Vector-based Ranking Techniques for Identifying the Topical Anchors of a Context

Aditya Ramana Rachakonda
aditya.ramana@iiitb.ac.in

Srinath Srinivasa
sri@iiitb.ac.in

International Institute of Information Technology
Bangalore, INDIA

Abstract

Terms in textual documents tend to occur more in contexts to which they are related. We can exploit this bias by modelling terms and their cooccurrences as graphs, resembling the abstract semantic memory structures in human beings. Given a context in the cooccurrence graph, we identify the topical anchors of the context. A context is defined as a collection of terms and topical anchors are those terms whose semantics represent the topic of the whole context. In graph-theoretic terms, a topical anchor is the node which is central to the context subgraph.

In this work we propose different vector-based ranking techniques and rank the nodes in a subgraph to identify topical anchors. Experiments were conducted on 100 distinct human defined contexts. The topical anchors generated were evaluated against those chosen manually by 86 volunteers and the results show that the algorithm correctly identifies the topical anchors 70% of the time.

1 Introduction

With advances in information retrieval (IR) it is not sufficient if the systems process text based on its lexical properties. IR systems are modelled to be efficient at solving lexical problems like identifying that *cars* and *car* are similar. But the humans interacting with the system are adept at handling semantic similarity and expect the machine to do so. It is common to find search queries like *best budget DSLR* or *good tourist destinations around Melbourne*. As semantics start playing an important role in human machine interaction, traditional IR systems that look for just terms in a document will be found wanting.

One of the reasons at the core of this problem is the assumption that terms in a document are independent of each other. Most IR systems simplify a document by looking at it as a bag-of-words or as a point in a vector space. It has been acknowledged that occurrence of terms in documents is not random and the cooccurrence properties of terms can significantly improve the quality of results [6]. Cooccurrence patterns in free text are a major source of semantic relatedness without recourse to linguistics.

15th International Conference on Management of Data
COMAD 2009, Mysore, India, December 9–12, 2009
©*Computer Society of India, 2009*

We addressed [20] a specific problem in mining semantics – that of identifying *topical anchors*. Topical anchors are terms which are central to a context and are used as representatives of a context when discussed in other contexts. For example, for the sentence, *"Brawn and Jenson Button celebrate Sunday's championship victory"*, the terms *Formula 1* and *motor racing* act as the words which best represent the context of the sentence. So, *Formula 1* should be the topical anchor of the sentence. Another example: A human reading a question like, *"What are the symptoms of type 2 diabetes and do oral insulin drugs have any side effects on the health of my pancreas?"*, can immediately understand that the question is posed in the *medical* context. In our work we primarily focus on finding the topical anchors like *medicine* given just a few words from a context to aid machine understanding.

Topical anchors are also useful in semantic search technologies where it is important to identify the context of a text or a document. For instance, automatic labelling of customer complaints with suitable topical terms, so that the complaints can be directed to the appropriate department, is a pressing need for many companies that offer web based support for their products. Similarly, as suggested in the example before, topical anchors can be used in online forums to determine the topic of a discussion thread and classify the thread accordingly – even if the topical anchor itself never appears in the thread.

To address this problem we borrow several concepts from cognitive sciences and IR. Cognitive linguists argue that the meaning of a term is not a property of the term but rather a property of the contexts in which it is used [9, 15]. A lexical term acts as a point of access to several associated semantic relationships with other terms[1]. Also, it is believed that the semantic memory in human beings is formed due to the co-activations of concepts in our mind [9, 12]. Latent Semantic Analysis [6, 3] and other semantics mining algorithms relate the cooccurrences of terms to certain semantic relationships.

Based on these models we build a cooccurrence graph, whose nodes are the terms and whose edges represent cooccurrence, as the basic data-structure for semantic relatedness. In our earlier work we posed the problem of topical anchors as a random walk on the cooccurrence graph and in this paper we extend the random walk using history vectors which are explained in detail later (see section 4). For the sake of completeness, the problem definition and the initial modelling have been included again in this paper briefly.

2 Related Literature

The problem of finding information which is representative of a context has been addressed in several domains. Cluster labelling algorithms identify words which could be used to represent a set of documents. Ontology discovery techniques attempt to identify the structure of the context by looking at several documents from that context. Text summarisation addresses the problem of finding the sentence which represents a document the most. In this section we will discuss some of the existing techniques in these areas and provide a motivation for finding topical anchors in text.

2.1 Ontology Discovery

Extracting structural semantics (ontology) from documents is very useful in determining the context of a document. There are several works on the discovery of ontology in free-text. They assume that the information in the corpus is structured into several topics and sub-topics, and the structure is available at least partly hand-crafted. They are usually run on a small and static set of documents. Most of these approaches utilise the help of a (human) Ontology Engineer [11, 14, 21] to verify and edit the ontologies. Nevertheless, they are automated to a large extent and perform considerably well on a small document corpus. Some techniques use noun-verb combinations in sentences to extract the underlying structure. ASIUM [11] is an ontology extractor which builds a bipartite graph of nouns and verbs to cluster related nouns based on their verbs. Reinberger et al [21] have proposed an algorithm which identifies whether a noun is a part of the subject or the object and uses that information along with the cooccurring verb to identify similar nouns.

Another method of identifying structure is to cluster concepts based on their similarities. Clerkin et al [4] have used the hierarchical concept clustering algorithm, COBWEB [12], to detect ontologies. It is an incremental algorithm which takes feature vectors as input and determines the best possible taxonomic organisation among all possible outcomes. Apart from these, tools like OntoEdit [14] are also available which act as a workbench for the Ontology Engineer. It learns an initial ontology based on a dictionary, and then uses this knowledge to help her merge or correct ontologies that are automatically extracted.

The drawback of *ontology discovery* techniques is the presence of an Ontology Engineer and in some cases the requirement for a partial taxonomy of the context. The completely automated algorithms are useful only for small contexts and generally are not scalable to include data from varied fields like the Web.

2.2 Cluster Labelling

Cluster labelling algorithms group a set of documents based on a similarity score between them and then identify phrases which are representative of the cluster. They pick commonly occurring phrases in the documents and compute a score for each phrase.

Geraci et al [13] compute the information gain as a phrase score, based on the occurrence or the non-occurrence of a phrase in a cluster. The information gain metric has a high value for a phrase whose occurrence is limited to a cluster. If a phrase occurs across clusters, it does not uniquely identify a cluster and hence its information gain is considered to be low. This metric is similar in usage to the χ^2 value used for feature selection in documents.

Stefanowski and Weiss [23] proposed a dimension reduction algorithm, based on *singular value decomposition* (SVD), to identify candidate phrases for cluster labelling. They perform phrase detection to extract phrases from documents and map these phrases along with the documents, onto a vector space. The dimensionality of the vector space is reduced using SVD. Finally, the phrases which are closer to the document clusters thus formed are used as their respective labels.

Mei et al [16] propose an algorithm for topic labelling of contexts, where each context is defined using a multinomial distribution of terms occurring in it. Their algorithm can be appended to the existing semantics mining techniques like Probabilistic Latent Semantic Analysis which defines a multinomial distribution over terms for a given context. As a part of training, they build a word distribution of labels over several different contexts. For any given query context, they identify a set of phrases called candidate labels and find the word distributions of each of the phrases

[1]Refer to Sahlgren's work [22] for a philosophical argument on "meaning is usage" and thus cooccurrences can be used to extract semantics.

in the trained model. Then they compute the KL divergence between the word distributions of the query context and the phrases. The phrase that matches the word distribution of the context the best is the label chosen for the context.

Cluster labelling algorithms look only for labels which occur in the text of the clusters and to do this they need a large enough cluster of documents to find a representative label. Although automatic topic labelling (ATL) [16] requires only a term distribution, a considerable set of documents in the context are needed to compute it to an acceptable degree of accuracy.

2.3 Text Summarisation

Text summarisation techniques address the problem of generating brief summaries for large text documents in a human readable form. They generally use *tf-idf* measures of words in a document and score different sentences in it. There are also techniques based on the centrality of a sentence in a document.

One such algorithm is LexRank [8]; it identifies the important and representative sentences in a document using random walks. All the sentences on a topic across documents are collected and the cosine similarity between them is measured. Based on the threshold on the cosine similarity, edges are added between nodes (nodes represent the sentences). A random walk on such a graph leads to central nodes that are sentences which represent the context best.

A variant of this approach called Biased LexRank [7], uses training data to compute a bias amongst nodes in the graph. The training data should be from the same domain as the document which is to be summarised. The bias distribution is used on top of the random walk probabilities to improve the quality of results.

Text summarisation is generally used to find a good summary of a document. But, a context is generally described in a collection of several documents and the sentences in the documents are usually document specific. For example, a generic summary could be, "Nadal defeats Federer at the Wimbledon" but rarely it describes the context like, "Federer and Nadal are tennis players".

In this paper, human generated labels and the recent paper on automated topic labelling [16] are used as baselines for comparison.

3 Topical Anchors

Given a document corpus consisting of many documents, the problem we address is the following: *Given a set of terms representing an unknown semantic context, find a term in the corpus that best labels the semantic context.*

In order to answer this question, we in turn adopt the following hypothesis: *Given a set of terms Q belonging to an (unknown) context C, the topical anchor is that term which has the highest probability of mention in any document or dialogue involving C.*

In this paper, we proceed to test this hypothesis based on a cooccurrence graph as the core data structure and *reachability* as the property of topical anchors.

3.1 Cooccurrence Graph

The basic data structure for this work is a *cooccurrence graph.* This is an undirected graph where nodes represent terms in the corpus and edges represent a cooccurrence count. Terms stored in the cooccurrence graph are noun phrases that are appropriately stemmed. Extracting noun phrases from text documents itself has a rich set of literature, which we shall not be addressing in this paper.

The cooccurrence graph G is a graph defined as, $G = (V, E, w)$, where V is the set of all nodes representing terms, E is the set of all edges representing cooccurrence and $w : E \rightarrow \mathbf{N}$ is a weight indicating the cooccurrence count. When two words a and b cooccur in a document, a node corresponding to each word is added to the cooccurrence graph, if such a node does not exist already. Also, if there is no edge between them, a new edge called $e_{a,b}$ is added, else the existing edge weight is incremented.

There are several ways by which cooccurrences are computed [5, 6, 19, 24] and choosing the best way is a problem on its own. In this work, the problem of finding topical anchors is posed in a manner independent of the actual technique used to generate the cooccurrence graph. The only property we require is that for any terms a and b which are linked in the cooccurrence graph, there should be a non-trivial semantic association between a and b.

3.2 Semantic Context

A "context" in a cooccurrence graph is any subgraph, containing terms and cooccurrences, which can be used to describe a specific semantic universe. Given a cooccurrence graph, it is not apparent what parts of the graph form semantic contexts.

Terms are extracted from an information source like a discussion thread or a document whose topical anchor we wish to identify. We call them *query terms* as they are used to query the graph in order to define a subgraph. In real-world applications, the query terms are expected to be automatically extracted from a piece of free text.

Given a term a in a cooccurrence graph G, the neighbourhood of a denoted by N_a is defined as the set of all nodes which share an edge with a. That is, the set of terms that are known to cooccur with a.

We then note that when terms from the same semantic context are used as query terms, there is an overlap in their neighbourhoods. Given this, suppose

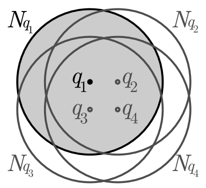


Figure 1: Context subgraph. The grey area represents the neighbourhood of query term q_1.

we are given a set of query terms Q that represent terms from an unknown context C, we say that C is a non-trivial semantic context if $\bigcap_{\forall q \in Q} N_q \neq \phi$.

When a set of query terms define a non-trivial context, for the purposes of determining topical anchors, we build a context subgraph whose node set V_C is calculated as:

$$V_C = \bigcup_q N_q, \text{ where } q \in Q \tag{1}$$

This is schematically shown in Figure 1.

In this work, the assumption that the context C is a non-trivial context is valid, because if the neighbourhood of two query terms does not have anything in common, then it would be difficult for humans to identify such a context and it will be impossible for an automated algorithm to identify a meaningful topical anchor. In fact, a good set of query terms from a single context would have a considerable overlap in their neighbourhoods.

3.3 Reachability and Centrality

As hypothesised earlier, a topical anchor of a context C is a term that is most likely to be mentioned in any document or dialogue involving terms in C. This hypothesis corresponds to calculating centrality of terms based on their reachability.

In the cooccurrence graph, a pair of cooccurring terms a and b are connected by an edge indicating the cooccurrence count. Using this, we can calculate the probability that b cooccurs with a as follows:

$$\rho(a \to b) = \frac{e_{a,b}}{\sum_{\forall x \in N_a} e_{a,x}} \tag{2}$$

Here $e_{a,b}$ is the cooccurrence count of the pair of terms a and b. Given a context C, a topical anchor t is one such term which has the highest probability of cooccurring with all terms in the context. If a random walker were to walk the cooccurrence graph the topical anchor can be thought of as the term which is most reachable from other terms for the random walker.

Random walking is a well-known technique on hyperlink graphs to compute an ordering of the nodes in the graph based on their reachability [1, 18].

For topical anchor computation, we start with a set of query terms Q, build a context subgraph C based on the terms in Q, and then use this subgraph to find the most reachable term.

A well-known algorithm for computing reachability over hyperlink graphs is PageRank [18]. An incremental equivalent of PageRank is the On-line Page Importance Computation algorithm or OPIC [1]. In OPIC, each node is initially assigned a *cash* value c_i. Then in an infinitely long process, nodes are randomly chosen, and their cash is uniformly distributed among all the outgoing links. Each node i also keeps track of the *history* of cash flow h_i through the node.

After every distribution, cash keeps getting accumulated at the target nodes till the node is picked for distribution. When distributing the cash, the current cash value is added to the cash flow *history* of the node.

The random walk probabilities are then computed as:

$$P(i) = \frac{h_i}{\sum_x h_x} \tag{3}$$

The OPIC computation is said to have reached a fixpoint when the ordering of nodes according to random walk probabilities stabilises.

In our earlier work [20], we computed topical anchors using a cash leaking random walk based on the OPIC model. We started with a set of query terms Q out of which we extracted a subgraph C that is representative of our context. Also, cash was distributed according to the probability of cooccurrence defined by ρ (equation 2), rather than with a uniform distribution.

In contrast to OPIC computations, for finding topical anchors, we need to compute centrality of nodes only within a subgraph C and not across the entire cooccurrence graph. So we computed OPIC on C, but allowed random walkers to walk out of C into the larger graph, thus discarding them. The random walk is explained in more detail in the next subsection.

3.4 Cash Leaking Random Walk

A characteristic feature of cooccurrence graphs over reasonably large corpora is their extremely small diameters. In addition, cooccurrence graphs contain a number of terms with very high degree. These could be commonly occurring nouns that are seen in almost every document, or polysemic terms that have different meanings in different contexts. In our experiments over a cooccurrence graph created from a Wikipedia dump, we found that a term like *United States* has edges to 78,738 other nodes, in other words more than 11% of the nodes in our data set. In two hops from *United States* 695,862 nodes were reachable i.e., more than 98% of the graph was covered.

Another example would be a word like *football* which is very popular and is connected with many other words. Any game and any player has a very high probability of being in the context of football at least once and hence will share an edge, however insignificant, with *football* in the cooccurrence graph. In fact, in our experiments, given a context around *Roger Federer* and *Rafael Nadal*, *football* was found to be more central to the context than *tennis*. The problem arises due to the differences in popularity of the two sports. *Football* occurs in a document corpus very rarely in the context of *tennis* but still due to its large number of edges, a tiny fraction of them will be large enough to act as a central node in the *tennis* context. The random walk should take into account not just the number of cooccurrence edges of each node in the context, but also the percentage of edges, from a node, which are in the subgraph. A word like *football* has a small percentage of edges coming into the *tennis* context and hence it should be penalised.

It might appear that a heuristic like TF-IDF would solve the problem but we found that TF-IDF picks important words yet those words are not the topical anchors [20].

To account for the elimination of nodes like *football* we propose a different method – that of a cash leaking random walk. The traditional random walk is usually performed on the whole graph G whereas a cash leaking random walk is run on a subset of the graph called C. The nodes in the subgraph C can participate in two types of edges; edges which lead to nodes inside the subgraph (E_C) and edges which lead to nodes outside the subgraph $(\overline{E_C} = E \setminus E_C)$.

A traditional random walk on the subgraph would ignore all edges other than E_C as shown in equation 4. In a cash leaking random walk, these edges in $\overline{E_C}$ are also considered but, all the cash that flows from nodes in the subgraph (V_C) to nodes outside the subgraph $(\overline{V_C})$ is deleted from the system as given by equation 5, such that the system continuously loses cash and hence the name.

$$c_{a \to b} = \frac{e_{a,b}}{\sum_i e_{a,i}} c_a, \text{ where } a, b, i \in V_C \quad (4)$$

$$c_{a \to b} = \frac{e_{a,b}}{\sum_i e_{a,i}} c_a, \text{ where } i \in V \text{ and } a, b \in V_C \quad (5)$$

One of the implications of such a random walk is that, the nodes which have a high percentage of cash flow into the subgraph, contribute more to the history of other nodes, than the nodes which have a low percentage of cash flow into the subgraph. The percentage computation is significant because the degree distribution between all nodes in the subgraph is not uniform, hence nodes that have a high degree into the subgraph tend to become important in a traditional random walk. In a cash leaking random walk their effect is normalised because the percentage of contribution into the subgraph is taken into account instead of the edge weights.

3.5 Discovering Topical Anchors

Technically, a topical anchor is the term which is most *reachable* from *all* the terms in a context. If a random walker starts from one of the nodes of the context subgraph in the cooccurrence graph and keeps visiting other nodes, the node which is visited most will be the topical anchor of the context. The history h_i of node i corresponds to the frequency of node i being visited and it enforces an ordering of all nodes in the context subgraph. The m nodes with the highest probabilities are then chosen as the topical anchors of the context where m is a user defined threshold.

However, when we begin our random walk with more than one query term, it is important to record how reachable is any given term from *all* the query terms. This is especially important if one or more query terms are polysems and may appear in two or more unrelated semantic contexts.

We address this issue in more detail in the next section.

4 History Vectors

The idea of using query terms to define a context can lead to a hidden bias towards some query terms. If query term q has a smaller degree in the cooccurrence graph than other terms in a query, then the neighbourhood of q will be smaller than that of other query terms. The cash with q gets distributed among lesser number of nodes, hence resulting in larger chunks of cash. This makes the neighbours of q comparatively richer than the rest of the nodes.

This leads to the curious problem that a smaller term q can "hijack" a larger query resulting in a skewed topical anchor. For instance, in Table 1, given the query terms *United States Dollar*, *Euro* and *West African CFA franc* over a cooccurrence graph on Wikipedia content, the topical anchor obtained by simple addition of history terms was *French Language*, with *currency* coming a distant fourth. The third query term *West African CFA franc* is a currency used in eight different countries in western Africa, and it is known well enough to be featured on Wikipedia, but enjoys far less representation than the two other currencies.

Another problem due to the usage of query terms for defining a context stems from the polysemy inherent in the query words. By picking the neighbourhood of a query term, the system does not discern the contextual difference in the neighbourhood of a polysemic query term. For example, the most important topical anchor for the query terms *Leaf*, *Fruit*, *Stem* and *Photosynthesis* was *Linguistics* due to the fact that *Stem* is also a common word in linguistics.

A combination of polysemy and a small neighbourhood could quickly lead to deterioration of the quality of topical anchors.

Query Terms	Topical Anchors
United States Dollar, Euro, West African CFA franc	French language, Guinea, Guinea-Bissau
Bayes, Euclid, Ramanujan, Bernoulli	Probability, Mathematics, Number
MIT, Stanford, IIT	University, Indian Institute of Technology, Bombay
Leaf, Fruit, Stem, Photosynthesis	Linguistics, Plant, Tree
Bernoulli, Poisson, Weibull, Binomial	Godwin, Norway, Harold Godwinson

Table 1: Examples with irrelevant topical anchors

The problem stems from the fact that some nodes in the graph get large amounts of cash from a single query term and hence end up in the top of the ranking. This problem can be addressed by measuring the cash flow separately from each query term. We can start by distinguishing between the cash of each query term by labelling the cash based on query terms from which it originates. By doing so we have a vector of cash at every node, a vector of cash flow between nodes and a vector of history at each of the nodes. The query terms form the dimensions of this vector space model of cash. The ranking to find the topical anchors is dependent on the *history vector* of each node in the subgraph and hence the name.

The history of cash flow through node i, denoted by a scalar h_i earlier, now becomes a vector of the form $\vec{h_i}$ whose dimensionality is $|Q|$, or the number of terms in the initial query. Topical anchors were computed by ordering the history values h_x in a descending fashion, but the problem becomes trickier when the history of a node is a vector. We propose several ways of ordering the vectors, and give the intuition behind each one of them in the rest of this section.

4.1 Projection

As the cash flow through each query term is independent, a vector space, whose orthogonal axes are described by the query terms, is constructed. Every history vector is a point in this space. The line of least bias $\vec{l}$ is defined as a line which is made of points equidistant from each of the axes. By definition this line passes through the origin. Projecting a vector onto $\vec{l}$ and measuring the distance from the origin to the projection can be used to find an ordering of the history vectors of the nodes. So, to obtain terms that are most reachable from all the query terms, we need to compare history vectors to the line of least bias and *project* each vector onto it as shown in figure 2(a). If $\vec{h_u}$ is the history vector of node u, then $\vec{h_u}\cos\theta$ would be the magnitude of projection onto $\vec{l}$, where θ is the angle between $\vec{l}$ and $\vec{h_u}$. The score of a node u is p_u, and it is a scalar using which the ordering of nodes is performed and in this case it is,

$$p_u = \vec{h_u}\cos\theta = \frac{\vec{h_u}\cdot\vec{l}}{\|\vec{l}\|} \tag{6}$$

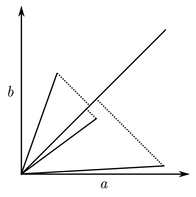
However, computing projections onto the line of least bias returns exactly the same results as using scalars for history. We explain this as follows.

Since $\vec{l}$ is the line of least bias, all components of $\vec{l}$ are the same. Without loss of generality, we can consider $\vec{l}$ as the vector $[1, 1, \ldots 1]^T$. The denominator in equation 6 is a constant and will not affect the ordering of the nodes and hence can be ignored. Given this, p_u in equation 6 can be rewritten as,

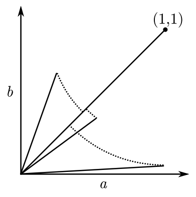
$$p_u = \sum_q h_u^q \tag{7}$$

where h_u^q represents the cash flow history from query term q through node u.

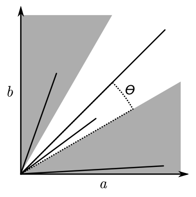
Examining p_u, which is computed using projection on $\vec{l}$, we can see that it is just the sum of cash history of all labels at a node and in the process the cash distinction created by labels and history vectors is lost. Computing the score using history vectors and projections onto $\vec{l}$ is exactly the same as not having the vectors in the first place. So, projecting vectors for ranking does not eliminate the inherent bias.



(a) History vectors projected onto the vector of least bias



(b) Euclidean distance between history vectors and best possible point on the vector of least bias



(c) Cosine thresholding of history vectors based on the angle made with the vector of least bias

Figure 2: Different history vector based orderings

We then tried to eliminate erroneous results by using two different types of penalties. In the first case, we penalised the history vectors by giving them a score which is directly proportional to their magnitude and inversely proportional to their angular distance from $\vec{l}$. In the second case we measured the magnitude of the history vector as the score, provided the angular distance from $\vec{l}$ is below a set threshold.

4.2 Euclidean distance in History Vectors

To give a score which is directly proportional to the magnitude and inversely proportional to the angular distance from $\vec{l}$, we compute the Eucledian or L_2 distance with an "ideal" vector.

We do this by normalising each history vector as follows. Let our result set be $T = \{\vec{h_1}, \vec{h_2}, \ldots, \vec{h_n}\}$. Let the term h_i^k denote the k^{th} dimension of history vector $\vec{h_i}$. This dimension is normalised as:

$$\mathbf{h_i^k} = \frac{h_i^k}{\max\limits_x h_x^k}$$

Given this, we can see that the point $\vec{1} = [1, 1, \ldots, 1]^T$ represents the "ideal" history vector where all dimensions have the maximum cash flow possible. The score for each result term would now be based on the L_2 distance from its own history vector to $\vec{1}$. This is shown in figure 2(b). The score for any given vector is:

$$p_u = R - L_2(\vec{r}, \vec{h_u}) \tag{8}$$

In equation 8, $L_2(\vec{x}, \vec{y})$ is the Euclidean distance between the points $\vec{x}$ and $\vec{y}$ and R is the magnitude of the ideal vector. This scoring ensures an ordering of all vectors but punishes nodes whose cash contribution is mainly from only one of the query terms as shown in figure 2(b).

4.3 Cosine similarity in History Vectors

Another method of penalising nodes whose cash history is chiefly contributed by only a few of the query terms is to set a threshold θ on the angle made between a node and the line of least bias $\vec{l}$ (see figure 2(c)). If the angle between a node and $\vec{l}$ is more than the set threshold θ, then that node is ignored from the topical anchor ranking, else its magnitude is used as the score. The threshold can either be set on the angle or equivalently it can be set on the cosine similarity measure between the vectors. The cosine similarity of two vectors separated by an angle of α, is a scalar value between 0 to 1 and is given by,

$$\cos\alpha = \frac{\vec{a}\cdot\vec{b}}{||\vec{a}||\,||\vec{b}||} \tag{9}$$

As shown earlier in equation 7, this is equivalent to,

$$p_u = \frac{\sum_q h_u^q}{||\vec{h_u}||} \tag{10}$$

Thresholding based on cosine similarity will remove terms that are skewed in their history vector components even if they have a small Eucledian distance from $\vec{1}$. However, the problem with such a technique is the manual setting of the threshold. It is very difficult to arrive at one optimal threshold which will work for several different contexts and corresponding query terms.

The improvement in the quality of results by using history vectors and penalties like *Euclidean* or *Cosine* is apparent from Table 2.

4.4 Comparison with other Random Walk Models

The cash leaking random walk is context dependent and is similar to some other random walk models [2, 10, 17] in this regard. Bookmark colouring [2], even though similar, is run on the whole graph and is designed to run on a hyperlink graph. This is a major difference because a hyperlink graph usually has a much higher diameter compared to a cooccurrence graph. In a cooccurrence graph a word could be connected to another word from a completely different context. For example, in our experiments we found that *Roger Federer* was linked to more than 25,000 nodes in two hops. To further illustrate the significance, *Roger Federer*, who is a tennis player, was linked to *feral child*, a child brought up by animals without any human contact, in two hops through 5 different paths (*Roger Federer* $\rightarrow$ *France* $\rightarrow$ *Feral child*). Context switching happens at extremely short path length in a cooccurrence graph. So, a technique like bookmark colouring would wander into several different contexts before settling down. Context subgraphs in cooccurrence graphs tend to be near cliques and this is not generally the case with other graph based models.

Betweenness centrality measures [10, 17] also use source s and destination t nodes in a fashion similar to query terms in the cash leaking random walks. The equivalence can be further seen when one of the nodes in between i will have the highest betweenness score and is equivalent to a topical anchor. But, the major drawback of such techniques is the modelling of the graph. In betweenness centrality, influence flow from one node to another is measured, but there is no such equivalence in topical anchors. Also connection subgraphs defined by Faloutsos [10] identify a subgraph around s and t and restrict the computation. Even here the similarity ends up being deceptive as connection subgraphs are designed to penalise nodes with a high degree whereas most topical anchors in a context subgraph are nodes with very high degree.

The major difference is the way fringe nodes on the subgraph are handled. Fringe nodes of a context C are nodes which have a higher degree of edges to nodes outside the context C, in $G \setminus C$, when compared to the degree of edges to nodes inside C. The semantic contribution of fringe nodes to C is not very important and such nodes are penalised. This is a problem unique to cooccurrence graphs and hence other techniques cannot be easily applied here.

Query Terms	Projection	Eucledian	Cosine
United States Dollar, Euro, West African CFA franc	French language, Guinea, Guinea-Bissau	Currency, Bank, France	Currency, Bank, France
Bayes, Euclid, Ramanujan, Bernoulli	Probability, Mathematics, Number	Mathematics, Mathematician, Euler	Mathematics, Mathematician, Probability distribution
MIT, Stanford, IIT	University, Indian Institute of Technology, Bombay	University, College, Technology	University, College, Science
Leaf, Fruit, Stem, Photosynthesis	Linguistics, Plant, Tree	Plant, Tree, Species	Plant, Tree, Species
Bernoulli, Poisson, Weibull, Binomial	Godwin, Norway, Harold Godwinson	Mathematics, Probability, Expected Value	Mathematics, Probability, Statistics

Table 2: Some example results (top 3) from our experiments

5 System Design

The system design is divided into two components, viz., the offline component and the online component as shown in the figure 3.

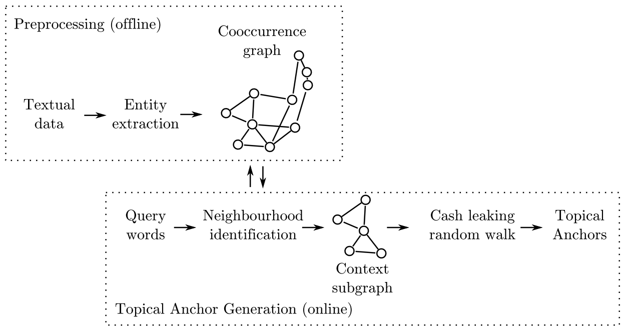


Figure 3: System design showing the offline computing and online computing components

The offline component consists of cleaning and parsing the text in a large text corpora, followed by entity extraction. After entities are extracted, the cooccurrences between them are measured and added into the cooccurrence graph. The cooccurrence graph consists of all the cooccurrences ever observed in the textual corpora. The whole offline model is performed only once and is incremental. Addition of new data corresponds to adding new edges or incrementing edges and is equivalent to having that data in the textual corpora initially.

The online computation starts by taking query terms either from a user or from a different program based on the application of topical anchors. Using the query terms and the cooccurrence graph, the neighbourhood is identified for each query. Using these neighbourhoods the context subgraph is built. A cash leaking random walk along with history ranking yields the topical anchors for the query.

6 Experimental Results

6.1 Data Set and Preprocessing

The experiments were run on a cooccurrence graph built using dump of the English language version of the open source encyclopedia, Wikipedia, obtained in December 2006. The dataset was cleaned by removing all the stub pages and in each of the pages the trivia sections, popular culture references, tables, info-boxes and general references were removed from the text. In the experiments, cooccurrence was measured between entities and not between all words. An entity in this experiment is generally taken to be a noun or a noun phrase which contributes semantically to a context.

The English language Wikipedia is organised in such a way that the first reference to an entity in a document is hyperlinked to the Wikipedia entry on that entity. So all the hyperlinks in a page were used to identify the entities that occur in page[2]. Usually only the first occurrence of an entity is hyperlinked, so all other occurrences of that entity in the document were programmatically marked to differentiate them from rest of the text.

A Wikipedia entry for an article on a topic is divided into sections and there are subtle variations in context across sections in the same document. In this experiment we used a modified version of document cooccurrence where each section was treated as a pseudo-document in itself. The title of the Wikipedia article belonged to all the sections and hence was appended as the title of each of these pseudo-documents and all the entities which cooccurred in that section were added to the cooccurrence graph by incrementing the edge between them.

Each section is treated as a set of entities and is added to the cooccurrence graph separately. If two words cooccur in three different sections in a document, then when the document is processed their cooccurrence is incremented by three. The number of times a word occurs in a section is completely ignored because the cooccurrence inside a section is taken to be

[2]This was done to reduce the amount of error in entity detection

bivalent, i.e., it is either present or absent.

There were 81,253 documents in all, giving rise to a cooccurrence graph with 709,220 nodes and 19,686,591 cooccurrence edges.

There are certain advantages and disadvantages in using an encyclopedic data set. The most important advantage of such a data set is the coverage of topics it provides and due to that topical anchors can be found across a vast number of topics. Another advantage of using Wikipedia in particular was the ease with which entities can be detected in wikitext[3]. The major disadvantage of an encyclopedic source is the lack of repetitiveness characteristic of real world data. Hence it was easy for *Roger Federer* and *football* to be related as the weight of the edge between *Roger Federer* and *tennis* is not being reinforced enough.

6.2 Experimental Setup

Two different experiments were conducted to validate the correctness of the topical anchors computed. For evaluation, the topical anchor engine was made available online[4] and volunteers were encouraged to give words from any semantic context and the topical anchors of the contexts were returned to them. Volunteers were also allowed to give feedback which was used in fine tuning and designing various aspects of the algorithms.

A set of 100 thus human generated queries were chosen and volunteers were asked to write down at most three topical anchors for each of the queries. There were a total of 86 volunteers of which 66 volunteers preferred to attempt all 100 questions and 20 volunteers preferred to answer 30 random questions from the hundred questions. The answers exhibited a considerable amount of synonyms and spelling errors. We cleaned the answers given, by dividing them into various semantic buckets. For example, with regard to *monarchy* the words *empire*, *dynasty*, *kingdom*, and *rule* are only minor differences in perception and hence were put in the same bucket. Words like *marine*, *ocean* and *sea* were put in their semantic bucket but words like *film* and *Hollywood* though related were kept apart. After mapping words into semantic buckets, the semantic buckets were termed as the topical anchors that the users chose. These user generated topical anchors were used to evaluate the topical anchors generated by the algorithm.

In the first experiment, the usage of history vectors was validated by comparing the three different ordering schemes for the history vector model, viz., projection, euclidean distance, cosine similarity thresholding described in sections 4.1, 4.2, 4.3 respectively. Table 2 shows some example queries and corresponding results from the three algorithms. Note: As projection is equivalent to the topical anchors generated with scalar history values, this table also shows the significant improvement resulting from the proposed algorithms using history vectors.

In the second experiment, Automated Topic Labelling (ATL) [16] was compared with the topical anchors generated by the euclidean distance history vector model. Ten different clusters of documents were taken and ATL was used to identify three labels for each cluster. A sample set of three to four query terms were taken from each of these clusters and were used to compute the topical anchors of the cluster. The results generated were compared using the user feedback as objective truth.

For all the experiments, we partitioned the topical anchors given by the users into *confidence intervals* based on the percentage of users endorsing a topical anchor. A confidence interval from x to $x + c$ is a bucket where $x\%$ to $(x + c)\%$ of users agree that the terms in the bucket are topical anchors for a query. For example, if 95% of the users answer *computer* for the query ,"*CPU, hard disk, monitor, mouse*" then we put *computer* into the 90-100 confidence interval. The confidence intervals are in steps of 10 starting from 40-50 and going up to 90-100. The user generated topical anchors which are not in any of the confidence intervals above 40 are ignored because of the lack of adequate support. The distribution of user generated topical anchors based on the confidence interval to which they belong is given in figure 4.

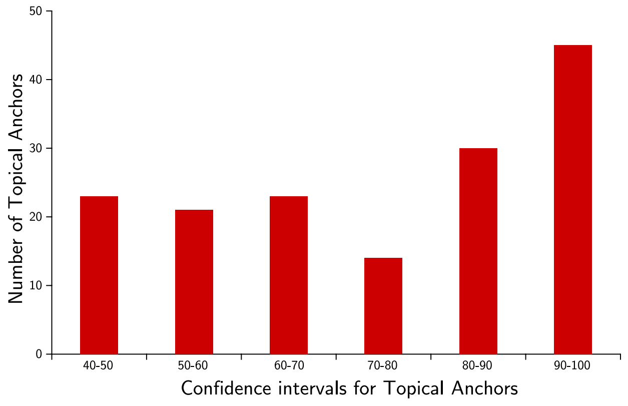


Figure 4: Distribution of User given Topical Anchors across different confidence intervals

With a confidence cut-off at 40, there are a total of 156 topical anchors, which are in confidence intervals of 40 and above, across all the 100 chosen queries. Some queries like, "*summer, winter, spring, autumn*" have just one anchor *season* and some others like "*Volt, Watt, Ohm, Tesla*" have several user given anchors like *unit, electricity, physics.*

In the experiments, *hit-rate* is defined as the number of anchors picked by an algorithm in a confidence

[3]It is a markup for representing text, images, data and hyperlinks on Wikipedia.

[4]Topical Anchors can be accessed from `http://tinyurl.com/topicalanchors`

interval out of the 156 assigned and it is used as the measure of comparison. Formally, hit-rate $HR_a = \sum_Q |t_a(Q) \cap t_u(Q)|$, where t_a is the set of topical anchors picked by the algorithm for a query Q and t_u is the set of topical anchors picked by a user for the same query.

In our earlier work [20], the validity of a cash leaking random walk was tested against a conventional random walk like OPIC [1] and with a baseline like TF-IDF. The topical anchors generated using all the algorithms were compared using the user feedback. We found that for the top ten topical anchors for each of the 100 contexts, `tfidf` and `opic` had an overall hit-rate of only 40 and 56 respectively, i.e., `tfidf` could only pick 40 correct topical anchors while the cash leaking random walk `cl` picked 137 topical anchors. To emphasise, `cl` with just the top topical anchor had a higher hit-rate of 63 than `tfidf` and `opic` with 10 topical anchors each (figure 5).

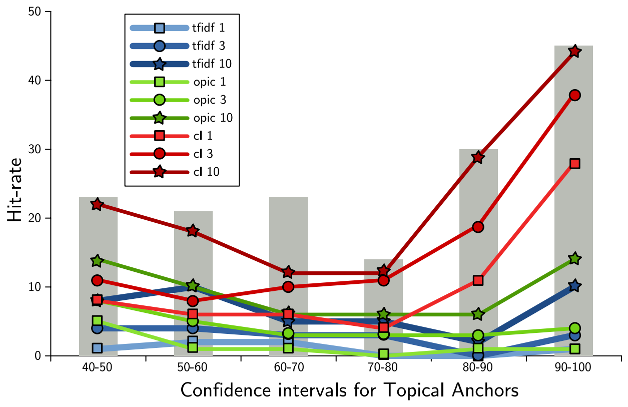


Figure 5: Comparison between TF-IDF and random walks with and without cash leakage

6.3 Experiment 1: Comparison of different History Vector models

In the first experiment we compared the performance of cash leaking random walks which use history vectors to negate bias inherent in the query terms. It has been shown in section 4.1 that using history vectors and projecting them onto the vector of least bias is exactly equivalent to computing a cash leaking random walk without the notion of history vectors. This method is called `Projection` from now on. The other two history vector models termed as `Euclidean` and `Cosine` were compared with `Projection` the same way as in Experiment 1.

The three different random walks were executed on each of the 100 queries to compute 10 topical anchors for each query. `Projection` as mentioned earlier had a hit-rate of 137 whereas `Euclidean` and `Cosine` had hit-rates of 149 each. As the topical anchors after the top three will not be so useful, they were eliminated from the plot.

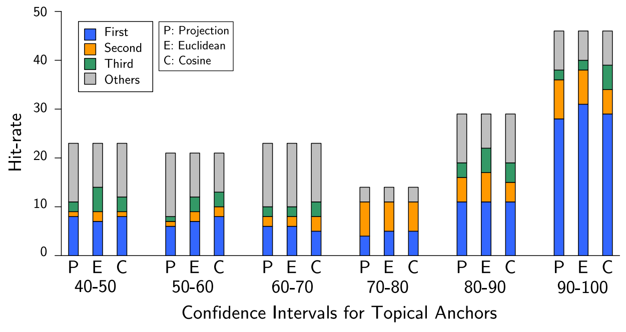
The topical anchors computed were grouped based on the rank given by the algorithm. In other words, we count the number of topical anchors ranked in the first place which are in the 90-100 confidence interval. The same process is repeated for the second and the third topical anchor and similarly for all other intervals. Figure 6(a) shows the hit-rate for each algorithm across all intervals.

The results in figure 6(a) are grouped by confidence intervals and hence every confidence interval has each of the three methods discussed in the paper. The vertical line for each method is divided into four regions. The bottommost region represents the hit-rate of highest ranked topical anchor by the algorithm. The regions above denote second highest, third highest and others which represent the topical anchors not picked in the top three positions.

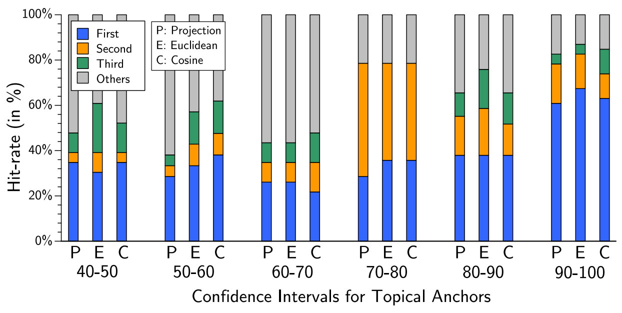
The others column on the 90-100 interval indicates that there are cases where the topical anchor was obvious to the user but the algorithms could not pick up that exact term – although it returned something relevant. For example, for the query, "*Mickey Mouse, Minnie Mouse, Donald Duck, Goofy*", the primary topical anchor chosen by the users was *cartoon* but the algorithm picked *Disney*.

We also found that, the more the users agree on a topical anchor, the better the results are with the algorithm. Figure 6(b) shows the hit-rate percentage i.e., the percentage of times the algorithm picked the topical anchors in the first, second and third places for each confidence interval. It can be seen that when almost all of the users agree on a topical anchor, i.e., 90-100 interval, the algorithms seem to perform better compared to topical anchors in other intervals. `Euclidean` picked the correct topical anchor in the first rank 62% of the time. Also we found that `Euclidean` with three topical anchors per query has an overall hit-rate percentage of 70.

There were five queries in the experiment where one of the query terms had a smaller neighbourhood when compared with the rest. For example, the query, "*Binomial, Poisson, Bernoulli, Weibull*", yielded no meaningful result when we used `Projection`. This has been true with all the five queries, and results from `Projection` deteriorate if there is any large difference in the degree of the nodes representing the query terms, indicating a bias. The results drastically improved with `Euclidean` which returned *probability* and *mathematics* as the results. `Cosine` too produced similar results but in certain queries it was unable to pick all the topical anchors as the number of query terms inside the threshold were lesser than the number of topical anchors of the query and it failed to pick other topical anchors.



(a) Hit-rate of the different history vector based algorithms



(b) Hit-rate percentage of the different history vector based algorithms

Figure 6: Experimental Results

Another example is the query "*MIT, Stanford, IIT*", where *IIT* being a node with a lesser degree resulted in the `Projection` choosing *Bombay* over *education, college* and *technology*.

Because of the problems of fixed thresholding with `Cosine`, where the threshold cannot be determined run-time, `Euclidean` worked the best among the three.

6.4 Experiment 2: Comparison of Cash Leaking Random Walks against Automatic Topic Labelling

In the second experiment we compared the topical anchors computed using a cash leaking random walk (CLRW) with a cluster labelling algorithm called Automatic Topic Labelling (ATL) [16]. This comparison is not as straight forward as the other experiments because the ATL algorithm takes as input a multinomial distribution of terms from a context. To compute this distribution we took 10 different clusters of documents and computed the multinomial distribution of terms on each one of them. From each cluster of documents we also took 4 important keywords and gave them as input to the CLRW (with Euclidean method of handling history vectors) algorithm to generate topical anchors.

ATL works by computing the KL Divergence between a context and a label. As we had already computed the multinomial distribution of terms for the context, the multinomial distribution of terms for each label was obtained from the cooccurrence distribution in our graph. KL Divergence was then computed and the winning phrases were chosen as the labels.

Users picked 14 representative labels across the 10 clusters and on them CLRW had a hit-rate of 10 whereas the ATL algorithm had a hit-rate of 8. Figure 7 shows the distribution of the topical anchors/labels across different confidence intervals and the number of them picked by each algorithm at first, second and third positions. The rest of the positions were not considered for the results.

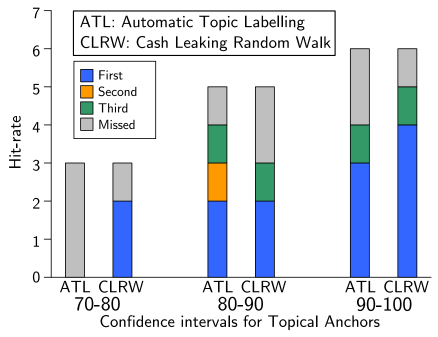


Figure 7: Comparison between Cash Leaking Random Walk and Automatic Topic Labelling

The results show that the topical anchors generated with just a few words from a context are comparable to, if not better than, labels of clusters generated by existing labelling algorithms.

But the importance of a Cash Leaking Random Walk is more prominent when the application has very little information of the context, like a few important words from a context as in a discussion thread in an online forum. All cluster labelling algorithms require a significant number of documents as a context to label it, and they would fail on a very short question like *"What are the symptoms of type 2 diabetes and do oral insulin drugs have any side effects on the health of my pancreas?"* whereas a Cash Leaking Random Walk can identify the context as *medicine*.

6.5 Results Summary

The user generated topical anchors were used as the basis for our evaluation. The topical anchors were collected from 86 volunteers and were then divided into different sets based on the percentage of users agreeing on each of those topical anchors.

In summary, we found that algorithms using history vectors efficiently, consistently outperformed a random walk which does not take query bias into account.

And, among the models using history vectors we found that Euclidean distance method outperformed the others. We also compared our model with an existing cluster labelling algorithm favourably.

7 Concluding Remarks

The notion that cooccurrence patterns amongst terms can be used to model certain semantic associations between them was well known. We hypothesised that the terms central to a context subgraph, represent the context best in other contexts and termed them as topical anchors. In this paper we define a cash leaking random walk and a vector-based ranking on top of it to identify topical anchors. Through a user evaluation and relevant experiments and a comparison with a recent cluster labelling technique, we corroborated our hypothesis.

There are still some problems to be addressed before topical anchors could be put to use in real-world systems. Currently we are working on issues like finding the optimum number of topical anchors to generate for a given context and the right number of query terms required before our results deteriorate. The results of the same will be published in the future.

References

[1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *Proceedings of the 12th international conference on World Wide Web*, 2003.

[2] P. Berkhin. Bookmark-coloring algorithm for personalized pagerank computing. *Internet Mathematics*, 2006.

[3] M. Ceglowski, A. Coburn, and J. Cuadrado. Semantic search of unstructured data using contextual network graphs, 2003.

[4] P. Clerkin, P. Cunningham, and C. Hayes. Ontology discovery for the semantic web using hierarchical clustering. *Semantic Web Mining Workshop*, 2001.

[5] I. Dagan, F. Pereira, and L. Lee. Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 1994.

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

[7] G. Erkan. Using biased random walks for focused segmentation. In *DUC*, 2006.

[8] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 2004.

[9] V. Evans, B. K. Bergen, and J. Zinken. *The Cognitive Linguistics Enterprise: An Overview*. 2006.

[10] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[11] D. Faure and C. Nédellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *LREC workshop on adapting lexical and and corpus resources to sublanguages and applications*, 1998.

[12] L. Fu. *Neural Networks in Computer Intelligence.* McGraw-Hill, Inc., 1994.

[13] F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani. Cluster generation and labeling for web snippets: A fast, accurate hierarchical solution. *Internet Mathematics*, 2006.

[14] E. Maedche and S. Staab. Learning ontologies for the semantic web. In *IEEE Intelligent Systems*, 2001.

[15] S. Mcdonald and M. Ramscar. Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, 2001.

[16] Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007.

[17] M. E. J. Newman. A measure of betweenness centrality based on random walks, 2003.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[19] M. Patel, J. A. Bullinaria, and J. P. Levy. Extracting semantic representations from large text corpora. In *Proceedings of the 4th Neural Computation and Psychology Workshop*, 1998.

[20] A. R. Rachakonda and S. Srinivasa. Finding the topical anchors of a context using lexical cooccurrence data. In *Procedings of the 18th ACM Conference on Information and Knowledge Management*, 2009.

[21] M. L. Reinberger and W. Daelemans. Is shallow parsing useful for the unsupervised learning of semantic clusters? In *Proceedings of the 4th Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2003)*, 2003.

[22] M. Sahlgren. Vector-based semantic analysis: Representing word meanings based on random labels. In *ESSLI Workshop on Semantic Knowledge Acquistion and Categorization*, 2001.

[23] J. Stefanowski and D. Weiss. Comprehensible and accurate cluster labels in text clustering. In *Proceedings of the 8th Large-Scale Semantic Access to Content Conference*, 2007.

[24] E. Terra and C. L. A. Clarke. Frequency estimates for statistical word similarity measures. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2003.