Discovery of Numerous Specific Topics via Term Co-Occurrence Analysis

Omid Madani
AI Center, SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
madani@ai.sri.com

Jiye Yu
Stanford University, EPGY
Stanford, CA 94305
jyu2009@stanford.edu

SRI Artificial Intelligence Center
Technical Note 569
March 24, 2011

Abstract

We describe techniques for the construction of term co-occurrence graphs and explore an application of such graphs to the discovery of tens of thousands of fine-grained, that is specific rather than broad, topics. A topic corresponds to a small dense subgraph in our work. We discover topics by randomized local searches (constrained random walks) initiated at each term (node) in the graph. The mined topics are highly interpretable, and reveal the different meanings of a term in the corpus. We explore document tagging via the induced topics, and demonstrate the information-theoretic utility of the topics when they are used as features in supervised learning. Such features lead to consistent improvements in text classification accuracy over the standard bag-of-words tfidf representation, using SVM classification, even at high training proportions when it is difficult to improve over the tfidf representation. We investigate the effect of various options and parameters, including window size during graph construction and variants of a tagging strategy, on the accuracy of classification. We explain how a layered pyramidal view of the term distribution helps in understanding the algorithms and in visualizing and interpreting the topics.

1 Introduction

Consider the task of obtaining quick insights into how a term is mentioned in a corpus of text (in news articles, emails, comments, tags, and so on). The term of interest may correspond to an entity, such as a person, an organization, a product, or an event, etc. (*e.g.*, "Bill Clinton", "religion",

"oil spill"). Reading each document that contains the term is one approach, but this can be very time consuming, and it may not readily reveal the salient "topics", *i.e.*, events, activities, roles, or in general the associations that the term (the concept corresponding to the term) participates in. Algorithms for quickly extracting such *specific* topics would find applications in intelligence analysis, discovery of user interests, document and corpus summarization, document tagging and routing, and so on. In this paper, we explore the use of co-occurrence graphs, built on top of terms (ngrams over words), for the discovery of such topics. Co-occurrence graphs find diverse uses in various information retrieval and language processing tasks [5, 2, 21, 23, 7]. In this paper, we first briefly present techniques for constructing such graphs. We discuss edge semantics, describe several variations to graph construction, and briefly explore the structural properties of the generated graphs. We then develop an application to fine-grained topic discovery. We describe algorithms for topic discovery and a few variations on document tagging using such topics.

We describe methods for constructing a type of co-occurrence graph wherein edge weights reflect conditional probabilities, or close variants, as we explain. We compute term co-occurrence information by sliding a window over each term appearing in a document (*e.g.*, [12, 21, 7]). During corpus processing, new edges reflecting co-occurrences are added to the graph. During corpus processing, or after all the documents are processed, certain edges are removed. We distinguish between three types of edge dropping, and explain the effects of each. Unlike other types of graphs, the nodes (terms) in the co-occurrence graph should not be treated the same way: the graph should not be viewed as "flat". We explain how consideration of term frequencies, in particular the *pyramid view* of the term distribution, finds several uses in algorithm design and in visualizing and making a better sense of the discovered topics. We also briefly explore the number of edges and density or connectivity of the generated co-occurrence graphs, in particular as a function of various parameters such as window size.

We mine the co-occurrence graph for relatively small dense subgraphs, or semicliques, which constitute our topics. While computing a maximum size (semi)clique is an NP-hard problem, we expected that mining for *maximal* semicliques, which we also anticipated would contain relatively few terms (*e.g.*, in the range of 10s), would be quite feasible and useful. We present a randomized local search strategy for semiclique generation (described in Section 4). We find that the induced semicliques, corresponding to groups of terms that have occurred in close proximity in several documents, are quite interpretable. They reflect a variety of subjects and events such as natural disasters, elections, sports events, group and organizational activities, and political controversies. Naturally, a given term, such as a person's name, can participate in several topics, often reflecting different senses of the term, or roles for the corresponding concept or entity. We explore the application of the topics as features for supervised learning, finding that the topics improve performance even at high training proportions. We also compare to latent Dirichlet allocation [1], and highlight the advantages of co-occurrence analysis in *fine-grained* topic discovery. We therefore provide evidence that topic discovery via co-occurrence analysis is an effective way to mine text collections, complementing the more commonly studied methods such as document clustering and existing term-document analysis techniques.

This paper is a significant expansion of our earlier publication [13], and contains improved performances in the classification section (Section 4.2.1). It is organized as follows. Section 2 describes the corpora and document processing. Section 3 describes graph construction and reports on some of the properties of the graphs generated. Section 4 presents topic (semiclique) mining, document tagging and our classification experiments. Section 5 discusses related work, and Section 6 concludes.

2 Data Sets and Preprocessing

Table 1 presents our data sets. The newsgroups data [9] contains roughly twenty thousand postings under twenty subjects (alt.atheism, sci.electronics, talk.politics.misc, talk.religion.misc, $\cdots$). The Reuters collection spans a year's worth of Reuters news articles [18]. The TDT5 corpus (Topic Detection and Tracking, for 2004, [10]), is a collection of news articles from various newswire sources (either in English or translated to English (from Chinese and Arabic). We lower cased words and changed numbers to NUM to tokenize. The vocabulary that we use includes both unigrams and bigrams and trigrams (phrases such as "new jersey"). We generated the ngrams via the same graph construction process, employing filters to keep the significant ngrams, which we explain in the next section. We did not use a stop list, and instead relied on statistical techniques to filter terms as appropriate in (sub)tasks such as edge selection and topic discovery.[1] Note that frequent terms can add some meaning to certain topics or improve readability when they are part of phrases (such as "William of Ockham"). The presence of ngrams (as later shown in the section on topic discovery), significantly improves the interpretability of the topics, specially because many of the topics are specific and focused around people and events (which are ngrams).

We only used the body portion of articles: for newsgroups, we skipped the subject and sender as well as footer or signature parts of the postings (the latter tended to have machine names or addresses, etc). Experiments were run on a Dual-Core AMD Opteron (25GB RAM, 2.8GHz). Document tokenization and graph construction was written in Java, and semiclique discovery was in written in C++.

3 Graph Construction

In this section, we first present the algorithm for graph construction, then explain and discuss the effects of edge dropping (zeroing certain edge weights) on potential "accuracy" losses (*i.e.*, the quality or coverage of topics found in our application), and present stability experiments and degree distributions. We discuss how thinking about term (document) frequency and its relation to term meaning and connections to other terms, and in particular, the "pyramid" view of the term distribution helps in understanding the techniques as well as visualizing the graph and its

[1] The choice of stop list, and more generally the particulars of tokenization, is domain/task dependent.

	N	#uni	$\|V\|$	$\|d\|$	user time
newsgroups	19k	96k	122k	208	10s of minutes
Reuters	800k	600k	730k	237	1 day
TDT5	280k	394k	466	350	half a day

Table 1: The data sets we use. N, #uni, $|V|, |d|$, and T, respectively denote the number of documents in corpus, the number of unigrams (words), the vocabulary size (including bigrams and trigrams), avg. number of term occurrences per document (or doc. length), and the range of (user) time it took to build the graphs (see Sections 2 and 3).

subgraphs. We report on the average number of connections and give an example neighborhood of a term in the graph.

Figure 1 presents our main graph construction algorithm. We first give a quick overview, then elaborate on several aspects further. The algorithm begins with an empty graph (no edges), and processes the corpus documents in a random order. The algorithm keeps and updates edge weights (or counts) for each term as well as a special term v_0. The weights are simply conditional probabilities, or a close variant, and are non-negative. In particular, $w_{v,u}$ is $P(u|v)$, meaning the probability that term u is seen sufficiently close to v, *i.e.*, within window of size L words on either side, given that a random position is picked from a randomly picked document, and v is seen in that position. By sliding a window of size L over each position (term occurrence) in the document, the counts are accumulated, for the eventual computation of the weights (conditional probabilities). After all documents are processed, the graph is pruned: edge weights that are not sufficiently statistically significant and those that do not pass a test of sufficiency of "informativeness" or "surprise-level", in particular point-wise mutual-information filter [15]) are dropped. In most of our experiments, L ranges in $[3, 40]$, which implies coverage from parts of sentences to several sentences and paragraphs (sentences are often, on average 10 to 20 words long). Explicit sentence and paragraph boundary detection may improve topic discovery.

3.1 More on Edge Weights

The weight $w^d_{v,u,i}$ denotes the value of term u for the ith occurrence of v in document d. Two approaches have been used in the literature for weighting the terms inside the window: Boolean (or plain) weighting and variant of proximity weighting. We implemented both options, and we describe both of them here. However, in our classification accuracy experiments of the next section, we didn't find evidence of superiority of one compared to the other.

If term u is outside the window of size L words on either side of v, then $w^d_{v,u,i} = 0$. In case of Boolean weighting, we simply have $w^d_{v,u,i} = 1$ when u occurs inside the window (whether u occurs once or multiple times within the window). In case of proximity weighting, we use a linear decreasing function of distance whose value is 1 when distance is 0 and 0 when distance is L.

GenerateGraph(L**,**W_0**,** DF_0**,** K**)**
/* process documents in a random order */
1. For each document d **in corpus:**
1.1 For each term v **in document** d**:**
1.1.1 $DF(v) \leftarrow DF(v) + 1$ **/* update document frequency*/**
/* compute the proximity value of term u wrt to v */
1.1.2 For each term u**,** $w^d_{v,u} \leftarrow \frac{\sum_{1 \le i \le tf^d(v)} w^d_{v,u,i}}{tf^d(v)}$
1.1.3 $w'_{v,u} \leftarrow w'_{v,u} + w^d_{v,u}$ **/* update cumulative edge value */**
1.1.4 OPTIONAL: If $DF(v) > DF_0$**, /* v seen sufficiently enough? */**
/* If so, drop its small connections (memory efficiency) */
1.1.4.1 For all neighbors u**, if** $\frac{w'_{v,u}}{DF(v)} < W_0$ **then** $w'_{v,u} \leftarrow 0$
/* After processing all documents, finalize */
/* graph weights (and, optionally, remove some more edges) */
2. For each pair of terms v **and** u **where** $w'_{v,u} > 0$**:**
2.1 $w_{v,u} \leftarrow \frac{w'_{v,u}}{DF(v)}$ **/* the edge weight */**
2.2 If $w_{v,u} < max(\frac{K}{DF(v)}, W_0)$**,** $w_{v,u} \leftarrow 0$**. /* insufficient evidence */**
2.3 If $\frac{w_{v,u}}{w_{v_0,u}} < 10$**,** $w_{v,u} \leftarrow 0$**. /* PMI filtering */**

Figure 1: The graph generation algorithm. $tf^d(v)$ denotes number of occurrences of v in document d, and $w^d_{v,u,i}$ denotes the proximity value of u with respect to v for the ith occurrence of v (in Boolean weighting, simply 0 or 1). The special (dummy) term v_0 is used in computing PMI stats (step 2.3). v_0 is placed at every word occurrence, and weights from it ($w_{v_0,u}$) are computed. Defaults: $L = 20, W_0 = 0.01, DF_0 = 50, K = 3$. See Sec. 3.

Thus, with l (length) denoting the number of words between u and v, $0 \le l \le L$ (in case of multiple occurrences of u within window, we use the shortest distance), we have $w^d_{v,u,i} = 1 - \frac{l}{L}$. Then, letting $tf^d(v)$ denote number of times term v appears in document d (term frequency of v), $w^d_{v,u} = \frac{\sum_{i=1}^{tf^d(v)} w^d_{v,u,i}}{tf^d(v)}$ is the (empirical) conditional probability or otherwise the average proximity of u to v in document d. We use a two-tier update: we first compute the weights (conditional probabilities or proximities) for each term in the document, then we average those weights for each term over all the documents:

$$w_{v,u} \leftarrow \frac{\sum_{1 \le d_i \le N} w^{d_i}_{v,u}}{DF(v)},$$

where the sum in the numerator goes over all the documents in the corpus, and DF(v) is the document frequency of v, *i.e.*, the number of documents in which term v appears ($DF(v) \le N$).[2]

[2]Note that appearances where v occurs in a larger ngram that is kept (such as v="new", in "new york") are not incorporated into the document frequency.

Another option is to perform the updates in one step. The two tier update lessens the effects of long documents.

An Example. Given the text fragment "She wants to travel to Toronto to attend CIKM", with "travel" being the word v at center of the window, and with $L = 3$, "to", "want", "Toronto", "she", and "CIKM" all obtain Boolean values of 1, and proximity values of respectively 1, 0.5, 0.5, 0 and 0 for proximity weighting. Assume the final weight $w_{v,u} = 0.1$, for v="sports" and u="team". In the Boolean weighting, this means that if in a randomly picked document we saw v, then with probability 0.1 (10% of the time), we'll see "team" within L positions of "sports". In our version of proximity weighting, $w_{v,u} = 0.1$ can mean at one extreme that 10% of the time we see u immediately next to v and otherwise we don't see u within window, or at another extreme, we see u with proximity value 0.1 to v (*e.g.*, 9 words from v, with $L = 10$) but 100% of the time with v. In practice, it's likely that the situation falls between these too extremes. Proximal weighting allows us to summarize two numbers (frequency of co-occurrence and proximity, both informative) in one.

In the optional step 1.1.4, certain edges may be dropped for memory efficiency. If the count on v is sufficiently large, and the computed weight is small (*e.g.*, the threshold is 0.01 in some of our experiments), the edge is dropped, implicitly setting its weight to 0 (however, the edge may be reinserted later).

3.2 PMI and Statistical Evidence

Step 2.2 and 2.3 drop many edges from the final graph. This removal step keeps statistically reliable and salient associations, and improves the quality of the topics extracted. Step 2.2 is a quick test of evidence for significance, where we use K=3 in our experiments. If a term appears only in one or two documents, the computed weights are inaccurate and in particular can be high for the nearby terms. The fraction $\frac{K}{DF(v)}$ guards against that possibility, while still including some such connections if the weights are high enough and the source vertex v has occurred sufficiently enough ($K = 3$). In step 2.3, we use connections to the special term v_0. The weights of the connection to v_0 is simply the prior (or marginal or unconditional) probability of observing v within a two-sided window of size L in case of Boolean weighting, or otherwise it's the marginal proximity value in case of proximity weighting. For instance, $w_{v_0,u}$ when u ="the" is 0.54 (with $L = 20$) in the newsgroup data set. This means that if a random document is picked, and within it a random position is picked, the proximity value of "the" is on average 0.54 (about 10 terms a way). "the" has document frequency of over 17k, appearing in almost all documents. We use these weights when applying a "statistic of surprise" test. In case of Boolean weighting, we are basically requiring that the point-wise-mutual-information or PMI be large enough [15]. We have empirically found that ratios above 5 work well in removing uninteresting colocations (see also Sect. 3.6), and we use a threshold of 10 in this paper.

3.3 Phrase Extraction

The use of bigrams, such as person names, and higher n-grams (phrases, common expressions) increase the interpretability of the topics found, and prevents the generation of "degenerate" topics that correspond to merely parts of names and such. We use the same algorithm, but with a one-sided window of size L=4 to generate candidate phrases, pairs of words v and u, such that $w_{v,u}$ is significant. We treated the pairs whose $w_{v,u} > 0.1$ and for which the edge weight passed the PMI condition as phrases and added them to the vocabulary. This process can be repeated to find longer phrases and expressions, but we went up to fourgrams for this paper.

3.4 Space and Run-Time Complexity of Graph Construction

The space complexity of graph construction is a direct function of the number of connections: $O(\deg |V|)$, where $\deg$ is the average degree of a term and $|V|$ is the vocabulary size. $|V|$ nears a million for Reuters (with up to 3-grams), thus with say $\deg = 1000$, we get a billion entries. Larger corpora (multiple years of Reuters or multiple news sources) then can exceed the memory capacity of a common machine. If we perform edge dropping, with a two-sided window, the maximum outdegree is $\frac{2L}{W_0}$ (4000 with $L = 20, W_0 = 0.01$). Section 3.7 reports on the actual outdegrees (far below this maximum). Use of the special v_0 (potentially connected to every term) only adds $|V|$ (number of terms) to the number of edges kept. Alternatively, for v_0, we could drop connections below a certain W' (say 0.0001), and use the conservative estimate of W' for them.

For each term v, we keep a hash mapping, initially empty, from term id to the edge information (the weight) (the edges outgoing from v). Sliding the window and updating term weights amount to look ups into this mapping (we use two mappings, one for document level statistics, and another for the corpus level). Thus, we incur a cost of $O(L)$ per term occurrence, but if we check weight magnitudes (step 1.1.4), we also incur a cost of $O(\deg)$ (the check could be done only periodically for each vertex). The last column of Table 1 shows the graph construction time ranges for unigrams (a single pass).

We note the memory size during graph computation was twice as high with $W_0 = 0$ compared to $W_0 = 0.01$, from under 15% of available memory to over 30% on TDT5 and Reuters (with $L = 20$). We next briefly discuss a few variations on algorithms and implementation of graph construction.

3.4.1 Discussion of Implementation Variations

We briefly discuss a few variations to the algorithms and implementation. For ngram extraction, we used the same code base and extra passes over the corpus (Section 3.3). We went up to fourgrams (three passes). An alternative for extracting higher n-grams (longer phrases and expressions) is to build and update an inverted index, mapping smaller n-grams to locations that they occur in. This can reduce the full cost of multiple passes over the data, and substantially speed up computing

	Avg Weight	WtDiff	MissFrac(.01)	MissFrac(.05)
20news	0.153	0.00006	0.016	0.0007
TDT5	0.152	0.0009	0.04	0.0013

Table 2: The first columns shows the average edge weight in the generated graph. The next three columns show respectively the average of difference in weight for corresponding edges from different runs (different random ordering of documents), and the ratio of edges missing from different runs, for those edges (with computed weight) greater than 0.01 and those edges with weight greater than 0.05.

frequencies for and selecting higher n-grams. An alternative to a minimum weight threshold for edge dropping during graph construction is a maximum outdegree per term, and periodic edge dropping when the outdegree reaches a limit. This may also allow for keeping smaller weights compare to a fixed weight threshold. The utility of such depends on application. Finally, graph construction can easily be distributed or partitioned/phased (in each phase, compute the edges for a subset of the vertices), because computing the outedges for each term can be done independently of the same computation for other terms. The phased approach can also mitigate memory constraints during graph construction: in each phase the edges incident to only a fraction of the terms need be computed.

3.5 Edge Dropping and Weight Estimation

The optional step of dropping low-weight edges during processing of the documents (step 1.4.4, for memory efficiency) can cause inaccuracies in edge weights.[3] Here, we are interested in approximation effects of dropping edges. Thinking in terms of Boolean weighting (simple conditional probabilities) simplifies the analysis. If we are interested in conditional probabilities of say mostly greater than $W = 0.05$, then dropping edges below a threshold of $W_0 = 0.01$ may not introduce substantial inaccuracy. The probability that an edge is dropped diminishes quickly as the ratio $k = W/W_0$ increases. The probability that an edge of true weight W is not seen in $1/W_0$ consecutive observations is $(1 - W)^{1/W_0}$ (with $W_0 = 0.01$ and $W = 0.05$, this is 0.1, and with $W = 0.1$ it is 0.01). Table 2 shows the average edge weights, and difference in edge weights, in 5 graphs computed from different random orderings of documents ($W_0 = 0.01$ throughout this paper), as well as the fraction of edges missing, for those edge weights above threshold of 0.01 and 0.05 respectively. We see that the losses are relatively small (one could also run the algorithm multiple times and generate multiple graphs, then average the graphs).

[3]Where the gold-standard in this analysis is weights computed without any edge dropping. Note that a corpus is finite and the empirical conditional probabilities are themseleves approximations of "true" probabilities. If the content of a corpus is nonstationary, then the problem of what true probabilities are becomes more complex.

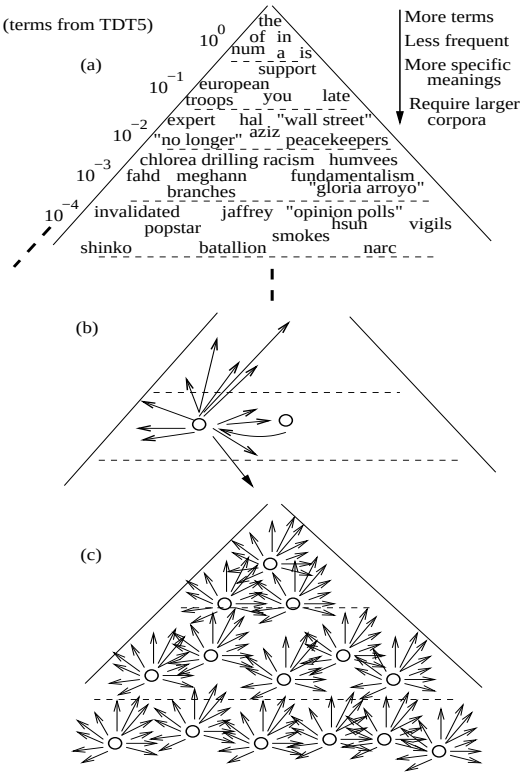

Figure 2: The layered pyramidal view of terms. (a) A selection of terms in TDT5 placed in the pyramid according to their document frequency (df). (b) Terms with similar df tend to reciprocate directed edges, and due to edge dropping during graph construction, terms tend to connect to other terms within layer or in higher layers. (c) In approximate graph construction, the lower level (less frequent) terms carry most of the burden of keeping the connections to higher level terms.

3.6 The Pyramid View

We now consider the term frequencies and how frequency relates to which edges are kept. Terms (single words and meaningful phrases) are approximately distributed according to Zipf's law, the number of terms growing rapidly with decreasing frequency. The terms can therefore be viewed as residing in a pyramid, where a few most frequent terms reside at the top (Figure 2(a)). Now, we note that if two terms have roughly the same frequency, or equal (nonzero) marginal probabilities, then their conditional probabilities must be equal: $P(u) = P(v) \Leftrightarrow P(u|v) = P(v|u)$. More generally, with $k > 0$,

$$P(u) = kP(v) \Leftrightarrow P(u|v) = kP(v|u) \tag{1}$$

(as $kP(v|u) = k\frac{P(v,u)}{P(u)} = k\frac{P(v,u)}{kP(v)} = P(u|v)$).

The implications of equivalence 1 are several for the way the algorithm builds the graph within the pyramid: edges are reciprocated and have a similar weight when the terms are in the same tier of the pyramid, *i.e.*, when the terms have close frequencies (Figure 2(b)).[4] Moreover, with the edge dropping using the minimum threshold W_0, outgoing edges of a term tend to connect it to terms in the same or above layers more than to terms in the lower layers (Figure 2(b) and (c)).

Note also that (again, from equivalence 1) given the knowledge of the term frequencies and an approximate weight for one direction (say $w_{v,u}$), the other direction ($w_{u,v}$) can be computed approximately as well. Thus the algorithm is not losing substantially by removing downward edges. In a sense, the algorithm leverages the lower frequency terms, of which there are many, to "support" (carry the burden of) keeping the connections to higher frequency terms, for the purposes of efficiency.

An effect that dropping edges by PMI (or similar constraints) has is that it tends to cut edges that go all the way to top tiers (most frequent words). Terms at the top tier are so general and frequent that they have very little meaning and limited utility in many tasks. The algorithm also removes edges that are not statistically significant, due to the source vertex not being seen sufficiently enough (those terms residing in the bottom layer of the pyramid).

If we drop edges during graph generation, one final type of information is lost: "non-salient" edges, *i.e.*, those edges that connect terms in the same tier, but have true weight below W_0, are also likely dropped. We expect that two terms having similar frequency but that colocate only say one in hundred occurrences are probably not useful as members of the same topic. We should also note that with lower term frequencies, we need substantially larger corpora to estimate weights with adequate statistical significance, thus lowering W_0 by itself does not improve coverage or effectiveness (a larger corpus is required). Of course, the space complexity of the graph increases linearly as W_0 is lowered (fixed overhead with each term occurrence, when W_0 is fixed).

3.7 Term degrees, Connectivity, and Graph Structure

Figure 3 shows the average outdegrees as a function of term frequency (proportion of documents that the term occurred), over those terms that had at least one edge ($L = 20, W_0 = 0.01$, proximity weighting). For all 3 datasets, we show outdegrees before PMI dropping is performed (before step 2.3 in Figure 1), and for the newsgroup set, we show the averages after PMI dropping as well. We observe that for terms with few occurrences, the averages are small, mainly since we drop edges due to lack of statistical evidence. After that, we see a quick increase, but later again the degrees subside somewhat. If we used PMI dropping, overall outdegrees drop significantly.[5] The terms with the highest outdegree in newsgroups were: "toronto" (deg=201,df=228), "crime"(192,294), and "print" (187,219). On TDT5 they were: "michael ballack" (233, 205) (German soccer player),

[4] If we drop edges during graph generation, it is possible that an edge in one direction is kept while the other is absent, even for two terms with similar frequency.

[5] The bins in the last few proportion ranges have only 10s of terms, and the slight increase towards the end (before PMI dropping) are due to variance of averaging over a small sample.

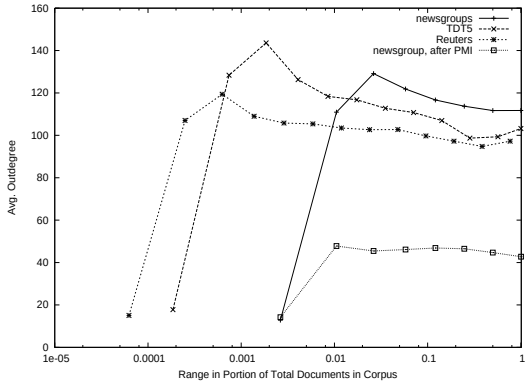

Figure 3: Average outdegrees against average term frequency for L=20 and proximity weighting. The x-axis is bucketed, and the average frequency of the terms whose frequency fall in the bucket marks the x-coordinate. Average outdegree is initially small (due to terms that are not seen enough) but after hitting a maximum, it dips somewhat as frequency increases. For the top three plots, degree averaging is done before PMI filtering of edges, *i.e.*, before step 2.3 in Figure 1. For the bottom plot, averaging is done after, on the newsgroup data set.

"paul scholes"(232, 269) and "andriy shevchenko" (231,235). And for Reuters they were: "squad goalkeepers" (218,172), "midfielders" (217,339), and "nearbys" (218,172). The uninformative (stop) term "of" has outdegree around 100 in all three data sets. On the newsgroup data set, we could run graph generation without edge dropping, and of course we obtained very high outdegrees (thousands for all the stop words).

Tables 3 and 4 display the graph sizes (after PMI filtering) and the number of terms (up to 3grams) with at least one edge, as a function of window size L and other parameter settings. With increasing window size, the number of associations (edges) can potentially increase, but PMI filtering controls such growth. We observe that each doubling increases the number of edges by about 50%, while the number of terms with some edge only increases slighlty (by 5%). Thus the degree of existing terms must be increasing significantly (the co-occurrence graph becomes denser with increasing L). As expected Boolean weighting yields more edges than proximity weighting for the same window size. Proximity weighting of size L may be comparable to Boolean weighting of $L/4$. Increasing the window size to the whole document substantially increases the number of edges (tested on newsgroups). Note that while average document length is roughly 200, there are 300 documents with length over 1000, and 87 with length over 2000 terms, in newsgroups.

We treat the graph as undirected when we mine for candidate topics (Section 4): as long as an edge in either direction exists (passing the minimum PMI threshold and a minimum weight), the two vertices are considered connected. Note that the conditional probabilities follow equation 1, and PMI is symmetric. The average degrees (which now includes both indegree and outdegree in the original directed graph) increases as a function of document frequency of the terms. Table

20 newsgroups, $L \rightarrow$	3	10	20	20 ($W_0 = 0$)	20 (prox)	40	40 (prox)	All
vert. edges > 0	22k	25k	26k	26.6k	20k		21k	33k
edges	140k	310k	473k	600k	139k	695k	203k	3.7 mil

Table 3: After filtering edges via PMI threshold value of 10, and minimum weight value of 0.01, the number of edges and number of vertices with at least one edge is shown, for various window sizes (Boolean window, minimum weight value of 0.01, unless specified), on the newsgroups data set. The total number of terms was 136k, but only a fraction of terms keep an edge. Many vertices are too infrequent (below frequency 3) to obtain an edge. Increasing window yields significantly more edges ("All" refers to Boolean window covering all the document).

Reuters, $L \rightarrow$	5	20 (proxim)	40
vertices with some edge	206k	191k	237k
edges	2.56mil	2.8 mil	10.5 mil

Table 4: Number of vertices (with at least one edge), and edges, for a few choices of L (similar to Table 3). Total number of terms (vertices) was 750k for Reuters.

5 shows degree sizes for a few term frequency ranges. Note the degree increases somewhat with higher frequency. We also looked at the number of neighbors within radius 2, and for vertices with several hundred edges, the number was in the thousands on the newsgroups, and 10s of thousands for Reuters, for terms with frequency at minimum in the 10s. We note that the number of terms with at least one edge is roughly 20-30k on newsgroups, and roughly 200k in Reuters. Thus almost all terms with frequency in the 10s reach almost all other terms with the same minimum frequency constraint within 3 edges (paths of no longer than 3 edges). Indeed, we briefly explored the connectivity of the generated (undirected) graphs, for several window sizes on both newsgroups and Reuters data sets. All the graphs generated had a single large connected component, and then several (10s of) tiny components (*i.e.*, single edges or connected groups with no more than a handful of vertices, containing low frequency terms).

3.8 An Example Neighborhood

Figure 4 shows the neighborhood for "christianity" from processing the newsgroup data set, for those neighbors whose edge weight are greater than 0.05 (and pass the PMI constraint). Showing the neighbors in a hierarchy helps understand the generality of terms in the corpus. As may be expected, edges going upward have higher weights than the corresponding reverse edge. Other neighbors of "christianity" (having weight below 0.05) were: (islam,df=145), (religions,147), (compatible,205), (homosexuality,134), (nature,358), (jesus,534), and (nothing,1273).

$DF(v) \rightarrow$	10-50	50-100	≥ 200	≥ 1000
newsgroups (L=10)	18	59	131	202
newsgroups (L=40)	26	85	187	243
Reuters (L=5)	8.7	29	130	240
Reuters (L=40)	42	125	474	818

Table 5: Degree (after making the graph undirected) as a function of document frequency and window size on newsgroups and Reuters (Boolean Window).

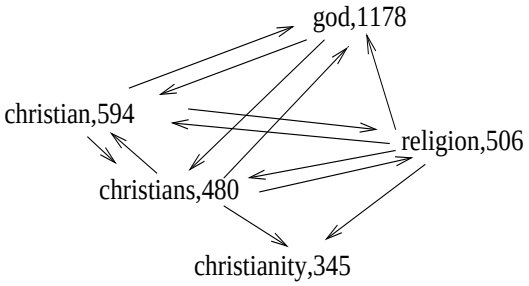

Figure 4: The immediate neighbors of "christianity", with connection weight of at least 0.05 from "christianity" to them, from processing the newsgroups data set. All the edges among the neighbors are shown (passing 0.05 threshold). The edges from christianity are implicit (not shown). The document frequencies are shown next to each term. In this corpus, "god" appears in more contexts (or has more meanings) than the other terms shown.

4 Semicliques: Discovery and Use

In this paper we are exploring treating sets of highly-interconnected terms in the co-occurrence graphs as candidate topics. Although it is generally difficult to assign hard boundaries to groupings of terms, *i.e.*, deciding what terms to include in the group and what not, such sets can reflect the events or topics that the articles in the corpus discuss (implicitly or explicitly). Such topics can then find applications n document tagging and routing, discovery of user interests, summarizing a corpus, and so on. In this work, we explore (maximal) semicliques in the co-occurrence graph as topics. A semiclique is a subgraph wherein every node is connected to a high fraction of other nodes in the subgraph.

In the first part of this section, we first present our generic procedure for finding semicliques (or semiclique/topic mining), and then report on statistics of the semicliques extracted, timings, stability, an statistical analysis of significance, and example semicliques. In the second part, we explain and motivate our procedure for tagging the documents by the semicliques, and report on coverage statistics (*e.g.*, the fraction of the corpus tagged by the semicliques), followed by classification experiments, where we use the semicliques tagging documents as extra features to boost

GenerateSemiCliques($G(V, E)$, k, r, τ)
1. $Q \leftarrow \emptyset$**. /*** Q **is the global set of semicliques */**
2. For each term v **repeat until k consecutive failures.**
2.1. Initialize: $S \leftarrow \{v\}$**. /* S is a candidate semiclique */**
2.2. Repeat until no more additions possible: /* attempt to grow S ***/**
2.2.1 Pick a u **at random from those neighbor of** S **for which /* examine neighbors of** S ***/**
2.2.1 the addition of u **preserves the** r**-connectivity of** S**:** $S \leftarrow S \cup \{u\}$**.**
2.3. If $|S| \geq 3$**, add** S **to** Q **and reset failures if non-redundancy conditions are met (see Section 3), otherwise increment number of failures (go to 2.1).**

Figure 5: Pseudo code for topic generation. A topic is a maximal semiclique. Each term in a semiclique S (a set of terms) is connected to at least $\lceil r(|S|-1) \rceil$ other terms in the semiclique ($r \geq 0.5$ in our experiments). Edges are undirected and unweighted here: It is assumed there is a connection as long as an edge in one direction in the co-occurrence graph passes filters (PMI and min edge weight of 0.01). A newly generated semiclique is added to Q if no other semiclique S' has high overlap with it ($\frac{|S \cap S'|}{min(|S|,|S'|)} \leq 0.5$), or otherwise S has higher size (in which case the others with high overlap are removed from Q).

supervised learning performance. The supervised learning experiments give us a way to compare variations to the techniques and the choice of parameters. Each part concludes with a discussion section.

4.1 Semiclique Generation

For the rest of the paper we make the co-occurrence graph undirected, and furthermore ignore the weights. In the undirected graph, the vertices edge u and v are connected, as long as an edge exists between u and v (after PMI and minimum weight filters for edge removal) in either direction. A semicliques, for us, is a maximal set of terms with the r*-connectivity* property: by r-connectivity we mean each term (vertex) is connected to at least a fraction r of the other terms in the semiclique. That is, semiclique S has r-connectivity iff every vertex in S is connected to $\lceil r(|S|-1) \rceil$ other vertices in S. We used $r \geq 0.5$ (default of $r = 0.6$), as a simple way of ensuring that we generate fairly cohesive and specific semicliques (see Discussion Section 4.1.7). Maximal means no additional vertex can be added without violating r-connectivity.

The algorithm for generation of semicliques given a co-occurrence graph is given in Figure 5. Here, unless otherwise specified, this is the graph output using the algorithm of Figure 1 (with application of PMI filtering, and removing edge weights below 0.01). The graph is treated as undirected, and we do not use the edge weights in clique discovery. Later, we discuss the effects of parameters such as window size. Let Q denote the set of kept semicliques, initially empty. Terms are sorted in a random order.[6] Repeatedly, a term is picked as a seed from this list, and several

[6]It is also conceivable that in many tasks, one may be interested only in the topics that a fixed term participates in, in which case, only semiclique generation from a single term would be of interest, and one can skip whole corpus

iterations of randomized semiclique growth are conducted (described below) to create a candidate (maximal) semiclique S. Two tests are performed to determine whether S can be added to Q. If it's not added to Q, we have a failure. Candidate generations are repeated until k consecutive failures (to add to Q) occurs. After k consecutive failures, the algorithm picks the next term in the randomly ordered list and repeats. We next describe the generation of a candidate semiclique, then the tests for addition to Q.

4.1.1 An Iteration of Semiclique Generation

Each iteration of candidate semiclique generation begins with initializing a semiclique S with the seed term v. Let neighbors of S be $Neib(S) = \{v \in V | v \notin S, \exists u \in S, (u, v) \in E\}$. Let $U \subseteq S$ be the subset containing those neighbors whose addition to S doesn't violate the r-connectivity of S.[7] If U is empty, the semiclique growth process is stopped, a maximal semiclique has been found, and the candidate S is examined for addition to Q. If U is not empty, a vertex from U is picked at random and added to S. The random walk is constrained in the sense that only the vertices from U (and not all of $Neib(S)$) can be added to S. The tests needed to create U are performed efficiently, by keeping track of the number of connections of vertices (inside and outside of S) to vertices in S. An alternative to semiclique generation via a random walk is a deterministic (or "greedy") approach. In fact, our first implementation of mining for semicliques was a deterministic one: the terms in $Neib(S)$ are examined in the order of their degree, and the first to satisfy r-connectivity is picked. So larger but fewer semicliques may be obtained. With this option, we generate at most one semiclique per term. Under the random-walk option, the terms in $Neib(S)$ are examined in a random order, but semiclique generation is attempted many times. Thus many more (nonredundant) semicliques can be generated from a given seed, and for the whole corpus (*e.g.*, twice or more). We have obtained higher accuracy boosts with the substantially more semicliques generated via random walks (Section 4.2.2).

4.1.2 Addition to the Global Set

Once a maximal semiclique S is generated, there are basically two conditions to meet for adding S to Q. Our first condition is simply a minimum threshold on size (of 3), as we are interested in sizeable candidate topics. The other condition stems from the desire to avoid creating redundant topics. We utilize a set overlap measure to avoid redundancy:[8] for two semicliques S and S' (treated as sets of terms), $J(S, S') = \frac{|S \cap S'|}{\min(|S|, |S'|)}$. If the overlap exceeds the threshold of $o = 0.5$,

generation.

[7] Note a violation can occur if u is not connected to sufficiently many members of S, or if once added to S, another vertex $v \in S$ fails to be sufficiently connected.

[8] The Jaccard coefficient is another option: $J(S, S') = \frac{|S \cap S'|}{\min(|S|, |S'|)}$, but it yields somewhat redundant sets (*e.g.*, a semiclique could be (almost) a subset of another. An alternative way to achieve this goal is to compare the sets of documents that the semicliques tag.

L=3	excess 39 table 160 salt 55 sodium chloride 4 strongly 158 sodium chloride 4 react 50 table 160
L=10	symptoms 96 salt 55 causes 229 sodium chloride 4 table 160 strongly 158 excess 39 react 50 msg 75 negotiating 35 armenians 156 table 160 backers 4 brought 335 parties 125 israelis 120 negotiating 35 hostilities 8 table 160 secure 236 negotiating 35 table 160 attacking 90 israel 361 israelis 120 arabs 156 ...
L=40	folx 4 uncomfortable 37 chest 58 react 50 salt 55 excess 39 sugar 42 symptoms 96 table 160 (cont.) strongly 158 sodium chloride 4 rapid heartbeat 3 msg 75, ... negotiations 40 table 160 contains 232 territories 49 gaza 58 implemented 111 israelis 120 lebanon 73 table 160 negotiating 35 attacking 90 intervene 32 drawer 28 table 160 documents 125 armenians 156 turkey 123 signed 87 ...

Table 6: Example semicliques containing "table" from the newsgroups dataset, for several choices of L (Boolean window during graph construction, r=0.6). Each line is a single semiclique (unless it begins with cont.). Each term is followed by its document frequency. $L = 3$ yields fewer and smaller semicliques, and the terms in the small semicliques may not provide sufficient context to understand the meaning of the semiclique. We need higher L to discover many of the topics hidden in the data. We observe that there are at least two senses of "table" in the data ("table salt", and "negotiating table").

it is considered high overlap. Let D denote the set of those semicliques (already in Q) that have high overlap with the newly generated S. If D is empty or if S has more nodes than any member of D, S is added and the members of D are removed from Q. Otherwise, S is dropped (a failure). Computing intersections with semicliques in Q is achieved efficiently, by each term pointing to the clique ids (the members of Q) that it appears in (using a hash set). If S is not added to Q, the number of consecutive failures is incremented, otherwise, it is reset to 0.

4.1.3 Example Semicliques

We have observed that many of the semicliques found are quite interpretable. The terms are indicative of what the topics might be. Searching (grep) on familiar terms in a file of semicliques reveals many interesting and suggestive topics. In the newsgroup data set, the number of semicliques that had the word "table" in them for respectively L=3, L=10, and L=40 (Boolean window, $r = 0.6$), were 2, 8, and 29. See table 6. With $L = 3$, we get smaller and fewer candidate topics. Higher L is better suited to topics mining. We observe at least two senses of "table": one related to salt (table salt) and occurring in a health-related topic, and another surrounding conflicts and negotiations (*e.g.*, negotiation table). Examining documents that contain (most of) the terms in the topics verifies our expectations on what the candidate set might be about (health issues and conflicts).

Table 7 shows a few example semicliques containing the word "girl", and Table 8 shows a few example semicliques containing the word "clintons" in the Reuters dataset.

L=3	sadly 58 girl 86 small 885 terrible 108 head 544 her 705 struck 62 girl 86 boys 77 old 1249 boy 182 little 1615 girl 86 ...
L=10	serious injuries 9 sadly 58 five 324 several 1024 overpass 16 hit 489 girl 86 hair 79 child 287 she 660 girl 86 her 705 wife 175 ...
L=40	serious injuries 9 hit 489 kids 208 cool 179 rocks 66 seat 115 struck 62 drunken 20 five 324 girl 86 (cont.) overpass 16 sitting 183 throw 259 beltway nosed 8 sadly 58 .. (cont.) cars 400 washington dc beltway snot 8 .. ball 193 girl 86 hit 489 overpass 16 threw 95 pitches 58 civilians 117 families 75 girl 86 hundreds 152 armenians 156 women 381 cradled 3 slaughter 65 (cont.) innocent 213 children 596 massacre 98 fallen 66 ...

Table 7: Example semicliques containing "girl" from the newsgroups dataset, for several choices of L (Boolean window during graph construction, r=0.6). The number of semicliques containing "girl" was 6, 19, and 79 for L=3, L=10, and L=40 respectively.

4.1.4 Size of Output, and Timing and Complexity

The number and average size of semicliques generated are shown in table 9 for the newsgroups dataset (r was set to 0.6). As we increase the window size, the number and average length of the semicliques increases. This is expected as the degrees and egde-density grows in the co-occurrence graph with larger window size. Proximity weighting within window size L is comparable to Boolean weighting of roughly $L/4$, and deterministic (greedy) generation yields considerably fewer semicliques.

On the whole Reuters corpus, with $L = 5$, we obtain 446k semicliques (42 hours), with average size of 5.9, and with $L = 40$, more than 2 million semicliques are generated, with average size of 8 (100 hours). We note that further speed up of our implementation is possible. For istance, one could first enumerate all the cliques of size 3 and begin each clique generation iteration from a 3-clique chosen at random.

The time complexity of the generation algorithm is dependent on the average degree of the terms, and neighborhood sizes, within a radius of the seed term. Inside semiclique growth, the amount of work done is $O(|S| + d)$, where d denotes the degree of the most recently added term, and $|S|$ denotes the current size of the growing semiclique. Thus one run of maximal semiclique gernation takes $O(|S|(|S| + d))$ or $O(d|S|)$ (since $d > |S|$), where $|S|$ denotes the average semiclique size, and is in the 10s, while d can be in the 10s or 100s. We attempt about $O(k)$ such generation attempts, thus for $|V|$ terms, we obtain a complexity of $O(kd|S||V|)$.

L=5, 24 semicliques containing "clintons"
partners 14726 clintons 209 whitewater 297 scandal 5461 lady hillary rodham 225 mcdougals 34 legal 18095 tangled 180 whitewater 297 clintons 209 known 21489 failed 24163 mcdougal 76 clintons 209 starr 196 whitewater 297 former 53515 arkansas 1462 mcdougals 34 (cont.) clinton 10600 susan mcdougal 55 ...
L=40, 73 semicliques containing "clintons"
sentence 3601 clintons 209 affair 3095 daughter 3059 mother 3978 betrayed 334 her husband 1368 love 3091 starr 196 album 398 clintons 209 columbia 2303 aides 2857 clintons 209 blair 1851 aide 3284 wife hillary 231 dinner 1632 george bush 620 clinton 10600 palace 1901 majorca 44 castle 698 guests 1197 moorish 27 king 7698 clintons 209 queen 2370 hillary rodham 225 investigators 3155 clintons 209 prosecutors 4771 testimony 3223 testified 1210 kenneth starr 66 allegations 7267 (cont.) susan mcdougal 55 lawyer 5979 prosecutor 3986 perjury 205 probe 4748 whitewater 297 attorney 6859 (cont.) fbi 1559 investigation 11380 testify 1193 prosecution 2606 starr 196 prison 6356 fraud 4974 (cont.) lied 456 mcdougal 76 wrongdoing 1320 trial 10786 alleged 11100 conspiracy 1776 grand jury 578 ...

Table 8: Example semicliques containing "clintons" from the Reuters dataset. The number of semicliques containing "clintons" was 24 and 73 for L=5 and L=40 respectively.

newsgrps	L=3	L=10	L=20	L=20 (proxim)	deterministic (L=20)	L=40	L=40, $r = 1$	L = ALL
$\|Q\|$	11k	18k	44k	10k	5k	59k	23k	624k
avg. $\|S\|$	4.0	4.7	5.7	4.9	4	6	4.8	11

Table 9: Number of semicliques generated $|Q|$, and average size (number of terms), under several window sizes (Boolean window, and random walk, r =0.6, r-connectivity, and k=100).

4.1.5 Stability

The semiclique generation method is randomized, and we have observed that as we increase the number of consecutive failures k, the number of semicliques generated ($|Q|$) initially increases, and eventually stabilizes. However, k may need to be increased to 100s for stabilization, specially for a relatively sizeable window size L, such as $L = 40$. With $L = 40$, $k = 10, 50, 100, 200$, and 400, we obtain respectively 29k, 50k, 59k, 68k, and 75k semicliques. Thus the growth in $|Q|$ gradually stabilizes with increasing k. Another natural question concerns the stability of the structure of the semicliques produced. One question here is whether the topics produced from different runs are similar.[9] If the topics remain mostly the same or similar, this is evidence that the topics may be

[9]Note that due to our overlap constraint and randomization, we expect to obtain different sets of semicliques from one run to another. If all the maximal semicliques were created, or if the generation algorithm was fully deterministic, there would be no variation.

	$k = 100$	$k = 400$	r=1.0
$\tau \geq 0.9$	0.22	0.28	0.90
$\tau \geq 0.8$	0.36	0.46	0.91
$\tau \geq 0.7$	0.56	0.58	0.94
$\tau \geq 0.6$	0.79	0.84	0.96
$\tau \geq 0.5$	0.99	1.0	0.99

Table 10: Proportion of semicliques generated in one run (with $L = 40$), that had overlap τ greater than the given minimum thresholds ($0.9, 0.8, \cdots, 0.5$), with some semiclique generated from another run, on the newsgroups dataset, under different generation parameters k and r (number of consecutive failures k and r in r-connectivity), where $k = 100$ and $r = 0.6$ unless specified. Increasing k and in particular the connectivity r yields more stable semicliques.

special and significant in some sense (see also next next section 4.1.6). If they vary substantially, it may be that the algorithm discovers different "views" of the data each time, but it could also mean that the topics are not so "special".

Table 10 shows several stability scores, where stability is measured in the percentage of semicliques generated in one run which have overlap with some semiclique in another run exceeding a threshold. Formally, let Q_1 and Q_2 denote the set of semicliques generated from two runs (using different randomization seeds). Then for a given threshold τ (such as 0.9), let $Q_\tau = \{S \in Q_1 | \exists S_2 \in Q_2$, such that $|S \cap S_2|/|S| \geq \tau\}$. For each τ, we are reporting $|Q_\tau|/|Q|$. The reported proportions remain stable under several runs. We have observed that as k is increased and as we increase the connectivity ratio r (towards requesting pure maximal cliques), stability goes up. On the other hand, as L goes up, and/or if the average degree of vertices is increased, stability goes down (there are more ways to create r-connected semicliques).

4.1.6 An Analysis of Expectation and Statistical Significance of Semicliques

Here, we derive a bound on the probability of obtaining a (semi)clique of certain size, and the number of semicliques of such size that one might expect, under a random graph assumption. Therefore, assume the occurrence graph was a random graph, *i.e.*, given E edges, a random set of the vertex pairs (from $\binom{|V|}{2}$ possible pairs) is chosen and the pairs are connected. Let p denote the probability of a connection between a pair of vertices. Consider subsets of size s, from a total of $n = |V|$ nodes (vocabulary of size n). Given a set of s nodes, the probability of e edges (randomly occurring) among them is $\binom{\binom{s}{2}}{e}(1-p)^{\binom{s}{2}-e}p^e \leq s^{2\min(e,s^2-e)}p^e$ (where we used $(1-p) \leq 1$, and $\binom{a}{b} \leq a^b$). We can simplify further for pure cliques. There, there is only one combination, and the probability is $p^{\binom{s}{2}}$ ($e = \binom{s}{2}$). There are $\binom{n}{s}$ such s node subsets. Then the expected number of subsets (semicliques) with e edges is not greater than $\bar{C} = \binom{n}{s}s^{2e}p^e \leq n^s s^{2e}p^e$. We observe that $\bar{C}$ also bounds the probability that we obtain at least one such semiclique in a random graph. For the

case of pure cliques, we get no more than $n^s p^{\binom{s}{2}}$.

Given a total of E edges in the graph, with $\binom{n}{2}$ possible pairs to connect, with the assumption of random connections, we obtain: $p = \frac{E}{\binom{n}{2}}$. We have $\binom{a}{2} \geq a^2/2.5$, for $a \geq 5$. Thus, for cliques of size $s \geq 5$, we get $n^s p^{\binom{s}{2}} \leq n^s \frac{(2.5E)^{s^2/2.5}}{n^{4s^2/5}} = n^{s-4s^2/5}(2.5E)^{s^2/2.5}$. With $s = 5$, and from Table 3, using $n > 2.5 \times 10^4$, and $E < 10^6$ ($L \leq 40$), we obtain that an upper bound on the expected number of cliques of size 10 or above: $(2.5 \times 10^4)^{5-25}(2.5 \times 10^6)^{25/2.5} = 2.5^{-10}$. Thus, obtaining even one clique of size 5 or higher is highly significant, under the random graph assumption. Of the almost 23k cliques generated with $L = 40, r = 1$ (Table 9), 1300 cliques had size at least 10, and more than 7000 had size at least 5. For lower set sizes and when we relax to semicliques $r < 1$, we need to take more care in our approximations when providing the bounds. In most cases however, the expected number of semicliques obtained from a random graph is far smaller from what we obtain from mining the co-occurrence graph, as expected. The pattern of connectivity in the co-occurrence graph is far from random. We note that one could take into account the specific vertex degrees in assessing significance (those with high degree increase the probability that a (semi)clique containing them occurs in the graph, and lower the significance of such a grouping).

4.1.7 Discussion on Semiclique Generation

Imposing a fixed threshold on connectivity is a starting point, and improvements in several direction are possible. The choice of a fixed and relatively high threshold is motivated by simplicity and a desire to find relatively small and topically cohesive and specific groupings. Imposing a fixed threshold is somewhat arbitrary however: for instance, depending on the order the neighbors are examined, one of a pair of vertices may be added but not the other. As mentioned in the outset of this section, imposing hard (and non-arbitrary) boundaries may be difficult or impossible in many cases. The goal of creating more stable semicliques and/or term groupings that have soft boundaries should prove fruitful. Observation of the many semicliques generated (Tables 6 to 8) suggests that it should be possible to join a significant number of them, or design algorithms that create larger and more topically distant groupings. The challenge is striking a good balance, or providing an easily controllable knob, between generality and specificity of the generated topics, while avoiding the creation of semicliques that span multiple unrelated topics (that lose cohesiveness).[10] Inducing hierarchical groupings (in a specific to general manner, or vice versa) via graph mining is another relevant future direction.

We did not use the weights (conditional probabilities) in semiclique generation. Taking into account the edge weights (such as conditional probabilities, or the undirected intersection probabilities) at some point during topic generation may also be useful. Constraining based on the conditional probabilities may yield topics with terms that tend to be in similar tiers of the term frequency pyramid, but this depends on the details of how the edge weights are used in topic creation.

[10] Topicality and cohesiveness are, of course, fuzzy concepts.

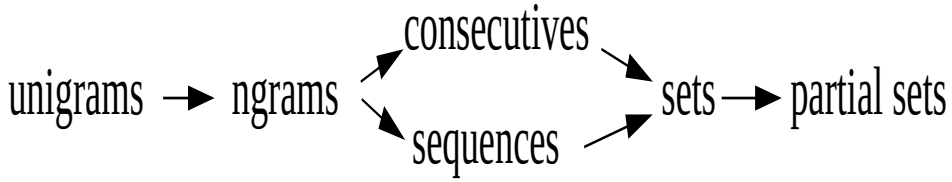

Figure 6: A hierarchy of regularities over text. From left to right, the constraints on the regularity is relaxed: in ngrams (most constained), the elements need to appear consecutively and in order, while sets require only that the elements appear within a defined (possibly small) region of text. Sequences and consecutives are obtained by relaxing either of consecutiveness or ordering. Partial sets are the least constrained: not all the elements of the set (but a sufficient subset) need to appear. One can envision repeating this composition (*e.g.*, to construct ngrams over sets), or to also learn distributions over various parameters.

Semicliques are just one kind of regularity discoverable from text. Another is statistically significant ngrams (that may reflect proper names, phrases, co-locations, common expression, etc). Figure 6 shows a hierarchy of a few kinds of regularities that one can discover from text, roughly from the most constrained (ngrams) to the least constrained. Constructing ngrams using unigrams as the elements (or lower-order ngrams) requires both consecutiveness of the elements (the unigrams need to be adjacent) and sequencing (the unigrams' occurrences need to satisfy a temporal order). Relaxing the sequencing (ordering) constraint yields "consecutives" (*e.g.*, "ab" or "ba", *i.e.*, "a" and "b" need to be adjacent, but their occurrence need not follow an order). Relaxing the consecutiveness but keeping ordering yields sequences (*e.g.*, "a" should come after "b", but not necessary immediately). Relaxation of both yields what we are calling *sets* in the above picture, where there is no ordering or sequencing constraints on the members, but all members should appear in a constrained vicinity. Finally, we reach what we have referred to as *partial sets*, where not all members of the regularity need to appear for the regularity to be detected. One can recursively repeat the construction, for example to look for ngrams over sets, and to obtain higher level regularities. Of course, the details of the constructs, such as what the extent of a region of detection of a regularity is (for the unconstrained regularities), are application dependent. Semiclique mining of the co-occurrence graph, as we have described, is just one way of inducing candidate sets and partial sets.[11] One can envision methods that, similar to ngram extraction, require a few passes over the corpus, and mine for statistically significant sets of co-occurring terms.

[11]Note that not all terms of a discovered semiclique are guaranteed to appear in a single document. Semicliques are only suggestive: that a significant number of documents may contain a sufficient fraction of the terms in the semiclique.

4.2 Tagging Documents with Topics

We seek measures of the relevancy of a topic corresponding to a semiclique to a document. Intuitively and simply, the higher the fraction of the members that appear in the document, the more relevant the topic is to the document. There are other considerations: perhaps the terms in a semiclique should be treated differently, and the frequency of those terms in the document and where they occur in the document should be factors too (*e.g.*, do they appear close to one another?). Since we constrain the members of semicliques to connect to over 50% of others, degree of connectivity within a semiclique, and the edge weights, may not be an important factor.

Consider the following semiclique discovered from processing newsgroups:

```
(girlfriend,37)(wife,205)(she,477)(her,510)
```

We believe that all the terms in the semiclique add some meaning to the suggested topic or reinforce the meaning, and thus are useful. It is, however, evident that the first two terms are more important in constraining or grounding the meaning of the semiclique than the last two more general terms. The df (document frequency) numbers reflect this importance. In Section 4.2.1, we will also see that taking frequency into account when tagging can be useful.

For determining the tag value of topic S on a document d (both treated as sets), we simply consider the terms in the intersection of the document and the topic ($d \cap S$). Thus, we ignored factors such as the location of the semiclique terms in the document and the connectivity or edge weights within the semiclique. We experimented with Boolean (plain) weighting, two kinds of log (document) frequency weighting, and inverse document frequency weighting. The tag value is given by:

$$\frac{\sum_{v \in d \cap S} f(v)}{\sum_{v \in S} f(v)}, \tag{2}$$

where $f(v) = 1$ for Boolean weighting, $f(v) = \log(N/df(v))$ for log inverse document frequency, $f(v) = 1/\log(df(v))$ for inverse log document frequency (inv. log frequency, our default choice), and $f(v) = 1/df(v)$ for (plain) inv. (document) frequency weighting, where N is corpus size (number of documents).

Boolean tagging does not distinguish among the terms (ignores the informativeness of the terms), and we find that it can yield inferior performance results compared to the other tag weighting methods in some of our classification experiments. In terms of bias or sensitivity to term frequency, (plain) inv. frequency tagging is the most sensitive (most biased in favor of the lowest frequency terms in the semiclique). Next is inv. log frequency, *i.e.*, using the $f(v) = 1/\log(df(v))$ formula, followed by log inv. $f(v) = \log(N/df(v))$ (as N dampens the sensitivity to df), and then Boolean tagging (which is insensitive). For a given threshold on tag value (*e.g.*, 0.1 or 0.5), Boolean tagging yields significantly more tags per document on average than the other tag weighting methods, as expected. For example, on the newsgroups dataset, with minimum tag threshold value of 0.1, we obtain 28 topics tagging a document on average, and 92% of the documents have

	0.1	0.3	0.5	0.7
Boolean	72 (98%)	66 (98%)	6.8 (71%)	4.5 (59%)
$\log(N/df)$	70 (98%)	30 (94%)	8.9 (78%)	3.7 (55%)
$1/\log(df)$	70 (98%)	28 (93%)	8.6 (78%)	3.7 (55%)
1/df	28 (76%)	14 (76%)	7.2 (76%)	3.9 (60%)

Table 11: Average number of tags per document with different minimum thresholds for various tag weighting methods, with a requirement of at least two terms in the intersection ($|d \cap S| \geq 2$), on the newsgroups dataset ($L = 20$, proximity weighting). The percentage of documents with at least one tag is shown in parentheses. We used the 10k semicliques generated with minimum size requirement of 3 terms, connectivity of 0.6, and maximum overlap allowance of 0.5 ($N = 2000$). As we move from top row to bottom (from Boolean to inv. df weighting), the tag counts and coverage numbers decrease in general, but there are exceptions.

at least one tag, while with Boolean tagging, the average number of tags per document goes up to 72 per document and almost all documents (98%) have a tag. Table 11 shows the tagging statistics for a few parameter settings on the newsgroups data set (semicliques generated using $L = 20$ and proximity window). With Boolean window, and/or with increasing L, the number of documents tagged, and the average number of tags per document, increase as expected. Using semicliques generated from Boolean window and $L = 40$, the average number of tags per document is 300 (min value of 0.1), and 98.4% are tagged.

4.2.1 Boosting Classification Performance

Labeled data has been used in various ways to assess the performance of unsupervised methods (when external commonly agreed criteria are often unavailable). For instance, in document clustering, the clusters induced are sometimes compared to human categorization of the dataset. While the goal(s) in unsupervised tasks are often not completely in alignment with the supervised tasks, such comparisons can provide further indirect but more objective evidence that certain algorithmic choices are beneficial. Here, we use the induced semicliques as feature to *augment* the standard tfidf feature set. We used linear SVM training in our experiments (liblinear [6]). We find that the classification performance improves and, importantly, we obtain further insights into the utility of choices made in graph and semiclique generation, and in tagging.

If semicliques capture many of the topics discussed in a corpus, we expect that some such semicliques would align well with the often high-level classes assigned to the documents by human labelers (“sports”, “politics”, etc). By “aligning” well, we mean that a considerable portion of documents that a semiclique tags would belong to one or relatively few classes. Not all semicliques may align well with the high-level topics assigned by humans. Of those that do, they may not tag sufficiently many documents to add extra value over the standard features (the terms), or they may also add redundancies and noise. We find here that the semicliques do indeed improve

accuracy consistently, in particular at high proportion training set sizes, where the standard bag of words features are very difficult to beat when using SVMs. This is additional evidence that the semicliques are capturing useful regularities in the data.

We now describe our default experimental setting. We used tfidf values for the original term features (which gave the best performance and was hardest to improve upon, compared to Boolean feaure values, as well as raw frequency values). The terms were from the same vocabulary (up to fourgrams) that semicliques were built on. For co-occurrence graph construction, we used (by default) $L = 40$ (and Boolean window), $r = 0.6$ for connectivity, and minimum edge weight of 0.01 for semiclique generation. We used inverse log df weighted tage values by default and compared to plain Boolean score (Equation 2). To construct an instance vector, we kept only those topic features with value at least 0.1, and having at minimum two members in common with the document. We scaled the clique feature values by 10 (discussed below). We then L2 normalized the feature vectors, for SVM training. The regularization parameter C for the SVM was set at $C = 10$. We report on the Max F1 performance score.[12]

4.2.2 Table of Newsgroups Results

Table 12 shows the performance gains when we augment with semiclique features, under two training proportion on the newsgroups data. We note that out of 20 comparisons in each trial, respectively 14, 15, and 16 wins or more are significant at the $p \leq 0.1$, $p \leq 0.05$, and $p \leq 0.01$ confidence levels in pairwise sign tests. All our performance results are averages over 10 trials. If we sum the the number of wins over the ten train-test splits, the confidence level goes to $p \leq 0.01$. The table also suggests that increased window size and the use of inverse log df for feature tags boost the performance gain. The choice of threshold on minimum tag value can be important. Especially for Boolean tagging, relatively high threshold is required to get good results. Too many tags (features) per document may overwhelm the learner. Sign tests confirm the significance of the difference between $L = 3$ vs $L = 40$ and between Boolean and inv. log df tagging. For example, for all the 10 trials, the average Max F1 (averaged over the 20 classes) for inv. log df tagging was higher than that of Boolean tagging (which leads to significance of $p \leq 0.001$). We took the best threshold on tag value for each tagging strategy (0.3 for inv. log df, and 0.7 for Boolean).

The average of Max F1 over the 20 classes, without the semiclique features, at 80% training portion was 0.87, thus an improvement of 1% represents about 10% reduction in the error. We observe improvements in both precision and recall at the Max F1 score, *i.e.*, the improvement in Max F1 score due to augmenting with semiclique features seems not to be confined to increase in precision only or recall only. The number of wins for either recall or precision was lower than the number of wins for the combined Max F1 score.

As noted in Section 4.1.7 (see Figure 6), both ngrams and semicliques represent kinds of regularities in the data. The kind of small semicliques that we discover are very close to the level

[12] Max F1 is the harmonic mean of precision and recall [15].

Training Portion →	0.8		0.5	
L and min tag value ↓	# Wins (std)	# Boost (std)	# Wins (std)	# Boost (std)
$L = 3$, 0.1	12(±2)	0.26 (± .002)	11.9 (±3)	0.24(±0.16)
$L = 3$, 0.3	13.1(±1.8)	0.43 (±0.16)	13.6(±2.5)	0.37 (±0.2)
$L = 3$, 0.5	14.1(±1.1)	0.40 (±0.07)	16.4(±1.5)	0.40 (±0.1)
$L = 3$, 0.7	12.5(±1.9)	0.31 (±0.10)	16(±0.77)	0.37 (±0.04)
$L = 40$, 0.1	14.7 (±1.7)	0.88 (±0.2)	16.6(±1.3)	1.0(±0.14)
$L = 40$, 0.3	**17.3 (±1.4)**	**1.0 (±0.16)**	**18.4(±1.0)**	**1.1(±0.18)**
$L = 40$, 0.5	16.2 (±1.4)	0.70 (±0.10)	16.9 (±1.3)	0.73(±0.11)
$L = 40$, 0.1 (bool values)	12 (±1.7)	0.46 (±0.20)	11.6 (±1.8)	0.32 (±0.15)
$L = 40$, 0.3 (bool values)	12.3 (±1.2)	0.52 (±0.18)	12.7 (±1.3)	0.43 (±0.16)
$L = 40$, 0.5 (bool values)	14.5 (±2.0)	0.69 (±0.16)	14.9 (±2.0)	0.65 (±0.17)
$L = 40$, 0.7 (bool values)	14.6 (±1.5)	0.52 (±0.09)	15.5 (±2.2)	0.48 (±0.09)
$L = 40$, 0.9 (bool values)	13.3 (±1.3)	0.28 (±0.09)	12 (±2.1)	0.24 (±0.07)
$L = 20$, 0.3	15.8 (±1.9)	0.90 (±0.16)	17.8 (±1.2)	0.8 (±0.11)

Table 12: Performance boosts on newsgroups, with approximately 20k docs and 20 classes. The results are averages over 10 trials. In each trial, a random 80% or 50% of documents are kept for training, rest are held out, and avg of Max F1 is computed, with and without the semiclique features.

of ngrams, in that they are specific, in the sense that they tag (or "occur") in relatively few documents.[13] In one experiment, we noted that the addition of bigrams and then trigrams and fourgrams each improved Max F1 score over just using unigrams. At training proportion of 0.5, the overall average boost was 0.5 (from 0.836 to 0.841) when adding up to fourgrams to unigrams. We conclude that the addition of semicliques (on top a vocabulary that includes up to fourgrams) yields another unit of boost on average.

Table 13 displays a few variations to the default setting. The general pattern when we vary parameters such as r-connectivity, minimum edge weight for semiclique discovery (0.01 by default) or PMI, was smooth change in performance as a function of the level of change in parameter. r=0.6 seemed to give best performance, compared to both higher r and lower r.[14] In general, the more semicliques generated (preserving nonredundancy), with increase in L, the higher the performance boost. We did not search for best L, but setting L to whole document still seemed to provide a little improvement over $L = 40$, and Boolean window appeared better then proximity window (for the same L value) in these tests, again the main cause maybe due to the large number of semicliques generated. It appears that both small and large semicliques add to the performance boost.

[13]They can even be regarded as more specific, since we require a sufficient number of the members to appear in a document. This depends on the details of the tagging procedure.

[14]A lower $r = 0.5$, yielded fewer but larger semicliques. We didn't experiment with lower r values, as the connectivity condition on the semicliques does guarantee cohesiveness.

Training Portion →	0.8	
min size 6	15.3 (±1.7)	0.69 (±0.16)
max size 5	15.7 (±1.5)	0.58 (±0.12)
r=0.5	15.5 (±1.1)	0.59 (±0.07)
r=1.0	14.3 (±1.6)	0.65 (±0.2)
SVM C=1	15.5 (±1.2)	0.82 (±0.013)
Deterministic gen.	14.4	0.37
Scale 1	3.5 (mostly tied)	0.04
Scale 20	13.7 (±1.7)	0.8 (±0.19)
Scale 100	1.4 (±1.7)	-3.3 (±0.4)
All doc. window, r=0.6	**17.3** (±1.3)	**1.2** (±0.17)
All doc. window, r=0.7	15.7 (±2.1)	0.72 (±0.15)

Table 13: Number of wins and performance boost under several changes to the default setting (default: Boolean window, L=40, r=0.6, k=100, scaling semiclique feature values by 10, C=10 for the linear SVM regularization parameter). Top two rows show the effect of dropping semicliques with size less than 6 and size greater than 5, respectively. Next two rows report on changing connectivity. Deterministic gen. refers to the deterministic generation of semicliques (Section 4.1.1). When we scale the semiclique feature values by 100 (effectively removing the influence of the tfidf term features), we observe noticeable underperformance. The regular term features are also needed for best performance. Increasing window size to the whole document keeps the performance boost.

Scaling the semiclique feature values (default of 10) yields superior results than no scaling, but scaling them too much, in effect nullifying the baseline tfidf features, leads to degradation. This may be due to the possibility that semiclique features do not sufficiently cover, that is have a significant tag value for, all the documents.

Other variations we tried included setting the min edge weight thershold to 0 for semiclique discovery (led to no significant change in Max F1 performance) or increasing to 0.05 and 0.1 (gradual degradation). Our initial default tagging strategy was the use of (plain) inverse frequency, though with the right threshold, we found some improvement using the inv. log frequency variant (the latter is not as highly sensitive to the frequencies). However, we did not conduct systematic statistical tests.

Figure 7 shows the Max F1 performance curves, on class 1, alt.atheism, from newsgroups, where each point is the average of 10 trials. We observe the consistent superior performance of the augmented feature set, *i.e.*, when using semiclique feature values together with the standard tfidf term features.

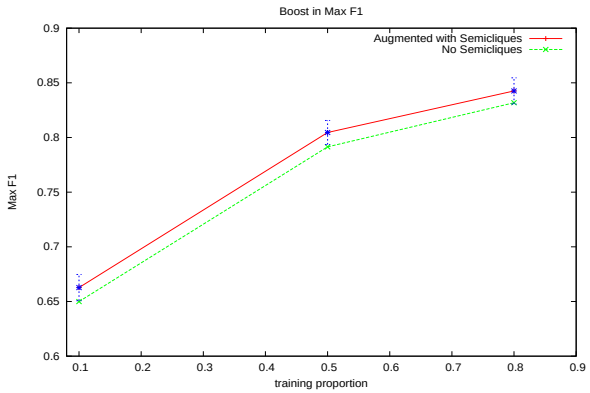

Figure 7: Max F1 performance for a few training portions, when using standard tfidf features versus tfidf features together with semiclique features, on alt.atheism from newsgroup. Semiclique features improve performance even at larger training sizes. The performance boost does not seem to change, for the range of proportions tested. Each point is the average of 10 random splits of data.

4.2.3 Results on Reuters

In the Reuters data set the documents have been labeled by several hundred industry, topic, and geographic codes [18]. The class distribution is skewed, and a document can belong to multiple categories, unlike newsgroups. We ran our experiments on the top 30 (most frequent) and bottom 30 of the topic categories (out of just over 100 topic categories). To save time, we limited our experiments to a random sample of 40k documents (out of the total 800k). There was no noticeable performance boost on the bottom 30 (mostly ties), but we observed average boost exceeding 1% on the top 30 (similar to newsgroups), and the number of wins (average of 25 and higher, out of 30) is highly significant. See Table 14. In the top 30, the most frequent class labeled just under 50% of the documents, and the lowest labeled around 2.5% of the documents (over 1000 documents). In the bottom 30, the most frequent labeled only 0.3% of the documents (130 documents or fewer), and the least frequent labeled less than 10 documents. The very small number of documents may explain the no performance change results for bottom 30.

We observe here that increase in L has some effect in performance boost, but different strategies for tag value (inverse log df versus Boolean) appear to yield similar performance boosts. The best threshold on (inv. log. df) tag value was obtained at the minimum threshold of 0.1, and it's possible that lowering this threshold may improve results somewhat. We saw both improvements in precision and recall at the Max F1 score, but the number of wins for either recall or precision was lower than the number of wins for the combined Max F1 score. We experimented with using semicliques derived from processing only the 40k docs, or from processing all the 800k docs. We didn't note a noticeable different in boost or number of wins using either.

Top 30 topical classes of Reuters	0.8 Training portion	
$L = 5$, 0.1	25.9 ($\pm$1.8)	0.91 ($\pm$0.10)
$L = 5$, 0.3	25.5 ($\pm$1.2)	0.87 ($\pm$0.08)
$L = 40$, 0.1	26.5 ($\pm$1.7)	1.16 ($\pm$0.19)
$L = 40$, 0.3	25.8 ($\pm$1.9)	0.94 ($\pm$0.17)
$L = 40$, Bool 0.1	26.8 ($\pm$1.2)	1.215 ($\pm$0.20)
$L = 40$, Bool 0.3	26.7 ($\pm$1.6)	1.31 ($\pm$0.19)
$L = 40$, Bool 0.5	27 ($\pm$1)	1.15 ($\pm$0.15)
$L = 40$, Bool 0.7	22.8 ($\pm$2.6)	0.39 ($\pm$0.08)

Table 14: Number of wins and performance boost on top 30 topic classes on a subset of 40k documents of Reuters, as before, averaged over 10 trials of 80-20 splits (see Table 12). The average Max F1 over the 30 classes without the semiclique feature was 0.80.

4.2.4 Comparisons with Latent Dirichlet Allocation

The latent Dirichlet allocation (LDA) modeling technique assumes roughly that each document is generated by a mixture of topics, from a set of t topics, and the LDA task is to recover such hidden topics. One attraction of the approach, similar to ours, is that a document need not belong to a single topic. We explored using LDA to generate topics and compared boosts in performance.[15] The topics were learned on the same feature representation (*i.e.*, up to fourgrams). As in the above, we augmented the feature set (tfidf), with the topic features. Topic feature values, which are probabilities in $[0, 1]$, were scaled by 10 in the instance vectors, as was done above for semiclique features, to have an effect. We experimented with higher and lower scaling, and 10 was best (more on scaling below).

Several performance comparisons are given in Table 15. We experimented with different number of hidden topics t ($t = 20, 40, 100, 200, 300, 600$) on newsgroups. We observe that at higher training proportions, semiclique features yield higher performance boosts. With increasing t, the increase from use of LDA features also appears to grow, supporting our observations that the more topics, the better. However the time complexity of creating the LDA topics becomes prohibitive when we increase t. We ran $t = 30$ up to 60 iterations (for smaller t, under 100 iterations was sufficient), and for $t = 600$, we ran only 10 iterations (each iteration takes longer with increasing t). The poor performance of $t = 600$ suggests that a reasonable number of iterations (10s) is needed for significantly better performance. At smaller training proportions, the the fewer LDA features show more benefit. We experimented with scaling of 1 and 100. Both yielded inferior performances. Note that when scaling by 100, the effect of the tfidf features is close to 0. We found that at very low training proportions of 2% and 5%, scaling by 100, and using numbr of topics fewer than t=100, does improve average Max F1, respectively by 3.0 and 7.0 at 5% and 2%

[15] We used the version written in C, available from Blei at cs.princeton.edu/ blei/lda-c/index.html [1]

Training Portion →	0.8	0.5	0.1
	Newsgroups		
semicliques	17.3, 1.0	18.4, 1.1	18.4, 1.4
LDA t=100	11.8, 0.15	14.6, 0.45	18.8, 2.1
LDA t=200	12.2, 0.16	14, 0.35	16.6, 1.10
LDA t=300*	13.3, 0.3	14.9, 0.41	18, 1.29
LDA t=600*	11.6, 0.12	14.4, 0.25	17, 0.45
Reuters Subset (40k docs), top 30 Topic classes, proportion 0.8			
semicliques	26.6, 1.2		
LDA 200	21.8, 0.18		

Table 15: Improvements (number of wins and absolute increase in Max F1), via augmenting tfidf features, from using LDA topics, or semiclique features, under different trainng proportions (0.8, 0.5, 0.1). The average number of wins in 10 trials (out of 20 topics for newsgroups, 30 for Reuters subset), and the improvement in Max F1 is given. Semiclique features, being closer to the term level, tend to sustain the improvement edge at higher training portions. LDA with fewer number of topics, is better at lower training portions. LDA with t=300 and 600 were stopped respectively after 60 and 10 iterations, while others were run till completion (just under 100 iterations).

training, compared to only using tfidf features.

On the Reuters dataset, on the bottom 30 least frequent topical categories, LDA features did not show any performance boost, similar to the case for semicliques. We only tested LDA at number of topics t=200 on Reuters, due to the time constraints, and on the top 30 topical categories. We again observe an advantage with semiclique features (Table 15).

In the process of recovering the hidden topics via LDA, parameters are learned for each topic and term pair. Thus, unless sparse LDA variants are developed or the vocabulary is substantially reduced, the memory consumption for large data sets with 100s of thousands of terms and number of desired topics exceeding thousands, becomes very costly. The required runtime also grows as might be expected. Each iteration of variational EM took around an hour for $t = 100$ topics, and increased to close to four hours for $t = 600$ on the newsgroup data. The percent improvement in document likelihood fell below 1% only after 10s of iterations. We allowed "convergence" (EM likelihood improvement dropping to 0.0001) for $t \leq 200$ (which took several days for t=200), on the 20k and 40k documents of newsgroups and Reuters respectively.

4.2.5 Discussion of Tagging and Classification

We studied boost in performance of classification and observed the relative benefits of a few strategies for semiclique creation and tag scoring in this setting. Observe that in our experiments, the number of classes is relatively few (20 for newsgroup, and about 100 for Reuters), and the classes are relatively general. The task imposes its own bias on the techniques most appropriate. We an-

ticipate that if somewhat broader semicliques are discovered, the supervised learning performance may attain additional boost. As we increased window size L, and thus as average semiclique size and number of cliques increased, we observed increased performance.

Our simple intersection based tagging score increases in tag value roughly linearly as the number of terms in the intersection of the document and semiclique is increased (for simplicity, assume Boolean tag scores). This strategy is akin to requiring a conjunction (an "AND") or a majority occurrence. For relatively sizeable and broader semicliques, for example one with say 30 terms, it is counter intuitive to require that a majority of the member terms need to appear in the document before the tag value (the topical relevance) exceeds 0.5. A more lenient scoring mechanism, more an akin to a disjunction, that rapidly rises in value when a critical but relatively smaller intersection size is reached may be more appropriate once a semiclique's size surpasses a handful of terms (a sigmoidal shape for the scoring function). One possibility to achieve this is, roughly, to assume the topics are discovered by an efficient co-occurrence graph mining technique, but then to impose a generative model, the term weights (the parameters) for such topics (that determine the tagging scores) are learned via a technique akin to an LDA inference.

5 Related Work

Term co-occurrence (or co-location) relations find a number of applications in diverse tasks, such as semantic distance computation, synonymy, query expansion, and so on (*e.g.*, [5, 2, 12, 21, 16, 23, 3, 19, 7]). Our work focuses on studying the application to fine-grained topic discovery, and spans several dimensions of exploring graph structure, topic extraction, tagging, and topic utility as features in supervised learning. In this context, we precisely defined semantics of edge weights and explained some of the effects of various types of edge removal. We highlighted the utility of the pyramid (layered) view of the generated graphs and subgraphs. The use of inverse document frequency (idf) weighting of terms is wide spread in information retrieval, such as in improving relevance and similarity scores [22, 17]. For an account of the observation that the more frequent the word, the more meanings it tends to have, as well as possible causes for such phenomena see for example [14]. See also [20] who use pairwise associations and term frequencies for discovering subsumption (IS-A) relations. For synonymy discovery, smaller window sizes (less than 5) have proved most useful [19, 7]. We find here that reducing window size L lowers the number and coverage of the topics found. An application to query disambiguation by grouping related words at query time is described in [23], though their termp grouping is not based on edge density, and the edge weight semantics as well as the evaluations are different.

LSI is an elegant matrix method based on term-document co-occurrence patterns [4], and together with the probabilistic extensions, these methods have become very versatile and are well-studied (*e.g.*, [8, 1]). As we saw with LDA, these methods appear to be more appropriate for discovering relatively few (low 100s) and relatively broad topics, and interpretation and efficiency can be an issue. We aimed at finding many relatively specific topics. An interesting direction is

designing effective methods that span the two extremes. Increasing window size (which eventually becomes whole document), and relaxing topic connectivity (we required 50%) may push the semicliques towards broader themes. Speeding up LSI and related techniques has been a research challenge. Techniques such as random projections (*e.g.,* [19]), have been shown to be effective, while introducing some approximation. We view such methods as largely-orthogonal, designed for computing *semantic similarity*, when words are represented as feature vectors (the initial dimensions may be documents, or terms with proximity weights as computed here).

There is substantial work on computing semicliques (or quasicliques), dense subgraphs, clustering, and communities in networks. Here, it is very important to allow for overlapping groupings to support multiple senses and events, unlike much (graph) clustering work. On the other hand, unlike some applications [11], we do not require enumerating all the maximal semicliques. As we mentioned, the co-occurrence graph of terms is best viewed as hierarchical, reflecting term frequencies, not flat.

6 Summary and Future Directions

We presented techniques for computing graphs of co-occurrence relations, wherein edge weights represent conditional probabilities of term presence within a window. We explored extracting dense subgraphs, in particular maximal semicliques, as candidate topics. We showed the utility of the layered pyramid view of terms in aiding the interpretation of topics and understanding the algorithms for graph generation and topic discovery/tagging. The proposed approach yields numerous fine-grained topics that tend to be easily interpretable collections of terms. We investigated topic tagging and showed that the topics could augment the standard features to improve supervised learning performance.

There are many directions for future work. We assumed a static corpus, while in many realistic situations new documents are added over time and old documents may not be useful anymore. Online updates to the edges of the graph, or periodic graph reconstruction may be needed. We explored associations based on simple proximity, ignoring other potentially useful but more costly structural (*e.g.,* syntactic) information. As discussed, the semiclique discovery and tagging techniques presented should be extensible in a number of ways, for instance, to create more stable and/or possibly broader topics. Finally, recovering and presenting the possible relations among the terms in the discovered semicliques, *i.e.,* what the associations may correspond to (*e.g.,* who, what, how, when, ..), in part via natural language processing, is another promising direction.

Acknowledgments

Many thanks to Shahin Saadati for his assistance in using Java, Wen Wang for providing the TDT5 data, and to Erik Yeh and other AIC staff for discussions and suggestions. This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under

Contract No. FA8750-07-D-0185/0004, in the context of research on an adaptive cognitive system (CALO), under DARPA's PAL (Perceptive Assistant that Learns) program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Air Force Research Laboratory (AFRL).

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[2] P. F. Brown, V. J. Della-Pietra, P. V. deSouza, C. J.Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[3] I. Dagan, L. Lee, and F. C. Pereira. Similarity-based models of word cooccurrence probabilities. In *Machine Learning*, pages 34–1, 1999.

[4] S. Deerwester, S. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 1990.

[5] L. Doyle. Indexing and abstracting by association. System Development Corporation, Unisys Corporation, 1962.

[6] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9, 2008.

[7] D. Freitag, M. Blume, J. Byrnes, E. Chow, S. Kapadia, R. Rohwer, and Z. Wang. New experiments in distributional representations of synonymy. In *CoNLL*, 2005.

[8] T. Hofmann. Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI*, 1999.

[9] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[10] LDC. Topic detection and tracking. In `http://www.ldc.upenn.edu/Projects/TDT/`, 2005.

[11] J. Li, K. Sim, G. Liu, and L. Wong. Maximal quasi-bicliques with balanced noise tolerance: Concepts and co-clustering applications. In *SIAM Data Mining*, 2008.

[12] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behaviour Research Methods, Instruments, and Computers*, 2, 1996.

[13] O. Madani and J. Yu. Discovery of numerous specific topics via term co-occurrence analysis. In *ACM Conference on Information and Knowledge Management*, 2010.

[14] D. Manin. Zipf's law and avoidance of excessive synonymy. *Cognitive Science*, 32(7), 2008.

[15] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[16] M. Patel, J. A. Bullinara, and J. P. Levy. Extracting semantic representations from large text corpora. In *Fourth Neural Computation and Psychology Workshop*, 1997.

[17] S. Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60, 2004.

[18] T. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 - from yesterday's news to tomorrow's language resources. In *Third International Conference on Language Resources and Evaluation*, 2002.

[19] M. Sahlgren. Vector-based semantic analysis: representing word meanings based on random labels. In *In Semantic Knowledge Acquisition and Categorisation Workshop*, 2001.

[20] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *SIGIR*, 1999.

[21] H. Schutze and J. O. Pedersen. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management: an International Journal*, 33, 1997.

[22] K. Spark-Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28, 1972.

[23] A. Veing and P. van der Weerd. Conceptual grouping in word co-occurrence networks. In *IJCAI*, 1999.