Community Topic: Topic model inference by consecutive word community discovery

Eric Austin and **Osmar R. Zaïane**
University of Alberta
Alberta Machine Intelligence Institute
Edmonton, Alberta
eaustin@ualberta.ca
zaiane@ualberta.ca

Christine Largeron
Université Jean Monnet
Hubert Curien Laboratory
Saint-Etienne, France
largeron@univ-st-etienne.fr

Abstract

We present our novel, hyperparameter-free topic modelling algorithm, Community Topic. Our algorithm is based on mining communities from term co-occurrence networks. We empirically evaluate and compare Community Topic with Latent Dirichlet Allocation and the recently developed top2vec algorithm. We find that Community Topic runs faster than the competitors and produces topics that achieve higher coherence scores. Community Topic can discover coherent topics at various scales. The network representation used by Community Topic results in a natural relationship between topics and a topic hierarchy. This allows sub- and super-topics to be found on demand. These features make Community Topic the ideal tool for downstream applications such as applied research and conversational agents.

1 Introduction

Topic modelling discovers 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 and 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 all fields and domains seek to better understand the main ideas and themes of document collections too large for a human to read and summarize. This requires topics that are interpretable and coherent to the human users.

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. One key component 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 (Dziri et al., 2019). However, more can be done with topics to improve a conversational agent given the right topic model. It can be used to detect and control topic drift in the conversation so that the agent's responses make sense in context. If the user is engaged with the current topic, then the agent can stay on topic or detect sub-topics to focus the conversation. The agent can detect super-topics to broaden the range of conversation. 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.

The features that make a topic model useful for a conversational agent are the same that make it useful as a tool of applied research. The topics must be coherent and interpretable to be useful to a researcher and for an agent's response to fit into a conversation. A measure of relatedness between topics allows for a natural flow to exploration and conversation. 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.

The most widely used topic model, Latent Dirichlet Allocation (LDA), lacks many of these features and has other drawbacks. The number of topics must be specified, requiring multiple runs with different numbers of topics to find the best topics. It performs poorly on short documents. Different runs on the same corpus can produce different topics, especially if the order of the documents is different (Mantyla et al., 2018). Common terms can appear in many different topics, reducing the uniqueness of topics (Nan et al., 2019).

Neural networks 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 (Burkhardt and Kramer, 2019) 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 (Sakr et al., 2021), it seems natural to take a network-based approach to topic modelling. Our topic modelling algorithm, Community Topic (CT), mines communities from networks constructed from term co-occurrences. These topics are collections of vocabulary terms and are thus interpretable by humans. The network representation provides a natural topic structure and hierarchy. The topics themselves form a network with connections of varying strength between the topics and on which super-topics can be mined. Each topic is a sub-graph that can be mined to find sub-topics. Our algorithm can run quickly on simple hardware which makes it ideal for researchers from all fields for exploring a document collection.

In this paper, we review related work on topic modelling. We describe our algorithm, how it constructs term co-occurrence networks, and how it mines topics from these networks. We empirically evaluate our algorithm and compare it to LDA as a standard benchmark as well as a recently developed clustering approach based on word embeddings. Our results show that our approach is able to find more coherent topics in a shorter period of time with more stable results while also providing a natural topic structure and hierarchy.

2 Related Work

Topic modelling emerged from the field of information retrieval and methods for document indexing, query matching, and classification. 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 to better understand large document collections (Hoyle et al., 2021) in fields as varied as political science (Isoaho et al., 2021) and bioinformatics (Liu et al., 2016).

One early approach was Latent Semantic Analysis (LSA) (Deerwester et al., 1990). LSA decomposes the term-by-document matrix to find vectors representing the latent semantic structure of the corpus. These vectors relate terms and documents and can be viewed as topics, although they are uninterpretable. Another method based on matrix decomposition is Non-negative Matrix Factorization (NMF) (Lee and Seung, 1999). Unsatisfied with the lack of a solid statistical foundation to LSA, researchers developed Probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 1999) which has a generative probabilistic model of the data with the topics as the latent variables.

One major drawback of pLSA is that the topic mixture is estimated separately for each document. To remedy this, researchers developed Latent Dirichlet Allocation (LDA) (Blei et al., 2003). 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. The probability of a topic z given a document d, $p(z|d)$, is a multinomial distribution over the 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.

The number of topics must be specified. LDA maximizes the probability of the observed corpus assuming that it was generated by the hidden latent variables. This computation is intractable (Blei et al., 2003) but can be approximated by variational inference (Jordan et al., 1999) or Markov chain Monte Carlo (Jordan, 1999). The topics are probability distributions over terms which are interpretable to human users. The trained model can discover the topic mix of unseen documents.

While LDA has been extremely successful and is widely used, there have been many attempts to improve upon it. Researchers have tried promoting named entities to become the most frequent terms in the document (Krasnashchok and Jouili, 2018).

In (Yang et al., 2016), the authors use a two-step LDA process to identify and re-weight words that are topic-indiscriminate. To improve the performance of LDA on tweets, the authors of (Mehrotra et al., 2013) pool tweets into longer documents based on various schemes such as common author and same hashtag. The MetaLDA model (Zhao et al., 2017) incorporates document and word meta information such as document labels, WordNet synonyms (Miller, 1995), and word embeddings (Mikolov et al., 2013). The author-topic model (Steyvers et al., 2004) extends LDA by conditioning the topic mixture on document author. The Correlated Topic Model (CTM) (Blei and Lafferty, 2006a) models the correlations between topics. The Dynamic Topic Model (Blei and Lafferty, 2006b) allows for the modelling of topic evolution over time. The Hierarchical LDA model (HLDA) (Griffiths et al., 2003) allows for a hierarchy of topics using a tree structure. A flexible generalization of LDA is the Pachinko Allocation Model (PAM) (Li and McCallum, 2006). Like HLDA, PAM allows for a hierachy of topics but this hierarchy is represented by a directed acyclic graph rather than a tree of fixed depth, allowing for a variety of relationships between topics and terms in the hierarchy.

In recent years, new types of topic models have emerged based on neural networks and deep learning. Some of these methods remain close to the LDA framework while others are completely different approaches. The Embedded Topic Model (ETM) (Dieng et al., 2020) combines word embeddings trained using the continuous Skip-gram algorithm (Mikolov et al., 2013) with the LDA probabilistic generative model. Another approach is to use deep neural networks to learn the probability distributions of a generative probabilistic model. This can be done using a variational autoencoder (VAE) (Kingma and Welling, 2014; Kingma et al., 2019). There have been many VAE-based topic models developed, including the neural variational document model (NVDM) (Miao et al., 2016), the stick-breaking variational autoencoder (SB-VAE) (Nalisnick and Smyth, 2017), ProdLDA (Srivastava and Sutton, 2017), and Dirichlet-VAE (Burkhardt and Kramer, 2019). 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 (Hoyle et al., 2021).

Other approaches use the word embeddings learned by a neural network but do not use the probabilistic generative model framework. The top2vec algorithm (Angelov, 2020) clusters document vectors learned by the doc2vec algorithm (Le and Mikolov, 2014). 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 (McInnes et al., 2018). Then dense clusters are found using HDBSCAN (Campello et al., 2013). 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.

The approach closest to ours is Vec2GC (Rao and Chakraborty, 2021). Like top2vec, this algorithm uses doc2vec to learn document embeddings. Vec2GC creates a network of the documents where edges exist between documents that have a cosine similarity over a certain threshold. Community mining is then applied to the network to find communities of related documents. Our approach differs in that it finds interpretable communities of terms, i.e. topics, rather than groups of similar documents. Our approach does not rely on learning embeddings with a neural network and computing pairwise similarities but uses the co-occurrence information present in the documents themselves.

3 Preliminaries

3.1 Networks and Communities

A comprehensive review of network theory is beyond the scope of this work and we refer the reader to (Newman, 2018). We define sufficient terminology to be able to understand our algorithm.

A network is represented by a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. 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. 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 **internal degree** of a vertex v_i, denoted k_i^{int}, is the number of edges that connect v_i to another vertex of the same community. 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. $k_i^w = \sum_{v_j \in V} w_{i,j}$. The **internal weighted degree** of a vertex v_i, denoted $k_i^{w,int}$, is the sum of the weights of all edges that connect v_i to another vertex of the same community. The **embeddedness** of a vertex v_i is k_i^{int}/k_i. The **weighted embeddedness** of a vertex v_i is $k_i^{w,int}/k_i^w$.

Community structure is the tendency of networks to consist of 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. There is no single formal accepted definition of a community or how dense the connections must be to form a community. Certainly a fully connected group of vertices, i.e. a clique, would constitute a community, but communities need not be so densely connected. We are interested in finding all of the communities of the network. This global partitioning of the network into communities is called **community detection**. Many different community detection algorithms have been developed over the years and are reviewed in (Coscia et al., 2011; Fortunato, 2010; Fortunato and Hric, 2016).

3.2 Datasets

We use three datasets to evaluate the different topic modelling approaches: 20Newsgroups[1], Reuters-21578[2], and BBC News[3]. The 20Newsgroups dataset consists of 18,846 posts on the Usenet discussion platform which come from 20 different topics such as “atheism” 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” and “copper”. The BBC News dataset consists of 2225 articles in five categories: “business”, “entertainment”, “politics”, “sport”, and “tech”.

3.3 Preprocessing

We use spaCy[4] to lowercase and tokenize the documents and to identify sentences, parts-of-speech (POS), and named entities. We only detect noun-type entities 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 (Schofield and Mimno, 2016) found that they 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 (Hoyle et al., 2021), we remove terms that appear in fewer than $2(0.02|d|)^{1/log10}$ documents. It was shown in (Martin and Johnson, 2015) 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 (Chen et al., 2008). This is intuitive as adjectives and verbs can be used in many different contexts, e.g. 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 will compare the quality of topics with and without this filtering as different algorithms may be more sensitive to the presence of generic terms. Even with nouns there are issues with polysemy, i.e. words with multiple meanings and thus multiple different common contexts. To help with this problem, we use Gensim[5] to extract meaningful n-grams (Bouma, 2009). An n-gram is a combination of n adjacent tokens into a single token so that a term such as “microsoft_windows” can be found and the computer operating system can be distinguished from the windows of a building. We apply two iterations so that longer n-grams such as “law_enforcement_agencies” can be found.

3.4 Term Co-occurrence Networks

The network that we construct from a corpus has terms as vertices. An edge exists between a pair of vertices v_i and v_j if the terms t_i and t_j 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

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html

[2] https://huggingface.co/datasets/reuters21578

[3] https://www.kaggle.com/competitions/learn-ai-bbc/data

[4] https://spacy.io/

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

a fixed-size sliding window over a document.

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. An alternative weighting scheme is to use normalized pointwise mutual information (NPMI) between terms (Eq. 1). This adjusts for the frequency of the terms and assigns high values to terms that co-occur more frequently than expected.

$$NPMI(t_i, t_j) = \frac{log\frac{p(t_i,t_j)}{p(t_i)p(t_j)}}{-log(p(t_i, t_j))} \qquad (1)$$

The edges can be thresholded, i.e. those edges whose weights fall below a certain threshold are removed from the network. For the count co-occurrence networks, we use 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, a threshold of > 0.35 removes a similar number of edges which are presumably the low information edges.

4 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 3.3. 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. Very few of the NPMI edge weights are negative so the impact of this thresholding on network structure is small. After the network is constructed, CT applies a community detection algorithm 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 (i.e. community) is sorted so that the most important and relevant terms for the topic come first and the topics are returned.

Algorithm 1 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 $Topics$
```

5 Empirical Evaluation

We conduct empirical evaluations both to determine the best hyperparameters for CT as well as comparing Community Topic to two other topic modelling approaches, LDA and top2vec. The code and data for the experiments in this section are available at a public GitHub repository: *https://github.com/eric-austin/topic_modelling*.

5.1 Evaluation Metrics

To compare different topic models, we use two coherence measures: C_V (Röder et al., 2015) and C_{NPMI} (Aletras and Stevenson, 2013). These measures both calculate the similarity of terms of the same topic, with more similar terms leading to higher coherence scores. The C_V measure compares the context vectors of two terms found using a 110-term sliding window over the test corpus, while C_{NPMI} computes pairwise NPMI computed using a 10-term sliding window. Both measures have been shown to correlate with human judgements of topic quality with C_V having the strongest correlation (Röder et al., 2015). Even though C_V has stronger correlation that C_{NPMI} with human evaluations, C_{NPMI} is more commonly used in the literature (Hoyle et al., 2021), 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. We use Gensim[6] to compute both measures. Each dataset has a train/test split. We train all models on the train documents and eval-

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

uate using the test documents. We use the standard 110-term window for C_V and 10-term window for C_{NPMI}.

5.2 Hyperparameter Combinations

We train on three datasets 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 (WT) (Pons and Latapy, 2005) and Leiden (Traag et al., 2019) with resolution parameters of 1, 1.5, 2, and 2.5. The Leiden resolution parameter controls the scale of discovered communities with larger values of the parameter finding more, smaller communities. We have previously evaluated many different community detection algorithms and while other algorithms perform better on synthetic benchmark networks, WalkTrap and Leiden were the two that worked best on the term co-occurrence networks. Other common community detection algorithms struggled to find distinct topics. 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.

5.3 Hyperparameter Evaluation

CT has several hyperparameters that can be set, but we desire an algorithm with as few hyperparameters to tune as possible. We thus conduct a series of experiments to determine whether there are good default values for term ordering, co-occurrence window, edge weight, and thresholding.

Ranking the terms is important both for topic labelling and evaluation. 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 CT are groups of vertices so we use the properties of the vertices to rank the terms by importance. We found that ranking terms in the topics by internal weighted degree $k_i^{w,int}$ produced the highest coherence scores. Table 3 in the appendix presents full results.

Filtering out non-noun POS tended to improve the coherence scores when using Leiden but did not have a significant effect on WT. Results are presented in Table 4 in the appendix.

The different co-occurrence windows definitions did not have a significant effect on any coherence scores, as shown in Table 5 in the appendix.

Table 6 in the appendix shows that WT perform best with raw count edge weights and no thresholding but also performs well with non-thresholded NPMI edge weights. Leiden performs well with either count or NPMI edge weights and with or without thresholding.

The Leiden algorithm has a resolution parameter that controls the size of detected communities. Table 7 in the appendix shows that Leiden performs better with a smaller resolution parameter which results in finding larger communities.

These results do not suggest a single best hyperparameter combination across community detection algorithms. The best hyperparameters differ by corpus when using WT. Fortunately, with Leiden a single combination of noun-only POS filtering, sentence co-occurrence window, NPMI edge weights, and no thresholding worked well on all datasets. This is a major point in favour of Leiden as using it turns CT into a hyperparameter-free algorithm. When using Leiden, CT's coherence scores were not quite as strong as with WT, but not having to tune hyperparameters outweighs a slight increase in automated coherence scores, especially give the questions that have been raised in recent years about the reliability of these metrics (Hoyle et al., 2021; Doogan and Buntine, 2021). Full results for each algorithm are given in Tables 8 and 9 in the appendix.

Before declaring a best community detection algorithm to use in CT, we want to consider factors other than just the automated coherence scores. The run time of the algorithms and their stability also impact the choice. We will now evaluate these factors and compare CT with LDA and top2vec.

5.4 Topic Modelling Algorithm Comparisons

We compare the best coherence scores achieved by CT using WT and Leiden to those achieved by top2vec and LDA. We ran LDA on all datasets with both noun-only POS filtering and no filtering for 5, 10, 20, 50, 100, and 200 topics. We ran LDA for 2000 iterations with symmetric dirichlet prior of $\alpha = 1/\text{number of topics}$ and used the best hyperparameters for each dataset. The top2vec algorithm does not have hyperparameters to tune.

We can see from Table 1 that CT produces more coherent topics than both LDA and 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
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 1: Best coherence scores achieved by all algorithms on all datasets.

		Noun-only, threshold > 2		No POS filter, no threshold	
		Time	$\mathrm{C_V}$	Time	$\mathrm{C_V}$
CT	Build net.	3.12 ± 0.02		3.12 ± 0.01	
	Sorting	0.07 ± 0.00		0.08 ± 0.01	
	WT	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
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 2: Run times and stability of algorithms on 20Newsgroups corpus. All times in seconds.

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 demonstrates the sensitivity of the community detection algorithm run times to the size of the networks. Results of this experiment are presented in Table 2.

We can see that the run times of LDA and top2vec are unaffected by the POS filtering that reduces the number of tokens in each document. The network creation and topic sorting steps of CT 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 the fastest, taking only 50 ms on the smaller network while WT takes under 2 seconds. On the larger network, the run times of the algorithms increase by about one order of magnitude. This only takes Leiden up to about half a second while WT takes over 20 seconds. LDA takes about 7 seconds on both corpora and top2vec takes about 65 seconds. On the smaller network, the total run time of CT is comparable to LDA with WT and about twice as fast using Leiden; both LDA and CT are much faster than top2vec. On the larger network, CT is still fastest with Leiden, but slower than LDA with WT. CT is still faster than top2vec.

In addition to comparing the coherence of the topics, we evaluate the use of the discovered topics for clustering the documents. To cluster the documents with the CT topics, 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 highest proportion. We perform this clustering on the BBC corpus with noun-only POS filtering. CT discovers 5 topics using Leiden with resolution parameter 1.0, sentence co-occurrence, NPMI edge weights, and no thresholding. 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. The document clusters found using the CT topics are much closer to the article categories than those found with LDA as measured by Normalized Mutual Information, a standard clustering quality measure. The CT clustering achieves a NMI of 0.790 while the LDA clustering only scores 0.098. The topics produced by LDA have a lot of overlap of top terms, with general terms such as "year" and "government" appearing in most topics. CT has no overlap between topics, making the distinctions between the topics of a document clearer.

5.5 Topic Hierarchy and Relationships

A major advantage of the network representation is 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, i.e. sub-topics. Super-topics can be found by applying community detection to the network of topics, where vertices represent a topics and edges are aggregated from the connections between individual terms.

WT struggles to find sub- and super-topics. This may be due to the high density of the community sub-graphs, which are the denser parts of the original graph by definition, and the topic network, which tends to be fully connected. However, CT using Leiden is able to find both sub- and super-topics

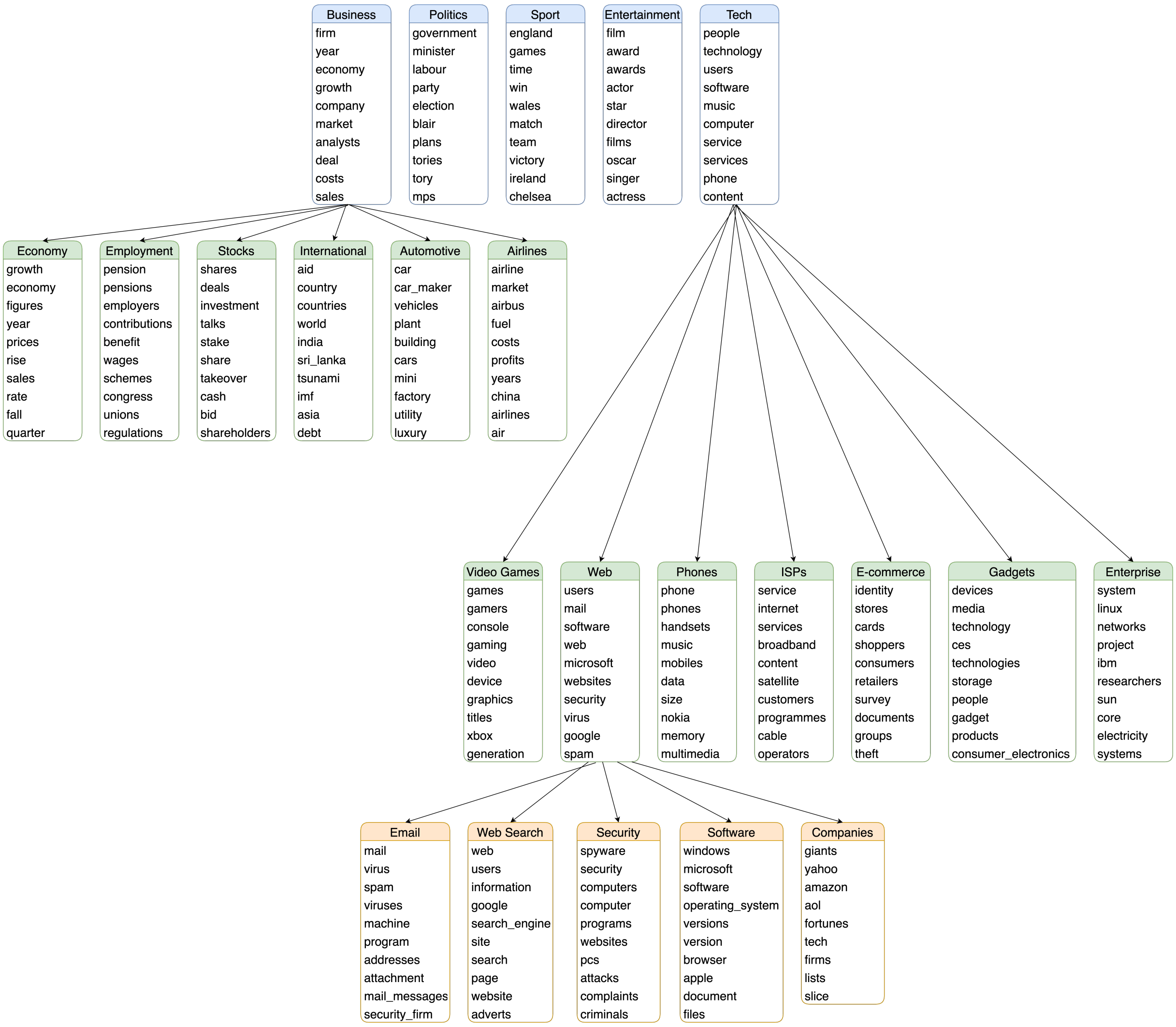

Figure 1: Hierarchy of BBC corpus topics found by iteratively applying CT algorithm using Leiden.

at multiple levels. Using a resolution parameter of 1, five large topics corresponding to the article categories are found on the BBC corpus. Applying CT with Leiden again to the "Tech" topic finds 7 sub-topics such as "video games", "the web", and "cellphones". "The web" sub-topic produces another set of 5 sub-sub-topics such as "email", "web search", and "internet security." This hierarchy can be seen in Figure 1. With a resolution parameter of 2, CT with Leiden initially finds a set of 48 small topics. Performing community detection on the network of topics results in 9 super-topics, 5 of which are large and correspond to the article categories. These super-topics are shown in Figure 2 in the appendix.

As CT with Leiden provides the richest topic hierarchy, finds communities of different sizes as desired, works well on all datasets with the same set of CT hyperparameters, and is extremely fast, we conclude that it is the best community detection algorithm to use in CT.

6 Conclusion

We have presented our novel topic modelling algorithm, Community Topic. We have conducted a thorough empirical evaluation of the algorithm to determine that it works best and needs no hyperparameter tuning with the Leiden community detection algorithm. CT discovers topics with higher coherence scores than LDA and top2vec. It is hyperparameter free and automatically discovers the number of topics with the user able to set the scale of the topics using the Leiden resolution parameter. The discovered topics have a natural network hierarchy and relationships, allowing for the discovery of sub- and super-topics as desired. It is time and resource efficient, requiring no special hardware and discovering topics in less time than LDA and top2vec and much less time than the hours of GPU training required by VAE-based models (Hoyle et al., 2021). CT produces topics with no redundancy, a known issue with LDA and neural topic models. These features make it an ideal tool for downstream applications such as conversational agents and corpus exploration by researchers.

In the future, we will empirically evaluate CT against other models that provide a topic hierarchy. As many community detection algorithms failed to find quality topics on the co-occurrence networks, we will investigate ways to improve both the algorithms and the network representation. While automated coherence metrics give some idea of the topic quality, we plan to integrate CT into a conversational agent to truly test the coherence of the topics and the quality of the topic structure.

References

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 13–22.

Dimo Angelov. 2020. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*.

David Blei and John Lafferty. 2006a. Correlated topic models. *Advances in Neural Information Processing Systems*, 18:147.

David Blei and John Lafferty. 2006b. Dynamic topic models. In *Proceeding of the 23rd International Conference on Machine Learning*, pages 113–120.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, 30:31–40.

Sophie Burkhardt and Stefan Kramer. 2019. Decoupling sparsity and smoothness in the dirichlet variational autoencoder topic model. *Journal of Machine Learning Research*, 20(131):1–27.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172. Springer.

Jiyang Chen, Osmar R Zaïane, and Randy Goebel. 2008. 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*, volume 1, pages 725–729. IEEE.

Michele Coscia, Fosca Giannotti, and Dino Pedreschi. 2011. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):512–546.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. 2020. Topic Modeling in Embedding Spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453.

Caitlin Doogan and Wray Buntine. 2021. 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*, pages 3824–3848.

Nouha Dziri, Ehsan Kamalloo, Kory Mathewson, and Osmar R Zaïane. 2019. Augmenting neural response generation with context-aware topical attention. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 18–31.

Santo Fortunato. 2010. Community detection in graphs. *Physics Reports*, 486(3-5):75–174.

Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics Reports*, 659:1–44.

Thomas Griffiths, Michael Jordan, Joshua Tenenbaum, and David Blei. 2003. Hierarchical topic models and the nested chinese restaurant process. *Advances in Neural Information Processing Systems*, 16.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57.

Alexander Hoyle, Pranav Goel, Andrew Hian-Cheong, Denis Peskov, Jordan Boyd-Graber, and Philip Resnik. 2021. Is automated topic model evaluation broken? the incoherence of coherence. *Advances in Neural Information Processing Systems*, 34.

Karoliina Isoaho, Daria Gritsenko, and Eetu Mäkelä. 2021. Topic modeling and text analysis for qualitative policy research. *Policy Studies Journal*, 49(1):300–324.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Michael Irwin Jordan. 1999. *Learning in Graphical Models*. MIT press.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Diederik P Kingma, Max Welling, et al. 2019. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392.

Katsiaryna Krasnashchok and Salim Jouili. 2018. 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)*, pages 247–253.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196. PMLR.

Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Wei Li and Andrew McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 577–584, New York, NY, USA. Association for Computing Machinery.

Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. 2016. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus*, 5(1):1–22.

Mika V Mantyla, Maelick Claes, and Umar Farooq. 2018. Measuring lda topic stability from clusters of replicated runs. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–4.

Fiona Martin and Mark Johnson. 2015. More efficient topic modelling through a noun only approach. In *Proceedings of the Australasian Language Technology Association Workshop 2015*, pages 111–115.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. 2013. 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*, pages 889–892.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736. PMLR.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Eric Nalisnick and Padhraic Smyth. 2017. Stick-breaking variational autoencoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. 2019. Topic modeling with wasserstein autoencoders. *arXiv preprint arXiv:1907.12374*.

Mark Newman. 2018. *Networks*. Oxford University Press.

Pascal Pons and Matthieu Latapy. 2005. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*, pages 284–293. Springer.

Rajesh N Rao and Manojit Chakraborty. 2021. Vec2gc–a graph based clustering method for text representations. *arXiv preprint arXiv:2104.09439*.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 399–408.

Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid Aref, Marcelo Arenas, Maciej Besta, Peter A Boncz, et al. 2021. The future is big graphs: a community view on graph processing systems. *Communications of the ACM*, 64(9):62–71.

Alexandra Schofield and David Mimno. 2016. Comparing apples to apple: The effects of stemmers on topic models. *Transactions of the Association for Computational Linguistics*, 4:287–300.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. 2004. Probabilistic author-topic models for information discovery. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 306–315.

Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):1–12.

Kai Yang, Yi Cai, Zhenhong Chen, Ho-fung Leung, and Raymond Lau. 2016. Exploring topic discriminating power of words in latent dirichlet allocation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2238–2247.

He Zhao, Lan Du, Wray Buntine, and Gang Liu. 2017. Metalda: A topic model that efficiently incorporates meta information. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 635–644.

A Appendix

A.1 Term Ordering

Table 3 shows the coherence scores by ordering scheme. Internal weighted degree tended to perform best across algorithms and datasets.

Ordering	Coherence	Leiden	WalkTrap
Internal Weighted Degree	C_V	**0.533 ± 0.002**	**0.545 ± 0.007**
	C_{NPMI}	**-0.058 ± 0.004**	**0.041 ± 0.007**
Internal Degree	C_V	0.521 ± 0.002	**0.545 ± 0.007**
	C_{NPMI}	**-0.064 ± 0.004**	**0.042 ± 0.007**
Weighted Degree	C_V	0.458 ± 0.002	0.492 ± 0.006
	C_{NPMI}	-0.146 ± 0.005	-0.020 ± 0.011
Degree	C_V	0.450 ± 0.002	0.489 ± 0.006
	C_{NPMI}	-0.150 ± 0.005	-0.023 ± 0.010
Weighted Embeddedness	C_V	0.470 ± 0.003	0.481 ± 0.006
	C_{NPMI}	-0.277 ± 0.004	-0.223 ± 0.011
Embeddedness	C_V	0.470 ± 0.003	0.484 ± 0.006
	C_{NPMI}	-0.295 ± 0.004	-0.245 ± 0.011

Table 3: Average scores for each community detection algorithm by ordering scheme ± the standard error of the mean. Bold indicates best result for each algorithm.

A.2 POS Filtering

Table 4 shows that filtering out non-noun POS can improve coherence scores, but not for WT.

POS	Coherence	Leiden	WalkTrap
All	C_V	0.515 ± 0.003	**0.554 ± 0.010**
	C_{NPMI}	-0.082 ± 0.006	**0.051 ± 0.012**
Noun only	C_V	**0.549 ± 0.003**	**0.537 ± 0.009**
	C_{NPMI}	**-0.035 ± 0.006**	**0.031 ± 0.009**

Table 4: Average scores for each community detection algorithm by parts-of-speech filtering ± the standard error of the mean.

A.3 Co-occurrence Window

Table 5 shows that there is no statistically significant difference between the three different co-occurrence window definitions.

Window	Coherence	Leiden	WalkTrap
Sentence	C_V	0.533 ± 0.004	0.541 ± 0.011
	C_{NPMI}	-0.069 ± 0.008	0.042 ± 0.013
Sliding 5	C_V	0.530 ± 0.004	0.542 ± 0.012
	C_{NPMI}	-0.049 ± 0.007	0.037 ± 0.014
Sliding 10	C_V	0.535 ± 0.004	0.553 ± 0.011
	C_{NPMI}	-0.057 ± 0.008	0.044 ± 0.012

Table 5: Average scores for each community detection algorithm by co-occurrence window ± the standard error of the mean.

A.4 Weights and Thresholding

Table 6 shows that WT perform best with raw count edge weights and no thresholding but also performs well with non-thresholded NPMI edge weights. Leiden performs well with either count or NPMI edge weights and with or without thresholding.

Weight	Threshold	Coherence	Leiden	WalkTrap
Count	> 0	C_V	0.521 ± 0.004	**0.577 ± 0.016**
		C_{NPMI}	-0.045 ± 0.008	**0.113 ± 0.012**
	> 2	C_V	**0.534 ± 0.004**	0.537 ± 0.009
		C_{NPMI}	**-0.003 ± 0.006**	0.051 ± 0.016
NPMI	> 0	C_V	**0.535 ± 0.005**	**0.557 ± 0.012**
		C_{NPMI}	-0.081 ± 0.010	0.071 ± 0.007
	> 0.35	C_V	**0.541 ± 0.005**	0.509 ± 0.013
		C_{NPMI}	-0.104 ± 0.008	-0.070 ± 0.011

Table 6: Average scores for each algorithm by weight type and threshold ± the standard error of the mean.

A.5 Leiden Resolution Parameter

Table 7 shows that Leiden performs better with a smaller resolution parameter which results in finding larger communities.

Resolution	Coherence	Leiden
1.0	C_V	**0.562 ± 0.005**
	C_{NPMI}	**0.037 ± 0.005**
1.5	C_V	**0.552 ± 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 7: Average scores for CT with Leiden for various resolution parameters ± the standard error of the mean.

A.6 Best Results for Community Detection Algorithms

Tables 8, and 9 show the best scores achieved by CT using Leiden and WT, respectively. While WT tends to achieve the highest scores, it has a different set of best hyperparameters on each dataset.

A.7 Topic Hierarchy

Figure 2 shows the aggregation of an initial set of small topics found with a high resolution parameter into super-topics by applying Leiden on the network of topics.

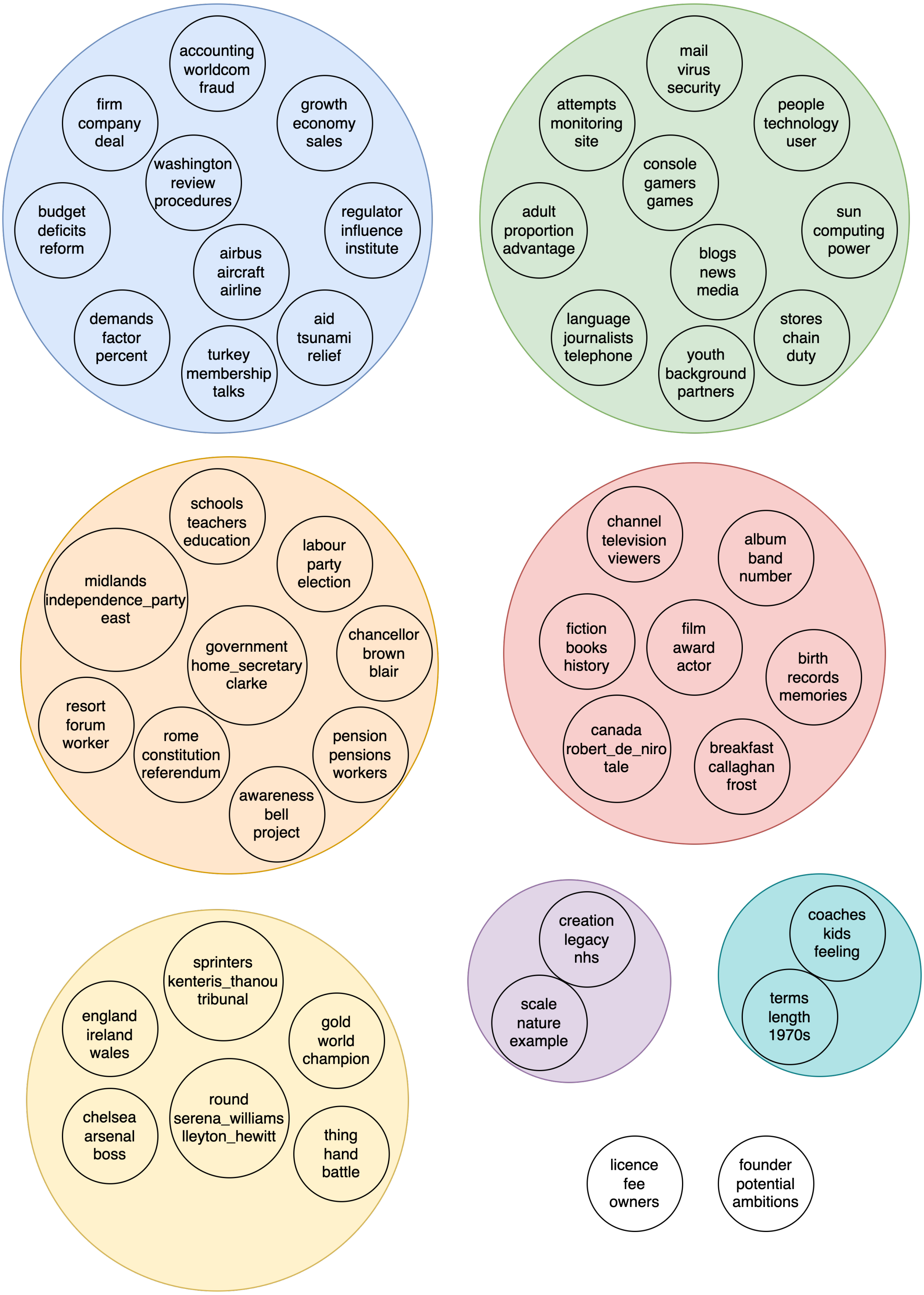

Figure 2: Super-topics found by applying community detection on network of small topics.

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 8: Best coherence scores using Leiden. Average results on all datasets and results for each 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	0.632 (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 9: Best coherence scores using WT. Average results for all datasets and results for each 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.