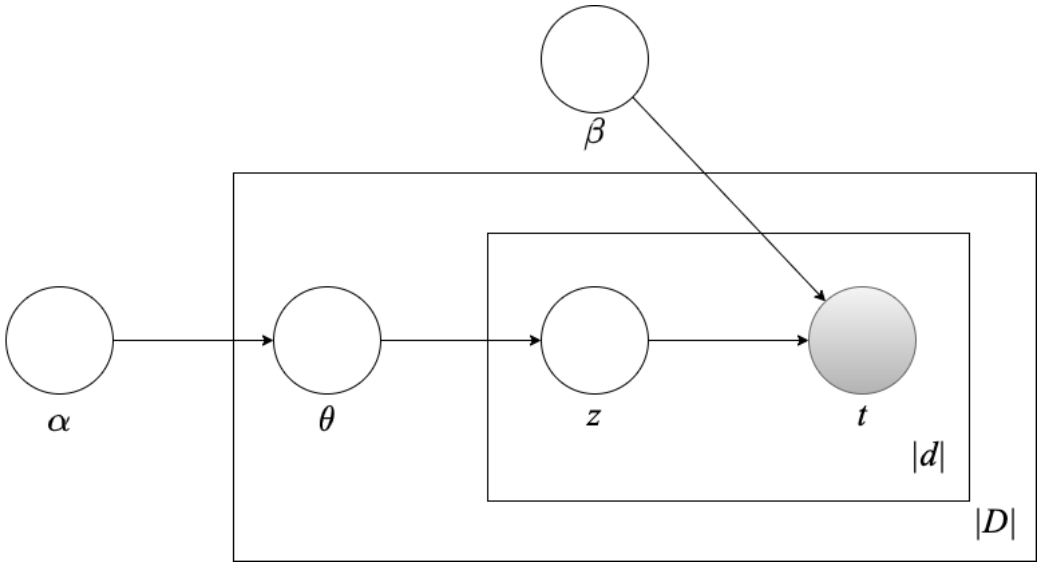


Figure 2.2: Probabilistic graphical model of LDA. For each document d in the corpus D, the parameters θ of a multinomial distribution over topics is sampled from a Dirichlet prior distribution parameterized by α. For each term position in d, a topic z is sampled from the multinomial distribution. Given z, a term t is sampled from a multinomial distribution over all terms parameterized by β. Shaded circles indicate observed variables while white circles are hidden latent variables.

While LDA has been extremely successful and is widely used across many disciplines, there have been many attempts to improve upon it while maintaining the hierarchical probabilistic model framework. The most straightforward attempts at improvement have been those that do not modify the LDA algorithm itself but rather modify, augment, or re-weight the data. Researchers have tried first detecting named entities, i.e. specific real-world object such as

a person e.g. Donald Trump, in the documents and giving increased weights in $\mathbf{X}$ to those terms so that they become the most frequent terms in the document [52]. They found that this tended to improve the quality of detected topics over baseline LDA. In [115], the authors use a two-step process to identify and re-weight words that are topic-indiscriminate i.e. those words that occur in many topics and thus cannot be used to distinguish between topics. First, LDA is run once with the original TF-IDF term-by-document matrix $\mathbf{X}$. From the learned topic model, those words that occur in many topics are idenfied and their frequency weights discounted. Then LDA is run again with the re-weighted matrix and a new topic model is learned. The authors found that this tended to outperform baseline LDA. One weak point of LDA is its performance on corpora that consist of short documents as the algorithm relies on term-document co-occurrences therefore short documents provide less co-occurrence information. To improve the performance of LDA on Twitter posts (tweets), the authors of [69] pool tweets into longer documents based on various schemes such as common author, similar posting time, and same hashtag (a hashtag is a special label on a tweet that indicates topic or context, e.g. #blacklivesmatter).

Twitter is a rich source of information but LDA struggles with short documents such as tweets. This motivation has prompted researchers to go beyond simple strategies such as pooling tweets to be able to analyse twitter topics. One attempt is both a data augmentation method and a modification of the LDA model called MetaLDA [120]. This method can incorporate document and word meta-information such as document labels, WordNet synonyms [73], and word embeddings e.g. word2vec [72]. This information is used to condition the Dirichlet prior of the topic distribution, whereas with standard LDA the same prior distribution is used for each document. Another approach is Twitter-LDA [121] which has an author-specific topic distribution for each Twitter user and restricts documents/tweets to be about a single topic.

There have been many other hierarchical probabilistic models that are extensions or variations of the basic LDA model. Twitter-LDA is itself a variation on an earlier algorithm that conditions the topic mixture on document author

called the author-topic model [107], although the author-topic model allows for multiple authors per document and multiple topics per document which is less appropriate for Twitter data.

In LDA, the topic for each term position of a document is sampled independently. However, it seems likely that the topics that appear together in a document are somehow related to each other. The Correlated Topic Model (CTM) [7] substitutes a logistic normal distribution that models the correlations between topics for the Dirichlet prior. This allows for more expressiveness than LDA and the learned correlations between the topics can aid in the exploration of the document corpus.

The authors behind the Correlated Topic Model also developed the Dynamic Topic Model [8] which allows for the modelling of topic evolution over time. In this model, the time span over which the documents in a corpus were published is discretized into a series of periods. A separate LDA-like model is trained on the documents of each different time period, but these models are chained together so that the topics learned in a given time period influence the topics learned in the next time period.

Another variation created by a team including two of the original LDA authors is the Hierarchical LDA model (HLDA) [38], so called not because it is a hierarchical probabilistic model, which all the models in this section are, but because it allows for a hierarchy of topics using a tree structure. When each term is generated, rather than sampling from a multinomial distribution over all topics, a path is followed down the topic tree and a topic is sampled from a multinomial distribution over only those topics encountered along that path. These topics will be conceptually related but with topics higher in the tree being more general and topics further down the tree being more specific. An example of such a tree is illustrated in figure 2.3.

A flexible generalization of LDA is the Pachinko Allocation Model (PAM) [61]. Like HLDA, PAM allows for a hierachy of topics but this hierarchy is not restricted to being a tree of fixed depth. Like CTM, PAM allows for correlations between topics but these are not restricted to pairwise correlations. This is because the PAM represents the topic structure as a directed acyclic

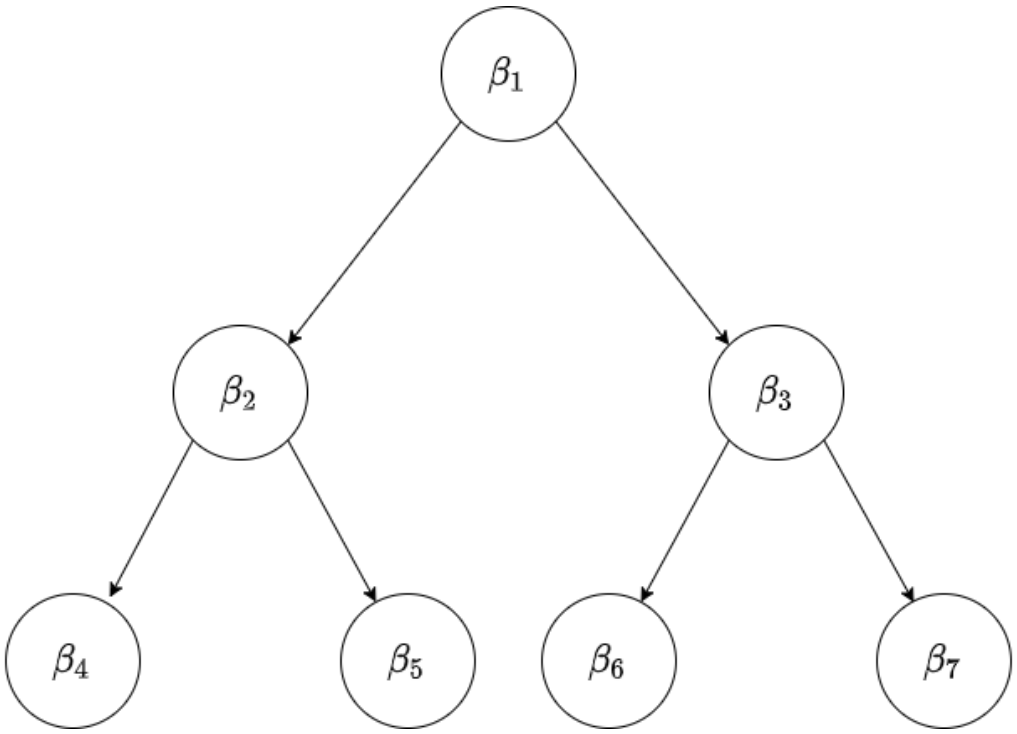


Figure 2.3: Tree hierarchy of topics in the Hierarchical LDA model. Each term is generated by a topic that is sampled from those encountered along a path from the root to a leaf, e.g. β_1, β_2, β_5. These topics are related in that the topic parameterized by β_2 is a sub-topic of the topic parameterized by β_1, and the topic parameterized by β_5 is a sub-topic of the topic parameterized by β_2.

graph (DAG) where each leaf vertex is a term in the vocabulary and non-leaf vertices are topics. The topic nodes model correlations between their child vertices, which can be both term vertices and other topic vertices. This allows for modelling a variety of relationships between topics and a more flexible topic hierarchy. An example of such a DAG is illustrated in figure 2.4. This flexibility allows for a wide variety of possible model architectures, but also raises the problem of which to choose. The authors elect to evaluate a simple four level hierarchy with one root topic, a level of super-topics, a level of sub-topics, and the vocabulary terms. Each level is fully connected to the one below it and there are no other connections. Even though this architecture does not take full advantage of the flexibility offered by the DAG, their experiments show it models the text data better than HDP, CTM, and LDA.

All of the models described in this section are generative probabilistic mod-

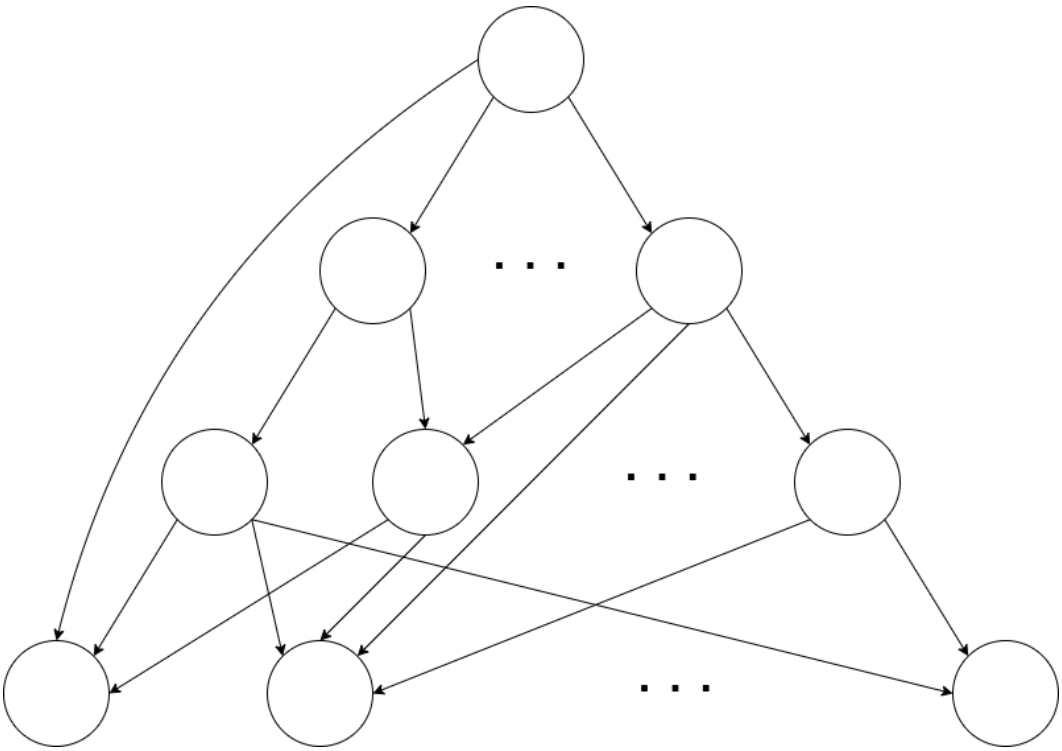


Figure 2.4: An example of a topic hierarchy DAG used by PAM. The leaf vertices at the bottom are the terms in the vocabulary. Non-leaf vertices are topics. There are multiple levels of super- and sub-topics but these topics can be distributions over both topics and terms and the nesting can span multiple levels.

els that can be seen as variations on the basic LDA framework. These models typically involve making the LDA model more complex to capture some aspect of how documents are actually created (e.g. authors, intra-document topic correlations) or how topics should be modelled (e.g. hierarchies). However, these more complex models do not always outperform vanilla LDA in all situations. For example, the MetaLDA model was designed to work better on tweet data than standard LDA, but in [25] the authors applied both models to a corpus of tweets and found that which model performed best depended on the number of topics chosen and the topics selected for evaluation. The added complexity and uncertain benefits of these models has meant that standard LDA has remained the most popular model among researchers using topic models in applied research [108]. Part of the uncertainty about which model performs best stems from evaluation metric issues which are reviewed in Section 2.1.4.

2.1.3 Recent Approaches

In recent years, new types of topic models have emerged. Many of these approaches are based on the huge advances in neural networks and deep learning that we have witnessed over the past decade. Some of these new methods remain close to the LDA framework while others are completely different approaches. Here we will review a few of these recent approaches.

One algorithm that marries LDA with representations learned by neural networks is the Embedded Topic Model (ETM) [23]. This model posits a similar generative process as LDA whereby a document is generated by first sampling a topic mixture, in this case from a logistic-normal distribution rather than a Dirichlet distribution, and then sampling a topic from that mixture and a term from that topic for each term position in the document. Where ETM fundamentally differs from LDA is how the terms are sampled given a topic. ETM uses word embeddings trained using the continuous Skip-gram algorithm [72] to represent the terms in the vocabulary. The word embeddings are vector representations learned by a neural network that capture linguistic patterns and semantic meanings. These embeddings can be learned during the training of ETM or the model can be provided pre-trained embeddings. The model also learns a vector representation for each topic in the same vector space. Given a topic z, a term t is sampled from a softmax distribution on $\mathbf{M}^T\alpha_z$ where $\mathbf{M}$ is the matrix whose columns are the word embeddings for each term in the vocabulary and α_z is the embedding of topic z. So the most likely terms for a topic are determined by the distance between the term and topic embeddings rather than simple co-occurrence counts. The topics learned by ETM are still probability distributions over terms so are interpretable to a human user but the model leverages the expressive power of a distributed term representation learned by a neural network.

Another approach is to use deep neural networks to learn the probability distributions of a generative probabilistic model, rather than variational Bayes or Markov chain Monte Carlo as in LDA and its variants. This can be done using a variational autoencoder (VAE) [49, 50]. There have been many

VAE-based topic models developed, including the neural variational document model (NVDM) [70], the stick-breaking variational autoencoder (SB-VAE) [77], ProdLDA [106], and Dirichlet-VAE [13]. These models discover topics that are qualitatively different than those found by traditional LDA, although there is debate as to whether they are truly superior [42].

Other approaches use the word embeddings learned by a deep neural network but are not based on generative probabilistic models. An algorithm presented in [28] first learns 200-dimensional word2vec embeddings [72] on the text corpus. Then, a document is placed in this vector space by taking the centroid of all the word embeddings of the terms in the document. Once all document vectors are calculated, they are clustered using the k-means algorithm. The produced topics are collections of documents rather than terms, which makes topics less interpretable but potentially still useful for data exploration.

The top2vec algorithm [3] is also based on embeddings learned by a neural network, in this case the term and document vectors learned jointly by the doc2vec algorithm [59]. The authors argue the the semantic embedding space itself is a continuous representation of topics; every point in that space is a topic and there is no finite number of discrete topics such as in LDA. Any vector in that space can be transformed into a probability distribution over all terms by taking the softmax of the dot product of that vector with the matrix of all word embeddings. To find the topics for collections of related documents, first the dimensionality of the document embeddings is reduced to two dimensions using the UMAP algorithm [68]. Then dense clusters are found using HDBSCAN [14]. The topic for a cluster of documents is the centroid of all those document vectors in the original embedding space and the most relevant terms are those whose embeddings are closest to the topic embedding.

Having seen that a great variety of topic models exist, the question naturally arises of how to compare them. We now turn our discussion to evaluation methods for topic models.

2.1.4 Evaluation Methods

There are three main approaches to topic model evaluation: human evaluation, intrinsic evaluation, and extrinsic evaluation.

Human Evaluation

Human evaluation is the gold-standard and is especially important given topic modelling's application as a tool in applied research. However, human evaluation has fallen out of favour in recent years [42] and most recent work relies solely on automated metrics. Given the extra effort, time, and expense related to human evaluation, even papers that do incorporate some human evaluation tend to have a very small number of evaluators, calling in to question the reliability of the results.

In [79], the authors provide a set of nine human evaluators with a rubric for evaluating topics on a three-point scale from "useless" to "useful" based on qualities such as "meaningful", "interpretable", and "easy-to-label". Another approach was taken in [15] where rather than ask for a subjective opinion the authors designed two tasks for the human evaluators to perform. The word intrusion task presented evaluators with five terms from a topic and one intruder from a different topic with higher rates of intruder detection indicating a more coherent topic. In the topic intrusion task, the evaluators were given part of a document and the top words from four topics, three of which were high probability topics for that document and one of which was not. If the low probability intruder topic could be reliably detected then model could be said to be doing a good job of finding the topics of a document. These experiments were each run with eight evaluators hired on Mechanical Turk[1]. Researchers working on a corpus of grant applications and journal abstracts from the National Institute of Health were able to collaborate with two domain experts from the National Institute of Neurological Disorders and Stroke [76]. The experts were able to classify and rate topics discovered by LDA and their ratings correlated positively with word intrusion task scores generated by ten

[1] https://www.mturk.com/

other expert evaluators on the same topics.

Intrinsic Evaluation

Given the effort and expense associated with human evaluation, researchers have sought to develop automated metrics that can measure the quality of a topic model that agree with human judgements. The first metric used in the original LDA paper was perplexity, borrowed from the field of language modelling:

$$perplexity(D) = \prod_{d \in D} \prod_{t \in d} \sqrt[|d|]{\frac{1}{p(t)}} \tag{2.6}$$

This is equivalent to the inverse of the geometric mean of the per-term likelihood of a sequence of terms with lower scores corresponding to language models that better model the documents. While perplexity was used early on for evaluating topic models, the metric does not actually tell us anything about the semantic content of the topics themselves. In fact, perplexity was found to be negatively correlated with human assessment of topic quality [15] and has fallen out of favour. Of course, this raises the question of whether generative probabilistic models are the best approach to topic modelling since these find topics that are the latent variables best able to reconstruct the original documents and the terms most important for that task may not be the most relevant to the thematic or conceptual information in a document.

Many other automated metrics have been developed that seek to measure the quality of the topics themselves. We would like to discover topics that represent an identifiable category and are meaningful, interpretable, and easily labelled; in other words, topics that are coherent. There have been many different coherence metrics developed, but all are based on the idea that terms from the same topic should be likely to co-occur together in some reference corpus. There are many variations based on different definitions of likeliness and co-occurrence and which reference corpus is used.

One of the first proposed measures is known as C_{UCI} [79] and is based on pointwise mutual information (PMI) between the top-N terms in the same

topic calculated using probabilities derived from the term co-occurrence counts collected by a sliding window over a Wikipedia corpus:

$$\text{PMI}(t_i, t_j) = \log \frac{p(t_i, t_j) + \varepsilon}{p(t_i)p(t_j)} \tag{2.7}$$

$$C_{UCI} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \text{PMI}(t_i, t_j) \tag{2.8}$$

where ε is a small smoothing factor added to avoid taking the log of zero. The authors show that this coherence measure positively correlates with the judgements of their human evaluators. This metric was improved upon to get C_{NPMI} by using normalized pointwise mutual information (NPMI) in place of PMI [2].

$$\text{NPMI}(t_i, t_j) = \frac{\log \frac{p(t_i,t_j)+\varepsilon}{p(t_i)p(t_j)}}{-\log(p(t_i, t_j) + \varepsilon)} \tag{2.9}$$

Another coherence metric is C_{UMass} [76] which is based on conditional probability rather than PMI/NPMI and calculates probabilities from co-occurrence using the original training documents as a reference corpus and using the entire document for a co-occurrence window.

$$C_{UMass} = \frac{2}{N(N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} \log \frac{p(t_i, t_j) + \varepsilon}{p(w_j)} \tag{2.10}$$

Given the proliferation of these coherence metrics, in [97] researchers developed a general framework to describe the entire space of possible coherence functions and evaluated their correlation to human ratings previously gathered in [2, 15, 58]. Interestingly, the coherence metric most highly correlated with human assessment of topic quality was a new measure found by the authors as they explored the space of coherence functions they defined. This metric, C_V, uses a sliding window of 110 terms on the Wikipedia corpus to create NPMI-based context vectors for the top-N terms in a topic. These context vectors are compared using cosine similarity to determine a topic's coherence. The use of context vectors means that this metric does not only check whether

two terms co-occur often, but also whether they co-occur with the same set of other terms.

Given that researchers have developed several automated coherence measures that correlate with previous human evaluations, it would seem that the problem of evaluating topic models is solved. However, some researchers have raised doubts about the reliability of these automated metrics. In [25], the authors show that various coherence metrics do not necessarily agree with each other and that the LDA topics they rate most highly is sensitive to the number of topics chosen and the reference corpus used for evaluation. Motivated by the lack of consistency in the use of these metrics and the very high scores achieved by neural topic models, the authors of [42] set out to use fresh human evaluations to check whether the previously reported correlation between automated metrics and human judgement still holds given the qualitatively different topics found by these new models. They found that human evaluators often judged topics produced by old-fashioned LDA to be superior even though the automated metrics score the neural topic models more highly. They conclude that measures designed for older models may be incompatible with new models and that evaluation of models using generic corpora like Wikipedia may not be appropriate given the use of topic models in specific research domains. So while these automated metrics may still have some use, their results should be taken with a grain of salt and we must not fall into the trap of using a single number to evaluate a model and declaring the model with the biggest number to be the best.

Extrinsic Evaluation

Topic models can also be evaluated by how well they perform at an external, downstream task such as sentiment detection [109], information retrieval [114], and document classification [90]. While topic model approaches could outperform methods based on earlier schemes such as BOW and TF-IDF, in recent years deep neural network algorithms have been the state-of-the-art for these tasks. The main use of topic models is now as tools for content analysis, exploring and understanding large document collections, assisting applied research

in various disciplines, and revealing new knowledge and insights. Evaluation of this function is much more difficult than with a task such as classification where standard benchmarks and metrics exists. Thus researchers typically fall back to intrinsic evaluation metrics such as coherence as a proxy for how well a topic model would facilitate this exploration and knowledge discovery.

One can ask questions about whether the features of the topic model would enable such exploration and knowledge discovery. Are the topics interpretable by human researchers? Are the discovered topics coherent? Are the topics related in some way so that connections between topics can be seen and exploration can be guided? Does the model discover some natural number of topics or must the user specify how many to find? Is there a natural hierarchy to the topics so that researchers can request more specific and finely-grained topics or broader, more general topics as they explore? The more questions that can be answered "yes", the more likely the topic model is to be a useful tool for researchers.

One new application area for topic models is in chatbots and conversational agents [26]. The qualities that would make a topic model a good research tool are similar to those that would make the model useful in a conversational agent. Topics that are coherent to a researcher would also help the agent provide coherent responses. A topic structure with connections that can guide exploration between related topics would also enable an agent to guide a conversation between related topics. Of course, evaluating a conversational agent is at least as hard a problem as evaluating a topic model as a research tool although there are competitions such as the Alexa Prize[2] that seek to do just that.

Now that we have reviewed the field of topic modelling, we move on to cover social network analysis, the second key area of research that supports our work, and its sub-field of community detection and search.

[2]https://www.amazon.science/alexa-prize/socialbot-grand-challenge

2.2 Social Network Analysis

The study of complex networks is a field that combines ideas from many different disciplines such as the social sciences, physics, mathematics, and computer science. Mathematicians have been studying graphs and their properties at least since Euler solved the Königsberg bridge problem in 1736 [29, 31]. Social scientists have the longest tradition of studying real-world networks that dates back to the empirical examination of social groups by Jacob Moreno in the 1930's [82]. Subsequently researchers from disciplines such as physics and computer science have applied themselves to the study of networks and have brought techniques from their fields to aid the analysis and the field has seen a rapid growth in the number and sophistication of methods.

A comprehensive review of network theory is beyond the scope of this work and refer the interested reader to an excellent text on the subject [82]. We present and define sufficient terminology to be able to understand our contribution to the field.

- A network is represented by a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges.
- We denote the number of vertices in the network as n, i.e. $|V| = n$.
- We denote the number of edges in the network as m, i.e. $|E| = m$.
- A network may be **unweighted**, in which case there is a binary alternative between the existence or non-existence of an edge $e_{i,j}$ between any two vertices $v_i, v_j \in V$ that indicates a relationship between those vertices.
- A network may be **weighted**, in which case an edge $e_{i,j}$ has an associated weight $w_{i,j}$ which is a numeric value that characterizes in some way the relationship between vertices v_i and v_j.
- A network may be **undirected**, in which case relationships between vertices v_i and v_j are symmetrical and represented by $e_{i,j} = e_{j,i}$

- A network may be **directed**, in which case an edge $e_{i,j}$ indicates a relationship from v_i to v_j which is distinct from the relationship from v_j to v_i represented by $e_{j,i}$.

- The **degree** of a vertex v_i, denoted k_i, is the number of edges connected to that vertex, i.e. $k_i = |\{e_{i,j} : v_j \in V\}|$.

- The **weighted degree** of a vertex v_i, denoted k_i^w, is the sum of the weights of all edges connected to that vertex, i.e. $k_i^w = \sum_{v_j \in V} w_{i,j}$. This is often referred to as *strength* in the literature, but this convention clashes with our own terminology introduced later in the paper so we use *weighted degree* to avoid confusion.

- A **connected triplet** consists of three vertices that are connected. The triplet is **open** if if is connected by two edges and **closed** if it is completely connected by three edges. A closed triplet is also called a **triangle**. An example is shown in Figure 2.5.

Social network analysis has revealed various properties that networks exhibit that we would not expect to see if connections between vertices were random or regular like a lattice [80]. Networks tend to exhibit the *small-world effect*: the average path length between any pair of vertices increases only as the log of the total number of vertices in the network. This explains the quick spread of, for example, a viral disease through a large population. Real networks also tend to exhibit a skewed degree distribution where there are many vertices of low degree but a long tail of a small number of vertices of a high degree. Different networks also exhibit differing degree correlations. In some networks there is positive degree correlation where vertices of high degree tend to be connected to other vertices of high degree and low with low (think of a social network in which popular people are friends with other popular people). In other networks there is a negative correlation where vertices of high degree will have more connections with low degree vertices and vice versa (think of a regional hub airport that has many connections to smaller airports in the region that are themselves not connected).

The two properties of networks that are most relevant to our work are transitivity and community structure. Transitivity refers to the tendency of triangles to form in real world networks. If v_j is connected to v_i and v_i is connected to v_k, then it is much more likely that v_j and v_k are connected than we would expect if connections were formed randomly. This is illustrated in Figure 2.5. The level of transitivity, i.e. the proportion of connected triplets that are closed to form triangles, differs from network to network and is measured by the clustering coefficient CL [6]:

$$CL = 3 \times \frac{\text{number of triangles}}{\text{number of connected triplets}} \tag{2.11}$$

Community structure is the tendency of a network to consist of different groups of vertices where the density of edges within the group is much higher than the density of edges between groups. These groups of highly-connected vertices are called communities. A network with a strong community structure is illustrated in Figure 2.6. There is no formal definition of a community and many different methods have been developed to try to find this structure which we review in the next Section 2.2.1. We note how in Figure 2.6 there are many triangles within each community but very few that span multiple communities. This insight into the connection between transitivity and community structure underlies our own novel approach to community detection which is described in Section 2.2.2.

2.2.1 Community Detection and Search

Often we are interested in finding all of the communities in the entire network. This global partitioning of the network into communities is called *community detection* or *community mining*. In other cases, we are only interested in finding the local community of a given vertex. This is called *community search*. While community detection methods were the first to be developed, the emergence of very large networks in our age of big data has given extra importance to local search methods that do not need to detect all of the network's communities.

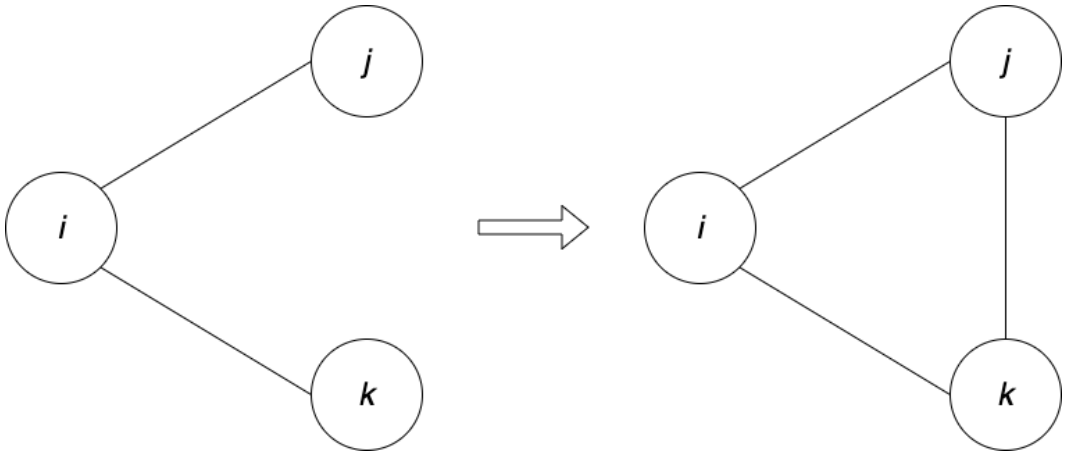


Figure 2.5: Many networks exhibit transitivity, which is the tendency for two vertices that have a common neighbour to themselves be connected. Here we see the closing of a triplet of vertices to form a triangle.

Many different community detection algorithms have been developed over the years. These algorithms vary across several dimensions and different researchers have developed different taxonomies to classify them. In [19], the authors classify algorithms according to the properties or features of the network used by the algorithm, for example those that use directly the density of the edges in a community, those that look for "bridges" as the boundary between communities, and those that look at how information is diffused within the network.

We prefer the classification introduced in [31] and updated in [33]. Algorithms based on optimization have received the most attention of researchers. These algorithms employ a function that measures the quality of a given set of communities and attempt to maximize the value of this function. By far the most popular quality function used is *modularity*, introduced in [83] along with a greedy optimization algorithm which we refer to as FastGreedy. Modularity compares the actual edge structure of the network to a null model where vertices have the same degree but edges are distributed randomly. It is defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{i,j} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j) \tag{2.12}$$

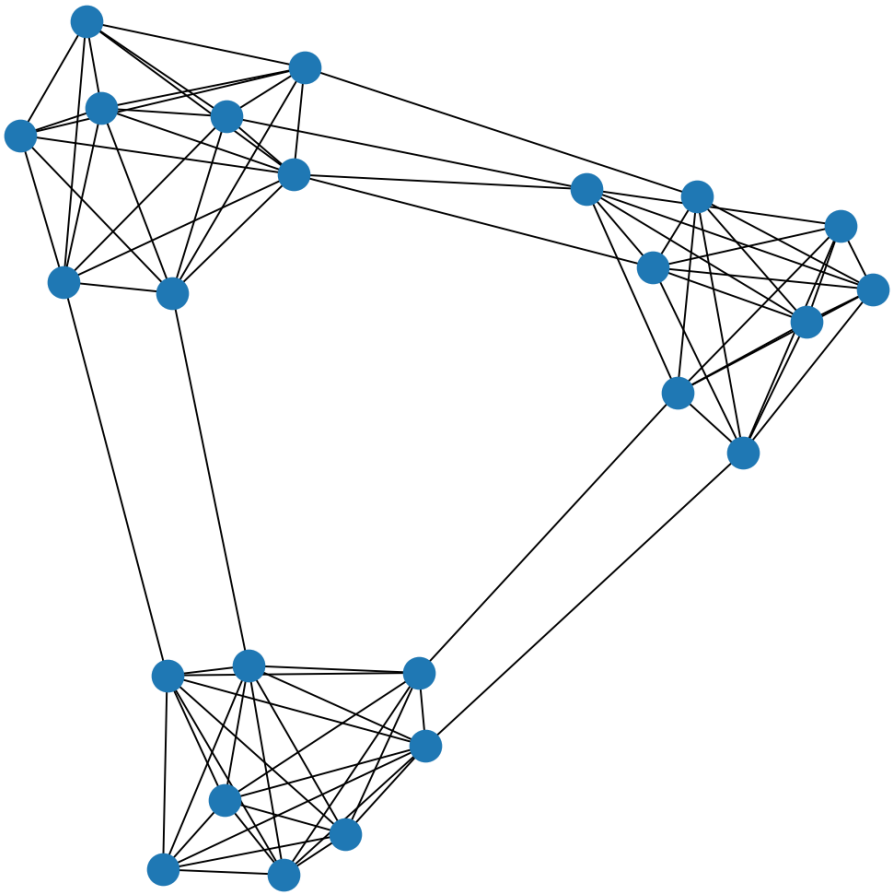


Figure 2.6: An example of a network that consists of three communities. We can see that the density of edges within a community is much higher than the density of edges between communities.

where $A_{i,j}$ is the adjacency matrix entry for v_i and v_j that is equal to 1 when $e_{i,j}$ exists and 0 when $e_{i,j}$ does not exist, C_i and C_j are the assigned communities of v_i and v_j, respectively, and δ is an indicator function that returns 1 when the two arguments are equal and 0 otherwise. The extension of this formula to the weighted case is straightforward; one simply uses the value $w_{i,j}$ from the weighted adjacency matrix and the weighted degree.

The Louvain algorithm [10] is another modularity-optimization algorithm that represented a large step forward in the speed and quality of community detection and is still one of the most popular algorithms in use today. The

algorithm begins with every vertex in its own community and proceeds in two phases that are repeated until no improvements to modularity are possible. In the first phase, the gain in modularity is calculated for each potential move of v_i into the community of each of its neighbours v_j with v_i being moved into the community that provides the greatest increase or not moving if no increase is possible. Then, a new network is constructed by combining each community into a single vertex and the edges between these new vertices have weight equal to the sum of all the combined edge weights and the process repeats on this new network.

The Leiden algorithm [112] is an improvement upon Louvain that improved upon both Louvain's speed and quality. Leiden integrates several previously proposed improvements to Louvain such as the smart local move [113], the fast local move [88], and the random neighbour move [110]. Leiden also employs an extra refinement step to guaranteed that detected communities are well connected, which is not always the case with Louvain.

Methods based on modularity optimization have been shown to be among the best performing community detection algorithms [116]. However, they suffer from two major issues: the resolution limit [32] and the field of view limit [54, 103]. The resolution limit means that small communities cannot be found and are combined into larger communities. The field of view limit is the opposite problem where large communities cannot be found and are broken up into smaller communities. Attempts have been made to mitigate these issues by introducing a resolution parameter [4] to the modularity formula that can be tuned to find communities of larger or smaller sizes. However, this adds a parameter to tune and does not completely solve the problem as using a value of the parameter that reduces the resolution limit will make the field of view limit worse and vice versa. Another way to avoid these problems is to use a different function for optimization that does not suffer from the resolution limit such as the constant Potts model [111].

Another category of algorithms are those that examine how a dynamical process, such as diffusion or a random walk, operates on the network. One such algorithm is WalkTrap [91] which is based on the idea that a short random

walk is likely to get stuck in a community as there are many more edges within the community than those that connect to another community and therefore the verticess encountered on a short random walk are likely to be part of the same community. The Infomap algorithm [98] is also based on random walks. It uses random walks as a model of how information would flow in the network and then uses information theory to find a code that can efficiently describe the information flow while maintaining unique "names" for important features of the network. These important features correspond to the network's communities and finding an efficient code corresponds to finding good communities. The Label Propagation algorithm [95] begins with each vertex assigned a unique label. At each step of the algorithm, each vertex takes the label of the majority of its neighbours, with ties broken randomly. Vertices that are densely connected quickly come to a consensus on their label and vertices with the same label are identified as a community.

Another category is spectral algorithms. These algorithms use the eigenvalues and eigenvectors of some matrix representation of the network to identify communities. The leading eigenvector algorithm of Newman [81] uses the modularity matrix to detect communities. However, spectral methods fail when the network is sparse, which is the case for many real world networks [33].

Community search methods are only concerned with finding the community of a given query vertex, not the community structure of the entire network. This is particularly useful for very large networks where partitioning the entire network would take too long or use too much memory. The tradeoff is that these methods do not have access to global information and must find a community using only local information gained by exploring the surroundings of the query vertex. This means that different algorithms taking different approaches must be used for community search than community detection. Community search algorithms can be grouped into three families: cohesiveness-base algorithms, motif-based algorithms, and optimization-based algorithms.

The first family includes algorithms that attempt to find a connected subgraph that contains the query vertex and satisfies a certain cohesiveness metric.

These metrics are typically relaxations of the clique, which is the maximally cohesive sub-graph where every vertex is connected to every other vertex. The metrics used include the k-core [105] which has been used in an algorithm able to find communities in large networks with different vertex types [30]. Another metric used is the k-truss [18] which has been used in an algorithm able to find communities in large networks that change dynamically over time [44]. A third metric is the k-edge connected component [36] which has also been used in algorithms able to find communities in large networks [43].

The second family of algorithms uses the presence of "motifs" [75], i.e. patterns of connections, to find communities. These motifs occur with higher probability in real complex networks than in random networks and are used as indicators of community structure. The Motif-based Approximate Personalized PageRank (MAPPR) algorithm [117] uses a modified version of PageRank [12, 89] and motif-based clusters to find the community of a query vertex.

As with community detection, there is a family of community search algorithms based on optimizing a quality function. Again, the most popular choice of quality function is modularity but with formula 2.12 adapted to calculate the local modularity of the query vertex and explored surroundings. Modularity R [17] and Modularity M [64] are two algorithms that use different local modularity functions to find the community of a given query vertex.

In the next Section 2.2.2 we review an algorithm that is able to perform both local community search and global community detection.

2.2.2 SIWO: Strong Inside, Weak Outside

The SIWO (Strong Inside, Weak Outside) algorithm was first introduced in [35] for global community detection. The algorithm has been subsequently refined and changed into a local search algorithm that can also perform global community detection. In this section, we describe the current version of the algorithm. As one of the contributions of this thesis is the extension of SIWO to work with weighted networks, we present the algorithm in sufficient detail that our contributions can be understood.

Like motif-based search methods, SIWO uses the presence of higher-order

edge structures, in this case triangles, in the network to identify community structure. However, it is a member of the optimization-base search family as it uses these triangles to compute a strength function for the edges and uses the greedy maximization of the total strength of the edges in a sub-graph to find the community. The following definitions are necessary for the description of the algorithm:

- The **support** of an edge $e_{i,j}$ is the number of triangles of which $e_{i,j}$ forms a part, in other words the number of common neighbours shared by the vertices v_i and v_j and is calculated by:

$$sup_{i,j} = |\{v_k \in V : e_{i,k}, e_{j,l} \in E\}| \tag{2.13}$$

- The **neighbourhood** of a vertex v_i is the set of its adjacent vertices:

$$V_i^N = \{v_j \in V : e_{i,j} \in E\} \tag{2.14}$$

- The **strength** of an edge $e_{i,j}$ quantifies the tendency for the edge to be an intra-community edge versus an inter-community edge, i.e. a high strength value indicates that $e_{i,j}$ connects two vertices in the same community whereas a low strength value indicates that $e_{i,j}$ connects two vertices in different communities. It is calculated by:

$$s_{i,j} = \frac{sup_{i,j}}{sup_{i,max}} + \frac{sup_{i,j}}{sup_{j,max}} - 1 \tag{2.15}$$

  where $sup_{i,max} = \max_{v_j \in V_i^N} sup_{i,j}$. Edge strength is bounded in the range [-1, +1]. It takes on negative values when $e_{i,j}$ is a part of relatively few triangles and positive values when $e_{i,j}$ is part of relatively many triangles.

  The strength of a community C is the sum of strengths of all the edges connecting vertices within the community:

$$s_C = \sum_{v_j, v_i \in C} s_{i,j} \tag{2.16}$$

- The **shell** of a community C is the set of vertices that are not in C but are connected to at least one vertex within C:

$$S_C = \{v_i \in V : v_i \notin C, \exists v_j \in C, \exists e_{i,j} \in E\} \tag{2.17}$$

This is illustrated in Figure 2.7 where the vertices in C are green and the vertices in S_C are purple.

- A **dangling vertex** is a vertex v_i that is connected to exactly one other vertex, i.e. $k_i = 1$. This is illustrated in Figure 2.7 by the red vertex.

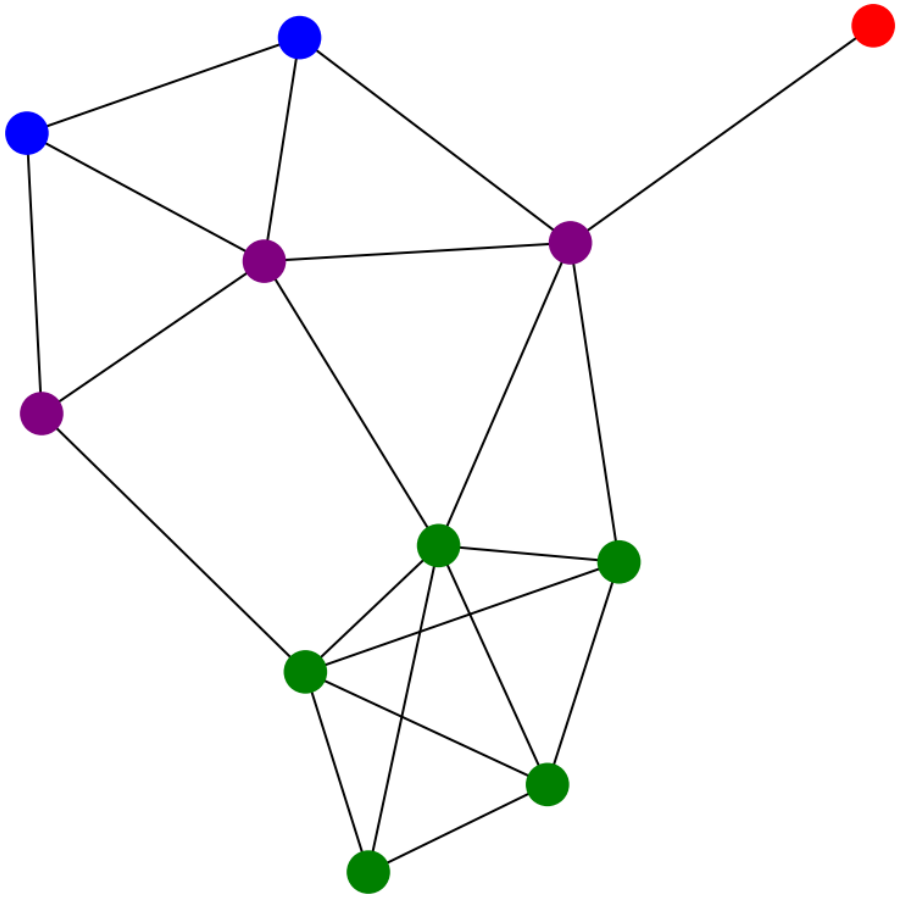


Figure 2.7: A community (colored in green) and its shell (colored in purple). Dangling vertices (colored in red) are special cases as they have no common neighbours with their single neighbour but can only logically belong to the community of that neighbour.

The SIWO algorithm begins by placing the query vertex into the community and all of its neighbours into the shell. Strength values must be computed in Equation 2.15 for all the edges connecting the community to the shell, which

Algorithm 1 SIWO

Require: Network G, query vertex v_q, Boolean $mergeOutliers$

```
C ← {v_q}                                  ▷ Community begins with only the query vertex
v_best ← v_q                               ▷ Track most recently added vertex
S_C ← ∅                                    ▷ Shell is initially empty
while True do
    S_C ← S_C ∪ (V^N_best - C)             ▷ Update shell set
    for e_{i,j} ∈ {e_{i,j} : v_i ∈ C, v_j ∈ S_C} do   ▷ Assign strength values
        if s_{i,j} not calculated then
            Calculate s_{i,j}              ▷ Includes computing required support values
        end if
    end for
    v_best ← argmax_{v∈S_C} s_{C∪{v}}      ▷ Best vertex adds the most strength
    if s_{C∪{v}} ≤ s_C then
        break                              ▷ Stop expanding community if no strength added
    end if
    C ← C ∪ {v_best}
end while
if mergeOutliers then
    for v_i ∈ S_C do                       ▷ Reform community by adding dangling vertices
        if k_i == 1 then
            C ← C ∪ {v_i}
        end if
    end for
end if
return C
```

involves calculating the necessary support values according to Equation 2.13. These values are saved and so only need to be calculated once. Adding a vertex from the shell to the community increases (or decreases) the total community strength by the sum of the strengths of all edges connecting that vertex to the community. Thus selecting the best vertex to add to the community is simply a matter of calculating this sum for each vertex and taking the maximum. In fact, the entire sum does not need to be calculated for each vertex in the shell set every iteration. The previous sum can be saved and updated only when a vertex is added to the community that connects to the shell vertex. Once the best vertex is found, it is added to the community only if its strength contribution is positive. If not, the search stops as no increase to community strength is possible. Optionally, any dangling vertices in the shell can be merged into

the community. The edge connecting a dangling vertex to a community will have a strength value of -1, as it will be part of no triangles, so the vertex will not be added to the community. This is desirable if the user wants to consider them outliers. However, a user may want to merge the dangling vertex into the community as this is the only larger community to which the dangling vertex could possibly belong. The SIWO algorithm is detailed in Algorithm 1.

The SIWO algorithm performs local community search, but can be adapted for global community detection by iteratively performing search on the parts of the graph that have not yet been added to any community. This introduces some stochasticity into the algorithm as the starting vertices for each community search are chosen randomly rather than provided by the user. This variant is called SIWO+ and is detailed in Algorithm 2.

Algorithm 2 SIWO+

Require: Network G, Boolean $mergeOutliers$
  $P \leftarrow \emptyset$ ▷ Set of detected communities
  $U \leftarrow V$ ▷ Set of nodes needing to be added to a community
  while $|U| > 0$ **do**
    $v_q \leftarrow$ vertex uniform randomly selected from U
    $C \leftarrow$ SIWO(G, v_q, $mergeOutliers$)
    $P \leftarrow P \cup \{C\}$
    $U \leftarrow U - C$
  end while
  return P

Both SIWO and SIWO+ have been shown to work well at their respective tasks. However, as defined they only work with unweighted networks. Extending these algorithms to work with weighted networks is one of the contributions of this thesis and is detailed in the next chapter. First, we review evaluation methods for community detection and search.

2.2.3 Evaluation Methods

There are many different methods for evaluating the quality of a clustering or community partition [20, 96]. We cover the most popular methods here. The

evaluation of a detection or search algorithm differs depending on whether the network being used has ground truth communities and, if so, whether these ground truth communities are reliable. When ground truth is available, it is possible to compare the detected community or communities to the ground truth to see how close the detected community structure is to the actual community structure.

When performing community search we only need to compare the detected community of the query vertex to the ground truth community of the query vertex. This can be done using precision which measures the proportion of vertices in the found cluster c that are true members of the ground truth community g:

$$\text{Precision}(c, g) = \frac{|c \cap g|}{|c|} \tag{2.18}$$

Recall measures the proportion of vertices in g that are correctly placed in c:

$$\text{Recall}(c, g) = \frac{|c \cap g|}{|g|} \tag{2.19}$$

These measures are typically combined into the F score:

$$F(c, g) = \frac{2 \times \text{Recall}(c, g) \times \text{Precision}(c, g)}{\text{Recall}(c, g) + \text{Precision}(c, g)} \tag{2.20}$$

Precision, recall and F score are all bounded within the range [0, 1] with higher values indicating a better found community.

When performing community detection, we need to compare the entire detected community structure to all the ground truth communities. To do so we use Normalized Mutual Information (NMI):

$$\text{NMI}(DC, GT) = \frac{-2 \sum_{i=1}^{|DC|} \sum_{j=1}^{|GT|} \mathbf{B}_{i,j} \log\left(\frac{\mathbf{B}_{i,j}|V|}{\mathbf{B}_i \mathbf{B}_j}\right)}{\sum_{i=1}^{|DC|} \mathbf{B}_i \log\left(\frac{\mathbf{B}_i}{|V|}\right) + \sum_{j=1}^{|GT|} \mathbf{B}_j \log\left(\frac{\mathbf{B}_j}{|V|}\right)} \tag{2.21}$$

where DC is the set of detected communities, GT is the set of ground truth communities, $\mathbf{B}$ is a confusion matrix whose rows correspond to the ground

truth communities and columns correspond to the detected communities, $\mathbf{B}_i$ is the sum over row i of $\mathbf{B}$, $\mathbf{B}_j$ is the sum over column j of $\mathbf{B}$, and $\mathbf{B}_{i,j}$ is the number of vertices found in both the i^{th} ground truth community and j^{th} detected community. NMI is bounded within the range [0, 1] with higher values indicating a detected community structure closer to the ground truth.

However, it is often the case that we do not have access to ground truth communities for a network. When working with synthetic networks such as those generated by the LFR benchmark [53, 55] the ground truth is always available. With real world networks, it can be much more difficult to determine the ground truth and even when it is available, there can be questions about its reliability. In some smaller social networks, such as the classic karate [118] and dolphin [65] networks, the researchers documenting the network have chosen to study well-defined communities so the ground truth is reliable. In other cases there is a real-world organizational structure that corresponds to a community structure in a network representation, such as the departments of a research institution or the conferences of a sports league [37]. In other cases it is more difficult to try to define a ground truth. Consider an online social network such as Facebook or Twitter. Ground truth communities can by constructed by forming communities out of users who have "liked" the same post or tweeted the same hashtag, but just because two users both clicked "like" on the new Disney movie trailer or tweeted the same "#blacklivesmatter" hashtag does not mean that they ever interact or have any connection themselves.

In cases where no reliable ground truth is available one must use different metrics that attempt to measure the intrinsic quality of the detected communities. One such measure is called silhouette and can be used to both provide a scalar measure of how well an object lies within its cluster as well as visualize the relative quality of clusters [99]. Silhouette was developed for evaluating clusterings and so uses the language of generic objects and dissimilarities. To calculate the silhouette score $s(i)$ for some object i one must calculate the average dissimilarity of i to all other objects in its own cluster A:

$$a(i) = \text{average dissimilarity of } i \text{ to all other objects in } A \tag{2.22}$$

For every other cluster $B \neq A$ one must calculate the average dissimilarity of i to all objects in B:

$$d(i, B) = \text{average dissimilarity of } i \text{ to all objects in } B \tag{2.23}$$

Then one takes the smallest average dissimilarity of all the other clusters:

$$b(i) = \text{minimum}_{B \neq A} d(i, B) \tag{2.24}$$

Then the silhouette score is calculated as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2.25}$$

The silhouette score is bounded in the range [-1, +1]. Scores close to +1 indicate that i is much more similar to the other objects in its own clustering than the objects in the next most similar clustering, indicating that it has been properly clustered. Values close to 0 indicate that i is about as similar to the objects in its own cluster as another cluster and could fit in either. Values close to -1 indicate that i is more similar to objects in another cluster and is therefore assigned to the wrong cluster.

When clustering points in a Euclidean space there is a clear dissimilarity measure to use: distance. However, for vertices in a network there is no such obvious choice. Several different possible choices for the distance between two vertices $dist(v_i, v_j)$ were proposed in [93, 94] including:

- the inverse edge weight,

$$dist(v_i, v_j) = \frac{1}{w_{i,j}} \tag{2.26}$$

- the maximum edge weight minus the edge weight,

$$dist(v_i, v_j) = \max_{v_x, v_y}(w_{x,y}) - w_{i,j} \tag{2.27}$$

- the shortest path between the vertices,
- the adjacency relation,

$$dist(v_i, v_j) = \sqrt{\sum_{x \neq i,j} (w_{i,x} - w_{j,x})^2} \tag{2.28}$$

- the neighbourhood overlap,

$$dist(v_i, v_j) = 1 - \frac{\sum_x w_{i,x} w_{j,x}}{\sum_x \left(w_{i,x}^2 + w_{j,x}^2 - w_{i,x} w_{j,x}\right)} \tag{2.29}$$

- the topological overlap,

$$dist(v_i, v_j) = 1 - \frac{\sum_{x \neq i,j} (w_{i,x} w_{j,x}) + w_{i,j}^2}{\min\left(\sum_x w_{i,x}^2, \sum_x w_{j,x}^2\right)} \tag{2.30}$$

- the Pearson correlation where μ_i is the average of all weights of edges adjacent to v_i and σ_i is their standard deviation,

$$dist(v_i, v_j) = 1 - \frac{\sum_x (w_{i,x} - \mu_i)(w_{j,x} - \mu_j)}{|V| \sigma_i \sigma_j} \tag{2.31}$$

- and the number of paths shorter than a certain length between them.

Now that we have reviewed social network analysis, community detection and search, and the SIWO algorithm, we move on to discuss the first contribution of this thesis: the extension of SIWO to handle weighted networks.

Chapter 3

SIWOw

The topological structure of a network, i.e. the arrangement of vertices and edges, can provide a great deal of information about the system represented by a network. However, many complex systems are defined not only by the presence or absence of connections between their elements but also by the intensity of those connections. When we represent these systems as unweighted networks for study and analysis, we are missing a crucial dimension of the underlying real system. A friendship in a social network is not simply a binary relationship but is characterized by an intensity of feelings and interaction which can be quantified and added as an edge weight. The various channels in an information network are not all equal but have varying capacities which make them more or less important to the flow of information. Often there will be a positive correlation between the areas of the network where there is a high density of connections and higher edge weights. However, the correlations between edge weights and topology can vary between networks and so taking the edge weights into consideration is crucial to understanding and analyzing the network and the system it represents [5].

There is a risk in network analysis of being misled when only considering network topology and ignoring edge weights, which is highlighted by the study of important airports using network betweenness centrality conducted in [85]. Betweenness centrality measures the importance of a vertex in a network using the number of shortest paths between pairs of vertices that run through that vertex. Using this measure on an unweighted airport network,

where edges represent the presence of a flight route between the two airport vertices, the authors found that the Anchorage, Alaska airport was the third most important in the world, after Frankfurt and Paris. Of course, it does not seem sensible that the airport of a relatively small and isolated city in the far northern reaches of North America would be one of the most important in the entire world. This is the result of ignoring edge weights. A route between Anchorage and Fairbanks, Alaska is not the same as a route between New York and London. The former will have less frequent flights and use smaller airplanes that can carry fewer passengers. In the case of an airport network, the "length" of a shortest path should be influenced by the number of passengers that can flow between the vertices. Using edge weights that reflect passenger capacity, the authors find that the three most important airports are London, Los Angeles, and New York.

Considering edge weights would also be key for our goal of applying social network analysis and community mining to term co-occurrence networks for topic modelling. There is a great difference between a pair of terms that co-occur in one sentence in a corpus of tens of thousands of documents and a pair of terms that co-occur in hundreds of sentences. Without assigning and using edge weights, this difference is ignored. Very few of the local community search algorithms discussed in the previous chapter are able to handle weighted networks but all of the discussed global community detection algorithms are. As presented in Chapter 2, the SIWO and SIWO+ algorithms are not able to handle weighted networks. Thus the first contribution of this thesis is to extend SIWO (and so SIWO+) to be able to work with weighted networks. This contribution provides the community with one of the few weighted community search algorithms, gives SIWO+ functionality that is expected for community detection, and allows us to use this algorithm for our community mining-based approach to topic modelling. We call the extended algorithms SIWOw and SIWOw+. In this chapter we explain our approach to handling weighted networks and present experiments that demonstrate the competitiveness of this approach.

3.1 Weighted Support

Many of the properties of edges, vertices, and networks have been generalized to the weighted case and many community detection algorithms have been extended to work with weighted networks. For example, the modularity formula in Equation 2.12 can handle weighted edges simply by using the weighted degrees k_i^w and k_j^w in place of k_i and k_j and having $m = \sum_{i,j} w_{i,j}$, *et voilà*, the Louvain and Leiden algorithms can handle weighted networks.

The SIWO algorithm does not only consider edges but higher-order structures, i.e. triangles, so the extension of the algorithm to handle edge weights is less straightforward. Fortunately, as we have already noted, there is a similarity between SIWO's support and strength notions and the clustering coefficent CL (formula 2.11). Researchers have already worked on several approaches for generalizing CL to the weighted case [5, 41, 84, 86, 119] so we are able to build off of their work to extend SIWO.

Given that there are several approaches, the authors of [102] compared these various weighted clustering coefficients both by qualitatively evaluating their properties and quantitatively evaluating their values on various networks. They concluded that rather than there being one ultimate version of the weighted clustering coefficient, the different measures capture different aspects of the networks. While none of these measures are perfectly suited for our purposes, some have properties that make them less appropriate for use in SIWO. The measures of [6, 86, 119] do not take into account the weights of all three edges of the triangle, only two. Our notions of support and strength are edge centric, not vertex centric, so we want to use all three edge weights, not only the two connected to a certain vertex. The measure of [41] is not equivalent to the unweighted value when all weights are equal to 1. This is problematic as we would like SIWO to find the same communities in the unweighted case and the equivalent weighted network with all edge weights equal to 1. This leaves the measure of [84] as the best fit but with two changes. We do not normalize the edge weights by the maximum edge weight of the network as that global information is not available to a local search algorithm, and we

do not normalize by the degree of the connected vertex as we are finding a value for an edge and the SIWO strength formula in Equation 2.15 includes normalization of the support values. The modification of SIWO to be able to handle edge weights is thus as simple as changing the support calculation from a counting of triangles to a sum over triangle values:

$$sup_{i,j} = \sum_{v_k \in V_i^N \cap V_j^N} f(w_{i,j}, w_{i,k}, w_{j,k}) \tag{3.1}$$

where f is one of the arithmetic mean, geometric mean, harmonic mean, and minimum. Most of the weighted clustering coefficients use the geometric mean. We would like to evaluate different functions as they have varying sensitivity to very small or very large outlier edge weights and may find different community structures.

Other than using Equation 3.1 in place of Equation 2.13 for the support calculation, the SIWOw (SIWOw+) algorithm remains identical to the SIWO (SIWO+) algorithm. The strength formula in Equation 2.15 still normalizes to values in the range [-1, +1] regardless of the magnitude of the edge weights. Now that we have a version of SIWO that can handle networks with weighted edges, we empirically evaluate the algorithm on both real-world and synthetic networks.

3.2 Weighted Karate Network

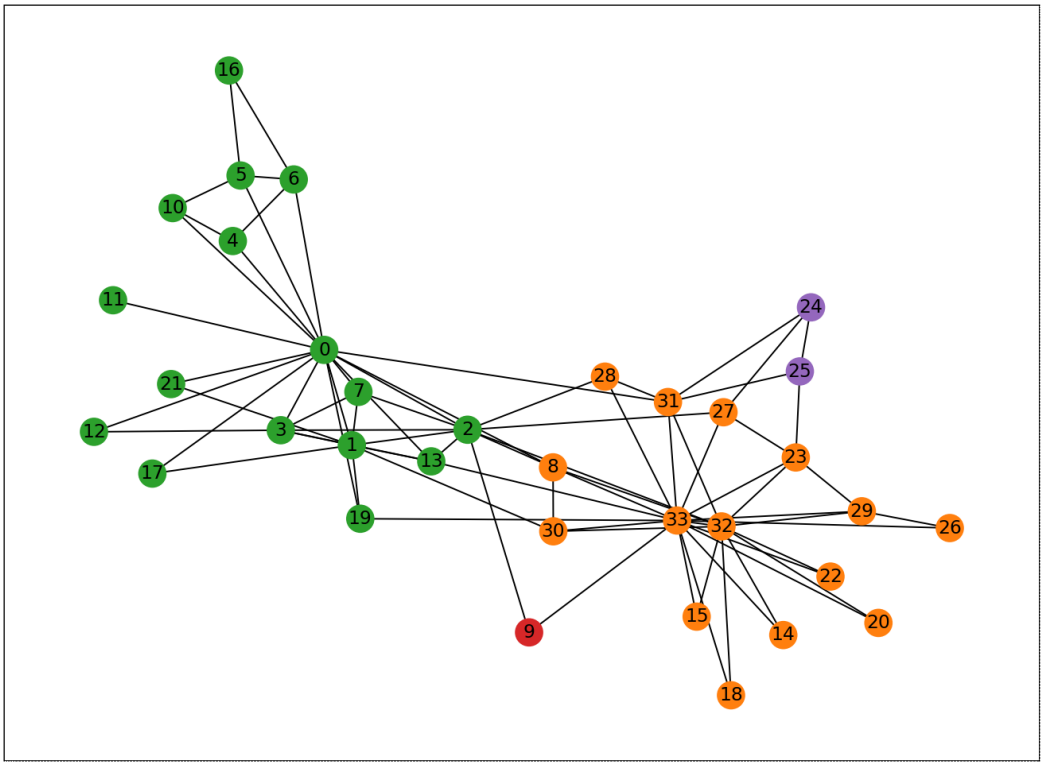
Unfortunately, there is a distinct lack of real-world networks with both weighted edges and ground truth communities. Fortunately, one of the oldest and most widely used networks in the literature, Zachary's karate network [118], has both. The karate network is a social network representing a university karate club that split into two new clubs following a dispute between the instructor and club president. The vertices in the network represent the individuals in the clubs. Edges exist between individuals who have personal relationships. The weights given to edges are the number of contexts in which the two individuals interact, e.g. if individuals 1 and 2 hang out on campus, attend the

same karate lessons, and meet at the student bar, then $w_{1,2} = 3$. The edge weights provide useful information about the intensity of the relationships and are thus valuable for detecting the communities of the social network.

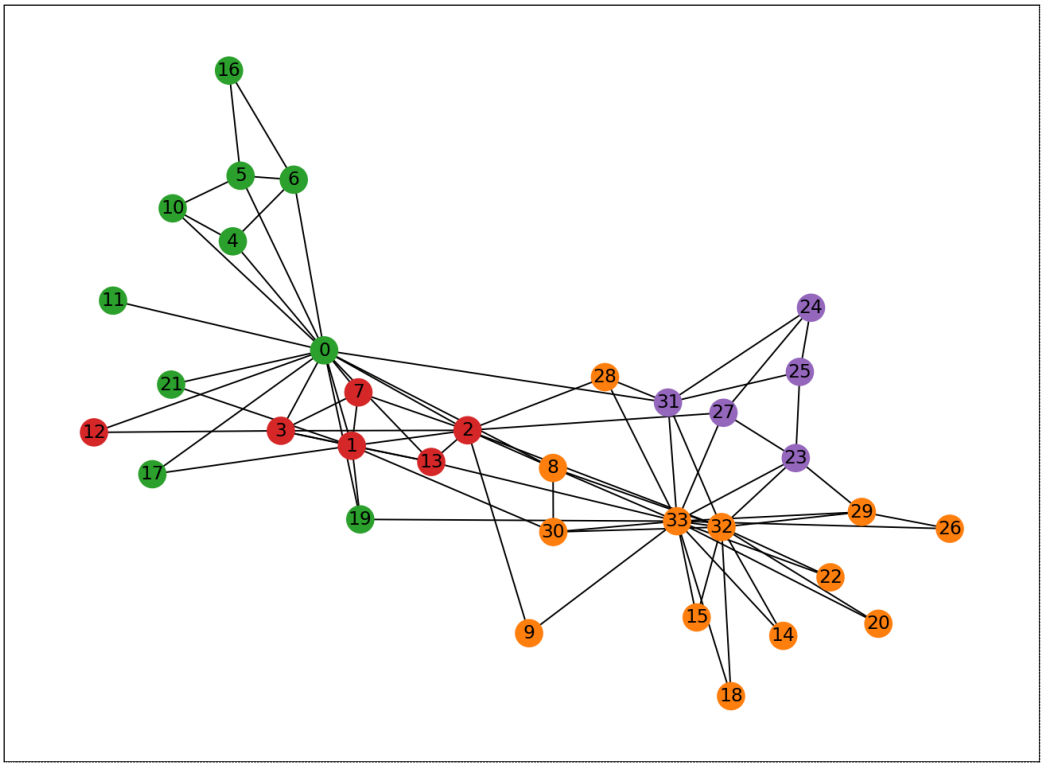
While some authors use this network as the sole dataset for evaluation [1], we believe that nothing definitive can be said about a community detection or search algorithm using the results from a single, small network. Also, many algorithms have an element of randomness and will give different results on the same network. Evaluations should be conducted over a variety of networks with multiple runs on each network to give the results some statistical power. We conduct these types of experiments in the next section. Here, we use a single run of several algorithms on the karate network to not only show that SIWOw+ can find high quality communities but also to be able to visualize the results and analyze some of the qualitative differences between the communities found by SIWOw+ and other algorithms.

We compare SIWOw+ (using the geometric mean) to Louvain, Leiden, and Infomap. We use SIWOw+ as the network is small and thus global community detection presents no issues and most weighted algorithms are also global community detection algorithms. The performance in terms of NMI is presented in Table 3.1. We can see that SIWOw+ does the best job of finding communities that are close to the ground truth. However, one run on one network is far from enough evidence to declare SIWOw+ "better" than the others. Infomap performs nearly as well as SIWOw+. Louvain and Leiden have very similar performance that is significantly below that of SIWOw+ and Infomap. Since the karate network is small, we can go beyond comparing summary metrics and visualize the results of the community detection to get a sense of how the algorithms work, where they fail, and what their differences are.

The communities found by each of the four algorithms are illustrated in Figures 3.1 and 3.2. In Figure 3.1a we can see the community structure detected by SIWOw+. SIWOw+ does a good job detecting the community of the karate instructor, coloured green. The only missed vertex is v_8, which has been incorrectly assigned to the community of the club administrator, coloured orange. However, all the algorithms make this mistake and v_8 actually has more

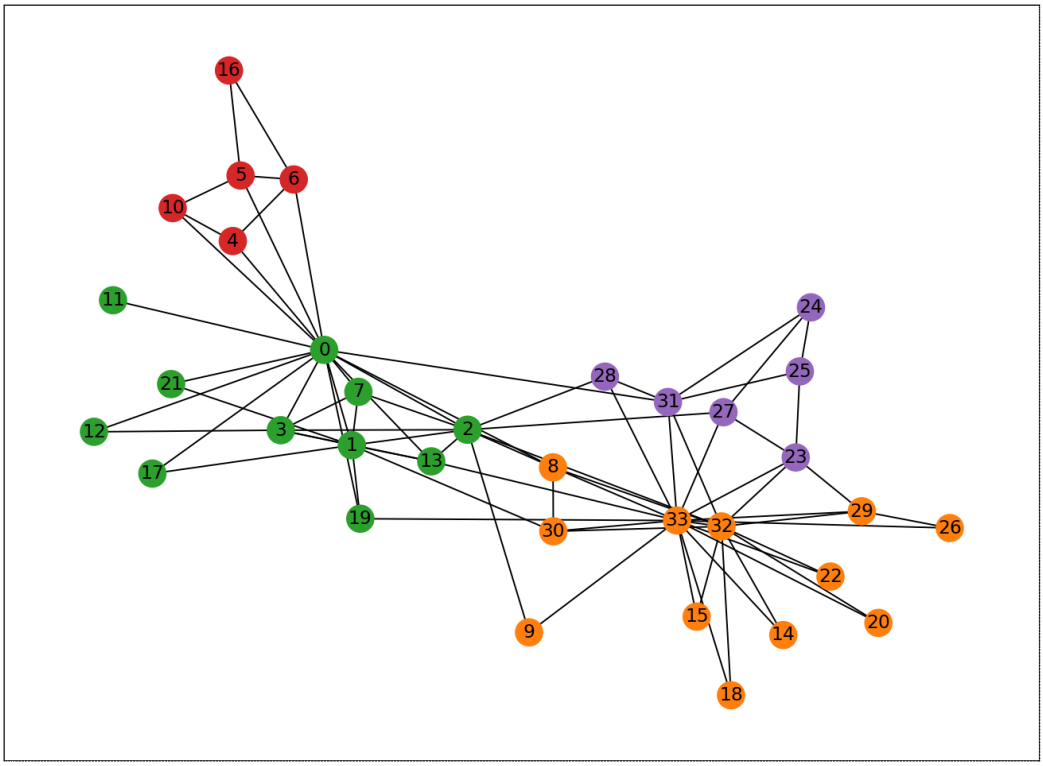


(a) SIWO communities

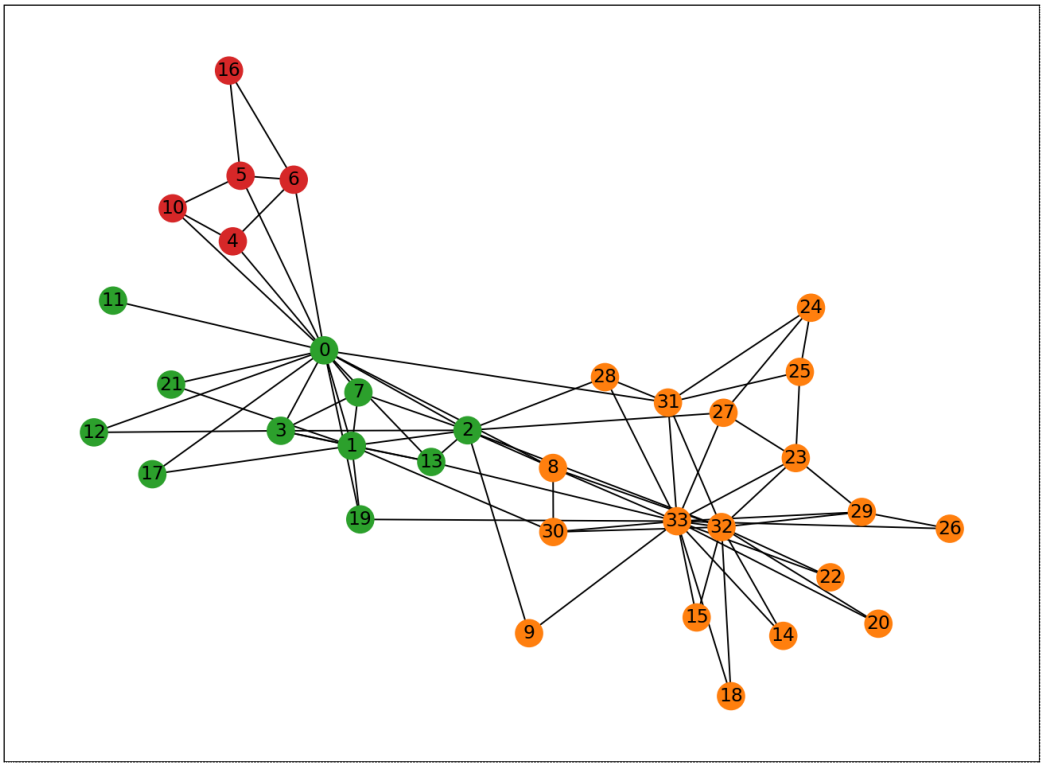


(b) Louvain communities

Figure 3.1: Community structure of weighted karate network detected by SIWO and Louvain



(a) Leiden communities



(b) Infomap communities

Figure 3.2: Community structure of weighted karate network detected by Leiden and Infomap

Algorithm	NMI
SIWOw+	0.697
Louvain	0.587
Leiden	0.588
Infomap	0.691

Table 3.1: NMI scores of four different community detection algorithms on the weighted karate network

connections to the orange community than the green so this may be a case of a vertex that could belong to either community or a mistake in the ground truth. Vertex v_{11} is a dangling vertex and has been correctly assigned to the green community thanks to the merge outliers step of the algorithm. However, SIWOw+ has found two small communities, $\{v_9\}$ and $\{v_{24}, v_{25}\}$, that the ground truth includes in the orange community. SIWOw+ does not assign v_9 to either ground truth community as it is not connected to any vertices that are themselves connected, i.e. neither edge connected to v_9 forms part of any triangle. Thus $s_{9,2} = -1$ and $s_{9,33} = -1$ so adding v_9 to any community would reduce that community's total strength. The same is true for v_{24} and v_{25}. They each have multiple connections to the orange community but none of the edges form parts of any triangles.

These failure cases could be addressed by adding another post-processing merge step to SIWO+. In the case of v_9, the edge weights could be used as a tie-breaker to include it in a larger community. As $w_{9,33} = 3 > w_{9,2} = 2$, v_9 would be correctly added to the orange community, increasing NMI. In the case of v_{24} and v_{25}, a rule that merged a small community with a larger community if all the vertices in the small community are only externally connected to that larger community would correctly add them to the orange community, increasing NMI. However, adding these sort of rules adds complexity to the algorithm and while this type of information may be available once the global community detection is complete, when using SIWO as a local search algorithm the nature of the connections between vertices in the shell set and other communities is unknown. We also run the risk of myopically trying to optimize a single number and overfitting to a dataset. It may actually be beneficial for our

downstream application of community mining, topic modelling, that these types of peripheral vertices are excluded from larger communities. Topics are meant to represent concepts or ideas and should be easy for humans to identify and label. Adding in terms that are only loosely connected to the core concept dilutes the topic and makes it less interpretable and coherent. So while one can think of ways to modify the algorithm to force these types of vertices into larger communities, for the purposes of topic modelling we view this as a feature, not a bug.

The communities found by the Louvain algorithm are illustrated in Figure 3.1b. We can see that Louvain breaks up each ground truth community into two smaller communities. Like SIWOw+, Louvain does not include v_{24} and v_{25} into the orange community, but also groups in three more vertices with them, even though those vertices have more connections to the orange community than v_{24} and v_{25}. The green community looks less coherent as a group and in this case the lower NMI score seems justified.

In Figure 3.2a, we see that the Leiden algorithm also splits each ground truth community into two smaller communities. Here the split of the green community makes more sense as the red vertices are only connected to one green vertex, v_0. As with Louvain, several of the vertices that should be in the orange community are in the purple community. This may be the result of modularity's resolution limit which makes it difficult to detect small communities such as $\{v_{24}, v_{25}\}$ as SIWOw+ did.

In Figure 3.2b, we see that the Infomap algorithm does a very good job at finding the orange community. Infomap splits the green community in two exactly as Leiden does. In Leiden's case, the modularity is lower if the red and green vertices are in the same community as there is a very low density of edges between them. In Infomap's case, a random walker is unlikely to visit both red and green vertices on the same walk as there is a bottleneck between them. It is SIWO's use of the higher-order structure of triangles that allows the algorithm to join them into one community. Most vertices in the red community have multiple common neighbours with v_0, so their connecting edges are strong and they should be in the same community. Similarly, most

other vertices in the green community have common neighbours with v_0, so their connecting edges are strong and they should be in the same community. Thus SIWO recognizes v_0 as a hub of a single community, which aligns with v_0's role in the real network as karate instructor and central figure of one of the two clubs.

Of course, only so much can be said about the quality of an algorithm when using a single dataset for evaluation. In the next section, we will conduct a more thorough empirical evaluation of several algorithms on a wide range of benchmark networks.

3.3 Evaluation on LFR Benchmark

Given the lack of real-world networks with ground truth communities, we must turn to synthetic substitutes for comprehensive algorithm evaluation. Early synthetic network generators such as that used in [37] created small networks with unrealistic structures such as equal sizes for all communities and equal degrees for all vertices. To remedy this, Lancichinetti, Fortunato, and Radicchi introduce their LFR benchmark generator [55] which is able to generate networks whose community size and degree distributions follow power laws and are thus more realistic. They later extended their benchmark to generate weighted networks [53]. While there are some properties of real networks that the generator does not capture, such as degree correlations, efforts to improve the benchmark have fallen short as it is difficult to control the various network properties simultaneously [87] and the LFR benchmark remains the standard. The dynamics of the complex systems represented by real-world networks are probably impossible to capture with a handful of equations so we cannot expect perfect realism from synthetically generated networks. Knowing this, we should evaluate our algorithms on a varied set of networks generated with different parameters to lessen the risk that our results are based on some unrealistic features of a particular network or parameter setting.

In this section, we evaluate SIWOw+ against eight other community detection algorithms. We are evaluating using community detection as most

community search algorithms cannot handle weighted networks and the ones that do are older and tend to perform less well on unweighted networks compared to their more modern peers. Most detection algorithms can handle weighted networks so evaluating SIWOw+ allows for a more comprehensive comparison. Also, our topic modelling application will use global community detection as vocabularies tend to be a few thousand or tens of thousands of terms, so the co-occurrence networks will be small enough that a local approach is not needed. As SIWOw+ is simply an iteration of the local SIWOw search, we can infer the performance of the search from the quality of the global detection.

3.3.1 Datasets

We generated a total of 1,530 weighted synthetic networks of 10,000 vertices using the LFR benchmark generator. Five networks were generated for each parameter combination of average degree in {15, 20, 25}, maximum degree in {50, 75, 100}, exponent for weight distribution β in {1.5, 2}, and topology mixing parameter μ_t and weight mixing parameter μ_w in {0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90}. The average and maximum degree control the density of the network as well as the distribution of community sizes. β controls how the edge weights vary with the degree of a vertex. The mixing parameters control the proportion of the edges and total edge weight of a vertex that are connected to vertices outside its own community, e.g. $\mu_t = 0.30$ means that on average 30% of a vertex's edges will connect to a vertex in a different community and $\mu_w = 0.60$ means that on average 60% of the total edge weights connected to a vertex will be on edges connected to vertices in a different community. A higher mixing parameter results in a less well-defined community structure and a more challenging network for community detection.

3.3.2 Algorithms

We compare SIWOw+ using the arithmetic mean, geometric mean, harmonic mean, and minimum against Infomap, WalkTrap, Label Propagation, Louvain,

Fastgreedy, Leading Eigenvector, and Leiden using both modularity and CPM optimization functions. Leiden has a resolution parameter that can vary the scale of the communities found to combat either the resolution or field of view limits. A simple grid search found the best values of 3.0 for modularity and 0.03 for CPM.

3.3.3 Results

Table 3.2 presents the results of the community detection of all algorithms on all 1,530 synthetic networks. The results are grouped by the resolution parameter so each column is an average over 90 networks at a particular value of $\mu_t = \mu_w$. The average modularity Q of the ground truth community is presented to illustrate the degradation of the true community structure as the mixing parameter increases which makes community detection more difficult. The average NMI plus/minus the standard deviation is given for each algorithm with the best score bolded. SIWOw+ with arithmetic mean, geometric mean, harmonic mean and minimum are denoted SIWOw+(a), SIWOw+(g), SIWOw+(h), SIWOw+(m), respectively. Leiden using modularity and CPM are denoted Leiden(m) and Leiden(c), respectively. We can see from the table that all the algorithms perform best when μ_w is small and that performance drops as μ_w increases, although the pattern of decline is different for the different algorithms. Infomap, SIWOw+, and WalkTrap begin as the best performers. Infomap remains the best performer across the range of μ_w values. SIWOw+ performs just as well as Infomap until $\mu_w = 0.40$ and remains a close second until $\mu_w = 0.65$. Past this point WalkTrap and Leiden outperform SIWOw+. The older algorithms, Fastgreedy and Leading Eigenvector, are by far the worst performers. It is interesting to see that the random walk-based algorithms, Infomap and WalkTrap, outperform the modularity-based methods, Leiden(m), Louvain, Fastgreedy, and Leading Eigenvector.

The difference between the different versions of SIWOw+ is small. For smaller values of μ_w there is no real difference. Only at $\mu_w = 0.65$ does the difference start to become significant. Interestingly, the ranking of the different means by performance matches their ranking by sensitivity to large

	$\mu_w = 0.10$ $Q = 0.888$	$\mu_w = 0.15$ $Q = 0.841$	$\mu_w = 0.20$ $Q = 0.792$	$\mu_w = 0.25$ $Q = 0.743$	$\mu_w = 0.30$ $Q = 0.693$	$\mu_w = 0.35$ $Q = 0.643$	$\mu_w = 0.40$ $Q = 0.594$	$\mu_w = 0.45$ $Q = 0.544$	$\mu_w = 0.50$ $Q = 0.494$
Infomap	$\mathbf{0.999 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	0.999 ± 0.000	$\mathbf{0.999 \pm 0.000}$	$\mathbf{0.998 \pm 0.001}$	$\mathbf{0.997 \pm 0.001}$	$\mathbf{0.992 \pm 0.002}$
SIWOw+(a)	$\mathbf{0.999 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.999 \pm 0.000}$	$\mathbf{0.998 \pm 0.000}$	0.995 ± 0.001	0.987 ± 0.001
SIWOw+(g)	$\mathbf{0.999 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.999 \pm 0.000}$	$\mathbf{0.998 \pm 0.000}$	0.995 ± 0.001	0.987 ± 0.001
SIWOw+(h)	$\mathbf{0.999 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.999 \pm 0.000}$	$\mathbf{0.998 \pm 0.000}$	0.995 ± 0.001	0.987 ± 0.001
SIWOw+(m)	$\mathbf{0.999 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	0.999 ± 0.000	$\mathbf{0.999 \pm 0.000}$	$\mathbf{0.998 \pm 0.000}$	0.995 ± 0.001	0.986 ± 0.002
WalkTrap	$\mathbf{0.999 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	0.999 ± 0.000	0.997 ± 0.001	0.994 ± 0.001	0.989 ± 0.002	0.981 ± 0.003	0.965 ± 0.004
Label Prop.	0.995 ± 0.000	0.992 ± 0.001	0.987 ± 0.001	0.981 ± 0.001	0.974 ± 0.001	0.968 ± 0.001	0.962 ± 0.002	0.957 ± 0.003	0.947 ± 0.004
Leiden(m)	0.966 ± 0.001	0.964 ± 0.002	0.963 ± 0.002	0.962 ± 0.002	0.960 ± 0.002	0.955 ± 0.002	0.951 ± 0.002	0.946 ± 0.003	0.937 ± 0.004
Louvain	0.924 ± 0.002	0.915 ± 0.002	0.907 ± 0.003	0.898 ± 0.003	0.888 ± 0.003	0.877 ± 0.003	0.865 ± 0.003	0.851 ± 0.004	0.831 ± 0.005
Leiden(c)	0.945 ± 0.006	0.925 ± 0.009	0.902 ± 0.011	0.877 ± 0.015	0.851 ± 0.018	0.821 ± 0.021	0.794 ± 0.024	0.765 ± 0.027	0.737 ± 0.028
Fastgreedy	0.773 ± 0.006	0.769 ± 0.005	0.778 ± 0.005	0.785 ± 0.004	0.773 ± 0.003	0.773 ± 0.003	0.762 ± 0.003	0.749 ± 0.004	0.721 ± 0.004
Leading eig.	0.639 ± 0.010	0.576 ± 0.009	0.520 ± 0.007	0.486 ± 0.007	0.394 ± 0.006	0.394 ± 0.006	0.354 ± 0.005	0.317 ± 0.005	0.274 ± 0.004
	$\mu_w = 0.55$ $Q = 0.445$	$\mu_w = 0.60$ $Q = 0.395$	$\mu_w = 0.65$ $Q = 0.345$	$\mu_w = 0.70$ $Q = 0.295$	$\mu_w = 0.75$ $Q = 0.246$	$\mu_w = 0.80$ $Q = 0.197$	$\mu_w = 0.85$ $Q = 0.148$	$\mu_w = 0.90$ $Q = 0.100$	
Infomap	$\mathbf{0.978 \pm 0.004}$	$\mathbf{0.956 \pm 0.008}$	$\mathbf{0.923 \pm 0.011}$	$\mathbf{0.874 \pm 0.015}$	$\mathbf{0.804 \pm 0.018}$	$\mathbf{0.705 \pm 0.019}$	$\mathbf{0.560 \pm 0.010}$	$\mathbf{0.494 \pm 0.006}$	
SIWOw+(a)	0.969 ± 0.004	0.944 ± 0.005	0.893 ± 0.007	0.790 ± 0.009	0.609 ± 0.008	0.400 ± 0.005	0.239 ± 0.003	0.167 ± 0.004	
SIWOw+(g)	0.968 ± 0.004	0.941 ± 0.006	0.885 ± 0.008	0.772 ± 0.011	0.577 ± 0.010	0.367 ± 0.006	0.225 ± 0.004	0.170 ± 0.004	
SIWOw+(h)	0.966 ± 0.004	0.938 ± 0.006	0.881 ± 0.009	0.762 ± 0.011	0.562 ± 0.011	0.358 ± 0.007	0.224 ± 0.004	0.172 ± 0.004	
SIWOw+(m)	0.966 ± 0.004	0.937 ± 0.006	0.877 ± 0.009	0.754 ± 0.012	0.549 ± 0.011	0.348 ± 0.007	0.224 ± 0.004	0.178 ± 0.005	
WalkTrap	0.940 ± 0.005	0.902 ± 0.007	0.845 ± 0.008	0.777 ± 0.009	0.702 ± 0.008	0.628 ± 0.005	0.557 ± 0.005	0.518 ± 0.006	
Label Prop.	0.926 ± 0.008	0.904 ± 0.011	0.870 ± 0.014	0.811 ± 0.018	0.655 ± 0.028	0.293 ± 0.029	0.205 ± 0.026	0.133 ± 0.022	
Leiden(m)	0.914 ± 0.008	0.891 ± 0.011	0.857 ± 0.014	0.801 ± 0.018	0.720 ± 0.021	0.598 ± 0.020	0.431 ± 0.009	0.370 ± 0.005	
Louvain	0.796 ± 0.009	0.758 ± 0.012	0.706 ± 0.015	0.624 ± 0.019	0.510 ± 0.021	0.351 ± 0.016	0.196 ± 0.004	0.142 ± 0.003	
Leiden(c)	0.705 ± 0.030	0.682 ± 0.031	0.656 ± 0.031	0.626 ± 0.030	0.582 ± 0.028	0.507 ± 0.023	0.403 ± 0.010	0.361 ± 0.008	
Fastgreedy	0.673 ± 0.007	0.611 ± 0.009	0.518 ± 0.011	0.392 ± 0.010	0.251 ± 0.006	0.151 ± 0.002	0.100 ± 0.001	0.074 ± 0.001	
Leading eig.	0.229 ± 0.003	0.187 ± 0.003	0.157 ± 0.002	0.129 ± 0.003	0.105 ± 0.003	0.086 ± 0.002	0.071 ± 0.002	0.057 ± 0.002	

Table 3.2: Average NMI scores $\pm$ the standard deviation computed over 90 runs, as a function of mixing parameters $\mu_w = \mu_t$. Bold numbers indicate the best score for each dataset. Modularity of the ground truth communities Q indicates level of community structure available for detection.

	$\mu_w = 0.10$ $\bar{N} = 339 \pm 6$	$\mu_w = 0.15$ $\bar{N} = 343 \pm 6$	$\mu_w = 0.20$ $\bar{N} = 345 \pm 6$	$\mu_w = 0.25$ $\bar{N} = 344 \pm 7$	$\mu_w = 0.30$ $\bar{N} = 344 \pm 6$	$\mu_w = 0.35$ $\bar{N} = 345 \pm 7$	$\mu_w = 0.40$ $\bar{N} = 343 \pm 6$	$\mu_w = 0.45$ $\bar{N} = 342 \pm 7$	$\mu_w = 0.50$ $\bar{N} = 345 \pm 6$
Infomap	**339 ± 6**	**343 ± 6**	**345 ± 6**	**344 ± 7**	**344 ± 6**	**346 ± 7**	**344 ± 6**	**345 ± 7**	**351 ± 6**
SIWOw+(a)	**339 ± 6**	**344 ± 6**	**346 ± 6**	**345 ± 7**	**346 ± 6**	**349 ± 7**	**350 ± 7**	**358 ± 7**	369 ± 7
SIWOw+(g)	**339 ± 6**	**344 ± 6**	**346 ± 6**	**345 ± 7**	**347 ± 6**	**349 ± 7**	**351 ± 7**	**357 ± 7**	371 ± 7
SIWOw+(h)	**340 ± 6**	**345 ± 6**	**347 ± 6**	**346 ± 7**	**347 ± 6**	**350 ± 7**	**353 ± 7**	360 ± 7	374 ± 7
SIWOw+(m)	**341 ± 6**	**346 ± 6**	**348 ± 6**	**348 ± 7**	**349 ± 6**	**352 ± 7**	**354 ± 7**	362 ± 7	376 ± 7
WalkTrap	**339 ± 6**	**343 ± 6**	**345 ± 6**	**345 ± 7**	**346 ± 6**	**352 ± 7**	**355 ± 8**	362 ± 7	370 ± 12
Label Prop.	329 ± 6	325 ± 6	312 ± 6	293 ± 6	278 ± 6	267 ± 6	256 ± 6	249 ± 6	250 ± 7
Leiden(m)	206 ± 4	203 ± 4	203 ± 5	202 ± 4	201 ± 4	196 ± 4	194 ± 4	190 ± 4	188 ± 4
Louvain	134 ± 3	116 ± 2	116 ± 2	108 ± 2	100 ± 2	93 ± 2	85 ± 2	78 ± 1	71 ± 1
Leiden(c)	236 ± 11	252 ± 12	230 ± 11	252 ± 12	270 ± 18	308 ± 22	343 ± 29	362 ± 31	400 ± 35
Fastgreedy	58 ± 1	55 ± 1	55 ± 1	58 ± 1	57 ± 1	54 ± 1	50 ± 1	47 ± 1	41 ± 0
Leading eig.	178 ± 8	111 ± 5	111 ± 5	102 ± 5	78 ± 4	61 ± 3	46 ± 3	36 ± 2	30 ± 2
	$\mu_w = 0.55$ $\bar{N} = 345 \pm 7$	$\mu_w = 0.60$ $\bar{N} = 345 \pm 7$	$\mu_w = 0.65$ $\bar{N} = 342 \pm 6$	$\mu_w = 0.70$ $\bar{N} = 345 \pm 7$	$\mu_w = 0.75$ $\bar{N} = 345 \pm 7$	$\mu_w = 0.80$ $\bar{N} = 344 \pm 6$	$\mu_w = 0.85$ $\bar{N} = 346 \pm 7$	$\mu_w = 0.90$ $\bar{N} = 345 \pm 6$	
Infomap	**354 ± 7**	**360 ± 7**	**363 ± 7**	**370 ± 8**	375 ± 8	**399 ± 8**	476 ± 9	482 ± 8	
SIWOw+(a)	386 ± 7	403 ± 8	400 ± 7	**363 ± 6**	297 ± 5	222 ± 5	171 ± 5	164 ± 6	
SIWOw+(g)	387 ± 7	405 ± 8	400 ± 7	**359 ± 7**	287 ± 6	217 ± 5	178 ± 5	176 ± 6	
SIWOw+(h)	392 ± 7	410 ± 8	406 ± 7	**363 ± 7**	287 ± 6	222 ± 5	186 ± 6	183 ± 7	
SIWOw+(m)	395 ± 7	413 ± 8	410 ± 7	**365 ± 7**	288 ± 6	228 ± 5	194 ± 5	192 ± 7	
WalkTrap	382 ± 18	398 ± 23	433 ± 31	538 ± 43	785 ± 62	1217 ± 82	1523 ± 73	1520 ± 79	
Label Prop.	254 ± 7	260 ± 7	273 ± 7	298 ± 8	**321 ± 16**	269 ± 28	**234 ± 32**	152 ± 28	
Leiden(m)	186 ± 4	183 ± 4	180 ± 4	177 ± 4	172 ± 4	166 ± 3	170 ± 3	177 ± 3	
Louvain	64 ± 1	57 ± 1	51 ± 1	44 ± 1	38 ± 0	33 ± 0	28 ± 0	27 ± 0	
Leiden(c)	446 ± 43	513 ± 54	554 ± 56	546 ± 55	522 ± 52	493 ± 48	500 ± 46	**513 ± 48**	
Fastgreedy	36 ± 0	30 ± 0	24 ± 0	20 ± 0	16 ± 0	14 ± 0	12 ± 0	12 ± 0	
Leading eig.	21 ± 1	16 ± 1	14 ± 1	15 ± 1	13 ± 1	18 ± 2	19 ± 2	17 ± 2	

Table 3.3: Average absolute number of communities found ± the standard deviation computed over 90 runs, as a function of mixing parameters $\mu_w = \mu_t$. Bold numbers indicate the closest to true number for each dataset. Average number of ground truth communities $\bar{N}$ ± standard deviation given as reference.

outlier values. SIWOw+(a) performs best and the arithmetic mean is the most sensitive to large values, while SIWOw+(m) performs worst and is completely insensitive to large outliers. For example, a triangle with three edge weights of 5, 5, and 100 would be given a value of 36.7 by the arithmetic mean, 13.6 by the geometric mean, 7.3 by the harmonic mean, and 5 by the minimum. Reducing the impact of these larger edge weights seems to reduce performance which makes sense when we expect there to be larger edge weights within communities.

Table 3.3 presents results for the same algorithms and networks in terms of the number of communities detected. We can compare the number of detected communities to the number of ground truth communities $\bar{N}$ to get one perspective on how the different algorithms struggle to find the ground truth. The modularity-based algorithms find fewer than the true number of communities which is the result of the resolution limit. This is true even of Leiden(c) with a resolution parameter of 3.0 which makes the algorithm favour smaller communities. This is because the parameter was selected based on NMI. Higher values for the resolution parameter would help find the smallest communities, but would also cause the algorithm to break up the larger communities at an overall cost to performance. While the scale at which the modularity function defines communities can be tuned, this is a good reminder that it cannot simultaneously be tuned for large and small communities and highlights the importance of testing algorithms on realistic and varied synthetic networks. In contrast, Leiden(m) finds too many communities even though its resolution parameter of 0.03 is tuned for finding larger communities. Again, these networks have communities of varying sizes and the parameter cannot be tuned for both large and small communities at once. The random walk-based algorithms are closer to the ground truth but eventually they find too many communities, with the older WalkTrap algorithm doing worse than Infomap. The changes for SIWOw+ and Label Propagation are non-monotonic. The number of communities found by Label Propagation decreases, then increases, then decreases again as μ_w increases. The number of communities found by SIWOw+ increases and then decreases. As the mixing parameters increase,

the number of triangles in a community will decrease so SIWOw+ is more likely to find sub-communities of the remaining pockets of triangles. As the community structure deteriorates further, there becomes even less of a distinction between inter- and intra-community connection patterns so the algorithm struggles to distinguish communities.

3.3.4 SIWOw+ vs SIWO+

Given that the weighted version of SIWO still depends on the presence of triangles and that the difference between the different mean functions is small, one could ask the question of whether considering edge weights adds much to the way SIWO works. To answer this, we conducted another experiment on a different set of LFR networks. These networks of 5,000 vertices were all generated using an average degree of 20, a maximum degree of 50, a μ_w of 0.30, β in $\{1.5, 2\}$, and μ_t in $\{0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70\}$. 20 different networks were generated for each parameter combination for a total of 360 networks. The different β allow for different distributions of edge weights. By keeping μ_w constant we force a majority of the weight to remain on intra-community edges. As μ_t increases, there will be fewer intra-community edges and more inter-community edges. Thus the weights remain a strong signal of a community throughout all the networks while the topology provides less information about community structure. Comparing the performance of SIWOw+(a), which is most sensitive to the edge weights, to unweighted SIWO+ allows us to see how much added information the edge weights provide to our algorithm.

The results of this experiment are presented in Figure 3.3. When the value of μ_t is small the performance of SIWOw+(a) and SIWO+ are very similar. As SIWO+ is able to find an almost perfect community structure there is no room for the weights to offer improvement. As μ_t increases the performance of both decreases. However, the drop in performance of SIWO+ is much more severe and a big gap in performance opens up. This demonstrates that incorporating edge weight information does provide valuable information to SIWO over and above that offered by the network topology and that while edge weight values

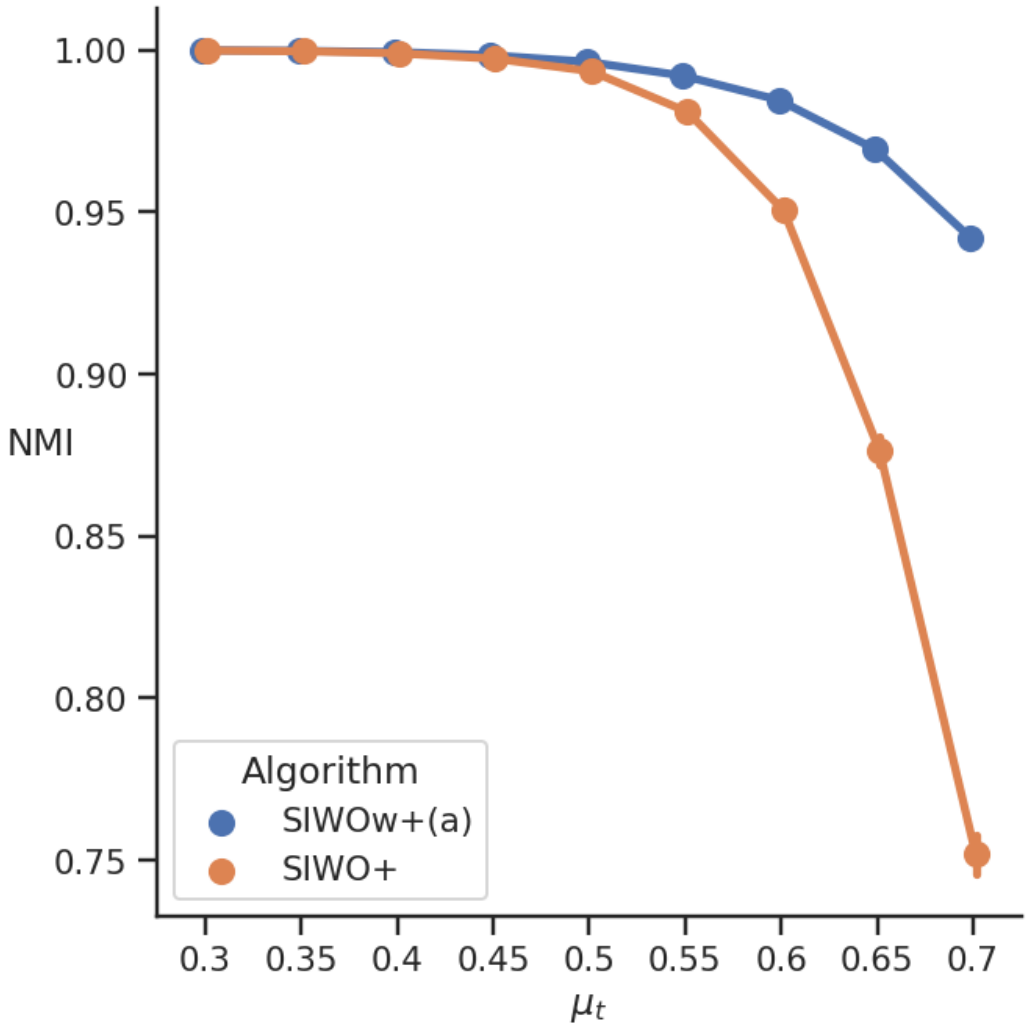


Figure 3.3: NMI score as a function of μ_t keeping $\mu_w = 0.30$ constant for SIWOw+(g) and SIWO+. The difference in performance is small when there is a strong topological community structure, but as μ_t increases the difference in performance becomes large.

and the density of connections are often correlated, we should take advantage of edge weight information whenever possible.

These experiments are not meant to provide a definitive ranking of community detection algorithms. Rather, our goal is to demonstrate the effectiveness of our extension of SIWO to the weighted case and to show that SIWOw+ is competitive with other modern, widely used community detection algorithms. Having shown this, we can also expect SIWOw to perform well on weighted local community search where there are far fewer competitors, although evalu-

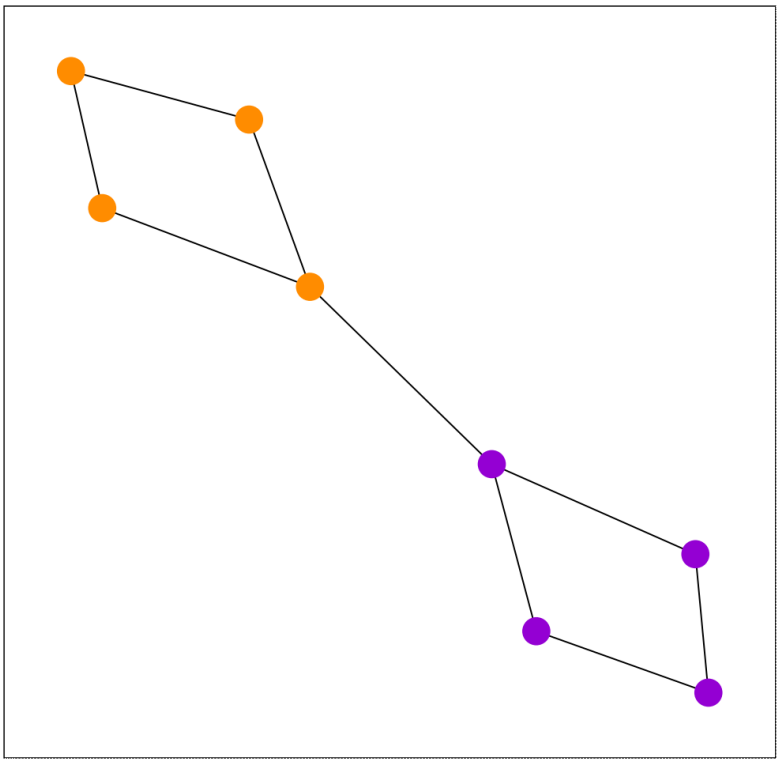
ation of this is beyond the scope of this thesis as our goal is to apply community detection to topic modelling. These experiments also highlight that different algorithms function in very different ways and their performance is impacted by the properties of the network. Moreover, which algorithm may be best to use can also depend on the goals of the analysis. If one wants to find all the small communities of the network then a modularity-based method would not be the best choice. However, if one were interested in grouping the vertices into fewer, larger communities and was planning on merging small communities anyway, then a modularity-based algorithm could be best. This is important to keep in mind as our ultimate purpose is to mine term co-occurrence networks for topic communities and the structure of those networks may differ from syntheric LFR benchmark networks.

Having evaluated SIWOw+ and seen the cases where the algorithm fails, it is natural to try to improve upon the algorithm although it is already among the best performing. In the next section we analyze two cases where SIWO is weak and one possible improvement.

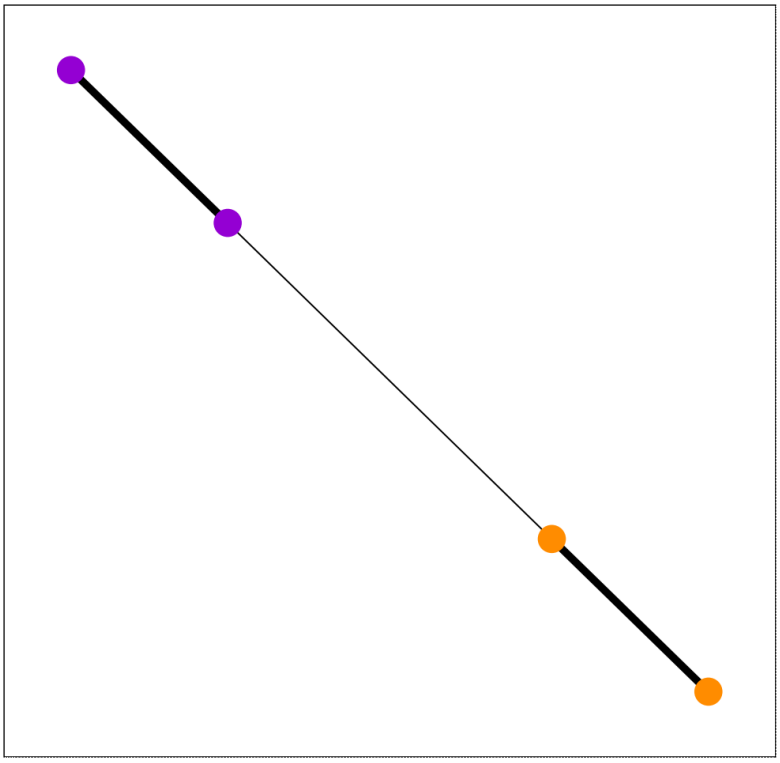
3.4 Min-Max SIWO

We have seen that SIWOw+ performs extremely well at detecting a network's community structure. However, there are cases where our approach fails. One case is very sparse networks. SIWO relies on triangles to determine which edges should lie within a community and which should lie between communities. In a very sparse network, there will be very few triangles even within communities. A simple example of such a network is illustrated in Figure 3.4a. A modularity-based algorithm such as Louvain is able to detect the two communities but it is impossible for SIWO to do so as there are no triangles in the network so every edge will have -1 strength. Some sort of small community merging step could correct this, but as we discussed with the karate network, it is not always desirable to do this if we want to identify the strong core of a community.

Another example can be seen in Figure 3.4b. Here the communities are only distinguished by the edge weights. The thicker edges connecting two



(a) Sparse community structure that lacks triangles.



(b) Network where only indication of community structure is edge weights.

Figure 3.4: Sparse networks that present a challenge to SIWO.

vertices within the same community have weights of 5 while the thin edge between the two communities has a weight of 1. Again, Louvain is able to distinguish these communities but it is impossible for SIWOw to do so.

Ultimately, SIWO fails in these cases as their communities do not conform with the idea of a community underlying the algorithm. SIWO was developed with a view that communities are collections of vertices that share many common neighbours and so looks for triangles as indications of community structure. So while SIWOw+ has proven to be a top performer on the LFR benchmark, it is not suitable for very sparse networks with few triangles that may nevertheless have some sort of community structure.

The opposite extreme of an extremely dense network can also present problems for SIWO. We came across this issue in our search for real-world weighted networks with ground truth communities. One commonly used network in the literature is the college football network introduced in [37] where edges connect teams that play each other. However, it is not possible to adapt this to be a weighted network as a pair of college teams plays each other at most once a season. We decided to construct our own network from a professional ice hockey league, the National Hockey League (NHL), where teams play each other multiple times per season. In fact, every team plays every other team at least twice so an edge would exist between each pair of vertices. The only way to distinguish different relationships between teams are the edge weights corresponding to the number of times the teams play each other. The NHL consists of 32 teams. It is divided into two conferences of 16 teams each. Each conference is divided into two divisions of eight teams each. Teams in different conferences play twice. Teams in the same conference but different division play three times. Teams in the same division play three or four times.

As the difference in edge weights is more distinct between the conferences than the divisions we expected the conference structure to be more easily detectable than the division structure. And indeed, Louvain is able to detect the conferences as communities. In contrast, SIWOw+ only finds one large community consisting of the entire network. Given the great performance of SIWOw+ on other networks this warranted some investigation. The reason

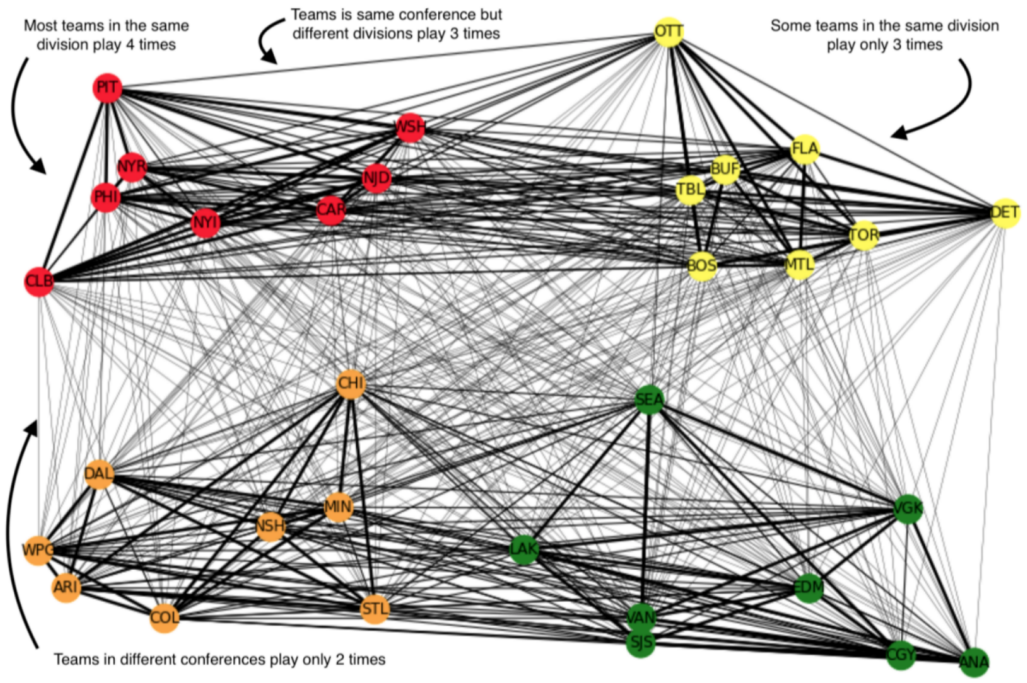


Figure 3.5: Network representing the National Hockey League with edge weights corresponding to the number of times teams play each other. The league is divided into two conferences with two divisions in each conference.

for this is that when there are so many connections even the edges connecting vertices in different communities have a high support value. Unless there is a very large difference in weights all of the support values will be close to the maximum. In this case, two teams in the same division have support of 92 while two teams in a different conference will have support of 73.3. Plugging these values into Equation 2.15 gives us a strength value of 0.6 for edges between teams in different conferences. If the teams in the same division played each other 30 or 40 times instead of 3 or 4 times then this would not be a problem but the effect of the slightly higher triangle values for within division teams is small compared to the cumulative effect of the large number of slightly lower value inter-division and inter-conference triangles.

We proposed modifying the strength Equation 2.15 to use min-max normalization. As currently formulated, Equation 2.15 normalizes the support value of an edge using the range $[0, sup_{i,max}]$. In most cases, that is a realistic range. However, in a completely connected network like the NHL network,

all of the support values cluster near the upper end of the range. Modifying strength to use min-max normalization results in the following updated strength formula:

$$s_{i,j} = \frac{sup_{i,j}}{sup_{i,max} - sup_{i,min}} + \frac{sup_{i,j}}{sup_{j,max} - sup_{j,min}} - 1 \tag{3.2}$$

where $sup_{i,min}$ is the minimum support value of any edge connected to v_i. In many cases this will be 0, but in the NHL network this will be the support value for the inter-conference games.

Using this strength formula did allow SIWOw+ to find the four divisions of the NHL as a community structure, which seemed a success. However, in testing on other networks this new strength formulation did not perform any better that the original version and oftentimes worse. The reason for this is that most networks are not nearly so dense as the NHL network and the parts of the network that are very densely connected are the communities themselves. An example of this is illustrated using the Les Misérables network [51] in Figure 3.6. This is a network of characters who co-appear in chapters of a novel. Vertices $v_{44}, v_{46}, v_{47}, v_{48}, v_{49}$, and v_{50} all co-appear almost exclusively together so clearly belong in the same community. The min-max normalization breaks these vertices up into separate communities, though, which hurts performance.

Clearly this is a case of overfitting a solution to the problems of a specific network. The min-max normalization allows for SIWOw+ to handle the NHL network, but this is an unusually dense network and the effects of this change hurt performance on many other more common network structures. Rather than expanding the capability of SIWO, min-max normalization changes it into an algorithm specialized only for super-dense networks. While this was an unsuccessful experiment, it does shed light on the difficulty of designing a community mining algorithm that works best on all networks.

In this chapter we have presented on of the contributions of our thesis: the extension of the SIWO algorithm to handle weighted networks. This has contributed one of the few local community search algorithms able to handle weighted networks. It has also added another global community detection

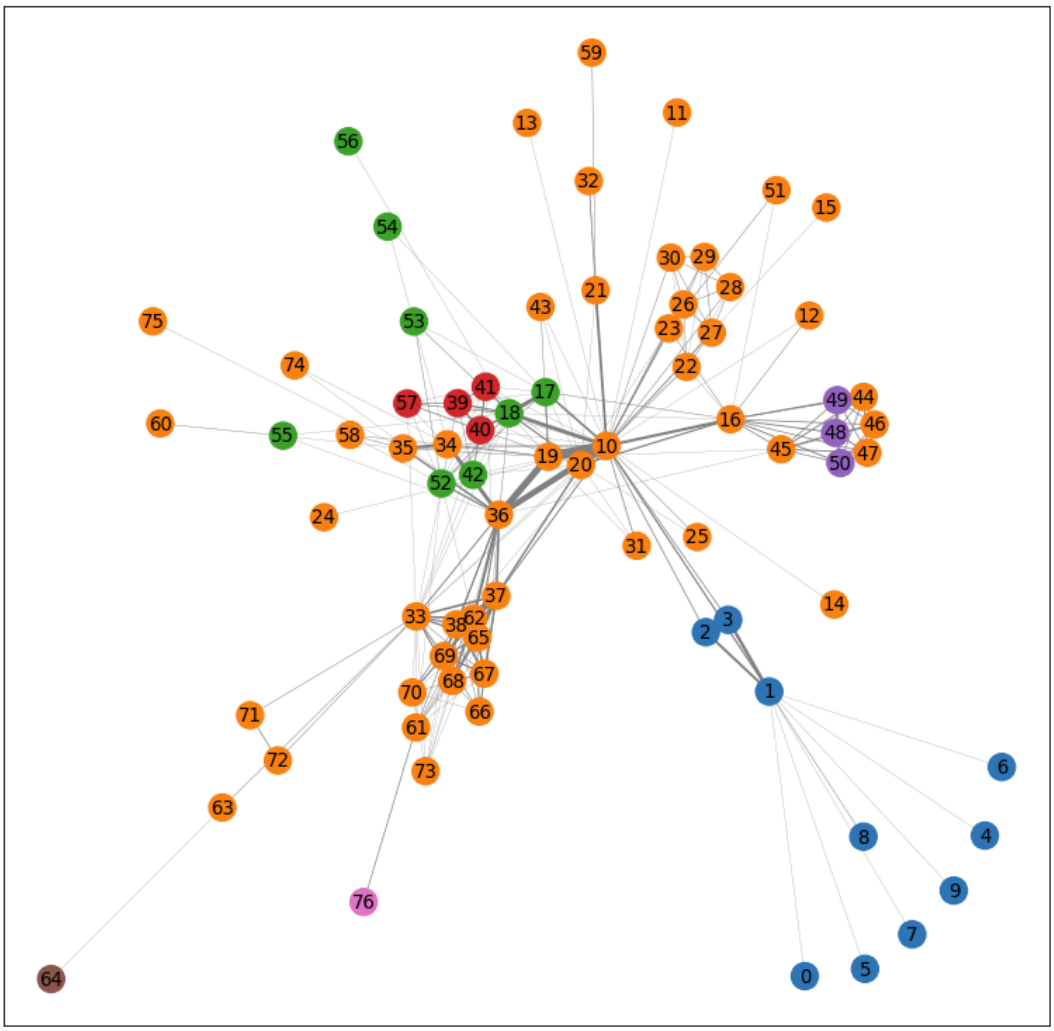


Figure 3.6: Network of characters who co-appear in the novel Les Misérables and the community structure detected by SIWOw+ using min-max normalization in strength formula.

algorithm for weighted networks that outperforms many of the most commonly used today, such as Louvain. Our experiments have shown the effectiveness of our approach on a range of networks. We have also noted the weaknesses of our algorithm, specifically very sparse and very dense networks. We keep in mind that the best performing algorithms in this chapter may not prove to be the best when it comes to applying community detection for topic modelling, to which we now turn.

Chapter 4

Community Detection for Topic Modelling

The field of topic modelling has been dominated by generative probabilistic models, in particular Latent Dirichlet Allocation. While LDA remains extremely popular, it suffers from several weaknesses. The number of topics must be specified but a natural number of topics for a corpus is often not known so many different values must be tried. The topics found by LDA are often not stable, i.e. repeated runs of LDA on the same data can generate different topics and the topics found are sensitive to the ordering of the documents in the corpus [66]. LDA assumes that terms in a document are exchangeable, i.e. their ordering does not matter, and thus does not consider the proximity of two terms within a document. LDA does not produce topics in a hierarchy and there is no natural relationship between the topics. LDA struggles with short documents such as tweets.

We reviewed several extensions of LDA in Chapter 2 that address some of these shortcomings. However, these models add complexity and there is no one model that addresses them all and the question still remains over whether the terms that are most important for reconstructing the original documents with a generative model are also the most important for finding coherent and interpretable topics. Given topic modelling's use as a tool in applied research outside of computing science, LDA's simplicity and the availability of thoroughly-tested and easy-to-use implementations has meant that LDA remains the topic model of choice.

In recent years, deep neural networks have achieved state-of-the-art performance in many areas including natural language processing. They have been used for sentiment analysis, document classification, translation, response generation, text summarization, and language modelling. It seems natural to apply the power of neural networks to the problem of topic modelling and indeed several of the approaches reviewed in Chapter 2 do just that. However, these neural models also suffer from weaknesses. Neural networks require large datasets to learn and must train for long periods of time on specialized hardware. Neural topic models can produce highly redundant topics [13]. Moreover, while neural topic models have achieved higher coherence scores than LDA, there are questions over whether these higher scores on automated metrics reflect a genuine improvement in topic quality [42].

Given these weaknesses, we have taken a different approach to the problem of finding interpretable topics from a corpus of documents. Our approach is based on a network representation of the documents and uses community detection to discover the topics. Networks are a flexible representation for interconnected data. Many different complex systems can be represented by networks from which communities with real-world meaning can be mined. For example, in biology protein complexes have been discovered from protein-protein interaction networks [62].

A network representation and community detection provide several advantages over other topic modelling approaches:

- **Discovery of the number of topics** - The topics are the discovered communities and the community detection algorithm finds the communities according to the network, not a user-specified hyperparameter.

- **Natural hierarchy** - The communities found are not only collections of vertices, but are themselves networks on which community detection can be applied to find sub-topics. The communities are also vertices in a network with edge weights being the sum of the edge weights of the member term vertices. Community detection can be applied to this topic network to find super-topics.

- **Natural relationship between topics** - Since the topics are vertices in a network, the edges and edge weights provide information on the relationships between topics. This can be used to guide exploration of a corpus in applied research or the topic of conversation for a chat bot. This information can also guide the merging of topics if the user desires a smaller number.

- **Time and resource efficient** - Processing the corpus to build the network and performing community detection is possible in a short time without specialized hardware.

- **Stability** - Different community detection algorithms have different levels of stochasticity but the communities found over different runs tend to be stable.

- **Minimal hyperparameters** - Many topic models have several hyperparameters to set, especially neural topic models. In contrast, community detection algorithms have few or none at all.

- **No redundancy** - LDA and neural topic models can find redundant topics, i.e. those that are very similar to other topics, and common words tend to appear in many topics. While finding overlapping communities is possible, we focus on detection algorithms that partition the network so that every vertex is in exactly one community and thus every term is in exactly one topic.

In this chapter we develop the second and third contributions of this thesis, the term co-occurrence networks and our community detection-based topic modelling algorithm. We evaluate the claimed advantages as well as the quality of the detected topics. We compare our algorithm against LDA as a standard benchmark as well as a recently developed algorithm based on word embeddings learned by a neural network. First, we describe how we get the network from a corpus and analyze some of the properties of these networks.

4.1 Term Co-occurrence Networks

The network that we construct from a collection of documents has terms as vertices. An edge exists between a pair of vertices if the terms co-occur. Co-occurrence can be defined in multiple ways. The first definition that we use is that two terms co-occur if they both occur in the same sentence. This is based on the assumption that two terms in the same sentence are more likely to be related than two terms in different sentences. This definition also results in an insensitivity to document length as the corpus could be split into documents of one sentence each and the resulting network would be unchanged. However, it is likely that two terms in adjacent sentences of the same document are also related so an alternative definition of co-occurrence is that two terms co-occur if they both occur within a fixed-size sliding window over a document. We will empirically evaluate topics found from networks based on both definitions to determine whether one definition is superior.

The weights of edges come from the frequency of co-occurrence. One method is to use the raw count as the edge weight. However, this does not adjust for the frequency of the terms themselves so more common terms will tend to have higher edge weights. This may impact the community detection algorithms. An alternative weighting scheme is to use NPMI (Equation 2.9). This adjusts for the frequency of the terms and assigns high values to terms that co-occur more frequently than expected. These edge weights encode more information than the raw counts and are bound to the range [-1, +1]. However, as we saw in Section 3.4, the performance of a community detection algorithm can be impacted by the relative magnitude of the edge weights so it is not obvious which of these edge weight formulations results in better topics so we empirically evaluate both.

Our notion of relatedness based on co-occurrence is supported by the distributional hypothesis. The distributional hypothesis was proposed in the field of linguistics and posits that words that occur in similar contexts tend to have similar meaning [39]. This relationship has been studied in human language understanding, for example in [74] where the authors show that there is a neg-

ative correlation between the subjective judgement of semantic similarity of pairs of nouns and the ease of discriminating between their sentence contexts.

This relationship between meaning and context has also been studied and exploited in computational linguistics and natural language processing. Methods reviewed in Chapter 2 such as LSA and NMF use the document context to produce vector representations of terms. Many other vector space models that attempt to capture word meaning have also been developed [27]. A major step forward occurred when neural networks were applied for learning these vector representations, commonly called word embeddings. Rather than devise a vector based on counts, the word2vec model learns the embedding through predictions. The word2vec model can learn either by predicting the surrounding terms in a given context window given a term (the skip-gram model) [72], or by predicting a term given the surrounding context (the continuous bag of words model) [71]. Many different forms of pre-training for neural language models also involves some form of prediction of terms given a context.

Given the relationship between co-occurrence and meaning, we expect there to be a greater density of edges and higher edge weights between groups of vertices in our network that represent terms with similar meanings. Community detection algorithms such as SIWO are able to find these densely-connected groups of vertices. The common concept or idea that connects the meaning of these related terms is a topic, so detecting a community in our network is really finding a topic in the document corpus.

We fully describe our algorithm and compare against two other topic modelling algorithms on a range of corpora later in this chapter. First, we discuss some common text preprocessing techniques that we perform on the corpus prior to building the network. Then we analyze some of the properties of the co-occurrence networks themselves and compare with the synthetic LFR networks used to benchmark the community detection algorithms in Chapter 3.

4.1.1 Preprocessing

There are many different methods for preprocessing a text corpus and we follow practices that have been found to work well for topic modelling in the literature. We use spaCy[1] to lowercase and tokenize the documents and to identify sentences, parts of speech, and named entities. We only detect noun-type entities, i.e. EVENT, FAC (buildings), GPE (geo-political entities), LOC (non-GPE locations), ORG (organizations), PERSON, PRODUCT, and WORK_OF_ART, which are merged into single tokens e.g. the terms "united", "states", "of", and "america" become "united_states_of_america". While stemming and lemmatization have been commonly used in the topic modelling literature, the authors of [104] found that stemming and lemmatizing do not improve topic quality and hurt model stability so we do not stem or lemmatize. We remove stopwords and terms that occur in $> 90\%$ of documents. Following [42], we remove terms that appear in fewer than $2(0.02|d|)^{1/\log 10}$ documents. This formula removes more terms from larger corpora, but the number of terms removed only grows proportionally to $\sqrt{|d|}$. It was shown in [67] that topic models constructed from noun-only corpora were more coherent so we detect and tag parts-of-speech to be able to filter out non-noun terms as in [16]. This is intuitive as adjectives and verbs can be used in many different contexts, e.g. we do not think of dogs and football as related even though we can say "the big dog" and "the big touchdown"; one can "play the piano", "play baseball", "play the stock market", and "play with someone's heart", but music, sports, finance, and romance are separate topics. However, we compare the quality of topics with and without this filtering as different algorithms may be more sensitive to the presence of generic terms and there may be some topical adjectives and verbs or n-gram combinations using them. Even using only nouns, there are still issues with polysemy, i.e. words with multiple meanings and thus multiple different common contexts. To help with this problem, we use Gensim[2] to extract meaningful n-grams using NPMI [11]. An n-gram is a combination of n adjacent tokens into a single token so that a term such as

[1] https://spacy.io/

[2] https://radimrehurek.com/gensim/

"microsoft_windows" can be found and the computer operating system can be distinguished from the windows you would find on a building. We apply two iterations so that longer n-grams such as "law_enforcement_agencies" can be found.

4.1.2 Datasets

We use three datasets to evaluate the different topic modelling approaches: 20Newsgroups[3], Reuters-21578[4], and BBC News[5]. The 20Newsgroups dataset consists of 18,846 posts on the Usenet discussion platform which come from 20 different topics such as "atheism", "motorcycles", and "hockey". The Reuters-21578 dataset consists of 21,578 financial articles published on the Reuters newswire in 1987 and have economic and financial topics such as "grain", "copper", and "interest". The BBC News dataset consists of 2225 articles published by the BBC in five categories: "business", "entertainment", "politics", "sport", and "tech".

4.1.3 Properties of the Co-occurrence Networks

Networks representing different real-world complex systems can have very different properties and structures. The co-occurrence networks that we generate from text documents may be very different from the synthetic LFR benchmark networks that we used to evaluate the community detection algorithms in Chapter 3. Here we examine two examples of co-occurrence networks and contrast their structure and properties with a network used in Chapter 3.

The two co-occurrence networks were created from the 20Newsgroups corpus preprocessed as described in Section 4.1.1. Their properties and those of the LFR benchmark network as described in Table 4.1. The vocabulary size and number of vertices was 2,929. Co-occurrence was defined based on sentences and one network had count edge weights while the other had NPMI edge weights. We also tried thresholding the edges, i.e. removing edges whose

[3]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html
[4]https://huggingface.co/datasets/reuters21578
[5]https://www.kaggle.com/competitions/learn-ai-bbc/data

	Count		NPMI		LFR
Threshold	n/a	> 2	n/a	> 0.35	n/a
$\|E\|$	389,904	56,195	389,904	71,681	123,246
Average k	266.2	38.4	266.2	48.9	24.6
Average k^w	470.9	198.3	68.1	20.0	945.6
CL	0.343	0.444	0.343	0.096	0.214

Table 4.1: Number of edges, average degree, average weighted degree and clustering coefficient for a term co-occurrence network with raw count edge weights with no thresholding and thresholding at 2, a network with NPMI edge weight with no thresholding and thresholding at 0.35, and a LFR benchmark network. The co-occurrence networks have 2,929 vertices each while the LFR network has 10,000 vertices.

weight is below a certain value. For the count network, we used a threshold of > 2 as co-occurrence once or twice in thousands of documents is likely noise rather than a relationship. This greatly reduces the number of edges in the network. For the NPMI network, we picked a threshold of > 0.35 to remove a similar number of edges which are presumably the low information edges. The LFR network of 10,000 vertices was generated with an average degree of 25, maximum degree of 100, mixing parameters $\mu_t = \mu_2 = 0.4$, and edge weight parameter $\beta = 2$.

We can see from the table that the original networks have many more edges than the LFR network even though they have fewer vertices. They are much more densely connected and have an average degree an order of magnitude greater than the LFR network. The thresholding greatly reduces the number of edges and average degree, although they remain more densely connected than the LFR network. There is a large difference between the average weighted degrees of the co-occurrence networks given the difference in scale of the edge weights. The maximum edge weight of the raw count network is 118; the maximum edge weight of the NPMI network is 0.992. The weights of the LFR network are closer to those of the count network but have an even greater range with a maximum weight of 392.6.

We also see that the unweighted clustering coefficient of the co-occurrence networks is appreciably higher than that of the LFR network. SIWO relies on the presence of triangles to detect communities so this could be helpful,

but as we saw in Section 3.4 when the network is too dense and there are too many triangles, SIWO struggles to detect communities, particularly if the differences between edge weights are not large. With thresholding, the clustering coefficient of the count network increases, indicating that the edges being removed are not those that are part of triangles. However, the clustering coefficient of the NPMI network drops significantly, so more of the edges being removed are those that form parts of triangles. This could negatively impact SIWO's ability to detect communities.

In addition to summary statistics like average degree, we can examine the distributions of these properties. We first examine the distribution of edge weights as this is the most obvious difference between the count and NPMI networks. Figure 4.1 shows the edge weight distribution of the LFR network. We see there are more edges with low weights than high weights. This is similar to the distribution of the count network shown in Figure 4.2a. However, the weights of the count network drop off much more quickly and they actually follow a power law distribution unlike the LFR weights. This can be seen in Figure 4.3 where the log-log plot has a close to linear slope for the count network but not the LFR network. The edge weight distribution for the NPMI network illustrated in figure 4.2b is even more distinct. It is a symmetrical, bell-shaped distribution. NPMI values are bound within [-1,+1] but we see that there are very few negative edge weights. This shows that two terms are likely to co-occur more often than chance conditioned on co-occurring in at least one sentence together, so the presence of an edge does carry meaningful information about relationships between terms.

When we apply thresholding to the co-occurrence networks we greatly reduce the number of edges. When we remove edges with weight 1 or 2 from the count network, we get the distribution in Figure 4.4a. The distribution looks extremely similar, which is a characteristic of power law, or "scale-free", distributions. The hope is that removing edges of low weight does not reduce the amount of information available for finding topics while making the community detection faster and more reliable. When we apply thresholding at > 0.35 on the NPMI network we remove a similar amount of what should

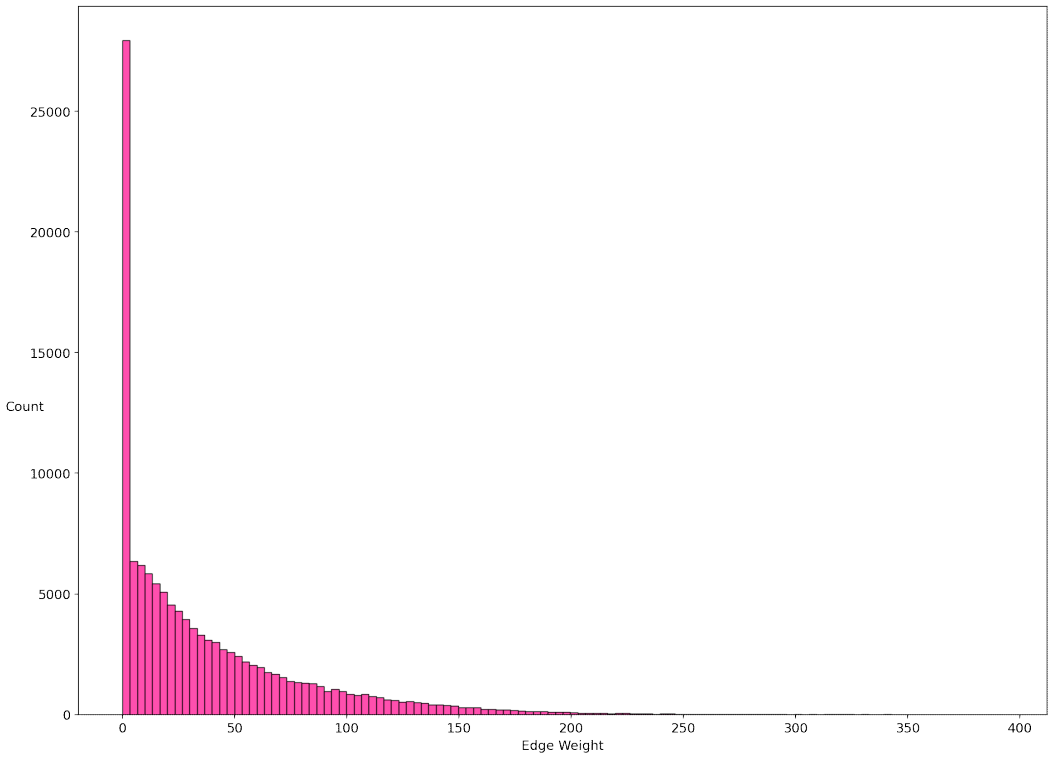
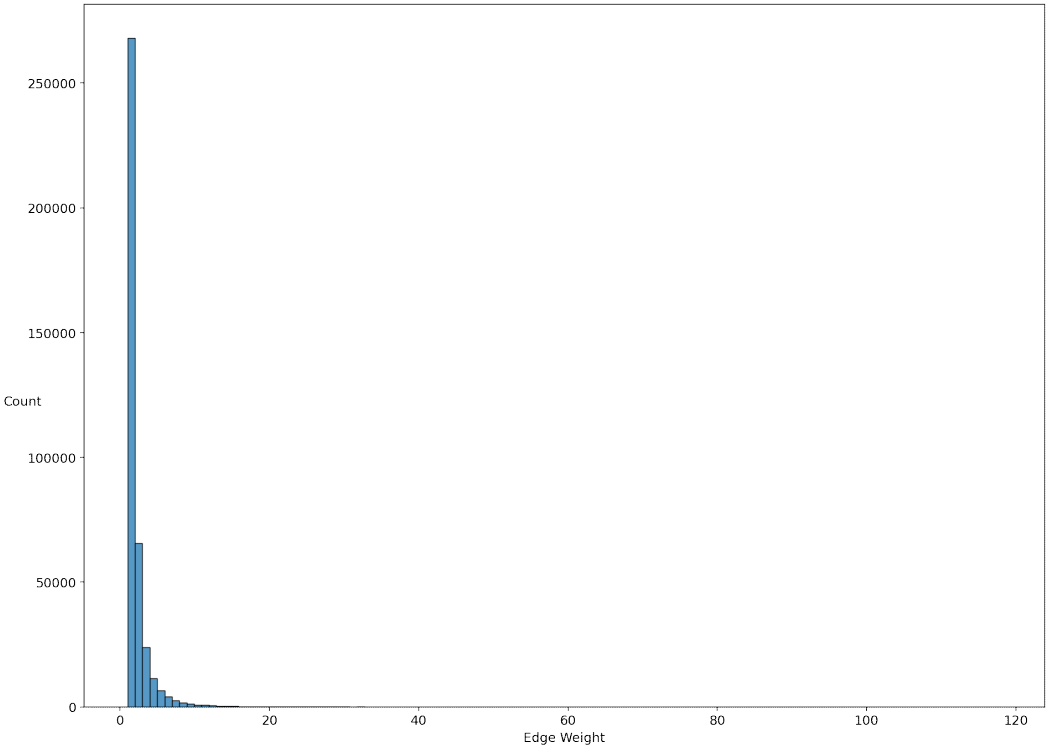


Figure 4.1: Edge weight distribution of LFR network.

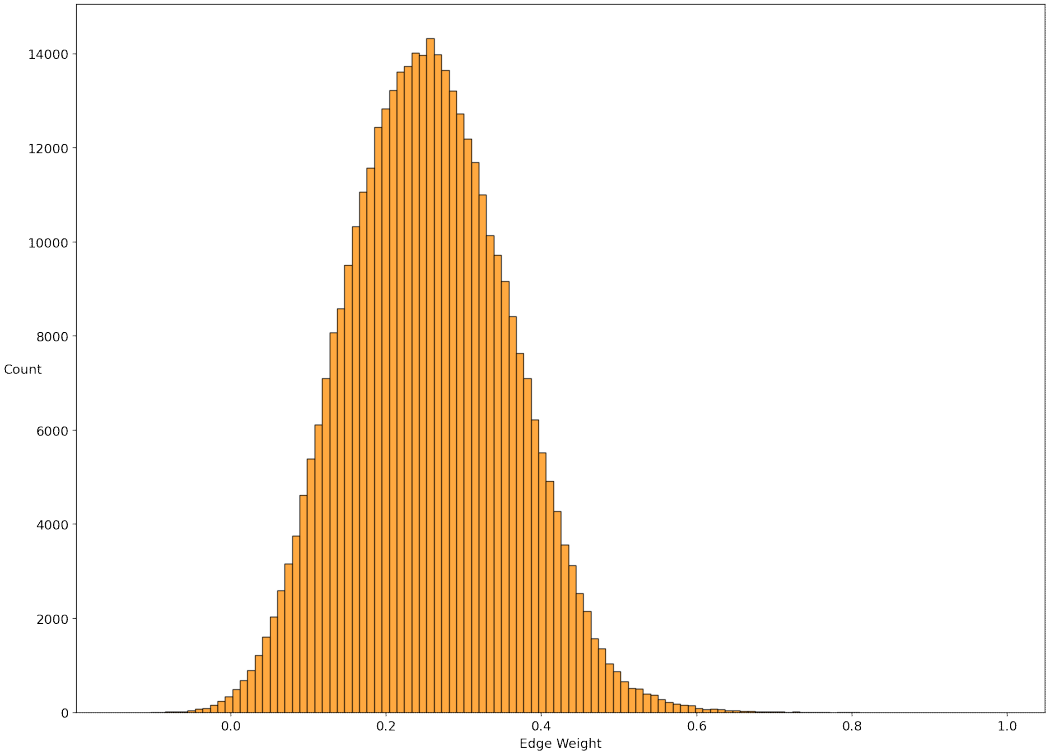
be the least important edges and get the distribution shown in Figure 4.4b. However, as the very different clustering coefficients in Table 4.1 show, the edges being removed are not the same.

We now look at the degree distributions. The LFR generator creates degree distributions according to a power law which we can see in Figure 4.5b. This is modeled on real world networks that exhibit this property. However, we can see regular gaps in the distribution which must be an artifact of the network generation process. The unweighted degree distribution for the count and NPMI co-occurrence networks are of course identical and illustrated in Figure 4.5a. This is not quite a power law distribution and there are very few edges with a very low degree, unlike the LFR network which has many dangling vertices. It is still heavily skewed with very few vertices having very high degrees.

After thresholding, the degrees fall across the distribution for both networks. The distribution of the count network becomes similar to that of the

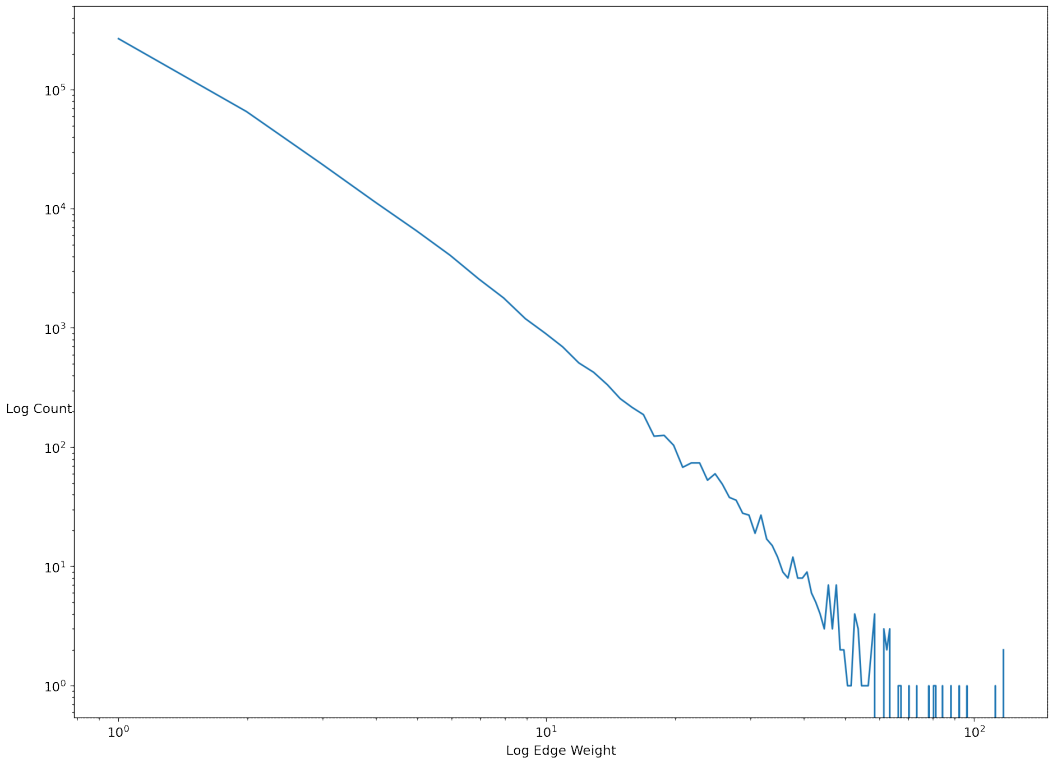


(a) Count co-occurrence network.

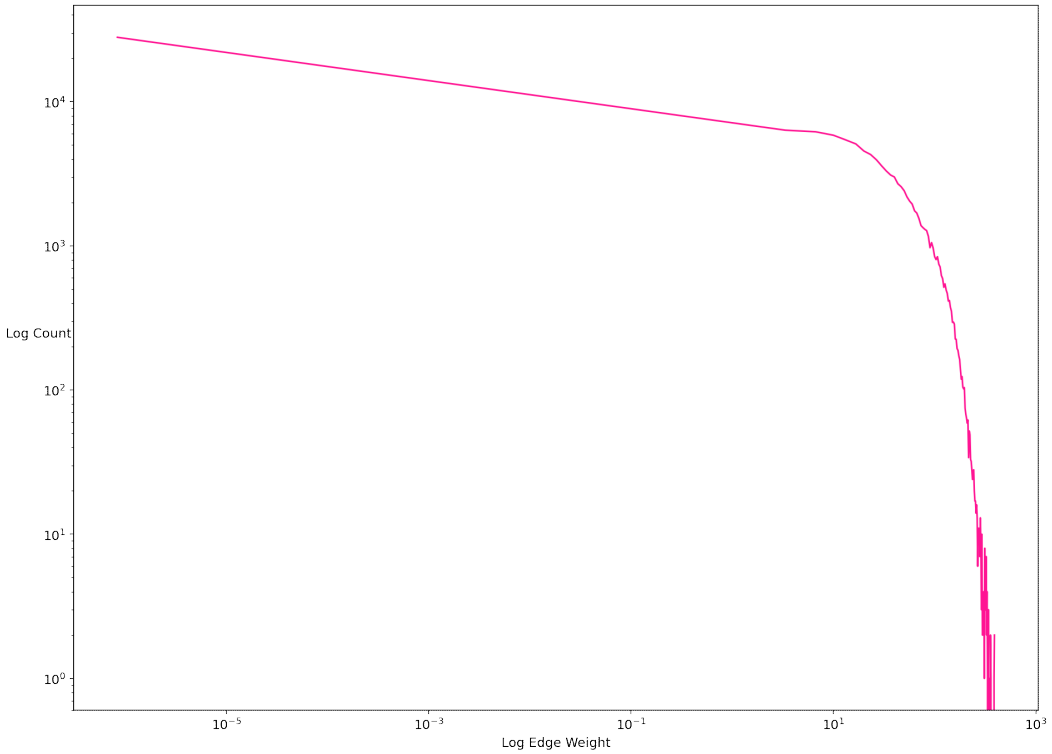


(b) NPMI co-occurrence network.

Figure 4.2: 20Newsgroups count and NPMI co-occurrence networks edge weight distributions with no thresholding.

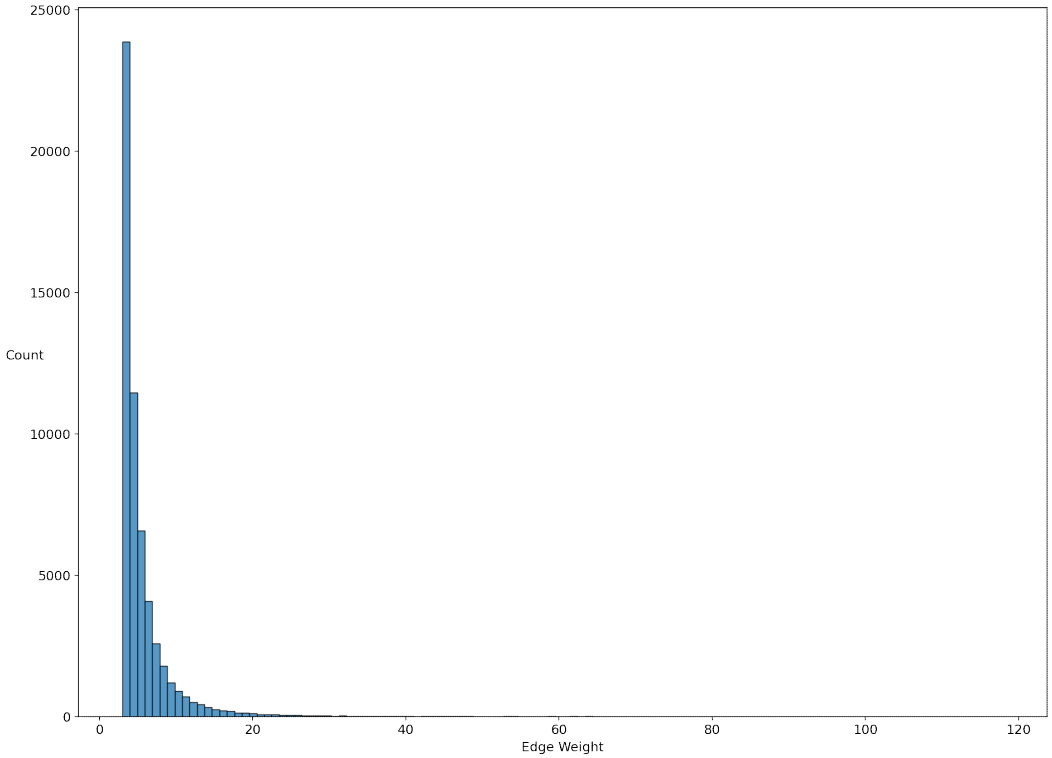


(a) Count co-occurrence network.

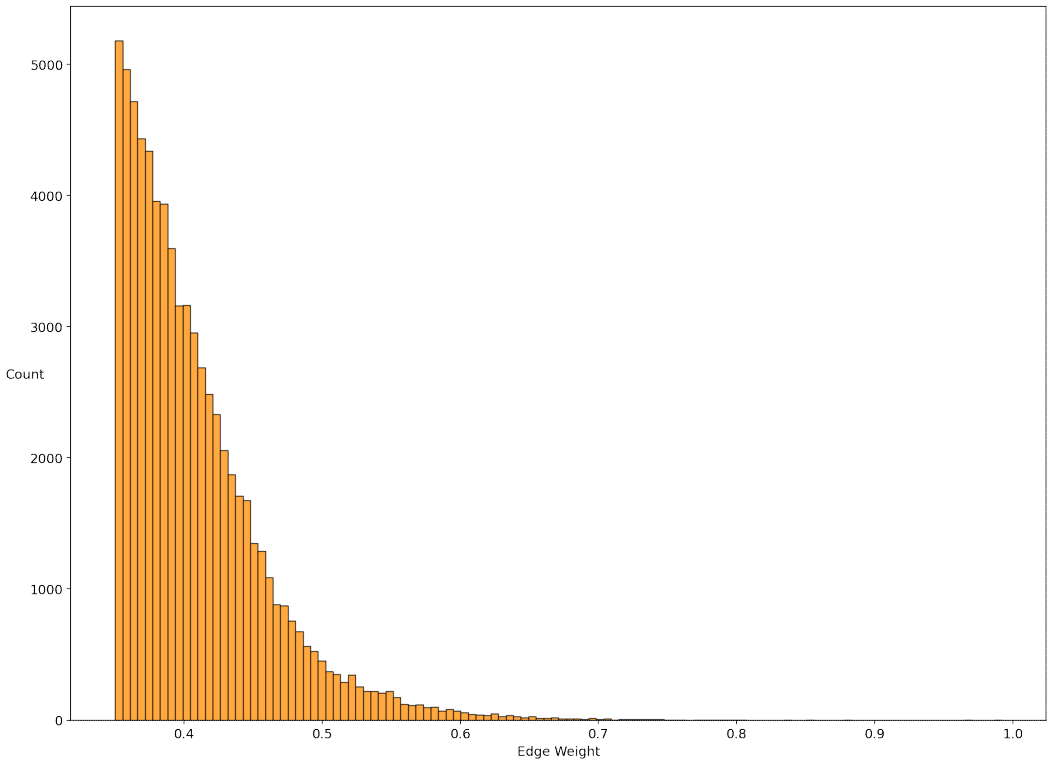


(b) LFR network.

Figure 4.3: Log-log plot of edge weight distributions of the count co-occurrence network and LFR network.

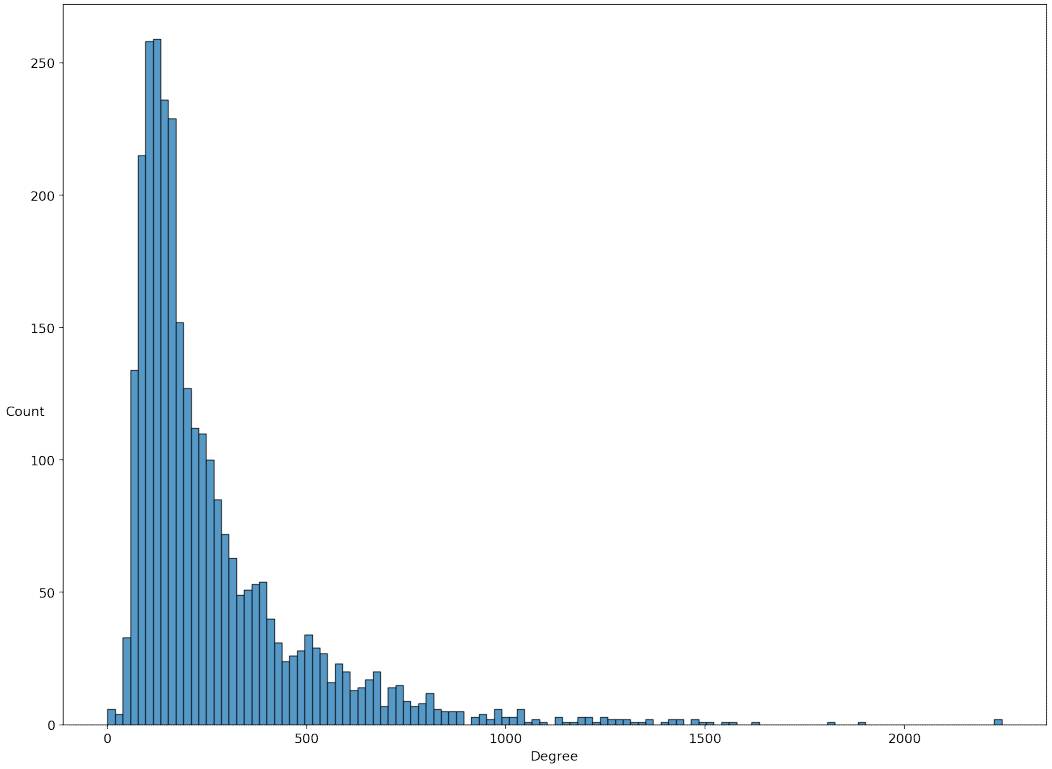


(a) Count co-occurrence network.

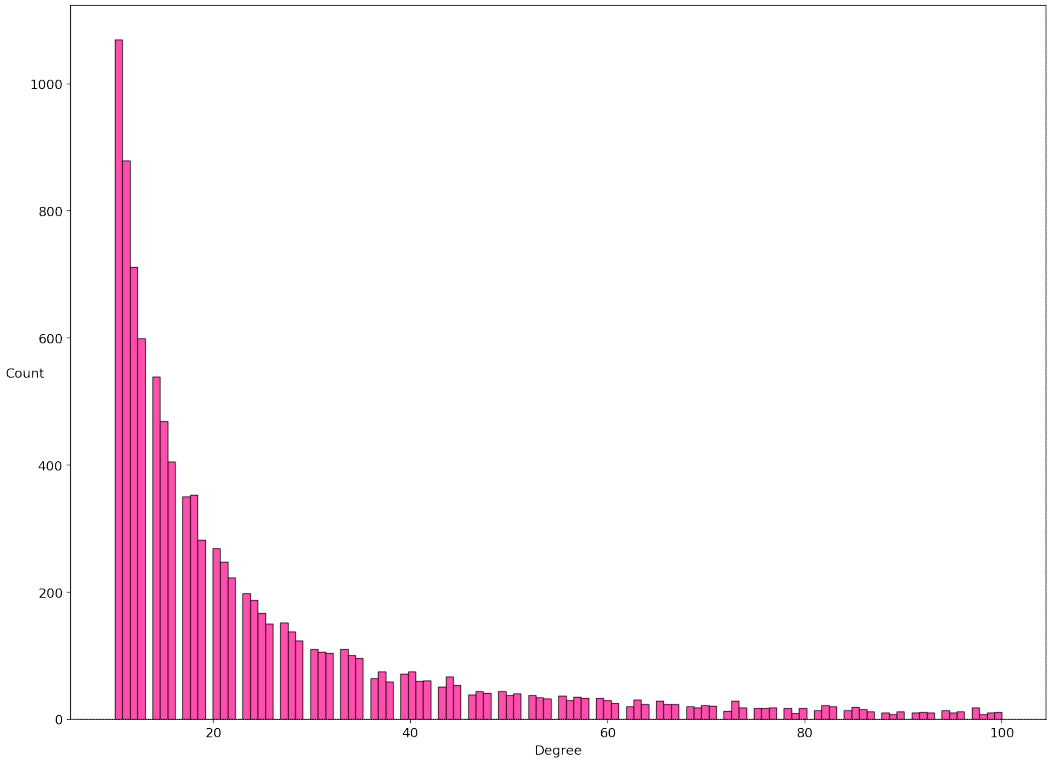


(b) NPMI co-occurrence network.

Figure 4.4: Count and NPMI co-occurrence networks edge weight distributions with thresholding at > 2 and > 0.35, respectively.



(a) Count and NPMI co-occurrence networks.



(b) LFR network.

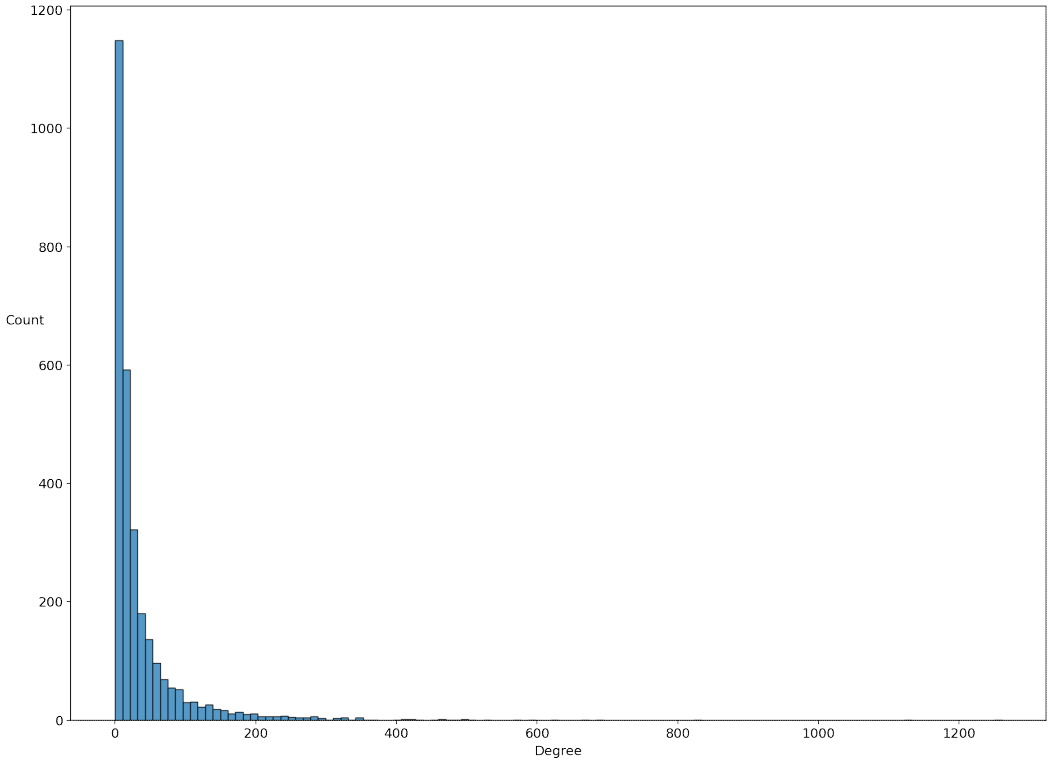
Figure 4.5: Co-occurrence and LFR networks unweighted degree distributions with no thresholding.

LFR network with many low degrees as we see in Figure 4.4a; many of the vocabulary terms have few frequent co-occurrences. We do not see this pattern with the NPMI network in Figure 4.4b. There is no long tail of high degrees and the distribution is close to symmetrical. Most vertices have both more and less important relationships once the NPMI formula is applied as their edge weights no longer correlate with term frequency.

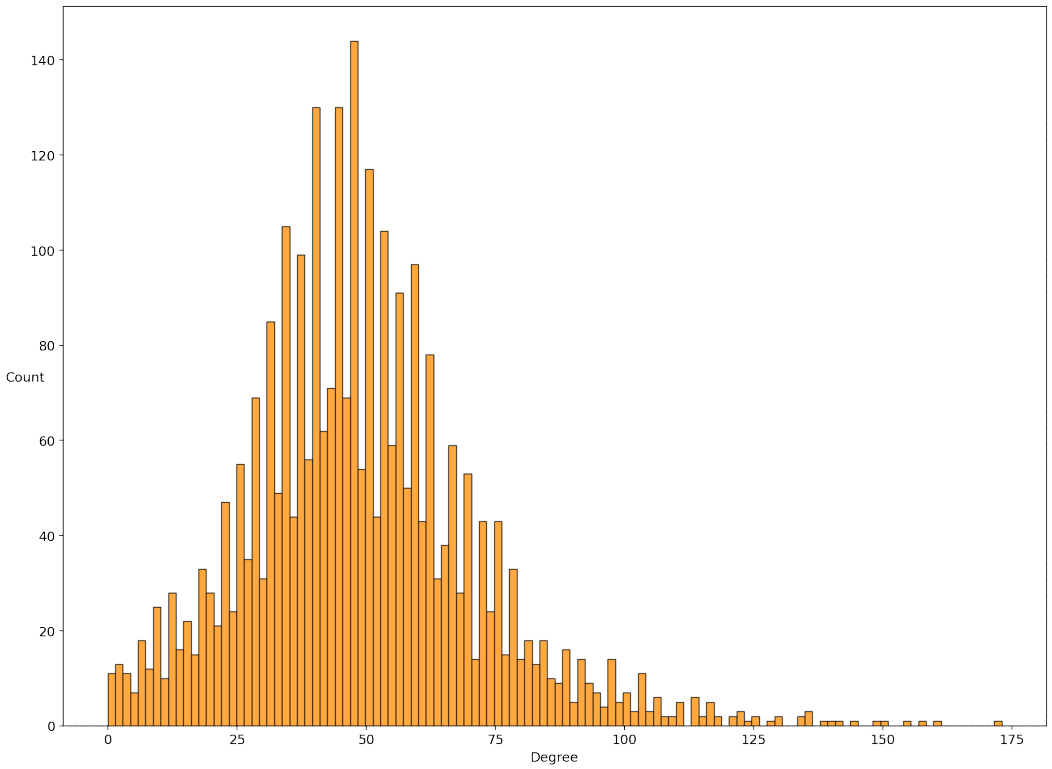
The weighted degree distributions of the count and NPMI networks are also quite different from each other and the LFR network. The LFR generator gives higher weights to edges connected to higher degree vertices according to the β parameter so the weighted degree distribution in Figure 4.7 follows a power law with an even more extreme skew than the unweighted degree distribution in Figure 4.5b. As with unweighted degree, the weighted degree distribution of the count co-occurrence network, illustrated in Figure 4.8a is not a true power law distribution as the most common weighted degree are not the lowest. We do see that the distribution is more skewed than the unweighted distribution, indicating that the edges connected to higher degree vertices do have higher weights as in the LFR network. This makes sense as terms that occur more frequently will be more likely to co-occur with a higher number of terms and to co-occur with each term more frequently. The weighted degree distribution for the NPMI co-occurrence network is shown in Figure 4.8b and is a less skewed version of the unweighted degree distribution, which follows from the fact that the edge weights are less than 1.

We can see the changes to the distributions when thresholding in applied in Figure 4.9. The count network's distribution becomes more similar to that of the LFR network as the majority of vertices now have the lowest weighted degrees. The distribution of the NPMI network, however, retains more of a bell curve. Again, this is consistent with vertices having a mix of low and high edge weights as removing low weight edges does not create a great bulk of low weighted degree vertices.

Clearly, the structures of the two networks are very different and we should expect them to contain different information about the relationships among the terms of the corpus. We can see this clearly when we examine the terms



(a) Count co-occurrence network.



(b) NPMI co-occurrence network.

Figure 4.6: Count and NPMI co-occurrence networks unweighted degree distributions with thresholding at > 2 and > 0.35, respectively.

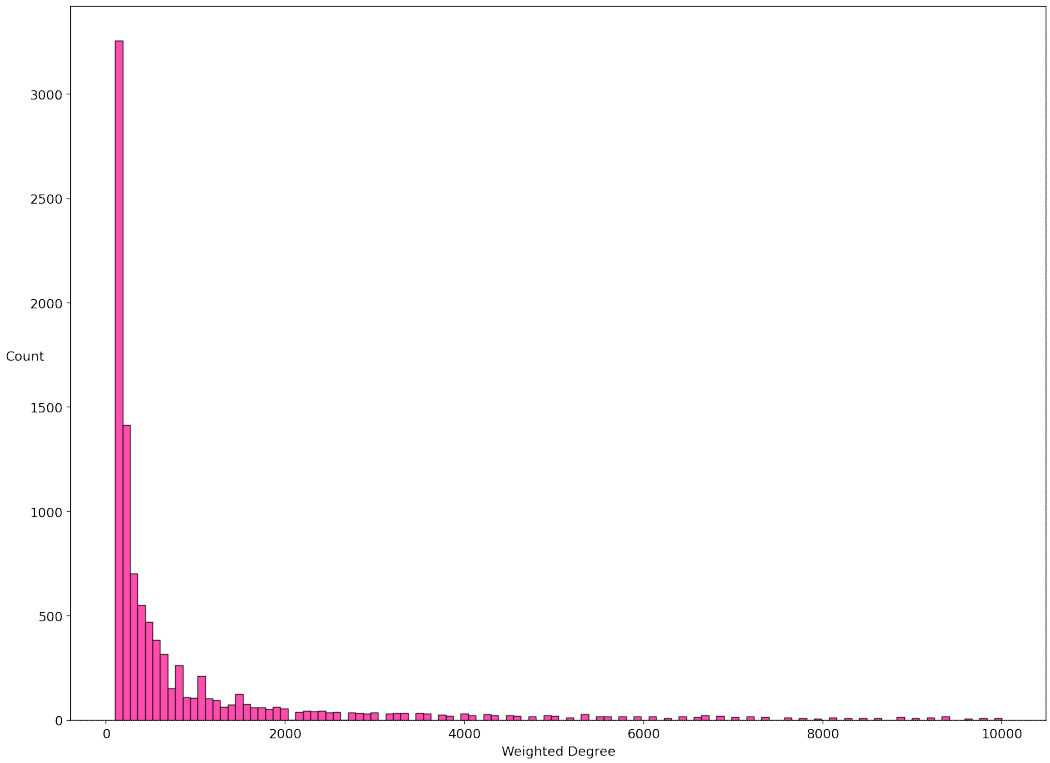
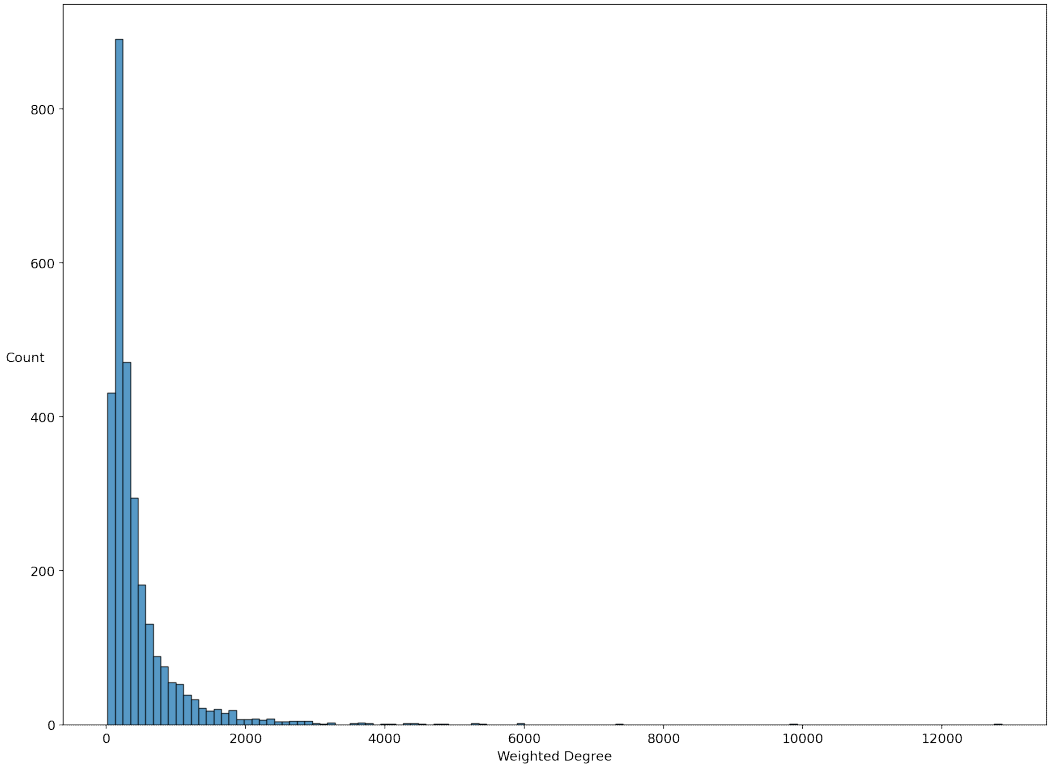


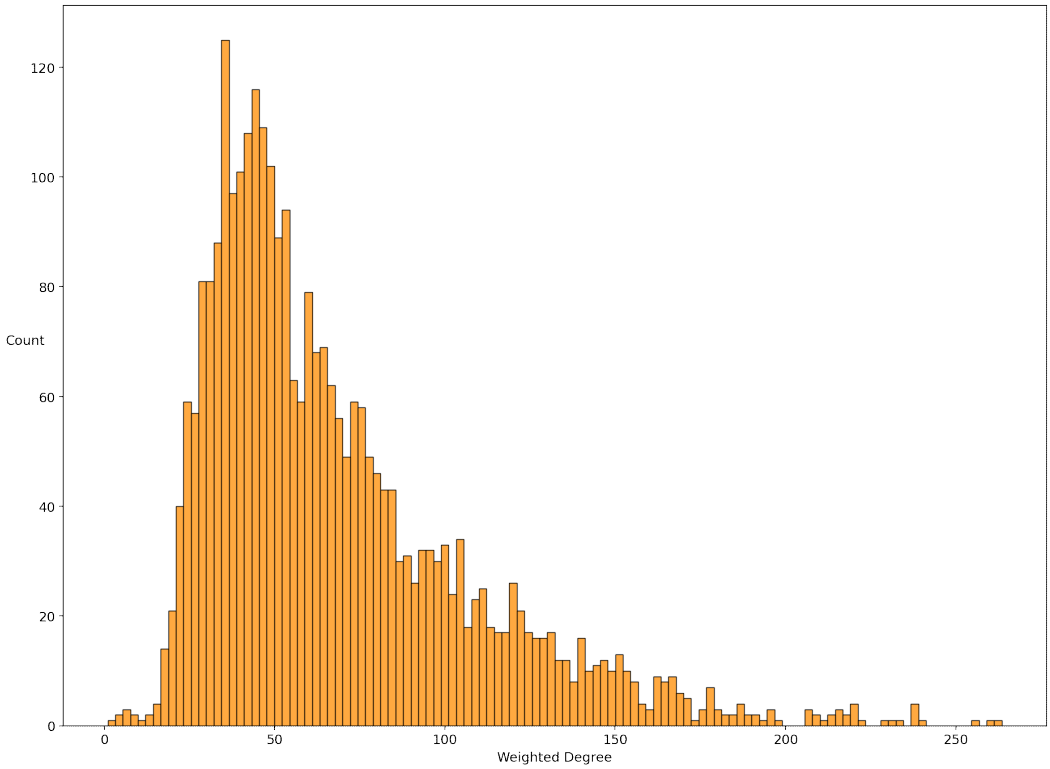
Figure 4.7: Weighted degree distribution of LFR network.

and relationships that are most important in each network. Table 4.2 shows the top 10 terms for both the count and NPMI co-occurrence networks when ranking by either unweighted degree or weighted degree. Rankings are shown before and after thresholding. The table also presents the top ten edges ranked by edge weight for each network. The top terms by unweighted degree are of course the same for both networks. These terms are quite general and are single words that could be seen in many different contexts. In the count network, six of the 10 top terms are the same when ranking by weighted degree. In the weighted degree ranking, the terms "program", "data", "file", and "god" seem a bit less generic than the terms they replace, "years", "case", "number", and "things". There is less overlap in the case of the NPMI network with only four of the 10 carrying over from the unweighted to weighted degree ranking. Again, the new terms might be considered less generic than the terms they replace, e.g. "space" is more specific than "things", but not by much.

After thresholding, the top terms when ranked by unweighted degree for the

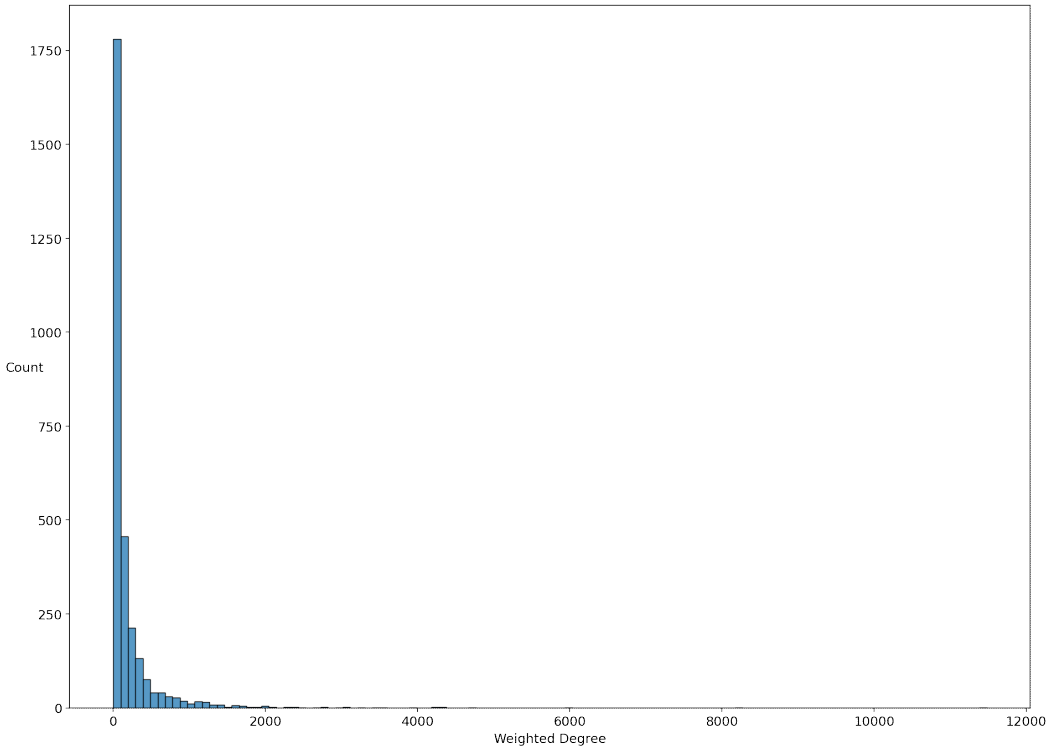


(a) Count co-occurrence network.

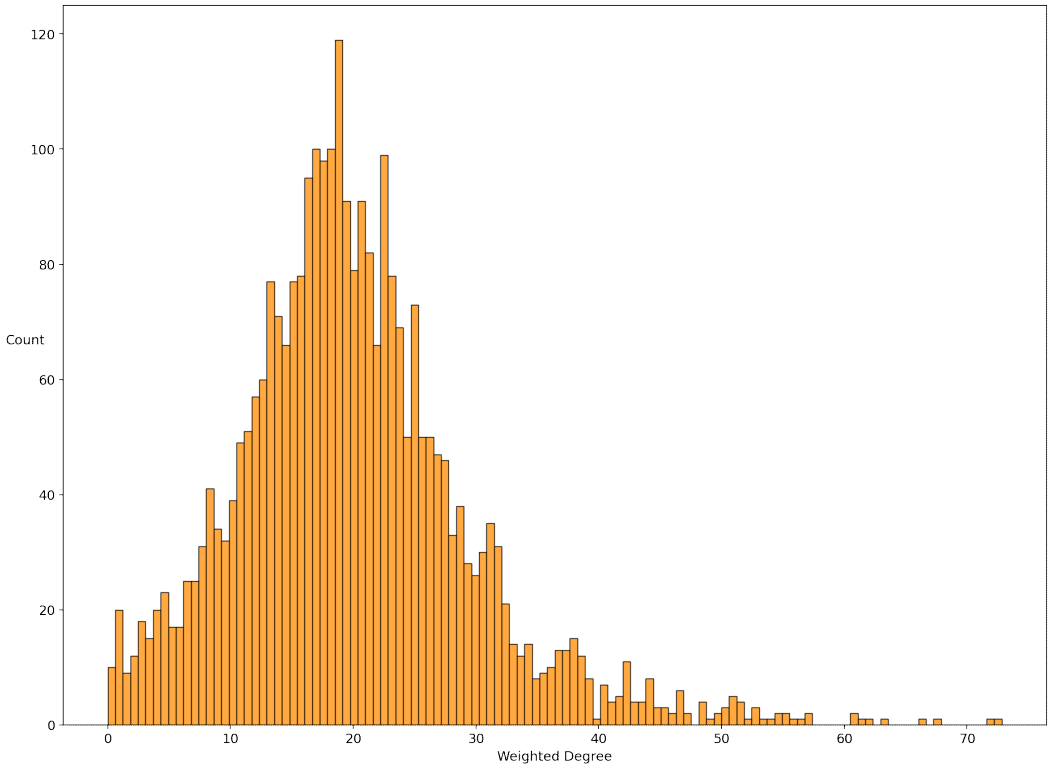


(b) NPMI co-occurrence network.

Figure 4.8: Count and NPMI co-occurrence networks weighted degree distributions with no thresholding.



(a) Count co-occurrence network.



(b) NPMI co-occurrence network.

Figure 4.9: Count and NPMI co-occurrence networks weighted degree distributions with thresholding at > 2 and > 0.35, respectively.

Count network				
No threshold		> 2		Top edges by $w_{i,j}$
Top terms by k	Top terms by k^w	Top terms by k	Top terms by k^w	
time	people	people	people	(people, god)
people	time	time	time	(people, time
way	system	system	system	(people, government)
system	way	way	information	(way, people)
information	information	information	file	(people, lot)
years	program	program	way	(people, things)
year	data	data	program	(people, world)
case	file	year	data	(jesus, god)
number	god	years	god	(file, program)
things	year	file	government	(bible, god)
NPMI network				
No threshold		> 0.35		Top edges by $w_{i,j}$
Top terms by k	Top terms by k^w	Top terms by k	Top terms by k^w	
time	system	new	san_diego	(gordon_banks_skepticism_chastity, intellect)
people	time	san_diego	new	(white_house_office_press_secretary, release_april)
way	data	observatory	observatory	(24x, speedstar)
system	people	telescope	telescope	(tor, det)
information	use	array	green	(francis, jagr)
years	control	transportation	array	(terrorists, drug_dealers)
year	information	green	transportation	(jet, propulsion)
case	space	research_center	cleveland	(prime, minister)
number	program	imaging	research_center	(int, char)
things	power	institute	imaging	(det, que)

Table 4.2: Top 10 terms ranked by unweighted and weighted degree for both the count and NPMI co-occurrence networks before and after thresholding along with the top 10 edges by weight for each network.

count network are extremely similar to the non-thresholded weighted degree ranking, which is expected as the low weight edges are exactly those that contribute to degree without contributing much to weighted degree. The top terms by weighted degree are very close to the top terms by degree and the non-thresholded top-terms by weighted degree, which tells us that the thresholding of the count network does not have a major impact on the most important vertices in the network.

As in the count network, thresholding the NPMI network causes the top terms by unweighted and weighted degree to become more similar. Unlike the count network, the top terms after thresholding the NPMI network are completely different than those with no thresholding. The terms become much less general, e.g. "telescope", and include some n-gram terms, e.g. "research_center". When we use NPMI for edge weights, the low edge weights are those that indicate a weak relationship which removes many of the edges connected to generic terms. With raw count edge weights, an edge weight of 5 may not be significant if connected to terms that occur hundreds of times while and edge weight of 2 could be significant connected to terms that occur

only a handful of times.

That the raw count and NPMI weights provide very different information on the importance of the relationships between terms can be clearly seen by looking at the top 10 edges by weight for each network in Table 4.2. Looking at the top edges for the count network, there is an obviously strong relationship between terms such as "god" and "bible" and "god" and "jesus". Some of the relationships are between terms that are so generic that it is hard to argue that their connections should be some of the most important for finding a topic, e.g. "people" and "things". The top edges for the NPMI network are completely different and are relationships one would expect to see in a topic given the categories of the dataset. The terms in edges like ("terrorists", "drug_dealers"), ("prime", "minister"), and ("jet", "propulsion") are quite obviously related and could come from categories related to guns, politics, and space science. ("tor", "det") and ("det", "que") are NHL teams and ("francis", "jagr") are NHL players which come from the hockey category. ("24x", "speedstar") comes from the name of a piece of computer hardware and ("int", "char") are data types; these could come from one of several computer technology related categories. ("white_house_office_press_secretary", "release_april") seems to come from a political category and an April release by the White House Office of the Press Secretary. The greatest weight is on the edge ("gordon_banks_skepticism_chastity", "intellect"). This seems to come from the signature of a user, Gordon Banks, who includes a quote by George Santayana: "Skepticism is the chastity of the intellect, and it is shameful to surrender it too soon." The n-gram extraction is picking up on this repeated pattern as is the edge weight between the terms.

Our analysis demonstrates that these co-occurrence networks do encode information about conceptually related terms and thus we should expect to be able to extract topics from the networks. The information carried by the edge weights differs greatly between the raw counts and NPMI. The effect of thresholding is also different for the two different types of networks. The topics found in each network may thus differ. The structure of the count network is much closer, but not identical, to that of the LFR networks while the NPMI

network is very different. The algorithms that performed best in Chapter 3 may not perform the best on the topic modelling task given these differences. We now present our community detection-based algorithm for topic modelling that operates on these networks.

4.2 Community Topic

We call our community detection-based topic modelling algorithm Community Topic. The algorithm takes in a corpus of documents that have been preprocessed as described in Section 4.1.1. First, a network is constructed from the document corpus. The user can select whether to use a sentence-based co-occurrence window or a sliding window of a fixed size. The user can also select whether to assign edge weights based on raw co-occurrence counts or NPMI and whether to threshold the edge weights. As a practical matter, the NPMI edge weights should at minimum be thresholded at 0 as negative edge weights cannot be handled by most community detection algorithms. As we see from Figure 4.2b, very few NPMI edge weights are negative so the impact of this thresholding on network structure is small. After the network is constructed, Community Topic applies a community detection algorithm such as SIWO or Leiden to find the communities in the network. Communities of size 1 or 2 are filtered out as outlier terms that belong to no proper topic. Finally, each topic is sorted based on vertex properties so that the most important and relevant terms for the topic come first and the topics are returned.

Sorting and ranking the terms in a topic is important as topics are commonly labelled by the top few terms and evaluation of topic model quality is usually done using only a fixed number of top terms from each topic. The topics produced by LDA are probability distributions over terms so the top terms are simply those with the highest probabilities. The topics produced by Community Topic are groups of vertices so we use the properties of the vertices to rank the terms by importance. There are several choices for which property to use. The degree k of the vertex can be used as the most central terms for a topic should occur relatively frequently and will thus occur with many

other terms rather than being related to only a few other terms. However, this ignores edge weights so weighted degree k^w can be used to rank terms instead. One problem with both k and k^w is that they make no distinction between connections within the same topic and between topics. As we saw in Table 4.2, the terms with the highest unweighted and weighted degrees are often the most generic which we would expect to connect to other terms in many different topics. We can thus use the internal unweighted and weighted degrees to rank vertices. These measures only take into account edges that connect to other vertices in the same community and so do a better job measuring the centrality of a term to a topic. These degree measures are all biased to more frequent terms, as are the probabilities of LDA. One could argue that more frequent terms are not necessarily the most representative but rather we should look at the proportion of a vertex's edges that connect to vertices within the same community. This is the embeddedness [56] of a vertex, defined as the internal degree divided by degree where either the unweighted or weighted degrees can be used. All these measures take advantage of the information in the network structure to measure the importance of a term.

Algorithm 3 Community Topic

Require: Preprocessed corpus D, parameters $window$, $weight$, $threshold$
  $G \leftarrow$ buildNetwork$(D, window, weight, threshold)$
  $Communities \leftarrow$ communityDetection(G)
  $Topics \leftarrow \{\}$
  for $community \in Communities$ **do**
    if $community$.length() > 2 **then**
      sort($community$)
      Topics.add($community$)
    end if
  end for
  return $Topics$

4.3 Parameter Evaluation

One of our goals is an algorithm for topic modelling that has a minimal number of hyperparameters for the user to have to tune. As described so far, our

algorithm can vary in the co-occurrence window, the edge weights, the weight threshold, the community detection algorithm (and any parameters for that algorithm), and the term ordering. Before we can compare to other topic modelling approaches, we first must evaluate whether different parameter settings have a impact on the quality of the topic model and if so which parameters work best.

4.3.1 Evaluation Metrics

To compare different topic models, we use two coherence measures: C_V and C_{NPMI}. Both measures have been shown to correlate with human judgements of topic quality with C_V having the strongest correlation [97]. Even though C_V has stronger correlation that C_{NPMI} with human evaluations, C_{NPMI} is more commonly used in the literature [42], possibly due to the extra computation required by C_V. We prefer the C_V measures as, in addition to being more highly correlated with human judgement, it considers the similarity of the contexts of the terms, not just their own co-occurrence. Synonyms such as the terms "movie" and "film" have similar contexts and co-occur with terms such as "director" and "actor", though they may not co-occur together frequently themselves as one would typically choose to use one term or the other. We use Gensim[6] to compute both measures. Table 4.3 shows that the scores are correlated when various Community Topic models are evaluated, but not perfectly so. We thus present scores from both. Each dataset has a train/test split. We train all models on the train documents and evaluate using the test documents. We use the standard 110-term window for C_V and 10-term window for C_{NPMI}.

Typically, topic coherence is evaluated by taking the top-N topic terms for fixed N, e.g. $N = 10$, and computing the measure only on those terms. It has been shown that the robustness of evaluation is improved by using several different values for N and averaging the scores for each [57]. We average over scores using $N \in \{5, 10, 20\}$.

To measure the quality of a topic hierarchy, we use two measures proposed

[6]https://radimrehurek.com/gensim/models/coherencemodel.html

		Pearson's r	p-value
20Newsgroups	Noun only	0.720	6×10^{-208}
	No filtering	0.588	2×10^{-121}
Reuters-21578	Noun only	0.575	5×10^{-115}
	No filtering	0.383	1×10^{-46}
BBC News	Noun only	0.460	1×10^{-68}
	No filtering	0.123	1×10^{-5}

Table 4.3: Pearson correlation coefficient between C_V and C_{NPMI}. A statistically significant positive relationship exists as indicated by the p-values, although the strength of relationship varies by dataset and with parts-of-speech filtering.

in [48]: topic specialization and hierarchical affinity. Topic specialization measures the distance of a topic's probability distribution over terms from the general probability distribution of all terms in the corpus given by their occurrence frequency. We expect topics at higher levels in the hierarchy closer to the root to be more general and less specialized and topics further down the hierarchy to be more specialized. Hierarchical affinity measures the similarity between a super-topic and a set of sub-topics. We expect higher affinity between a parent topic and its children and lower affinity between a parent topic and sub-topics which are not its children.

4.3.2 Community Detection Algorithms

The evaluation of the weighted community detection algorithms in Chapter 3 gave us some idea of their relative performance, but as we saw in Section 4.1.3 the structure and edge weights of the synthetic LFR networks are quite different than those of the count and NPMI co-occurrence networks. We cannot assume that the best performing algorithms will be the same on both sets of networks, and indeed our experiments show this. Infomap was the best performing algorithm in Chapter 3, but when applied to the co-occurrence networks tended to find only one or two large communities, which is clearly not what we desire from a topic model and precludes iteratively applying the algorithm to find sub-topics. Conversely, Leiden with the CPM objective function tended to find several hundred very small topics with only a few words

each. Decreasing the resolution parameter to induce larger topics did shrink the number of communities but did so by increasing the size of only one community, which is also not consistent with a good topic model.

We want the algorithm to be able to find a reasonable number of topics as a starting point, from which sub- or super-topics can be found. We thus limit our evaluation to SIWO, Leiden with the modularity objective function, and WalkTrap. Leiden has the tunable resolution parameter which we vary as well.

4.3.3 Parameter Combinations

We have three datasets. We train on each using both no parts-of-speech filtering and filtering all non-nouns. We create co-occurrence networks using both raw count and NPMI edge weights and threshold at 0 and 2 for the count networks and 0 and 0.35 for the NPMI networks. We use a sentence co-occurrence definition as well as sliding windows of size 5 and 10. We detect communities using Walktrap, SIWOw+(a), SIWOw+(g), and Leiden(m) using resolution parameters of 1, 1.5, 2, and 2.5. We try ordering topics by degree, weighted degree, internal degree, internal weighted degree, embeddedness, and weighted embeddedness. We evaluate with C_V and C_{NPMI} with top-$N \in \{5, 10, 20\}$. This gives us a total of 18,144 different evaluations which we use as data for comparing the various settings.

4.3.4 Term Ordering

The terms in a topic need to be ordered so that the most important can be used for labelling, evaluation, and human understanding. Vertices in a network have many properties that can be used to measure importance. In Table 4.4 we present the C_V and C_{NPMI} coherence scores for each community detection algorithm for six different term ordering schemes. The best performing for all algorithms are internal weighted degree and internal degree. These take into account only the connections within a community, not those between communities whereas weighted degree and degree would give undue importance to terms that connect to many terms in other topics. Weighted embeddedness

Ordering	Coherence	SIWOw+	Leiden	WalkTrap
Internal Weighted Degree	C_V	$\mathbf{0.478 \pm 0.003}$	$\mathbf{0.533 \pm 0.002}$	$\mathbf{0.545 \pm 0.007}$
	C_{NPMI}	$\mathbf{-0.131 \pm 0.005}$	$\mathbf{-0.058 \pm 0.004}$	$\mathbf{0.041 \pm 0.007}$
Internal Degree	C_V	$\mathbf{0.476 \pm 0.003}$	0.521 ± 0.002	$\mathbf{0.545 \pm 0.007}$
	C_{NPMI}	$\mathbf{-0.134 \pm 0.005}$	$\mathbf{-0.064 \pm 0.004}$	$\mathbf{0.042 \pm 0.007}$
Weighted Degree	C_V	0.471 ± 0.003	0.458 ± 0.002	0.492 ± 0.006
	C_{NPMI}	-0.146 ± 0.005	-0.146 ± 0.005	-0.020 ± 0.011
Degree	C_V	0.470 ± 0.003	0.450 ± 0.002	0.489 ± 0.006
	C_{NPMI}	-0.146 ± 0.005	-0.150 ± 0.005	-0.023 ± 0.010
Weighted Embeddedness	C_V	0.471 ± 0.003	0.470 ± 0.003	0.481 ± 0.006
	C_{NPMI}	-0.155 ± 0.005	-0.277 ± 0.004	-0.223 ± 0.011
Embeddedness	C_V	0.469 ± 0.003	0.470 ± 0.003	0.484 ± 0.006
	C_{NPMI}	-0.158 ± 0.005	-0.295 ± 0.004	-0.245 ± 0.011

Table 4.4: Average coherence scores for each community detection algorithm by ordering scheme $\pm$ the standard error of the mean. Bold indicates best result for each algorithm.

and embeddedness consider the proportion of internal connections, but by ignoring the absolute values they give undue importance to infrequent terms. This negatively impacts the C_{NPMI} scores more than the C_V scores as the C_{NPMI} is calculated with a smaller sliding window so is less likely to capture the co-occurrence of infrequent terms.

We see that the best ordering schemes are those that rank highly terms that are more frequent and more connected to other terms in the same topic, which matches our intuition of what an important term should be. We favour the weighted version as it has a slight edge in scores and incorporates more information even though the difference is very small or nonexistent.

The sorting scheme has a much smaller impact on SIWOw+ than Leiden or WalkTrap. We saw in Table 3.3 that SIWOw+ finds more, smaller communities than Leiden which finds fewer, larger communities. There will be a greater overlap between the terms being evaluated across the different ordering schemes when the communities are small and so the ordering scheme will have less of an impact on score. WalkTrap found a similar number of communities to SIWOw+ in Table 3.3 so the algorithm may be behaving differently on these co-occurrence networks. We will investigate this further when we examine the topics themselves.

For the rest of our experiments, we use only the evaluation scores found using the internal weighted degree ordering.

POS	Coherence	SIWOw+	Leiden	WalkTrap
All	C_V	0.472 ± 0.004	0.515 ± 0.003	$\mathbf{0.554 \pm 0.010}$
	C_{NPMI}	$\mathbf{-0.129 \pm 0.007}$	-0.082 ± 0.006	$\mathbf{0.051 \pm 0.012}$
Noun only	C_V	$\mathbf{0.483 \pm 0.004}$	$\mathbf{0.549 \pm 0.003}$	$\mathbf{0.537 \pm 0.009}$
	C_{NPMI}	$\mathbf{-0.132 \pm 0.008}$	$\mathbf{-0.035 \pm 0.006}$	$\mathbf{0.031 \pm 0.009}$

Table 4.5: Average coherence scores for each community detection algorithm by parts-of-speech (POS) filtering $\pm$ the standard error of the mean. Bold indicates best result for each algorithm.

4.3.5 Parts-of-Speech Filtering

Different parts-of-speech (POS), e.g. nouns, verbs, adjectives, serve different functions in language and may be more or less important to the topical content of a document. Some stop words such as "the" are so general and carry such little information about the content of a document that they are removed from the corpus prior to topic modelling as a matter of course. Research has also shown that removing non-noun terms can improve the quality of topics discovered with LDA [67]. We would like to check whether this is true of Community Topic. While non-noun terms may not be the most important terms in a topic, their presence or absence in a corpus could impact the structure of the co-occurrence networks and the discovered communities, for better or worse.

We can see in Table 4.5 that filtering out non-noun terms can have an impact on the quality of the discovered communities. The topics found by Leiden tend to be significantly more coherent when non-noun terms are filtered out. The topics found by SIWOw+ on the noun-only corpora are slightly more coherent when measured by C_V but there is no significant difference in C_{NPMI}. There is no significant difference in either measure for the topics found by WalkTrap.

Even without filtering, many of the top topic words are nouns and filtering non-nouns improves the running time so we prefer to keep only nouns. Still, it is encouraging that our approach is fairly robust to the presence of non-noun terms.

Window	Coherence	SIWOw+	Leiden	WalkTrap
Sentence	C_V	0.484 ± 0.005	0.533 ± 0.004	0.541 ± 0.011
	C_{NPMI}	-0.125 ± 0.009	-0.069 ± 0.008	0.042 ± 0.013
Sliding 5	C_V	0.473 ± 0.005	0.530 ± 0.004	0.542 ± 0.012
	C_{NPMI}	-0.137 ± 0.009	-0.049 ± 0.007	0.037 ± 0.014
Sliding 10	C_V	0.476 ± 0.005	0.535 ± 0.004	0.553 ± 0.011
	C_{NPMI}	-0.131 ± 0.012	-0.057 ± 0.008	0.044 ± 0.012

Table 4.6: Average coherence scores for each community detection algorithm by co-occurrence window $\pm$ the standard error of the mean. There are no significant differences between the co-occurrence windows for any of the algorithms for either coherence measure.

4.3.6 Co-occurrence Window

Community Topic operates on a term co-occurrence network, but there is no fixed universal definition of co-occurrence. We evaluate three different definitions to see whether they cause a large difference in the quality of communities found. The first definition of co-occurrence is terms occurring in the same sentence. This takes advantage of the information embedded in the sentence structure of the document; a sentence typically expresses at most one idea so terms in the same sentence will be related. However, adjacent sentences in the same document are likely to be related so terms in those sentences are likely to be related. Therefore a definition based on a sliding window over the document could capture information about term relationships missed by sentence co-occurrence. However, one must define the size of the sliding window. If the window is too large, there is a greater chance of unrelated terms co-occurring. If the window is too small, there is a chance that related terms will not co-occur. We evaluate sliding windows of size 5 and 10.

We can see in Table 4.6 that there is no significant difference between any of the co-occurrence definitions for any of the algorithms for either of the coherence measures. As we can see from Figures 3.1 and 3.2, communities do not consist solely of vertices that are all fully connected as in a clique. Vertices that are not themselves connected can be a part of the same community when they both connect to the same vertex. The community detection algorithms are thus robust to small changes in patterns of connections and edge weights

caused by the differenct co-occurrence definitions.

4.3.7 Edge Weight Type and Thresholding

We saw in Section 4.1.3 that the properties of the co-occurrence networks are very different depending on whether the raw count or NPMI edge weights are used and that these properties change when we use thresholding to remove edges with low weights. In Section 3.4 we saw that the performance of SIWOw+ is sensitive to the density of the network and the relative magnitude of the weights. SIWOw+ is not the only algorithm sensitive to the structure of the network, as illustrated by the fact that Infomap was the best performing on the LFR networks in Section 3.3 but fails to find distinct communities on most co-occurrence networks. We thus expect the edge weight type and thresholding to have a large impact on the performance of the community detection algorithms and topic quality, and indeed our experiments confirm this.

The results in Table 4.7 show that SIWOw+ performs best by the C_V measure on the count co-occurrence networks with no thresholding. By the C_{NPMI} measure, SIWOw+ performs best on both the count and NPMI networks without thresholding. It is interesting to see that thresholding has a negative impact on topic quality for both network types when Table 4.1 shows that the thresholding has the opposite impact on the clustering coefficient for the count and NPMI networks. The difference in performance for SIWOw+ on the count and NPMI networks shows that the magnitude of the edge weights is not having a major impact on the algorithm's ability to detect communities.

WalkTrap's scores are similar to those of SIWOw+. The performance on the non-thresholded networks is better with generally better performance on the count networks rather than the NPMI networks. The pattern is the opposite for Leiden where the resulting topics are more coherent when mined from the thresholded networks. Leiden performs equally well on the count and NPMI networks when measured by C_V but best on the count networks when measured by C_{NPMI}.

Weight Type	Threshold	Coherence	SIWOw+	Leiden	WalkTrap
Count	> 0	C_V	$\mathbf{0.517 \pm 0.008}$	0.521 ± 0.004	$\mathbf{0.577 \pm 0.016}$
		C_{NPMI}	$\mathbf{-0.072 \pm 0.010}$	-0.045 ± 0.008	$\mathbf{0.113 \pm 0.012}$
	> 2	C_V	0.474 ± 0.004	$\mathbf{0.534 \pm 0.004}$	0.537 ± 0.009
		C_{NPMI}	-0.113 ± 0.006	$\mathbf{-0.003 \pm 0.006}$	0.051 ± 0.016
NPMI	> 0	C_V	0.498 ± 0.004	$\mathbf{0.535 \pm 0.005}$	$\mathbf{0.557 \pm 0.012}$
		C_{NPMI}	$\mathbf{-0.085 \pm 0.007}$	-0.081 ± 0.010	0.071 ± 0.007
	> 0.35	C_V	0.422 ± 0.003	$\mathbf{0.541 \pm 0.005}$	0.509 ± 0.013
		C_{NPMI}	-0.254 ± 0.006	-0.104 ± 0.008	-0.070 ± 0.011

Table 4.7: Average coherence scores for each community detection algorithm by edge weight type and threshold $\pm$ the standard error of the mean. Bold indicates best result for each algorithm.

Mean	Coherence	SIWOw+
Arithmetic	C_V	0.476 ± 0.005
	C_{NPMI}	-0.136 ± 0.007
Geometric	C_V	0.479 ± 0.004
	C_{NPMI}	-0.126 ± 0.007

Table 4.8: Average coherence scores for the communities found using SIWOw+ with either the arithmetic or geometric mean $\pm$ the standard error of the mean. There are no statistically significant best results.

4.3.8 Community Detection Hyperparameters

SIWOw+ and Leiden have hyperparameters which can impact the results of the community detection. In the case of SIWOw+, the function used to combine the three edge weights of a triangle can be varied. In the case of the LFR networks, we did not see a significant difference between using different average functions until the network structure became very weak and the arithmetic and geometric means performed slightly better than the harmonic mean and minimum. We evaluate the arithmetic and geometric means in Table 4.8 and do not find a statistically significant difference, indicating that the topic modelling is robust to the choice of mean.

The Leiden algorithm can be used with modularity or CPM as the objective function. As previously discussed, the topics found using CPM were very poor so we do not consider it for Community Topic. Leiden also has a resolution parameter which impacts the size of communities detected and can combat either modularity's resolution limit (by increasing the parameter to detect smaller communities) or its field of view limit (by decreasing the parameter

Resolution	Coherence	Leiden
1.0	C_V	$\mathbf{0.562 \pm 0.005}$
	C_{NPMI}	$\mathbf{0.037 \pm 0.005}$
1.5	C_V	$\mathbf{0.552 \pm 0.005}$
	C_{NPMI}	-0.025 ± 0.007
2.0	C_V	0.519 ± 0.004
	C_{NPMI}	-0.097 ± 0.008
2.5	C_V	0.499 ± 0.003
	C_{NPMI}	-0.148 ± 0.008

Table 4.9: Average coherence scores for the communities found using Leiden with various resolution parameters ± the standard error of the mean. Bold indicates best results.

to detect larger communities). Leiden already finds rather large communities on the co-occurrence networks with the default resolution parameter of 1.0, so we experiment with increasing the parameter to find smaller communities and thus more specific topics.

In Table 4.9, we see that the coherence of the discovered topics decreases as the resolution parameter increases and the discovered topics become smaller and more specific. This may be an artifact of how topic model evaluation is conducted, using only the top few terms from each topic. This may hide the incoherence of larger topics, most of whose terms will not impact the evaluation but may not be related to the top topic terms. We will investigate this further in our comparisons with LDA and top2vec.

4.3.9 Community Detection Algorithm

Our evaluations in this section so far have revealed that at least one parameter choice for our Community Topic algorithm has a clear best choice. The ordering of terms in a topic by the internal weighted degree of the corresponding vertex in the network matches our intuition as to what makes a term important to a topic and empirically gives the most coherent results. In other cases, there is no clear best choice as the aggregate results are not sensitive to the parameter. The type of edge weight scheme to use and whether to threshold depend upon the community detection algorithm being used. In the results presented so far, the WalkTrap algorithm has tended to have the highest co-

herence scores, followed by Leiden and then SIWOw+. However, these results have been averaged over many different runs on different datasets with different parameters. To determine which algorithm performs best, we compare each using the best parameter settings for that algorithm. For each community detection algorithm, we find the best performing parameter combination both on the individual datasets and aggregated across datasets. We use C_V to determine the best combination but also present results for C_{NPMI}. In some cases C_V and C_{NPMI} agree but not always.

For SIWOw+, we can see the coherence results averaged across all three datasets as well as for each dataset individually in Table 4.10. The individual dataset results are presented with both the best parameters for the average, denoted P_{AVG}, and the best parameters for that individual dataset, denoted P_{DS}. The best parameter combination averaged across all three datasets is noun-only POS, arithmetic mean, sentence co-occurrence window, count edge weights, and no thresholding. These parameters are only the second best performing on the 20Newsgroups dataset, where slightly better results are achieved by not filtering non-noun POS but otherwise keeping the parameters the same. On the Reuters corpus, the best performing parameters are quite different than the best average parameters: no POS filtering, geometric mean, sliding window size 10, and NPMI edge weights with no thresholding. These different parameters make quite a difference to performance, especially when measured by C_{NPMI}. On the BBC dataset, the best performing parameters are close to the average best with only the geometric mean being the difference. This one change does make a difference to performance, though. That the best performing parameter combinations on two of the three datasets are similar both to each other and the best average combination is encouraging. However, the best combination on Reuters is very different and indicates that Commuity Topic is not as insensitive to these parameters as the aggregate results suggested, at least when using SIWOw+ as the community detection algorithm.

The ranking of the parameter combinations by C_V and C_{NPMI} tends to be close. The exception is the P_{AVG} parameters on the Reuters dataset. These

Coherence	Average	20NG		Reuters		BBC	
		P_{DS}	P_{AVG}	P_{DS}	P_{AVG}	P_{DS}	P_{AVG}
C_V	0.569 (1)	0.686 (1)	0.665 (2)	0.536 (1)	0.513 (6)	0.573 (1)	0.528 (6)
C_{NPMI}	-0.022 (2)	0.148 (1)	0.134 (2)	0.007 (2)	-0.138 (27)	-0.038 (2)	-0.062 (5)

Table 4.10: Best coherence scores when using SIWOw+ as the community detection algorithm. Average results for all three datasets as well as results on each corpus using both the best parameters for that corpus P_{DS} as well as the best parameters for the average P_{AVG}. The rank of that combination is given in parentheses next to the score.

Coherence	All Datasets	20NG		Reuters		BBC	
		P_{DS}	P_{AVG}	P_{DS}	P_{AVG}	P_{DS}	P_{AVG}
C_V	0.655 (1)	0.665 (1)	0.665 (1)	0.642 (1)	0.642 (1)	0.676 (1)	0.659 (2)
C_{NPMI}	0.114 (1)	0.106 (4)	0.106 (4)	0.113 (2)	0.113 (2)	0.028 (16)	0.122 (1)

Table 4.11: Best coherence scores when using Leiden as the community detection algorithm. Average results for all three datasets as well as results on each corpus using both the best parameters for that corpus P_{DS} as well as the best parameters for the average P_{AVG}. The rank of that combination is given in parentheses next to the score.

parameters get the 6^{th} best score by C_V but are middle of the pack by C_{NPMI}, 27^{th} out of 48. This is also the only case of the P_{AVG} parameters performing very poorly on a dataset for SIWOw+.

For Leiden, the best average parameters P_{AV} are noun-only POS, resolution parameter of 1.0, sentence co-occurrence window, NPMI edge weights, and no thresholding. We can see in Table 4.11 that these achieve the best C_V and C_{NPMI}. These parameters are also the best on the 20Newsgroups corpus by C_V, although they do not achieve the best C_{NPMI}. The same is true for the Reuters corpus. For the BBC corpus, the P_{AVG} parameters get the second best C_V and best C_{NPMI}. The P_{DS} parameters for BBC are quite different: no POS filtering, resolution parameter of 1.0, sliding window of size 10, NPMI edge weights, and thresholding at 0.35. These do not achieve a very high C_{NPMI}, however.

Compared to SIWOw+, the Leiden topics achieve worse scores on the 20Newsgroups dataset but better scores on Reuters and BBC. Also, Leiden performs well with the same set of parameters on all three datasets whereas SIWOw+ performs best with different parameter settings on each.

Coherence	All Datasets	20NG		Reuters		BBC	
		P_{DS}	P_{AVG}	P_{DS}	P_{AVG}	P_{DS}	P_{AVG}
C_V	0.632 (1)	0.759 (1)	0.720 (2)	0.621 (1)	0.576 (5)	0.683 (1)	0.598 (6)
C_{NPMI}	0.150 (1)	0.235 (1)	0.185 (5)	0.274 (1)	0.199 (3)	0.031 (11)	0.067 (6)

Table 4.12: Best coherence scores when using WalkTrap as the community detection algorithm. Average results for all three datasets as well as results on each corpus using both the best parameters for that corpus P_{DS} as well as the best parameters for the average P_{AVG}. The rank of that combination is given in parentheses next to the score.

WalkTrap performs best on average with P_{AVG} parameters of no POS filtering, sliding window of size 5, count edge weights, and no thresholding. These best average parameters are second best by C_V on the 20Newsgroups dataset, but the P_{DS} are only slightly different with the only change being the use of the sentence co-occurrence window. The P_{AVG} parameters are only the 5^{th} best performing by C_V on the Reuters corpus, but the P_{DS} best parameters differ only in thresholding the edge weights at 2. On the BBC corpus, the P_{AVG} parameters only achieve the 6^{th} best scores for both C_V and C_{NPMI}. The P_{DS} parameters are quite different with noun-only POS filtering, a sliding co-occurrence window of size 10, and NPMI edge weights with no thresholding.

Comparing the three algorithms, we can say that Leiden is the most robust as it achieves good C_V ans C_{NPMI} scores using the same set of P_{AVG} parameters. SIWOw+ performs worse on average than Leiden and worse on the Reuters and BBC corpora. However, SIWOw+ outperforms Leiden on both C_V and C_{NPMI} on the 20Newsgroups corpus. The fact that SIWOw+ performs best with different parameters on each dataset is a drawback. WalkTrap achieves the highest scores on 20Newsgroups and BBC, but like SIWOw+ it suffers from having a different best performing parameter combination on each dataset.

While using automated metrics such as C_V and C_{NPMI} is the only practical way to evaluate thousands of different topic models, we do not want to blindly follow what a single number is telling us. This is true when using coherence metrics on LDA models. Given the tendency of LDA and neural topic models to produce redundant topics, the authors of [78] introduce a uniqueness metric

to measure the overlap between topics as another view on topic model quality. This is not a concern with our approach as there is no overlap between topics. However, we want to examine the number of topics found, the sizes of these topics, the ability to find super- and/or sub-topics, and the proportion of terms that are included in the topics rather than discarded as outliers.

We examine the topics found on the BBC network as this corpus consists of news articles written for general consumption and so should be interpretable without needing any special familiarity with the corpus. The news articles come from five categories ("business", "entertainement", "politics", "sport", "tech"), but these are so broad that there should be many identifiable sub-topics.

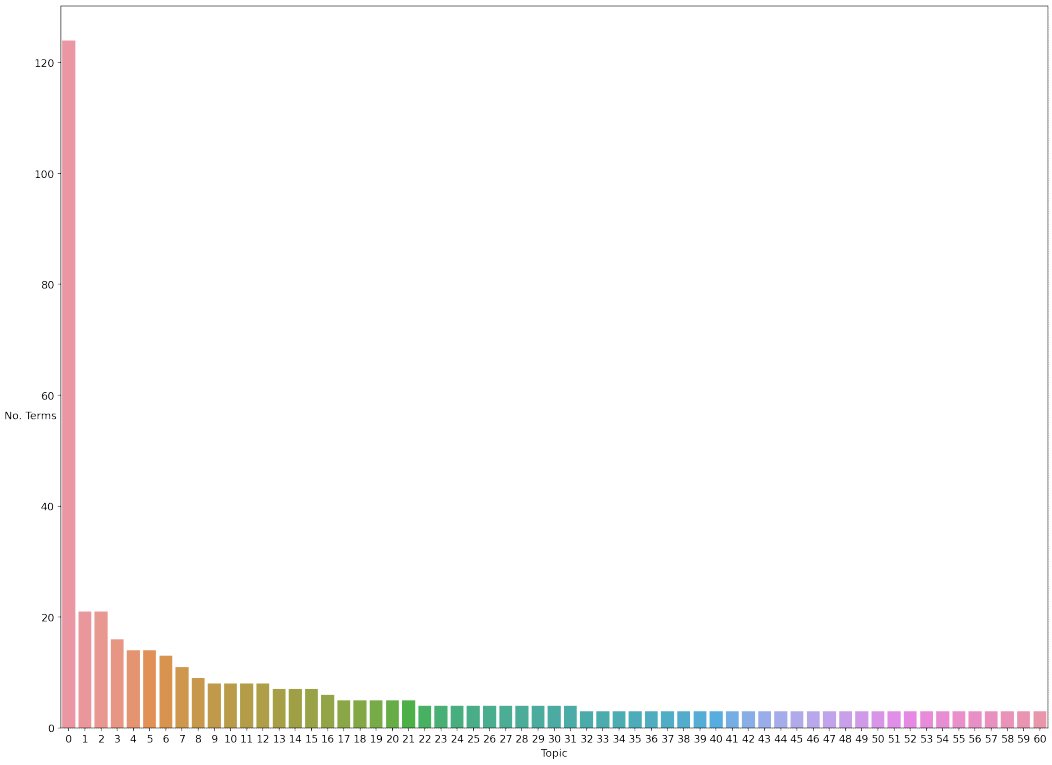


Figure 4.10: Distribution of community sizes found by SIWOw+. The algorithm detects many small communities and excludes a large proportion of terms as outliers.

As we see in Figure 4.10, SIWOw+ finds many small communities. The algorithm also eliminates the majority of terms as outliers. Of the 2,547 vocabulary terms, SIWOw+ discards 2,093 (82%) as outliers. While not every

term in the vocabulary is necessarily relevant to some definite concept or topic, it seems unlikely that $< 20\%$ of the vocabulary is relevant to the topics of the corpus, especially given that stop words and very common and uncommon terms have been removed already.

We can see the topics found by SIWOw+ in the first column of Table 4.13 labelled by the three top terms. On the whole, the topics are coherent and are recognizable as sub-topics of one of the five categories. They are perhaps too specific, for example topics 22 and 36 both relate to Russian oil and gas companies and 53 and 58 both deal with police.

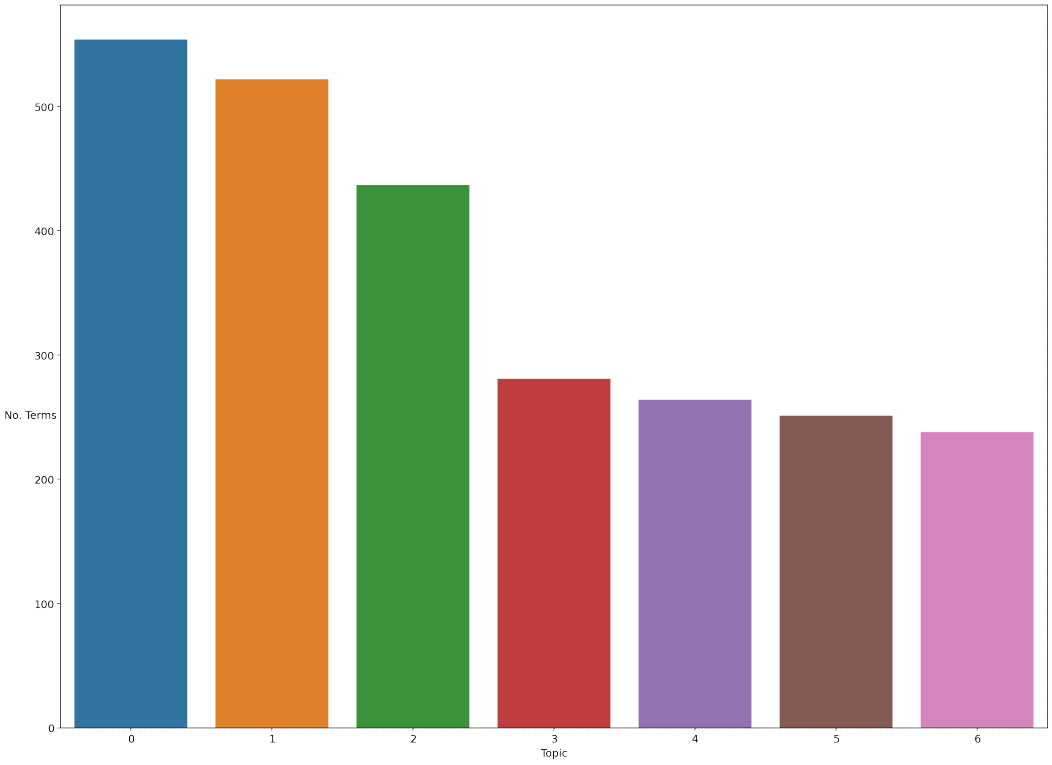


Figure 4.11: Distribution of community sizes found by WalkTrap. The algorithm detects fewer large communities and discards no terms as outliers.

WalkTrap achieves higher C_V and C_{NPMI} scores than SIWOw+. As we can see from Figure 4.11, it finds fewer, larger topics than SIWOw+. WalkTrap does not discard any terms as outliers, i.e. 100% of the terms in the vocabulary are part of a topic. The topics appear coherent and identifiable from the top three terms in Table 4.13. Topic 1 corresponds to the "business" category, topic 2 corresponds to the "politics" category, topic 4 corresponds to

SIWOw+	WalkTrap	Leiden (1.0)	Leiden (1.5)
1. year, people, time	1. growth, firm, economy	1. government, minister, labour	1. people, technology, users
2. game, opening, xbox	2. government, labour, minister	2. england, game, time	2. england, game, wales
3. number, week, meeting	3. people, year, time	3. firm, year, economy	3. film, award, actor
4. film, director, comedy	4. film, award, awards	4. people, technology, users	4. labour, election, party
5. firm, market, analyst	5. england, win, chelsea	5. film, award, awards	5. economy, growth, dollar
6. company, shareholders, investigation	6. users, technology, software		6. firm, company, deal
7. music, artists, itunes	7. team, france, player		7. police, law, court
8. labour, election, party			8. aid, countries, disaster
9. way, languages, colleague			9. olympics, athens, tribunal
10. nations, england, scotland			10. serena_williams, round, lleyton_hewitt
11. services, operators, music_download			11. pension, money, pensions
12. minister, chancellor, iraq_war			12. schools, teachers, children
13. court, judge, arbitration			13. channel, viewers, bbc
14. technology, intel, industry_experts			14. college, london, university
15. software, microsoft, windows			15. way, coaches, kids
16. growth, eurozone, explosion			16. mass, bills, names
17. chelsea, everton, mourinho			17. living, pundits, mini
18. coach, woodward, lions			18. identity, convention, documents
19. try, ball, corner			19. result, enthusiasm, time
20. group, consortium, merger			
21. london, headquarters, east			
22. yukos, sale, auction			
23. games, console, league			
24. oil, gas, barrels			
25. sprinters, katerina_thanou, iaaf			
26. economy, china_s, consumer_spending			
27. tel_aviv, chicago, athens			
28. sales, volume, utility			
29. information, port, techniques			
30. aid, countries, charities			
31. companies, business, fault			
32. home_secretary, detention, david_blunkett			
33. scurity, intelligence, security_firm			
34. report, mission, religion			
35. industry, consolidation, employee			
36. state, rosneft, gazprom			
37. players, squad, dressing_room			
38. britain, races, luck			
39. definition, dvds, tvs			
40. new_zealand, blacks, lions_tour			
41. service, provider, providers			
42. cabinet, allies, suspicion			
43. google, blogger, amazon			
44. benitez, liverpool, rafael			
45. wall, shot, stage			
46. day, boxing, green			
47. computer, methods, server			
48. mail, spam, attachment			
49. arsenal, fulham, arsene_wenger			
50. france, switzerland, chinese			
51. hodgson, charlie, lewsey			
52. parents, children, teachers			
53. law, enforcement, police_officers			
54. deal, iran, venture			
55. fraud, witness, tax_evasion			
56. phone, motorola, vodafone			
57. secretary, general, david_davis			
58. suspects, house_arrest, police			
59. consumer_electronics, ces, las_vegas			
60. dollar, slide, weakness			
61. china, beijing, washington			

Table 4.13: Topics labelled by top 3 terms found by each community detection algorithm on BBC corpus.

the “entertainement” category, and topic 6 corresponds to the “tech” category. Topics 5 and 7 both relate to the “sport” category, with topic 5 focused on soccer and topic 7 on other sports such as tennis and rugby.

Leiden achieved better coherence scores than both SIWOw+ and WalkTrap on the BBC corpus, and it found the fewest communities as we see in Figure 4.12. This may point to an issue with automated coherence metrics: a bias to large topics. Since only the top few terms are used in evaluation, the bulk of the terms in a topic are not considered. As we can see from Table 4.13, the five topics are coherent and identifiable and there is a nice correspondence between the five discovered topics and the five categories of articles.

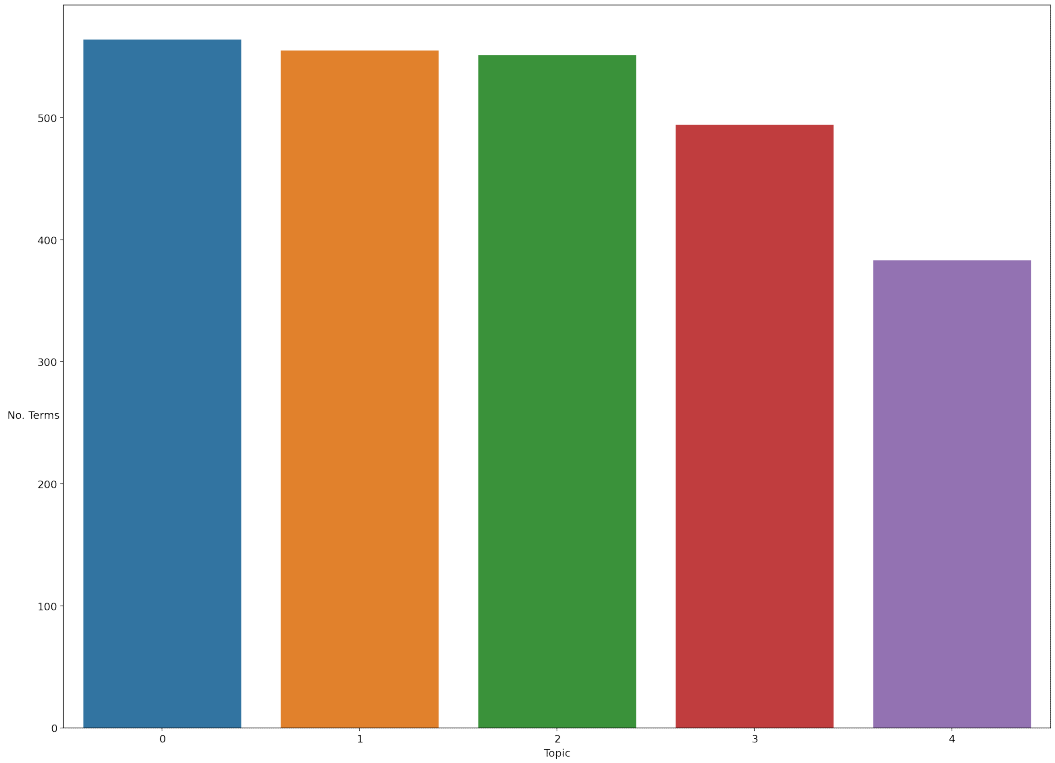


Figure 4.12: Distribution of community sizes found by Leiden with resolution parameter 1.0. The algorithm detects fewer large communities and discards no terms as outliers.

SIWOw+ finds many small topics while WalkTrap and Leiden find few large topics. Unlike SIWOw+ and WalkTrap, Leiden has a parameter to control the size of discovered communities. The best coherence scores for Leiden in Table 4.11 all came with the default resolution parameter of 1.0, but

if the automated metrics are biased to large topics then this may not be the best parameter to use. Keeping in mind the use case of a researcher exploring a corpus or a conversational agent talking to a human, a few very broad topics are less useful than many specific topics. Thus we examine the topics found by using a resolution parameter of 1.5. This finds 19 topics with a wide range of sizes, from several hundred terms to just a handful, illustrated in Figure 4.13.

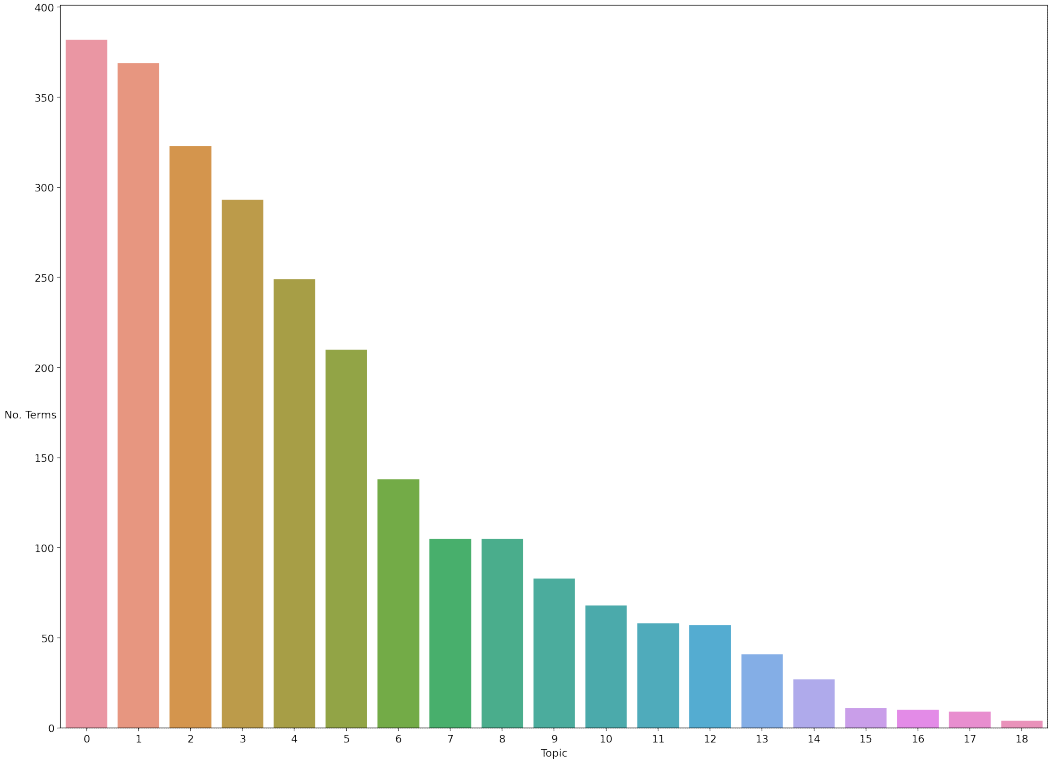


Figure 4.13: Distribution of community sizes found by Leiden with resolution parameter 1.5. The algorithm detects fewer large communities and discards only 5 terms as outliers.

As with the SIWOw+ topics, the top three terms for each topic in the fourth column of Table 4.13 are coherent and identifiable as a sub-topic of one of the five categories of article. While these topics achieve a lower automated coherence score, they may offer more insight and be of more practical use to a researcher or conversation agent.

Clearly, we cannot simply take the highest C_V or c_{NPMI} score and declare a best algorithm as the topics they produce are qualitatively very different.

Automated coherence may give us an idea of the quality of the topics, but questions have been raised here and in other work [25, 42] as to the reliability of these measures. We must also keep in mind the utility of the topic model in a given application area, the run time of the algorithm, the stability of the discovered topics, and the hierarchy of the topics. We next look at the topic hierarchy before comparing with LDA and top2vec on quality, run time, and stability.

4.3.10 Topic Relationships and Hierarchy

One of the major advantages of the network representation is that it provides a natural way to produce sub- and super-topics. A community is a sub-graph with its own network structure. Applying the community detection algorithm on the community sub-graph produces a new set of smaller communities, which are the sub-topics of the topic represented by the original community. The original detected communities also form a network where each vertex represents a community, edges exists between community vertices where there is at least one edge between a vertex of one community and a vertex of the other, and edge weights are the sum of the edge weights connecting vertices of each community. Super-topics can be found by applying community detection to this network to group related topics together.

SIWOw+ initially produces many small communities so finding sub-topics is not possible. Finding super-topics should be possible, however when we have tried applying SIWOw+ on the community network, only a handful of topics are grouped together. Some of these topics seem related, but others do not. As SIWOw+ takes the longest time, has the lowest coherence scores, only finds very small and specific topics, and is unable to find sub- or super- topics, we conclude that it is not an appropriate community detection algorithm for Community Topic on the datasets we tested on. This is surprising as it was one of the best performing algorithms both in our experiments of Chapter 3 and in previous work on unweighted community search. Other algorithms such as Infomap also performed well on the LFR benchmark and other real networks but failed to work on the co-occurrence networks. The network

structure analyzed in Section 4.1.3 must present uncommon difficulties for these algorithms.

WalkTrap achieved the highest coherence scores. Unlike SIWOw+, the topics it finds are large enough that there should be sub-topics to discover. Applying WalkTrap a second time on a community does indeed find another set of small communities. However, these sub-topics tend to consist of one large sub-topic and a handful of small sub-topics of only a few terms. Each community found on the original network is denser and has higher edge weights than the network as a whole, by definition. WalkTrap seems to struggle on these denser sub-graphs and does not find very plausible sub-topics. So even though WalkTrap finds topics that score very highly on automated coherence, the inability to access the hierarchical structure of the networks is a weak point.

Leiden is able to find sub-topics at multiple levels, illustrated in Figure 4.14. For example, the first run on the BBC corpus finds five topics corresponding to the five categories of articles. Applying Leiden to the "Business" topic sub-graph results in sub-topics about the economy, employment, the stock market, international trade, the automotive industry, and the airline industry. Leiden finds "Tech" sub-topics of video games, the web, cellphones, internet service providers, e-commerce, gadgets, and enterprise software. Even more specific topics can be found by mining the sub-topics. The "web" sub-topic yields topics about email, web search, security, software, and internet companies. This is not limited to what is illustrated in Figure 4.14. The "Sport" topic has a sub-topic about the olympics; the "olympics" sub-top has sub-topics about different sports and also doping scandals.

The modularity formula used by Leiden finds densely connected groups of vertices, where "dense" is relative to the vertex degrees. Thus Leiden can find more densely connected groups of nodes on very sparse or very dense networks. This enables more effective discovery of sub- and super-topics.

With the default resolution parameter of 1, Leiden finds five broad topics which can be mined for sub-topics. If we first search for narrow topics by using a higher resolution parameter, then we can instead find super-topics

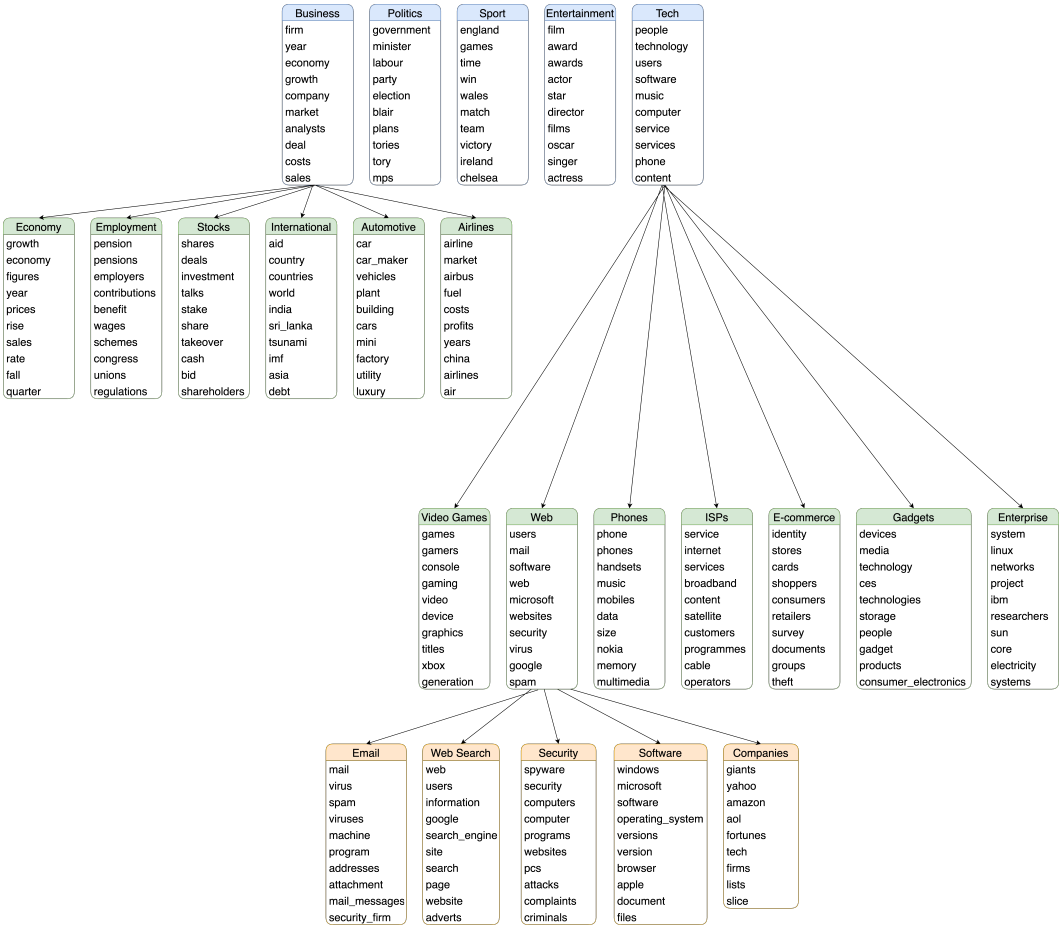


Figure 4.14: Hierarchy of BBC corpus topics found by iteratively applying Leiden algorithm. Topic labels assigned by authors.

by performing community detection on the network of communities. With a resolution parameter of 2, Leiden finds 48 topics on the BBC corpus. Performing community detection on a 48 vertex network of these communities with a resolution parameter of 2 only reduces this to 43 super-topics; as with SIWOw+, only a few vertices actually get combined into new super-topics. However, reducing the resolution parameter to 1.15 results in 9 super-topics. 5 of these roughly correspond to the 5 article categories, although they are slightly different from those found starting with a resolution parameter of 1. There are also 4 smaller super-topics. This is illustrated in Figure 4.15.

This ability to move up and down the topic hierarchy on the fly is the main advantage of using Leiden in Community Topic. Imagine a researcher first exploring a corpus. They could start with large topics and drill down

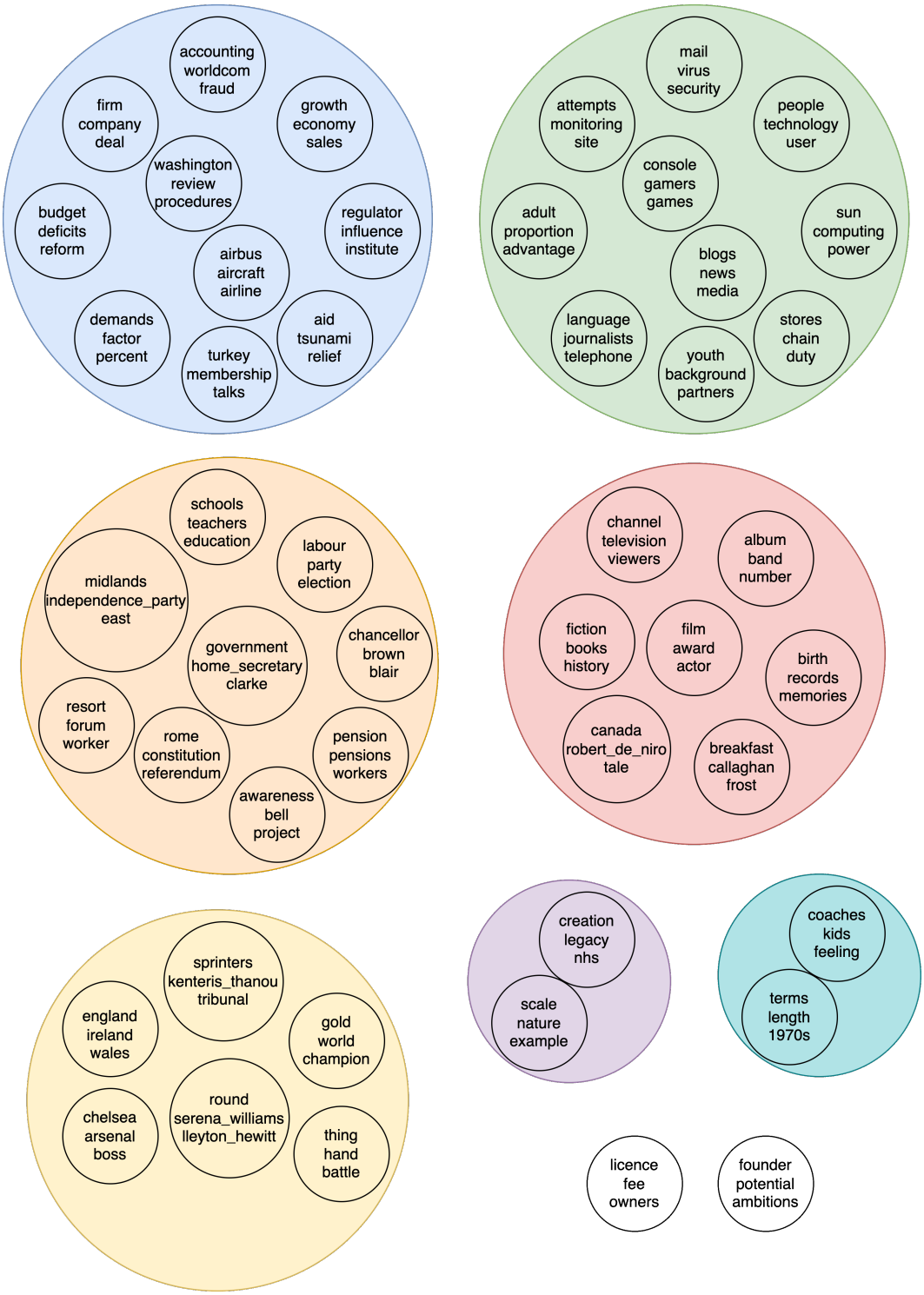


Figure 4.15: Super-topics found by applying community detection on network of small topics.

into the sub-topics of a topic of interest. Or they could start with many small topics, and find the super-topics to get more related terms to a topic of interst.

This mechanism could also be used by a conversational agent to guide the flow of a conversation, narrowing or broadening the topic of discussion as necessary.

Given that Leiden provides the richest topic hierarchy, is able to find communities of different sizes as desired using the resolution parameter, works well on all datasets with the same set of Community Topic hyperparameters, and is extremely fast, we conclude that it is the best community detection algorithm to use in Community Topic.

4.4 Topic Model Comparisons

We compare our Community Topic algorithm to LDA as a standard benchmark. We also compare to a recent algorithm based on word embeddings learned by a neural network, top2vec. We compare the algorithms on topic quality, as measured by C_V and C_{NPMI}, as well as run time and stability of topic quality over multiple runs.

4.4.1 Coherence

We run LDA on all three datasets with both noun-only POS filtering and no POS filtering for 5, 10, 20, 50, 100, and 200 topics. We run LDA for 2000 iterations with symmetric dirichlet prior of $\alpha = 1/\text{number of topics}$. Unlike [67], we do not find a significant difference in coherence for LDA based on POS filtering, shown in table 4.14. The best performing set of parameters for each dataset and the resulting coherence scores are presented in Table 4.15.

Coherence	**Noun-only**	**No POS filter**
C_V	0.415 ± 0.010	0.402 ± 0.011
C_{NPMI}	-0.076 ± 0.010	-0.068 ± 0.010

Table 4.14: LDA coherence by POS filtering. There is no significant difference between no POS filtering and noun-only filtering.

C_V and C_{NPMI} agree on the best performing parameters on the 20Newsgroups and Reuters datasets. However, on the BBC dataset the best performing model on C_V is the worst performing model by C_{NPMI}. On all three

	20Newsgroups	**Reuters**	**BBC**
Number of topics	10	5	200
POS filtering	No filtering	Noun-only	Noun-only
C_V	0.510 (1)	0.471 (1)	0.366 (1)
C_{NPMI}	0.027 (1)	0.025 (1)	-0.191 (12)

Table 4.15: Best LDA coherence on each dataset by C_V. C_{NPMI} also given. Rank out of 12 parameter combinations given in parentheses.

datasets, the coherence of the LDA models is much worse than that of the Community Topic models using any of the community detection algorithms.

The top2vec algorithm[7] does not take any parameters but like Community Topic finds the natural number of topics given the data. We present coherence results for all three datasets in Table 4.16.

	20Newsgroups	**Reuters**	**BBC**
POS filtering	No filtering	No filtering	Noun-only
C_V	0.625	0.532	0.638
C_{NPMI}	0.052	0.016	-0.023

Table 4.16: Best top2vec coherence on each dataset by C_V and C_{NPMI}.

The best results for all algorithms are consolidated and presented in table 4.17. The top2vec outperforms LDA by a large margin on C_V but the difference is smaller measured by C_{NPMI}. Community Topic using SIWOw+ outperforms top2vec on the 20Newsgroups but top2vec achieves similar or better scores on the Reuters and BBC datasets. Community Topic with Leiden and WalkTrap handily outperform top2vec and LDA. These results demonstrate that Community Topic can find topics that are more coherent than old-fashioned LDA and a modern algorithm based around neural network word embeddings.

As previously mentioned, we do not want to myopically follow a single metric when evaluating a topic model. That a topic model produces coherent topics is important, but we also must consider the run time, stability, and features of a model that make it useful for downstream applications.

[7]https://github.com/ddangelov/Top2Vec

Algorithm	Coherence	20Newsgroups	Reuters	BBC
Community Topic (WalkTrap)	C_V	**0.759**	0.621	**0.683**
	C_{NPMI}	**0.235**	**0.274**	**0.031**
Community Topic (Leiden)	C_V	0.665	**0.642**	0.676
	C_{NPMI}	0.106	0.113	0.028
Community Topic (SIWOw+)	C_V	0.686	0.536	0.573
	C_{NPMI}	0.148	0.007	-0.038
top2vec	C_V	0.625	0.532	0.638
	C_{NPMI}	0.052	0.016	-0.023
LDA	C_V	0.510	0.471	0.366
	C_{NPMI}	0.027	0.025	-0.191

Table 4.17: Best coherence scores achieved by all algorithms on all datasets.

4.4.2 Run Time and Stability

The run time of Community Topic is dominated by the run time of the community detection algorithm used, but also includes the time it takes to parse the corpus and build the network and the time to sort the communities. The creation of the co-occurrence networks is deterministic but the community detection algorithms have varying levels of stochasticity.

To compare the run times and stability of the algorithms over repeated runs, we ran 10 runs of each algorithm on the 20Newsgroups corpus with no POS filtering and noun-only filtering. The co-occurrence networks were created using the sentence co-occurrence window and count edge weights. The edge weights were not thresholded on the corpus with no POS filtering and were thresholded at > 2 on the noun-only corpus. This will demonstrate the sensitivity of the community detection algorithm run times to the size of the networks. Results of this experiment are presented in Table 4.18.

We can see that the run times of LDA and top2vec are virtually unaffected by the POS filtering that reduces the number of tokens in each document. The network creation and topic sorting steps of Community Topic are also the same for the larger corpus and network. However, the run times of the community detection algorithms are greatly affected by the size of the network. Leiden is that fastest, taking only 50 ms on the smaller network. WalkTrap takes under 2 seconds and SIWOw+ about 3 seconds. On the larger network, the

run times of the algorithms all increase by about one order of magnitude. This only takes Leiden up to about half a second. WalkTrap takes over 20 seconds and SIWOw+ close to a minute. LDA takes about 7 seconds on both corpora and top2vec takes about 65 seconds.

So on the smaller network, the total run time of Community Topic is comparable to LDA with WalkTrap and SIWOw+ and about twice as fast using Leiden; both LDA and Community Topic are much faster than top2vec. On the larger network, Community Topic is still fastest with Leiden, but slower than LDA with WalkTrap and about as slow as top2vec with SIWOw+.

		Noun-only, threshold > 2		**No POS filter, no threshold**	
		Time	C_V	**Time**	C_V
Community Topic	Network creation	3.12 ± 0.02		3.12 ± 0.01	
	Sorting	0.07 ± 0.00		0.08 ± 0.01	
	WalkTrap	1.88 ± 0.08	0.535 ± 0.000	21.34 ± 1.21	0.690 ± 0.000
	Leiden	0.05 ± 0.00	0.539 ± 0.039	0.55 ± 0.11	0.565 ± 0.022
	SIWOw+	3.11 ± 0.01	0.510 ± 0.000	56.47 ± 0.12	0.657 ± 0.000
top2vec		65.52 ± 3.54	0.516 ± 0.115	65.60 ± 3.45	0.535 ± 0.088
LDA		6.93 ± 0.13	0.483 ± 0.021	6.96 ± 0.09	0.492 ± 0.025

Table 4.18: Run times and stability of all algorithms on smaller filtered 20Newsgroups corpus with thresholding and larger non-filtered 20 Newsgroups corpus without thresholding. All times in seconds.

The C_V results in Table 4.18 are not meant to compare performance as these networks were not created with the best parameters for each algorithm. Rather, the standard deviation of the score indicates the stability of the topic models. LDA is more stable than top2vec, which has a high standard error, particularly on the noun-only corpus. Community Topic with WalkTrap and SIWOw+ is effectively deterministic. When using Leiden, however, the stability is comparable to LDA but better than top2vec.

4.4.3 Document Clustering

The primary goal for developing the Community Topic algorithm is to provide a topic model that discovers interpretable, coherent topics with a natural structure that facilitates navigation and exploration of the topics either by a researcher or an agent having a conversation. Providing features for document classification is not the purpose of Community Topic as deep learning

approaches such as recurrent neural networks and transformers have surpassed topic modelling-based approaches. However, our model must be able to determine the topics of documents as a researcher would want to find documents relevant to an interesting topic and a conversational agent must be able to determine the topic of the utterances of the interlocutor.

To cluster documents by topic, we first create a mapping from terms to topics which can be done in a single pass through the topics. The topic proportions of a document can be computed in a single pass over the document, counting the number of terms of each topic to get the topic proportions. We then assign the document to a topic cluster based on the topic with the largest proportion in the document. We perform this clustering on the BBC corpus, which has five document categories, with noun-only POS filtering. We use the topics discovered by Community Topic with the network constructed with a sentence co-occurrence window, NPMI edge weights and no thresholding as this produces five topics. We compare to LDA trained on the same corpus for five topics. LDA provides topic proportions for documents as well, and we take the top topic for each document as the cluster, just as with Community Topic. We compare the quality of the clusterings using Normalized Mutual Information. We can see in Table 4.19 that the similarity of the Community Topic clusterings is much greater than that of the LDA clusterings. The topics produced by LDA have significant overlap of top terms, with general terms such as "year" and "government" appearing in most topics. Community Topic has no overlap between topics, making the distinctions between the topics of a document clearer.

	Community Topic	**LDA**
NMI	0.790	0.098

Table 4.19: Document clustering NMI performance of Community Topic and LDA on BBC dataset.

We have shown that Community Topic is able to find topics that achieve far better coherence scores than LDA. Community Topic is able to find these

topics in a shorter time and with similar stability when using Leiden. Community Topic is effectively deterministic with WalkTrap and SIWOw+, although these algorithms take longer. While top2vec finds more coherent topics than LDA, it does not match the coherence scores of Community Topic, takes significantly longer and is much less stable over repeated runs. Community Topic also outperforms LDA on clustering the documents of the BBC corpus. Community Topic also provides a natural topic hierarchy and allows the user to move up and down the hierarchy by finding sub- and super-topics on the fly.

4.4.4 Hierarchical Topics

We compare CT to two probabilistic graphical topic models, HLDA and PAM[8]. As the implementation of PAM only allows for two non-root topic layers in the hierarchy we generate a three-level hierarchy for each algorithm for fair comparison, where level 0 is the root topic of all terms in the corpus, level 1 are the super-topics, and level 2 are the sub-topics. PAM requires the number of super- and sub-topics to be specified. We used the number of topics discovered by CT at each level for PAM.

HLDA produces topics at both levels that are probability distributions over vocabulary terms and are thus compatible with our evaluation metrics without modification. CT produces a list of terms ranked by the internal weighted degree. To calculate specialization and affinity, we convert these to probability distributions by dividing each value by the sum of the values. The super-topics discovered by PAM are distributions over sub-topics. We convert these to distributions over terms by taking the expectation for each term in the sub-topics given the super-topic distribution over sub-topics. Each PAM super-topic distribution gives some non-zero probability to all sub-topics so we need a way to distinguish children from non-children. We do this by taking the top 6 most likely sub-topics as the children of a super-topic since we are positing a topic hierarchy with an average of 6 sub-topics per super-topic.

Using a Leiden resolution parameter of 1.0, CT finds 5 or 6 super-topics on all datasets and 5, 6, or 7 sub-topics per super topic and we use these average

[8]https://bab2min.github.io/tomotopy/v0.12.2/en/

values to guide the PAM model. HLDA finds hundreds of super-topics and about 3 times as many sub-topics. This tendency to find many small topics at all levels leads to poor performance on our evaluation metrics and leads to a poor hierarchy where it is common for a child topic to appear in more documents than its parent. PAM performs better, but benefits from using the number of topics discovered by CT.

CT is the fastest of the algorithms, finding the topic hierarchy in under 5 seconds on all datasets. HLDA takes between 30 seconds and 5 minutes while PAM ranges from 10 seconds to 2 minutes. All experiments were run on the same laptop with 2.7 GHz dual core processor and 8 GB RAM.

The coherence results are presented in Table 4.20. We can see that CT achieves the highest coherence scores on all datasets as measured by both metrics except for C_{NPMI} on the 20Newsgroup corpus where PAM comes out on top. PAM acheives the second highest scores in all other cases. HLDA is a distant third with much lower scores. This demonstrates that the topics found by CT will be more interpretable to a human user.

	BBC		20Newsgroups		Reuters	
	C_V	C_{NPMI}	C_V	C_{NPMI}	C_V	C_{NPMI}
CT	**0.641**	**0.079**	**0.645**	0.044	**0.702**	**0.182**
HLDA	0.448	-0.162	0.444	-0.133	0.451	-0.093
PAM	0.600	0.063	0.636	**0.090**	0.555	0.056

Table 4.20: Coherence scores for CT, HLDA, PAM on three document corpora. Bold indicates best score for each metric and dataset.

Figure 4.16 shows the specialization scores for each algorithm on the three datsets. We see that both the super-topics (level 1) and the sub-topics (level 2) found by HLDA have a very high specialization. This is consistent with the large number of topics found at both levels but does not match our intuition that topics higher in the hierarchy should be general. PAM produces general topics at level 1 and more specialized topics at level 2, however the super-topics are so general and similar to the overall frequency distribution as to not provide useful information for the user. CT also produces sub-topics that are more specialized than the super-topics. Unlike PAM, the super-topics are

themselves specialized and thus useful and informative themselves.

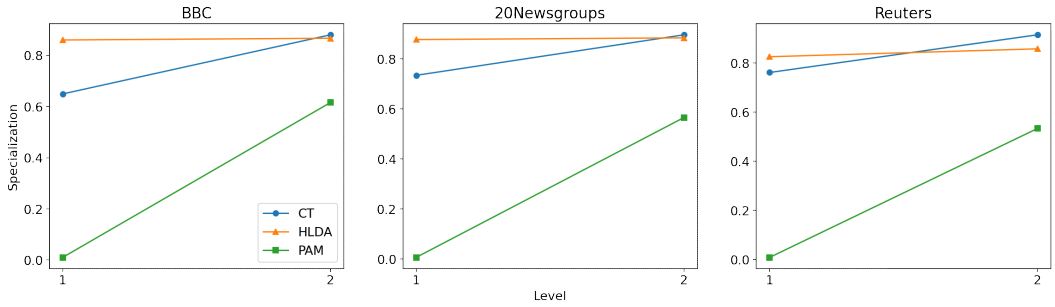


Figure 4.16: Topic specialization scores for CT, HLDA, and PAM on three corpora.

Figure 4.17 shows the hierarchical affinity scores for each algorithm on the three datasets. We see that HLDA has a higher affinity between parent topics and their children than non-children. However, the affinity is very low so the relationship between a super-topic and its sub-topics is very weak. PAM has the opposite problem with high affinities between parent topics and both child and non-child topics. This is because PAM super-topics are distributions over all sub-topics and is consistent with the super-topics being non-specialized. CT parent topics exhibit a high affinity with their children and zero affinity with non-children. This is because the sub-topics are a partition of the super-topic and thus do not overlap with any other super-topic.

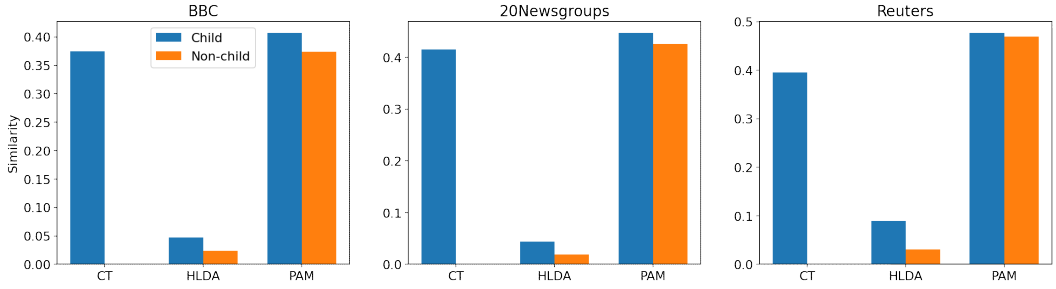


Figure 4.17: Hierarchical affinity scores between parent and children and between parent and non-children for CT, HLDA, and PAM on three corpora.

Our experimental results show that CT produces the most coherent and thus interpretable topics and the best topic hierarchy. CT topic hierarchies exhibit higher specialization for sub-topics than super-topics but with enough specialization at both levels to make the topics useful. CT super-topics have

a high affinity with their own sub-topics and no affinity with non-child sub-topics. CT is able to produce this coherent topic structure in less time than the other algorithms on commodity hardware.

4.5 Discussion

In this chapter, we presented the second major contribution of our thesis: Community Topic, our community detection-based community modelling algorithm. We have conducted a thorough analysis of the term co-occurrence networks constructed from document corpora. We have empirically evaluated the performance of Community Topic with various hyperparameters and community detection algorithms. We have compared Community Topic to LDA as a standard benchmark as well as the more recently developed top2vec algorithm.

We conclude that Community Topic works best with the Leiden algorithm. As Leiden performs well on all datasets with the same set of hyperparameters, we can use a sentence co-occurrence window, NPMI edge weights, no thresholding, and noun-only POS filtering as a standard and eliminate the need for hyperparameter tunig. Using Leiden also results in Community Topic being faster than both LDA and top2vec, and much faster than Variational Autoencoder-based topic models that require hours of training on special hardware [42]. Community Topic finds more coherent topics than LDA and top2vec. However, the greatest advantage may be the natural hierarchy afforded by the network representation with both sub- and super-topics able to be found using the same community detection. Community Topic achieves all of the goals set out at the beginning of the chapter:

- **Discovery of the number of topics** - Community Topic discovers the number of topics, and topics at different scales can be found by using the Leiden resolution parameter.
- **Natural hierarchy** - Community Topic can find sub- and super-topics from the network structure and original topics.

- **Natural relationship between topics** - The edge weights between the topics in the community network gives the degree of relatedness.
- **Time and resource efficient** - Community Topic is faster than LDA and top2vec.
- **Stability** - Community Topic is no less stable than LDA and much more stable than top2vec.
- **Minimal hyperparameters** - Using Leiden, Community Topic has no hyperparameters to tune. The user can use the Leiden resolution parameter to control the scale of discovered topics.
- **No redundancy** - Community Topic partitions the vocabulary into topics with no redundancy.

Chapter 5

Conclusion

In this thesis, we review the fields of topic modelling and social network analysis with the goal of combining them to develop a novel topic modelling algorithm, Community Topic, that overcomes the deficiencies of the most popular approach in use today, LDA. The first step to that goal is the extension of SIWO to the weighted case as the edge weights contain vital information in the term co-occurrence networks. SIWO has proven itself to be a top performer on unweighted networks so we have good reason to believe it would be a useful component of Community Topic.

We successfully extend SIWO into SIWOw to handle edge weights. We empirically demonstrate that this incorporating edge weights improves performance and that SIWOw is competitive with state-of-the-art algorithms for community detection.

We define and analyze term co-occurrence networks. We show that the properties of these networks are quite different from the synthetic and real networks previously used to evaluate SIWOw. This difference in structure may explain why some of the best performing community detection algorithms in Chapter 2, such as Infomap, completely fail on these networks and why SIWOw+, which outperforms Leiden and WalkTrap on the LFR networks, finds less coherent topics.

We develop and evaluate Community Topic, our novel approach to topic modelling. We first evaluate the various possible configurations and hyperparameter settings to determine that the Leiden algorithms works best. This

results in Community Topic being hyperparameter free as the same settings perform well on all tested datasets.

We show that Community Topic outperforms both the standard benchmark LDA as well as the recently developed top2vec, which relies on word embeddings learned by a neural network. Community Topic is faster and produces more coherent topics. It also provides a topic structure that can be utilized in downstream tasks such as corpus exploration and conversational agents. Sub- and super-topics can be found and there are relationships between topics which can all be used to guide a researcher exploring a corpus or an agent having a conversation.

5.1 Future Work

The fact that only one community detection algorithm, Leiden, is able to fully take advantage of the term co-occurrence networks to find coherent topics with a robust hierarchy indicates that there is still work to be done improving SIWO and other community detection algorithms, or at the very least adapting them to work better on the structure of the co-occurrence networks. Modularity based methods have weaknesses such as the resolution limit and field of view limit, but they are able to find communities for very sparse or very dense networks as they compare the actual network structure to a hypothetical random network of the same average density. Our attempt to adapt SIWO for denser networks, Min-Max SIWO, failed but there is no reason to think that it or other community detection algorithms could not be further improved to work better with Community Topic.

That most community detection algorithms struggle on the term co-occurrence networks could also be an indication that the networks themselves could be improved. We already took the step of converting raw counts to NPMI edge weights and while varying the co-occurrence window and thresholding do not have a material positive impact on topic quality, other improvements may be possible.

Part of the difficulty that community detection algorithms seem to have is

with polysemy as a single term with multiple meanings may connect strongly to terms in different topics. We attempt to combat this using n-grams, but other methods to disambiguate such homonyms could further improve Community Topic.

While we evaluated the quality of the topic hierarchy found by Community Topic on several measures, a more comprehensive evaluation of the quality of the hierarchy is possible. One possibility is to look at the specificity of the terms used for labelling a topic using an external reference such as WordNet. The label terms higher in the topic hierarchy should be more generic and those labelling deeper sub-topics should be more specific.

Of course, the problem of labelling the topics is another area of research. In our work, we have used subjective human judgement or the top few terms to label a topic. Investigating better methods for labelling would enable new forms of evaluation for the topic hierarchy which we claim is one of the best features of our algorithm.

Many of these future projects rely on some way to compare topic quality such as automated coherence metrics. However, these metrics are only a proxy for human judgements of topic quality. As we have shown, there are also features beyond topic coherence that make for a good topic model and should be considered and compared. The real test of a topic modelling algorithm is in its implementation and use as a tool. Integrating Community Topic and other algorithms into a conversational agent allows for a comparison of topic quality, run time, topic hierarchy, and the relationships between topics. This is much more difficult and involved than comparing coherence scores, but the proof of the pudding is in the tasting and we should not stop at automated metrics for evaluating our topic models.

References

[1] Deepak Bhaskar Acharya and Huaming Zhang. "Weighted Graph Nodes Clustering via Gumbel Softmax." In: *arXiv preprint arXiv:2102.10775* (2021).

[2] Nikolaos Aletras and Mark Stevenson. "Evaluating topic coherence using distributional semantics." In: *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*. 2013, pp. 13–22. URL: https://aclanthology.org/W13-0102/.

[3] Dimo Angelov. "Top2vec: Distributed representations of topics." In: *arXiv preprint arXiv:2008.09470* (2020). URL: https://arxiv.org/pdf/2008.09470.pdf.

[4] Alex Arenas, Alberto Fernandez, and Sergio Gomez. "Analysis of the structure of complex networks at different resolution levels." In: *New Journal of Physics* 10.5 (2008), p. 053039. DOI: 10.1088/1367-2630/10/5/053039.

[5] Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. "The architecture of complex weighted networks." In: *Proceedings of the National Academy of Sciences* 101.11 (2004), pp. 3747–3752. DOI: 10.1073/pnas.0400087101.

[6] Alain Barrat and Martin Weigt. "On the properties of small-world network models." In: *The European Physical Journal B-Condensed Matter and Complex Systems* 13.3 (2000), pp. 547–560. DOI: 10.1007/s100510050067.

[7] David Blei and John Lafferty. "Correlated topic models." In: *Advances in Neural Information Processing Systems* 18 (2006), p. 147. URL: http://www.cs.columbia.edu/~blei/papers/BleiLafferty2006.pdf.

[8] David Blei and John Lafferty. "Dynamic topic models." In: *Proceeding of the 23rd International Conference on Machine Learning*. 2006, pp. 113–120. DOI: 10.1145/1143844.1143859.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation." In: *Journal of Machine Learning Research* 3.Jan (2003), pp. 993–1022. DOI: 10.1016/B978-0-12-411519-4.00006-9.

[10] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. "Fast unfolding of communities in large networks." In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (2008), P10008. DOI: 10.1088/1742-5468/2008/10/p10008.

[11] Gerlof Bouma. "Normalized (pointwise) mutual information in collocation extraction." In: *Proceedings of GSCL* 30 (2009), pp. 31–40. URL: https://svn.spraakdata.gu.se/repos/gerlof/pub/www/Docs/npmi-pfd.pdf.

[12] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." In: *Computer Networks and ISDN systems* 30.1-7 (1998), pp. 107–117. DOI: 10.1016/s0169-7552(98)00110-x.

[13] Sophie Burkhardt and Stefan Kramer. "Decoupling Sparsity and Smoothness in the Dirichlet Variational Autoencoder Topic Model." In: *Journal of Machine Learning Research* 20.131 (2019), pp. 1–27. URL: https://jmlr.csail.mit.edu/papers/volume20/18-569/18-569.pdf.

[14] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. "Density-based clustering based on hierarchical density estimates." In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2013, pp. 160–172. DOI: 10.1007/978-3-642-37456-2_14.

[15] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-graber, and David Blei. "Reading Tea Leaves: How Humans Interpret Topic Models." In: *Advances in Neural Information Processing Systems*. Vol. 22. Curran Associates, Inc., 2009.

[16] Jiyang Chen, Osmar R Zaıane, and Randy Goebel. "An unsupervised approach to cluster web search results based on word sense communities." In: *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 1. IEEE. 2008, pp. 725–729. DOI: 10.1109/WIIAT.2008.24.

[17] Aaron Clauset. "Finding local community structure in networks." In: *Physical Review E* 72.2 (2005), p. 026132. DOI: 10.1103/physreve.72.026132.

[18] Jonathan Cohen. "Trusses: Cohesive subgraphs for social network analysis." In: *National Security Agency Technical Report* 16.3.1 (2008).

[19] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. "A classification for community discovery methods in complex networks." In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 4.5 (2011), pp. 512–546. DOI: 10.1002/sam.10133.

[20] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. "Comparing community structure identification." In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.09 (2005), P09008. DOI: 10.1088/1742-5468/2005/09/P09008.

[21] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. "Indexing by latent semantic analysis." In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407. DOI: 10.1002/(sici)1097-4571(199009)41:6<391::aid-asi1>3.0.co;2-9.

[22] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

[23] Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. "Topic Modeling in Embedding Spaces." In: *Transactions of the Association for Computational Linguistics* 8 (July 2020), pp. 439–453. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00325.

[24] Chris Ding, Tao Li, and Wei Peng. "On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing." In: *Computational Statistics & Data Analysis* 52.8 (2008), pp. 3913–3927. DOI: 10.1016/j.csda.2008.01.011.

[25] Caitlin Doogan and Wray Buntine. "Topic model or topic twaddle? Re-evaluating semantic interpretability measures." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 3824–3848. DOI: 10.18653/v1/2021.naacl-main.300.

[26] Nouha Dziri, Ehsan Kamalloo, Kory Mathewson, and Osmar R Zaïane. "Augmenting Neural Response Generation with Context-Aware Topical Attention." In: *Proceedings of the First Workshop on NLP for Conversational AI*. 2019, pp. 18–31. DOI: 10.18653/v1/W19-4103.

[27] Katrin Erk. "Vector space models of word meaning and phrase meaning: A survey." In: *Language and Linguistics Compass* 6.10 (2012), pp. 635–653. DOI: 10.1002/lnco.362.

[28] Fabrizio Esposito, Anna Corazza, and Francesco Cutugno. "Topic Modelling with Word Embeddings." In: *CLiC-it/EVALITA*. 2016. DOI: 10.4000/books.aaccademia.1767.

[29] Leonhard Euler. "Solutio problematis ad geometriam situs pertinentis." In: *Commentarii Academiae Scientiarum Imperialis Petropolitanae* 8 (1736).

[30] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. "Effective and efficient community search over large heterogeneous information networks." In: *Proceedings of the VLDB Endowment* 13.6 (2020), pp. 854–867. DOI: 10.14778/3380750.3380756.

[31] Santo Fortunato. "Community detection in graphs." In: *Physics Reports* 486.3-5 (2010), pp. 75–174. DOI: 10.1016/j.physrep.2009.11.002.

[32] Santo Fortunato and Marc Barthélemy. "Resolution limit in community detection." In: *Proceedings of the National Academy of Sciences* 104.1 (2007), pp. 36–41. DOI: 10.1073/pnas.0605965104.

[33] Santo Fortunato and Darko Hric. "Community detection in networks: A user guide." In: *Physics Reports* 659 (2016), pp. 1–44. DOI: 10.1016/j.physrep.2016.09.002.

[34] Eric Gaussier and Cyril Goutte. "Relation between PLSA and NMF and implications." In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2005, pp. 601–602. DOI: 10.1145/1076034.1076148.

[35] Shiva Zamani Gharaghooshi, Osmar R Zaıane, Christine Largeron, Mohammadmahdi Zafarmand, and Chang Liu. "Addressing the resolution limit and the field of view limit in community mining." In: *International Symposium on Intelligent Data Analysis*. Springer. 2020, pp. 210–222. DOI: 10.1007/978-3-030-44584-3_17.

[36] Alan Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[37] Michelle Girvan and Mark EJ Newman. "Community structure in social and biological networks." In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 7821–7826.

[38] Thomas Griffiths, Michael Jordan, Joshua Tenenbaum, and David Blei. "Hierarchical topic models and the nested Chinese restaurant process." In: *Advances in Neural Information Processing Systems* 16 (2003). URL: https://people.eecs.berkeley.edu/~jordan/papers/lda-crp.pdf.

[39] Zellig S Harris. "Distributional structure." In: *Word* 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520.

[40] Thomas Hofmann. "Probabilistic latent semantic indexing." In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1999, pp. 50–57. DOI: 10.1145/312624.312649.

[41] Petter Holme, Sung Min Park, Beom Jun Kim, and Christofer R Edling. "Korean university life in a network perspective: Dynamics of a large affiliation network." In: *Physica A: Statistical Mechanics and its Applications* 373 (2007), pp. 821–830. DOI: 10.1016/j.physa.2006.04.066.

[42] Alexander Hoyle, Pranav Goel, Andrew Hian-Cheong, Denis Peskov, Jordan Boyd-Graber, and Philip Resnik. "Is Automated Topic Model Evaluation Broken? The Incoherence of Coherence." In: *Advances in Neural Information Processing Systems* 34 (2021). URL: http://umiacs.umd.edu/~jbg//docs/2021_neurips_incoherence.pdf.

[43] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. "Querying minimal steiner maximum-connected subgraphs in large graphs." In: *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. 2016, pp. 1241–1250. DOI: 10.1145/2983323.2983748.

[44] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. "Querying k-truss community in large and dynamic graphs." In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. 2014, pp. 1311–1322. DOI: 10.1145/2588555.2610495.

[45] Karoliina Isoaho, Daria Gritsenko, and Eetu Mäkelä. "Topic modeling and text analysis for qualitative policy research." In: *Policy Studies Journal* 49.1 (2021), pp. 300–324. DOI: 10.1111/psj.12343.

[46] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. "An introduction to variational methods for graphical models." In: *Machine learning* 37.2 (1999), pp. 183–233. DOI: 10.1023/A:1007665907178.

[47] Michael Irwin Jordan. *Learning in Graphical Models*. MIT press, 1999.

[48] Joon Hee Kim, Dongwoo Kim, Suin Kim, and Alice Oh. "Modeling topic hierarchies with the recursive chinese restaurant process." In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. 2012, pp. 783–792.

[49] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014. URL: https://arxiv.org/pdf/1312.6114.pdf.

[50] Diederik P Kingma, Max Welling, et al. "An Introduction to Variational Autoencoders." In: *Foundations and Trends in Machine Learning* 12.4 (2019), pp. 307–392. DOI: 10.1561/9781680836233.

[51] Donald Ervin Knuth. *The Stanford GraphBase: a platform for combinatorial computing*. Vol. 1. AcM Press New York, 1993.

[52] Katsiaryna Krasnashchok and Salim Jouili. "Improving topic quality by promoting named entities in topic modeling." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2018, pp. 247–253. DOI: 10.18653/v1/P18-2040.

[53] Andrea Lancichinetti and Santo Fortunato. "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities." In: *Physical Review E* 80.1 (2009), p. 016118.

[54] Andrea Lancichinetti and Santo Fortunato. "Limits of modularity maximization in community detection." In: *Physical Review E* 84.6 (2011), p. 066122. DOI: 10.1103/physreve.84.066122.

[55] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. "Benchmark graphs for testing community detection algorithms." In: *Physical Review E* 78.4 (2008), p. 046110.

[56] Andrea Lancichinetti, Mikko Kivelä, Jari Saramäki, and Santo Fortunato. "Characterizing the community structure of complex networks." In: *PloS one* 5.8 (2010), e11976. DOI: 10.1371/journal.pone.0011976.

[57] Jey Han Lau and Timothy Baldwin. "The sensitivity of topic coherence evaluation to topic cardinality." In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 483–487. DOI: 10.18653/v1/N16-1057.

[58] Jey Han Lau, David Newman, and Timothy Baldwin. "Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality." In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 2014, pp. 530–539.

[59] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents." In: *International Conference on Machine Learning*. PMLR. 2014, pp. 1188–1196. URL: https://arxiv.org/pdf/1405.4053.pdf.

[60] Daniel D Lee and H Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization." In: *Nature* 401.6755 (1999), pp. 788–791. DOI: 10.1038/44565.

[61] Wei Li and Andrew McCallum. "Pachinko Allocation: DAG-Structured Mixture Models of Topic Correlations." In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 577–584. ISBN: 1595933832. DOI: 10.1145/1143844.1143917.

[62] Xiao-Li Li, Chuan-Sheng Foo, Soon-Heng Tan, and See-Kiong Ng. "Interaction graph mining for protein complexes using local clique merging." In: *Genome Informatics* 16.2 (2005), pp. 260–269.

[63] Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. "An overview of topic modeling and its current applications in bioinformatics." In: *SpringerPlus* 5.1 (2016), pp. 1–22. DOI: 10.1186/s40064-016-3252-8.

[64] Feng Luo, James Z. Wang, and Eric Promislow. "Exploring Local Community Structures in Large Networks." In: *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*. 2006, pp. 233–239. DOI: 10.1109/WI.2006.72.

[65] David Lusseau. "The emergent properties of a dolphin social network." In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 270.suppl_2 (Nov. 2003). DOI: 10.1098/rsbl.2003.0057.

[66] Mika V Mantyla, Maelick Claes, and Umar Farooq. "Measuring LDA topic stability from clusters of replicated runs." In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2018, pp. 1–4. DOI: 10.1145/3239235.3267435.

[67] Fiona Martin and Mark Johnson. "More efficient topic modelling through a noun only approach." In: *Proceedings of the Australasian Language Technology Association Workshop 2015*. 2015, pp. 111–115. URL: https://aclanthology.org/U15-1013.pdf.

[68] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction." In: *arXiv preprint arXiv:1802.03426* (2018). URL: https://arxiv.org/pdf/1802.03426.pdf.

[69] Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. "Improving lda topic models for microblogs via tweet pooling and automatic labeling." In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2013, pp. 889–892. DOI: 10.1145/2484028.2484166.

[70] Yishu Miao, Lei Yu, and Phil Blunsom. "Neural variational inference for text processing." In: *International Conference on Machine Learning*. PMLR. 2016, pp. 1727–1736. URL: https://arxiv.org/pdf/1511.06038.pdf.

[71] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781* (2013).

[72] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in Neural Information Processing Systems* 26 (2013). URL: https://arxiv.org/pdf/1310.4546.pdf.

[73] George A Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748.

[74] George A Miller and Walter G Charles. "Contextual correlates of semantic similarity." In: *Language and Cognitive Processes* 6.1 (1991), pp. 1–28. DOI: 10.1080/01690969108406936.

[75] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. "Network motifs: simple building blocks of complex networks." In: *Science* 298.5594 (2002), pp. 824–827. DOI: 10.1515/9781400841356.217.

[76] David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. "Optimizing semantic coherence in topic models." In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 2011, pp. 262–272.

[77] Eric Nalisnick and Padhraic Smyth. "Stick-breaking variational autoencoders." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017. URL: https://openreview.net/pdf?id=S1jmAotxg.

[78] Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. "Topic modeling with wasserstein autoencoders." In: *arXiv preprint arXiv:1907.12374* (2019). DOI: 10.18653/v1/P19-1640.

[79] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. "Automatic evaluation of topic coherence." In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2010, pp. 100–108.

[80] Mark Newman. "The structure and function of complex networks." In: *SIAM review* 45.2 (2003), pp. 167–256. DOI: 10.1137/s003614450342480.

[81] Mark Newman. "Finding community structure in networks using the eigenvectors of matrices." In: *Physical Review E* 74.3 (2006), p. 036104. DOI: 10.1103/physreve.74.036104.

[82] Mark Newman. *Networks*. Oxford University Press, 2018.

[83] Mark Newman and Michelle Girvan. "Finding and evaluating community structure in networks." In: *Physical Review E* 69.2 (2004), p. 026113. DOI: 10.1103/physreve.69.026113.

[84] Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. "Intensity and coherence of motifs in weighted complex networks." In: *Physical Review E* 71.6 (2005), p. 065103. DOI: 10.1103/PhysRevE.71.065103.

[85] Tore Opsahl, Filip Agneessens, and John Skvoretz. "Node centrality in weighted networks: Generalizing degree and shortest paths." In: *Social networks* 32.3 (2010), pp. 245–251. DOI: 10.1016/j.socnet.2010.03.006.

[86] Tore Opsahl and Pietro Panzarasa. "Clustering in weighted networks." In: *Social Networks* 31.2 (2009), pp. 155–163. DOI: 10.1016/j.socnet.2009.02.002.

[87] Günce K Orman and Vincent Labatut. "The effect of network realism on community detection algorithms." In: *2010 International Conference on Advances in Social Networks Analysis and Mining*. IEEE. 2010, pp. 301–305. DOI: 10.1109/ASONAM.2010.70.

[88] Naoto Ozaki, Hiroshi Tezuka, and Mary Inaba. "A simple acceleration method for the Louvain algorithm." In: *International Journal of Computer and Electrical Engineering* 8.3 (2016), p. 207. DOI: 10.17706/ijcee.2016.8.3.207-218.

[89] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank citation ranking: Bringing order to the web*. Tech. rep. Stanford InfoLab, 1999.

[90] Miha Pavlinek and Vili Podgorelec. "Text classification method based on self-training and LDA topic models." In: *Expert Systems with Applications* 80 (2017), pp. 83–93. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.03.020.

[91] Pascal Pons and Matthieu Latapy. "Computing communities in large networks using random walks." In: *International Symposium on Computer and Information Sciences*. Springer. 2005, pp. 284–293. DOI: 10.1007/11569596_31.

[92] Martin F Porter. "An algorithm for suffix stripping." In: *Program* (1980). DOI: 10.1108/eb046814.

[93] Reihaneh Rabbany, Mansoreh Takaffoli, Justin Fagnan, Osmar R Zaïane, and Ricardo JGB Campello. "Relative validity criteria for community mining algorithms." In: *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE. 2012, pp. 258–265.

[94] Reihaneh Rabbany, Mansoureh Takaffoli, Justin Fagnan, Osmar R Zaïane, and Ricardo JGB Campello. "Communities validity: methodical evaluation of community mining algorithms." In: *Social Network Analysis and Mining* 3.4 (2013), pp. 1039–1062.

[95] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. "Near linear time algorithm to detect community structures in large-scale networks." In: *Physical Review E* 76.3 (2007), p. 036106. DOI: 10.1103/physreve.76.036106.

[96] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi, and Elvia M Quiroz. "Internal versus external cluster validation indexes." In: *International Journal of computers and communications* 5.1 (2011), pp. 27–34.

[97] Michael Röder, Andreas Both, and Alexander Hinneburg. "Exploring the space of topic coherence measures." In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 2015, pp. 399–408. DOI: 10.1145/2684822.2685324.

[98] Martin Rosvall and Carl T. Bergstrom. "Maps of random walks on complex networks reveal community structure." In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123. DOI: 10.1073/pnas.0706851105.

[99] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: 10.1016/0377-0427(87)90125-7.

[100] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid Aref, Marcelo Arenas, Maciej Besta, Peter A Boncz, et al. "The future is big graphs: a community view on graph processing systems." In: *Communications of the ACM* 64.9 (2021), pp. 62–71. DOI: 10.1145/3434642.

[101] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.

[102] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo K. Kaski, and János Kertész. "Generalizations of the clustering coefficient to weighted complex networks." In: *Physical Review E* 75 2 Pt 2 (2007), p. 027105. DOI: 10.1103/PhysRevE.75.027105.

[103] Michael T Schaub, Jean-Charles Delvenne, Sophia N Yaliraki, and Mauricio Barahona. "Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities and the field-of-view limit." In: *PloS one* 7.2 (2012), e32210. DOI: doi.org/10.1371/journal.pone.0032210.

[104] Alexandra Schofield and David Mimno. "Comparing apples to apple: The effects of stemmers on topic models." In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 287–300. DOI: 10.1162/tacl_a_00099.

[105] Stephen B Seidman. "Network structure and minimum degree." In: *Social networks* 5.3 (1983), pp. 269–287. DOI: 10.1016/0378-8733(83)90028-x.

[106] Akash Srivastava and Charles Sutton. "Autoencoding variational inference for topic models." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017. URL: https://arxiv.org/pdf/1703.01488.pdf.

[107] Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. "Probabilistic author-topic models for information discovery." In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2004, pp. 306–315. DOI: 10.1145/1014052.1014087.

[108] Xiaobing Sun, Xiangyue Liu, Bin Li, Yucong Duan, Hui Yang, and Jiajun Hu. "Exploring topic models in software engineering data analysis: A survey." In: *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 2016, pp. 357–362. DOI: 10.1109/SNPD.2016.7515925.

[109] Ivan Titov and Ryan McDonald. "A joint model of text and aspect ratings for sentiment summarization." In: *Proceedings of ACL-08: HLT*. 2008, pp. 308–316.

[110] Vincent A Traag. "Faster unfolding of communities: Speeding up the Louvain algorithm." In: *Physical Review E* 92.3 (2015), p. 032801. DOI: 10.1103/physreve.92.032801.

[111] Vincent A Traag, Paul Van Dooren, and Yurii Nesterov. "Narrow scope for resolution-limit-free community detection." In: *Physical Review E* 84.1 (2011), p. 016114. DOI: 10.1103/physreve.84.016114.

[112] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. "From Louvain to Leiden: guaranteeing well-connected communities." In: *Scientific Reports* 9.1 (2019), pp. 1–12. DOI: 10.1038/s41598-019-41695-z.

[113] Ludo Waltman and Nees Jan Van Eck. "A smart local moving algorithm for large-scale modularity-based community detection." In: *The European Physical Journal B* 86.11 (2013), pp. 1–14. DOI: 10.1140/epjb/e2013-40829-0.

[114] Xing Wei and W Bruce Croft. "LDA-based document models for ad-hoc retrieval." In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2006, pp. 178–185. DOI: 10.1145/1148170.1148204.

[115] Kai Yang, Yi Cai, Zhenhong Chen, Ho-fung Leung, and Raymond Lau. "Exploring topic discriminating power of words in latent dirichlet allocation." In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 2238–2247. URL: https://aclanthology.org/C16-1211.pdf.

[116] Zhao Yang, René Algesheimer, and Claudio J Tessone. "A comparative analysis of community detection algorithms on artificial networks." In: *Scientific reports* 6.1 (2016), pp. 1–18. DOI: 10.1038/srep30750.

[117] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. "Local higher-order graph clustering." In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 555–564. DOI: 10.1145/3097983.3098069.

[118] Wayne W. Zachary. "An Information Flow Model for Conflict and Fission in Small Groups." In: *Journal of Anthropological Research* 33.4 (Dec. 1977), pp. 452–473. DOI: 10.1086/jar.33.4.3629752.

[119] Bin Zhang and Steve Horvath. "A general framework for weighted gene co-expression network analysis." In: *Statistical Applications in Genetics and Molecular Biology* 4.1 (2005). DOI: 10.2202/1544-6115.1128.

[120] He Zhao, Lan Du, Wray Buntine, and Gang Liu. "MetaLDA: A Topic Model that Efficiently Incorporates Meta Information." In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, pp. 635–644. DOI: 10.1109/ICDM.2017.73.

[121] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. "Comparing twitter and traditional media using topic models." In: *European Conference on Information Retrieval*. Springer. 2011, pp. 338–349. DOI: 10.1007/978-3-642-20161-5_34.