The Computer Journal Advance Access published March 15, 2011

doi:10.1093/comjnl/bxr024

Exploiting Wikipedia Knowledge for Conceptual Hierarchical Clustering of Documents

GERASIMOS SPANAKIS*, GEORGIOS SIOLAS AND ANDREAS STAFYLOPATIS

Intelligent Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, 15780, Zografou, Athens, Greece
**Corresponding author: gspana@ece.ntua.gr*

In this paper, we propose a novel method for conceptual hierarchical clustering of documents using knowledge extracted from Wikipedia. The proposed method overcomes the classic bag-of-words models disadvantages through the exploitation of Wikipedia textual content and link structure. A robust and compact document representation is built in real-time using the Wikipedia application programmer's interface, without the need to store locally any Wikipedia information. The clustering process is hierarchical and extends the idea of frequent items by using Wikipedia article titles for selecting cluster labels that are descriptive and important for the examined corpus. Experiments show that the proposed technique greatly improves over the baseline approach, both in terms of *F*-measure and entropy on the one hand and computational cost on the other.

Keywords: document clustering; document representation; Wikipedia knowledge; conceptual clustering

Received 14 July 2010; revised 28 January 2011
Handling editor: Franco Zambonelli

1. INTRODUCTION

Efficient document clustering has always been a research topic of great interest due to the fact that it combines aspects of machine learning (ML), information retrieval (IR) and natural language processing. Traditional clustering algorithms are usually based on the bag-of-words (BOW) approach [1] for representing documents. The two main disadvantages of the BOW model are that (a) it treats all words the same way, ignoring any syntactic or semantic relationship among them and (b) it introduces a large vector space and, hence, the curse of dimensionality. In this paper, we introduce a way to both enrich and compress document representation, with background knowledge provided by an ontology, in order to enhance performance in a clustering task.

Nowadays, Wikipedia has become one of the largest knowledge repositories and new content is being added to it daily by users around the globe. As a corpus for knowledge extraction, Wikipedia's advantages are not limited to its size and the easiness of article updating, but also comprise hierarchical category organization, dense link structure between articles, sense disambiguation capability based on the URLs, brief anchor texts and well-structured sentences.

This paper introduces an efficient document clustering technique using knowledge extracted from the Wikipedia corpus. The integration of Wikipedia articles into the document representation is performed by mapping one or more words of the document—forming a topic—to the corresponding Wikipedia article, creating what we call *document concepts* and using document concepts instead of a BOW. Wikipedia article features (ingoing/outgoing links etc.) are additionally used in the *concept frequencies* of each document, thus creating rich document representations. Moreover, the word disambiguation issue is addressed using a technique exploiting Wikipedia content and category structure. Finally, a conceptual hierarchical clustering (CHC) technique is implemented, based on important and frequent concepts of the corpus, which produces clusters with labels (conceptual clustering), informative of the content of the documents assigned to each specific cluster.

The remainder of the paper is organized as follows. In Section 2, related work dealing with text representation, ontology mining and document clustering is discussed. The proposed document representation using Wikipedia is described in Section 3. Section 4 describes the hierarchical

clustering process. The conducted experiments with the evaluation results are reported in Section 5. In Section 6, we conclude the paper and give directions for further improvement of our method.

2. RELATED WORK

There has been a growing amount of research in ways of enhancing text clustering by introducing external knowledge. Contextual information [2] was an early attempt to this goal. A two-stage model was introduced using features based on sentence categories as an alternative approach to the original vector space model which includes contextual information. At the first stage, a set of sentence categories was extracted and at the second stage it was used to describe a new vector space for document representation.

However, the most promising direction is the exploitation of external knowledge provided by ontologies. Emerging research in ontologies provides many sources, such as methods for enriching the analysis and visualization of documents by using semantic features [3]. Early work utilized the semantic lexicon WordNet [4] in order to enrich document representation. WordNet's rich hierarchical structure was used to attempt syntax-based disambiguation by assigning to each word a part-of-speech (POS) tag and by enriching the BOW data representation with synonyms and hypernyms. Apparently, the noise introduced by incorrect senses retrieved from WordNet developed to be a bottleneck for its use in document enrichment.

Wikipedia is another ontology which has been recently used in many applications involving IR [5], text categorization [6, 7] and text clustering [8, 9]. Gabrilovich and Markovitch [6, 10] propose a method to improve text classification performance by enriching document representation with Wikipedia concepts. The mapping between each document and Wikipedia concepts is achieved through a feature generator which acts like a retrieval engine. It receives a text fragment, which can be words, sentence, paragraph or the whole document, and outputs the most relevant Wikipedia articles to the text fragment. The titles of the retrieved Wikipedia articles are further filtered and those with high discriminative capacity are used as additional features to enrich the representation of the corresponding documents. Empirical evaluation shows that their method can greatly improve classification performance.

Wikipedia has also been applied for text clustering. Banerjee *et al.* [11] use a method similar to that applied in [6] for clustering short texts. Their method is different in the fact that they use query strings created from document texts to retrieve relevant Wikipedia articles. The titles of top-ranked Wikipedia articles serve as additional features for clustering Google news. Both methods of [6, 11] only augment document representation with Wikipedia concepts' content without considering the hierarchical structure of Wikipedia or other features of Wikipedia articles.

Wikipedia category information has also been utilized in [12] for text categorization and in [13] for text clustering. These methods extend the Wikipedia concept vector for each document with synonyms and associative concepts based on the redirect links and hyperlinks in Wikipedia.

All of the papers mentioned above, rely on existing clustering techniques [mostly k-nearest neighbors (k-NN) and hierarchical agglomerative clustering (HAC)]. In this paper, we extend the idea of frequent itemsets described in [14] and introduce a novel clustering technique, CHC. CHC takes into account both the relative significance of concepts—extracted by exploiting various features of the ontology—and their frequency in the corpus. This produces a hierarchical clustering tree by automatically extracting labels for each cluster, according to the content of the documents assigned to it.

3. DOCUMENT REPRESENTATION MODEL USING WIKIPEDIA

The framework of our method for exploiting Wikipedia in order to enrich document representation is presented in Fig. 1.

3.1. Concept extraction from Wikipedia

Our goal is to extract Wikipedia concepts which are described by one or more consecutive words of the document. For example, if a document contains the phrase 'data mining', it is desirable to extract both words as an entity and map it to the corresponding Wikipedia article, thus forming a document concept. Note that, in the same situation, the BOW method would have broken the previous semantic entity into the far more ambiguous singletons 'data' and 'mining'. However, the complexity of extracting all possible document N-grams in order to check whether or not they are mapped onto an existing Wikipedia article is too high, as mentioned in [12], and methods to reduce it until now relied only on restricting document topics to a specific domain [15].

In our approach, we overcome the bottleneck of N-grams by choosing to annotate each document's text with POS information using the TreeTagger tool provided by [16]. Wikipedia articles have descriptive titles, so it is not necessary to perform stemming or remove stop words during document preprocessing. After this procedure, we keep those consecutive words which are nouns and proper nouns (singular or mass or plural) along with prepositions, subordinating or coordinating conjunctions and the word *to* (POS tags in the Penn Treebank Tagset [17]). By grouping all consecutive words in the text having one of the previous POS tags, we perform full *noun phrase* extraction (e.g. we can extract both 'Barack Obama' and 'The President of the USA'), while reducing the computational cost of considering every N-gram of the text, including verbs, etc. The extracted noun phrases form our candidate concepts.

For each candidate concept, we automatically check 'on-the-fly' whether it exists or not as a Wikipedia article using the

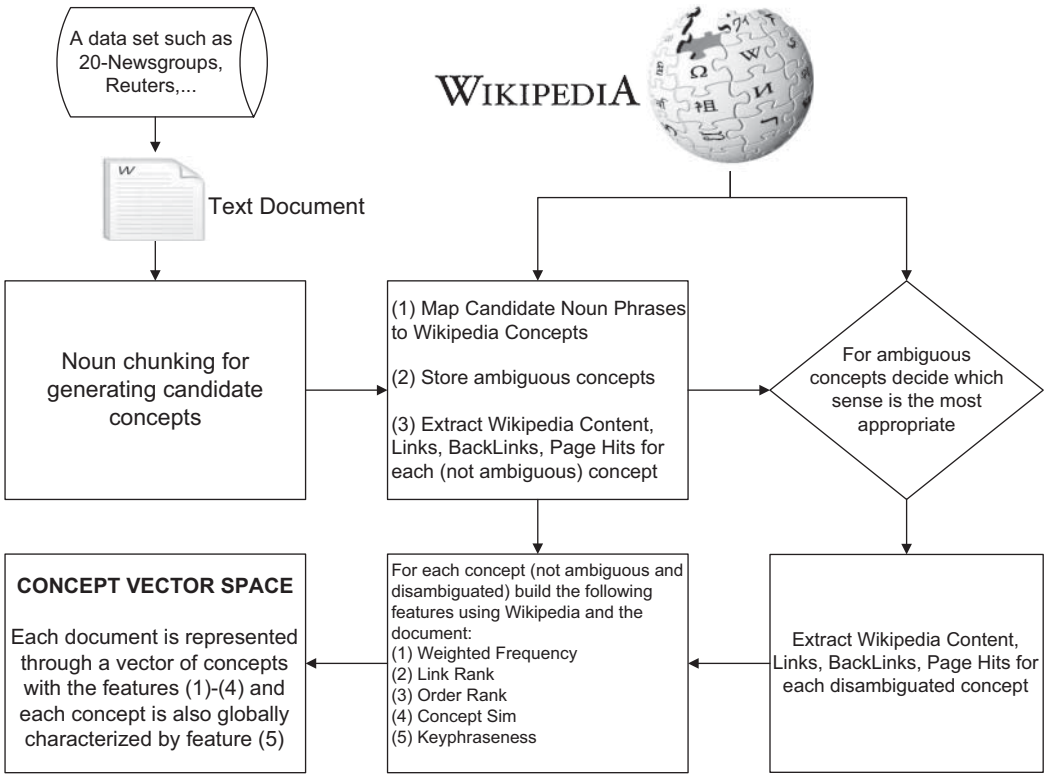

FIGURE 1. Representing a document by its Wikipedia matching concepts.

Wikipedia application programmer's interface (API).[1] If the concept has multiple senses (so there are multiple Wikipedia articles referring to the same noun phrase), we perform the disambiguation process described in the next section, in order to choose the most appropriate sense. Once we obtain a unique mapping between the candidate concept and Wikipedia, the concept is selected as a component of the document vector which is about to be formed. For example, we shall consider a text fragment from a document from a 20-newsgroups (20-NG) dataset. 20-NG [18] contains ∼20 000 documents of the full 20-NG collection of USENET news group articles. Each newsgroup belongs to a different category, with varying overlap between them: some newsgroups are very related (e.g. comp.os.ms-windows.misc and comp.windows.x) and others are not related at all (e.g. sci.electronics and alt.atheism). In the following text fragment (from document #59284 of the 20-NG dataset), the extracted concepts are shown in **bold**.

Depo Provera *was developed in the* ***1960s*** *and has been approved for* ***contraception*** *in many other* ***countries***. *The* ***UpJohn Company*** *of Kalamazoo, Mich., which will* ***market*** *the* ***drug*** *under the* ***name***, *Depo Provera Contraceptive Injection first submitted it for* ***approval*** *in the* ***United States*** *in the* ***1970s***. *At that* ***time***, ***animal studies*** *raised* ***questions*** *about its* ***potential*** *to cause* ***breast cancer***.

[1] See http://en.wikipedia.org/w/api.php.

It is obvious, how important to a text retrieval task is the ability to find concepts such as 'Depo Provera', 'UpJohn Company', 'United States' which would be broken into two words with no specific content (e.g. 'Depo' or 'Provera') or with other meanings (e.g. 'United' or 'States').

At the same time, using the Wikipedia API, for every selected concept i, we extract the features presented in Fig. 2. To clearly demonstrate the described features, consider Fig. 3 where an example of a *link*, a *backlink*, a *pagehit* and a set of *categories* are shown. Note that the *content* (which is not shown in this figure) refers to the whole text of the corresponding article.

After the extraction of the features described in Fig. 2 for every concept i in a document j, we combine them to form new document features, as described in the equations below, in order to form a richer document representation.

(i) Weighted Frequency ($Wfreq$) is defined by

$$\text{WFreq}_{j,i} = \text{size}_i * \text{frequency}_{j,i}, \quad (1)$$

where $size_i$ is the number of words that form concept i and $frequency_{j,i}$ stands for how many times concept i occurs in document j.

(ii) $LinkRank$ is a measure of how many links a concept has in common with the total of those contained in a document, thus it is a measure of the importance of the

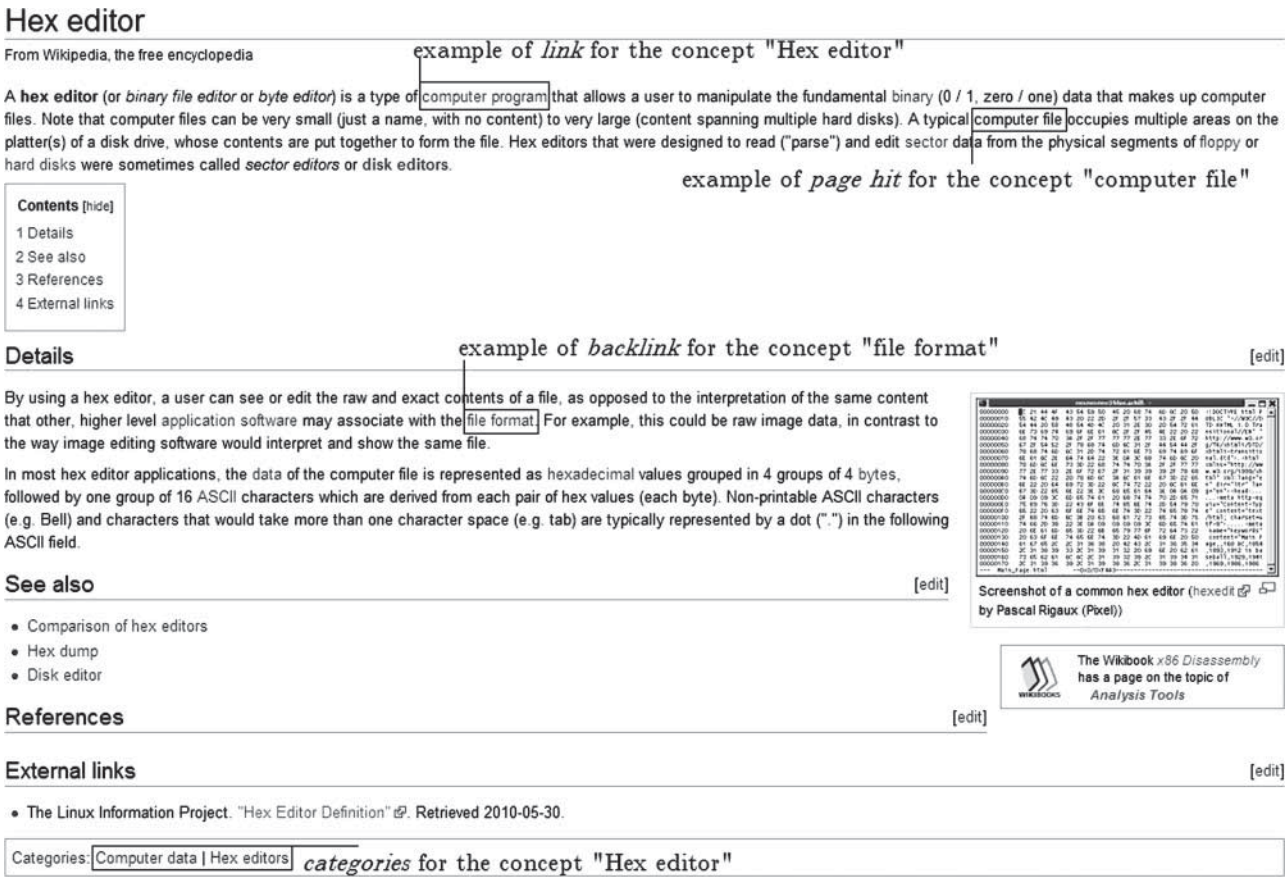

FIGURE 3. Example of Wikipedia concepts.

- $Content_i$: the corresponding Wikipedia article text
- $Links_i$: links from the corresponding article to other articles
- $BackLinks_i$: articles which have a link to the examined article
- $PageHits_i$: the articles in which the examined article (Noun Phrase) is simply present, either as link or not (plain text)
- $Categories_i$: the list of categories that the corresponding article belongs to

FIGURE 2. Concept features as extracted from the Wikipedia API.

concept to the document and is formally defined as

$$\text{LinkRank}_{j,i} = \frac{|\text{Links}_i \bigcap \text{Links}_{\text{Doc}_j}|}{|\text{Links}_{\text{Doc}_j}|}, \qquad (2)$$

where $Links_i$ is the set of Links of concept i, as defined in Fig. 2 and $Links_{Doc_j}$ is the set of Links of document j, defined as all the links of all concepts that represent document j.

(iii) *ConceptSim* is the similarity between the document and the article text of a concept contained in the document, computed in the classic term frequency − inverse document frequency (tf − idf) vector space, as defined in [1] and is given by the following equation:

$$\text{ConceptSim}_{j,i} = \cos(\mathbf{v}_j, \mathbf{v}_i), \qquad (3)$$

where $\mathbf{v}_j$ is the tf − idf vector of document j, $\mathbf{v}_i$ is the tf − idf vector of the Wikipedia article text corresponding to concept i and cos is the cosine function which computes the similarity between the two vectors.

(iv) $OrderRank$ is a measure which takes larger values for concepts that appear at the beginning of the document, based on the observation that important words often occur at the beginning of a document [19]. Formally, it is defined as

$$\text{OrderRank}_{j,i} = 1 - \frac{\text{arraypos}_i}{|j|}, \qquad (4)$$

where $arraypos$ is an array containing all words of the document in the order that they occur in the document, and $arraypos_i$ represents the position of the

first occurence of concept i in the array. If a concept consists of more than one word, then we take into consideration the position of occurrence of the first word of the concept and $|j|$ is the size of document j, i.e. how many words form the document.

Additionally, we define a global (document independent) measure for each concept defined as follows:

(v) Keyphraseness is a measure adapted from [20], which has a specific value for each different concept, regardless of the document we refer to, and is an indication of how much descriptive and specific to a topic a concept is. It is defined as

$$\text{Keyphraseness}(i) = \frac{\text{BackLinks}_i}{\text{PageHits}_i}. \quad (5)$$

A concept with high Keyphraseness value (i.e. most of its occurrences in Wikipedia are links to the corresponding article and not plain text) has more descriptive power than a concept with low Keyphraseness value, even if the latter may occur more times in Wikipedia, but less times as a link. Keyphraseness is normalized in the interval [0, 1], after the extraction of all concepts from all documents in the corpus, so that the highest Keyphraseness value is set to 1 and the lowest to 0.

3.2. Concept disambiguation process

If a candidate concept is polysemous, i.e. it has multiple meanings, it is necessary to perform word sense disambiguation to find its most proper meaning in the context where it appears. ConceptSim (as introduced by Equation (3)) is utilized to do explicit word sense disambiguation. It is therefore reminded that ConceptSim is based on the cosine similarity between the tf−idf vectors of the document and the examined concept, so the larger the value of ConceptSim, the higher is the similarity between the two corresponding text documents. Thus, the sense with the higher ConceptSim is the most appropriate for the examined document. However, in order to provide more accurate disambiguation results, we use in addition the categories that each concept belongs to, and we integrate it to the ConceptSim, creating a more robust measure of how similar a concept (c) is to the examined document (j), which will be called $SenseSim_{j,c}$ and is defined by the following equation:

$$\text{SenseSim}_{j,c} = \lambda * \text{ConceptSim}_{j,c} + (1 - \lambda) * \text{Dice}(\text{Categories}_c, \text{Categories}_{\text{Doc}_j}), \quad (6)$$

where $ConceptSim_{j,c}$ is given by Equation (3), $Categories_c$ shows the Categories of concept c as defined in Fig. 2, $Categories_{Doc_j}$ shows the Categories of document j, defined as all the categories of all (non-ambiguous) concepts that represent document j, λ is a parameter in [0, 1] to weight the text and category overlapping metrics and $Dice$ is the well-known co-efficient defined as

$$\text{Dice}(A, B) = \frac{2 * |A \bigcap B|}{|A + B|}. \quad (7)$$

TABLE 1. Disambiguation results for concept 'Client' in document #67480 of 20-NG collection.

'Client' senses	SenseSim
Client (computing)	0.0578
Client (ancient Rome)	0.0240
Client (band)	0.0170
Clients (album)	0.0168
Client (album)	0.0097

For instance, the 20-NG dataset, document #67480 belongs to the category comp.windows.x, and the concept "Client" in Wikipedia refers to several different senses of the word, as listed in Table 1. A small fragment of the context of document #67480 in which the word 'client' occurs is the following:

Since the server end is (or was) always at this end (California) it is faster to remotely run the client via DESQview X and have a short hop to the server than running the client locally and having a long hop to the server.

SenseSim measure is computed for each one of these senses, and the meaning with the larger value is selected to be part of the document representation (in this case it is obvious that the 'client' is used with its computing sense). To make more precise, the disambiguation process and to avoid enriching the document with not correct senses, it is possible to introduce a $SenseSim$ minimum threshold for replacing an ambiguous concept with its most proper sense (otherwise the concept will be dropped). In our experiments this threshold was set to 0.05.

3.3. Document representation

After completing the disambiguation process, we end up with a set of concepts that represent the document. Our goal is to construct a vector representation where each component corresponds to the importance of each concept in the document. As previously stated, each concept has four features related to the document, which are described by Equations (1) through (4), and one 'global' feature, which is described by Equation (5).

For instance, a small part of the document concept representation for the #67480 article from the 20-NG dataset is shown in Table 2. The measures WFreq, OrderRank, LinkRank and ConceptSim of the whole document are normalized in the interval [0, 1], as explained above for Keyphraseness. Note the ability to represent as a concept the words 'Word for Windows', which clearly refers to the well-known editor program, but which the BOW model would have broken into three words and would have led to a loss of descriptive value.

TABLE 2. Example representation vector [0, 1] normalized.

Concept	Wfreq	OrderRank	LinkRank	ConceptSim	Keyphraseness
Ethernet	0.3333	0.2919	1.0000	0.9499	0.6320
xserver	0	0.2948	0.4432	0.2759	0.3077
Traffic flow	0.3333	0.4711	0.2958	0.7869	0.1045
Word for Windows	0.6667	0.4032	0.3576	0.7278	0.0833
Mouse pointer	0.3333	0.3858	0.8342	0.7488	0.0460
Process (computing)	0	0.1647	0.4332	0.8365	0.0415
Client (computing)	0.6667	0.4350	0.4246	0.6661	0.0426
Network segment	0.3333	0.1055	0.7302	0.5041	0.7174
File server	0.3333	0.1604	0.4529	1.0000	0.6338

4. CONCEPTUAL HIERARCHICAL CLUSTERING

Generic clustering techniques have the disadvantage that, when applied to documents, they do not provide intrinsic textual descriptions of the clusters obtained, due to the fact that their algorithms were not designed specifically for text. On the other hand, existing conceptual clustering techniques for text are either known to be rather slow [21], or require extra steps to reduce the number of clusters [22].

Our clustering method extends the idea of frequent itemsets [14], aiming to provide a cluster description based on the Wikipedia concepts extracted from the corpus examined. The labels of each cluster are assigned during the clustering process using the Keyphraseness feature extracted from Wikipedia in order to define how descriptive a concept is for a cluster. The approach is hierarchical, i.e. the initial clusters are pruned and merged until a specific number of clusters given by the user is reached or until clusters become too dissimilar to be further processed.

Before proceeding with the clustering method, let us introduce some definitions:

(a) A *global important concept* is a concept that:
   (1) has a Keyphraseness value greater than a specific threshold, defined as *minimum keyphraseness threshold* and
   (2) appears in more than a minimum fraction of the whole document set, defined as the *minimum global frequency threshold*.

   A *global important k-concept-set* is a set of k global important concepts that appear together in a fraction of the whole document set greater than the minimum global frequency threshold.

(b) A global important concept is *cluster frequent* in a cluster C_m, if the concept is contained in some minimum fraction of documents assigned to C_m, defined as the *minimum cluster support*.

(c) The *cluster support* of a concept in a cluster C_m is the percentage of documents in C_m that contain this specific concept. The method consists of two steps. At the first step, initial clusters are constructed (based on the $Keyphraseness$ of concepts and on the frequency of concepts and concept-sets). At the second step, clusters get disjoint.

4.1. Initial clusters construction

In order to find the total weight of each concept in the document, we linearly combine the features of Equations (1)–(4). The final weight of concept i in document j is given by the following equation:

$$\begin{aligned}\text{Weight}(j,i) = {} & \alpha^{*}\text{WFreq}_{j,i} + \beta * \text{LinkRank}_{j,i} \\ & + \gamma * \text{OrderRank}_{j,i} \\ & + (1-\alpha-\beta-\gamma) * \text{ConceptSim}_{j,i}. \end{aligned} \quad (8)$$

The coefficients α, β and γ are determined by experiments and their value range is the interval [0,1]. In this way, we replace the usually sparse BOW model by a more compact concept model, which both reduces the vector space size (an important factor for processing large amounts of oversized documents) and enriches document features with external knowledge from Wikipedia.

Given the definitions (a) through (c) above, we can construct the initial clusters of the corpus. For every concept-set that corresponds to the restrictions of definition (a), we construct an initial cluster comprising all documents that contain this concept-set. It is obvious that initial clusters are not disjoint because one document may contain several global important concept-sets. The disjoining of clusters is carried out in the next section.

The *cluster label* of each cluster is defined by the global important concept-set that is contained in all documents assigned to the cluster.

4.2. Disjoining initial clusters

For each document, we define the most suitable initial cluster and attach the document only to this cluster. For this purpose, we

need to measure the 'goodness' of a cluster C_m for a document Doc_j by defining a function $\text{Score}(C_m \leftarrow \text{Doc}_j)$. The cluster with the highest value for this measure is selected as the unique cluster for the examined document. If there are more than one clusters that maximize the *Score* function, the one with the larger number of concept-sets at its label is chosen. The *Score* function is defined as follows:

$$\begin{aligned}\text{Score}(C_m \leftarrow \text{Doc}_j) &= \left[\sum_x \text{Weight}(j,x)\cdot \text{cluster\_support}(x)\right] \\ &\quad - \left[\sum_{x'} \text{Weight}(j,x')\cdot \text{Keyphraseness}(x')\right], \end{aligned} \quad (9)$$

where x represents a global important concept in Doc_j, which is cluster frequent in C_m, x' represents a global important concept in Doc_j, which is not cluster frequent in C_m, $\text{Weight}(j,x)$ is the weight of concept x in Doc_j as defined by Equation (8), $\text{Weight}(j,x')$ is similar as the above, $\text{cluster\_support}(x)$ is given by definition (c) and $\text{keyphraseness}(x')$ is given by Equation (5).

The first term of the score function rewards cluster C_m, if a global important concept x in Doc_j is cluster frequent in C_m. The importance of concept x in different clusters is captured by multiplying its weight in the document Doc_j by its cluster support in C_m. The second term of the function penalizes cluster C_m if a global important concept x' in Doc_j is not cluster frequent in C_m. The weight of x' in Doc_j is multiplied by its Keyphraseness value which expresses how important the concept is in general terms.

4.3. Building the cluster tree

In this stage, a cluster (topic) tree is constructed based on the similarity between clusters. In the cluster tree, each cluster (except from the cluster with the empty cluster label that lies at the top of the tree structure) has exactly one parent.

At this point, please recall that each cluster uses one global important k-concept-set as its label. Such clusters are called *k-clusters* below. In the cluster tree, the root node appears at level 0, which corresponds to the cluster with the label 'null' and collects all unclustered documents. The 1-clusters appear at level 1 and so forth for every level. The depth of the tree is equal to the maximum size of global important concept-sets.

The cluster tree is built bottom-up by choosing a parent at level $k-1$ for each cluster at level k. For each k-cluster C_m at level k, we first find all potential parents that are $(k-1)$-clusters and have a cluster label which is a subset of C_m's cluster label. There are at most k such potential parents. At the next step, we choose the 'best' parent. The *Score* function is used for this selection, but now we merge all documents in the subtree of C_m into a single document $\text{Doc}(C_m)$, and then compute the score of $\text{Doc}(C_m)$ against each potential parent. The parent with the highest score becomes the parent of C_m. All leaf clusters with no documents can be removed.

4.3.1. *Tree pruning*

A cluster tree can be broad and deep, depending on the minimum global threshold and the Keyphraseness values we define. Therefore, it is likely that documents are assigned to a large number of small clusters, which leads to poor accuracy. The aim of tree pruning is to merge similar clusters in order to create a more natural hierarchy for browsing and to increase clustering accuracy. Before introducing the pruning methods, we define a cluster similarity function, which is a key notion for merging and pruning procedures.

To measure the similarity between two clusters C_a and C_b, we treat one cluster as a document (by combining all the documents in the cluster) and measure its score using the *Score* function defined by Equation (7). The only differences are that the result has to be normalized so as to remove the effect of varying document size and that we have to compute both the similarity of C_a to C_b and the similarity of C_b to C_a. Formally, the similarity of a cluster C_b to C_a is defined as

$$\text{Sim}(C_a \longleftarrow C_b) = \frac{\text{Score}(C_a \longleftarrow \text{Doc}(C_b))}{\sum_x \text{Weight}(\text{Doc}(C_b),x) + \sum_{x'} \text{Weight}(\text{Doc}(C_b),x')} + 1, \quad (10)$$

where $\text{Doc}(C_b)$ stands for combining all the documents in the subtree of C_b into a single document, x represents a global important concept in $\text{Doc}(C_b)$, which is also cluster frequent in C_a, x' represents a global important concept in $\text{Doc}(C_b)$, which is not cluster frequent in C_a and $\text{Weight}(\text{Doc}(C_b),x)$ and $\text{Weight}(\text{Doc}(C_b),x)]$ are the weights of concepts x and x', respectively, in document $\text{Doc}(C_b)$

To explain the normalization by the denominator in (10), note that, in the *Score* function, the Cluster_Support and Keyphraseness take values in the interval [0,1]; thus the maximum value of the *Score* function would be $\sum_x \text{Weight}(j,x)$ and the minimum value $-\sum_{x'} \text{Weight}(j,x')$. So, after the proposed normalization, the value of Sim would be in the interval $[-1, 1]$. To avoid negative values for similarity, we add the term $+1$ and we end up with the above equation. Note that the range of the *Sim* function is [0,2].

The cluster similarity between C_a and C_b is computed as the geometric mean of the two normalized scores provided by Equation (8):

$$\text{Similarity}(C_a \longleftrightarrow C_b) = \sqrt{\text{Sim}(C_a \longleftarrow C_b) \times \text{Sim}(C_b \longleftarrow C_a)}. \quad (11)$$

The advantage of the geometric mean is that two clusters are considered to be similar only if both values $\text{Sim}(C_a \leftarrow C_b)$ and $\text{Sim}(C_b \leftarrow C_a)$ are high. The *Similarity* function has the same range as the Sim function, i.e. [0,2]. In our method, *Similarity*

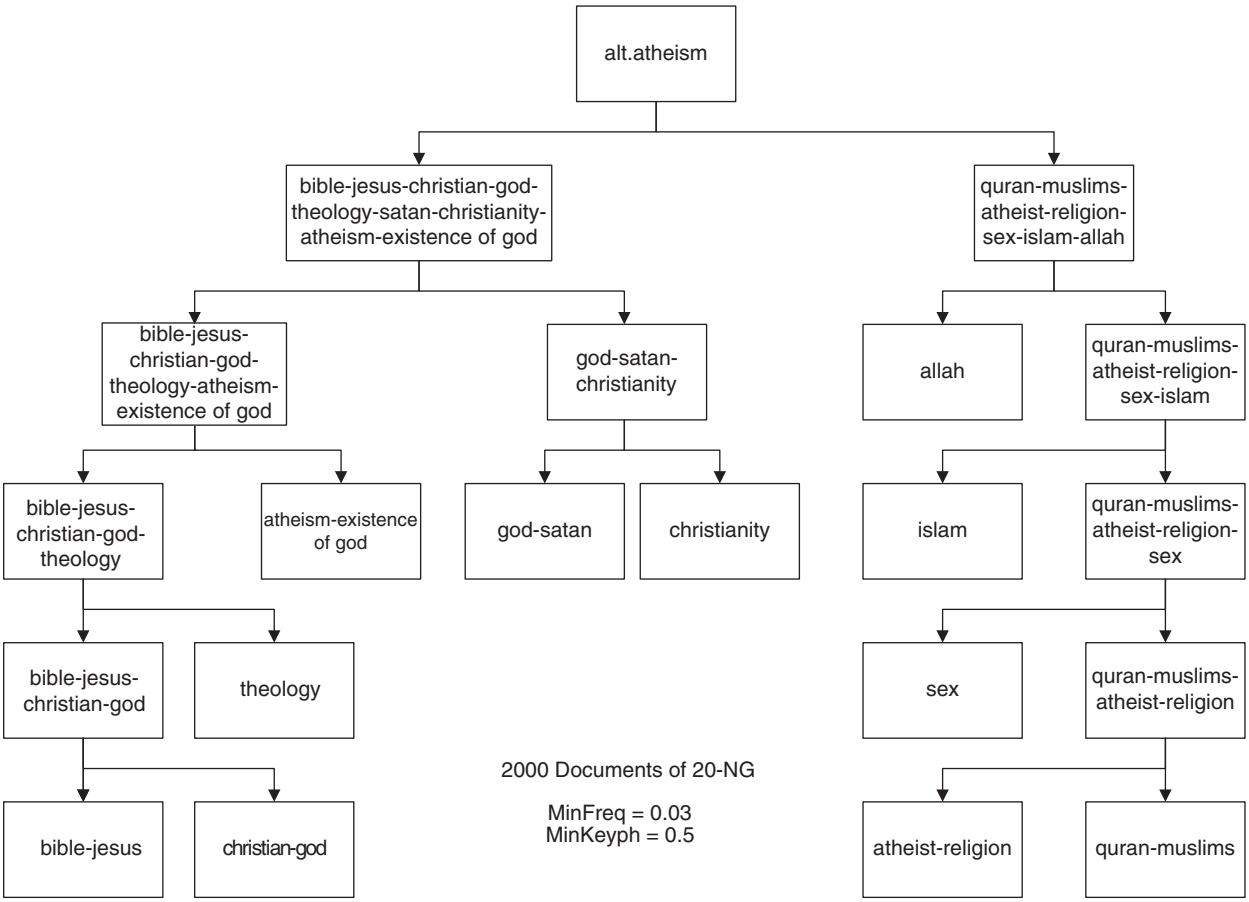

FIGURE 4. 20-NG alt.atheism category hierarchy example.

value 1 is considered the threshold for considering two clusters similar, although depending on the system requirements, this can be altered.

4.3.2. Child pruning and sibling merging

After the definition of the cluster similarity function, we are in a position to apply a pruning and merging technique in order to efficiently shorten the tree. The pruning criterion computes the $Similarity$ function between a child and its parent and is activated when the value of *Similarity* is larger than 1, i.e. the child is similar to its parent. The intuition behind this criterion is that, if a subtopic (e.g. God–atheism) is very similar to its parent topic (e.g. atheism), the subtopic is probably too specific and can be removed. We apply the criterion by scanning the tree in bottom-up order (up to level 2, since the root collects only unclustered documents), and if a cluster is pruned, its children become the children of their grandparent.

Sibling merging is a process applied to similar clusters at level 1 (recall that child pruning is not applied at this level). Each time, the *Similarity* value is calculated for each pair of clusters at level 1 and the cluster pair with the highest value is merged.

An example of the hierarchy created for the alt.atheism cluster of the 20-NG category is presented in Fig. 4. The figure shows how the small clusters merge until they form the final cluster, which will be described by all concepts that appear at labels of all clusters. Clusters with one concept as a label and no children (e.g. theology, sex, islam, etc) are, obviously, part of the initial cluster set. Child pruning has been applied to clusters with no children but with a label having more than one concept (e.g. Bible–Jesus, Christian–God, etc). Note that documents are organized in three subcategories with distinct topics: (a) Bible, Jesus, theology, existence of God (b) God, Satan and (c) Quran, Muslims, Islam.

Table 3 lists the labels of five 20-NG categories as produced by the CHC method. The important part is that although there is much overlap among the most important concepts (concepts that have both large Keyphraseness value and frequency) of each category, there is no overlap among the labels of the clusters, due to the high Keyphraseness value required for a cluster label. Notice that there are labels consisting of multiple words (such as ‘existence of God’ and ‘Window manager’). Also, there are labels which include acronyms (such as ‘Windows NT’ and ‘SCSI controller’).

TABLE 3. Cluster labels generated by CHC and most important concepts for some categories of 20-NG.

Category	Example of labels	Top-10 important concepts in terms of frequency and Keyphraseness value
alt.atheism	Atheism, Islam, existence of God, Quran	God, evidence, religion, atheist, Bible morality, Jesus, Satan, peace of God, death penalty
talk.religion.misc	Christianity, Branch Davidians	God, Jesus, Bible, Christians, Jews, Mormons, deity, Israelites, AMORC, Sermon
comp.os.ms-windows.misc	IBM, MSDOS, Windows NT, Unix	Windows, OS2, Microsoft, DOS, IBM, Memory, Mouse (computing), MSWindows, NDW, RAM
comp.sys.ibm.pc.hardware	SCSI controller, CMOS, Maxtor	Motherboard, SCSI, bus (computing), DOS, IRQ, ISA bus, CDROM, OS2, CPU, power supply
comp.windows.x	Window manager, SunOS, OpenWindows, xterm	SunOS, xterm, X-server, HP, bitmap, pixmap, event handler, source code, Xview, Xlib

5. EXPERIMENTS

We evaluated our method by comparing its effectiveness with two of the most standard and accurate document clustering techniques [23]: HAC and k-NN. HAC builds the corpus hierarchy from the individual documents by progressively merging clusters. The decision of which documents are merged to a cluster is taken using a metric about the distance (e.g. Euclidean). Moreover, the distance between clusters is measured in several ways, e.g. the maximum (or minimum or the mean) distance between the documents of each cluster. In the k-NN algorithm, each document is assigned to the cluster which is most common among its k nearest documents-neighbors (a distance metric is required, e.g. Euclidean distance). These methods are implemented with the use of CLUTO-2.0 Clustering Toolkit [24] and utilize the classic BOW representation for documents.

5.1. Datasets

Three well-known datasets were used for the evaluation of the proposed method: 20-NG, Reuters-21578 (both available from the UCI ML Repository[2]) and the Brown corpus (available from International Computer Archive of Modern and Medieval English[3]). 20-NG was fully described in Section 3.

Reuters-21578 [25] is the most widely used dataset for text categorization and clustering purposes. The collection documents appeared on the Reuters newswire in 1987 and were indexed with categories by several people. We chose to use articles that are uniquely assigned to exactly one topic (removing non-labeled data and documents without body) ending with $\sim 10\,000$ documents with more than 100 categories.

The Brown corpus [26] contains 500 documents published in 1961 representing written American English. Each document has more than 2000 words and the corpus covers a range of 15 genres (such as books on religion, skills and hobbies, academic texts, etc.).

[2]See http://kdd.ics.uci.edu/.
[3]See http://icame.uib.no/.

5.2. Evaluation criteria

To evaluate clustering quality, we adopt two quality measures widely used in text clustering bibliography [23], the *F-measure* and the *entropy*. The F-measure combines the *Precision* and *Recall* from the IR field. The precision, recall and F-measure of a cluster m with respect to a class l are defined as

$$P = \text{Precision}(l, m) = \frac{N_{l,m}}{N_m},$$
$$R = \text{Recall}(l, m) = \frac{N_{l,m}}{N_l}, \tag{12}$$
$$F(l) = \frac{2 \cdot P \cdot R}{P + R},$$

where $F(l)$ is the F-measure for class l, $N_{l,m}$ is the number of members of class l in cluster m, N_l is the number of members of class l and N_m is the number of members of cluster m.

With respect to class l, we consider the cluster with the highest F-measure to be the cluster that maps to class l, and its F-measure becomes the score for class l. The overall F-measure for the clustering result C is the weighted average of the F-measure for each class l:

$$F_C = \frac{\sum_l (|l| \times F(l))}{\sum_l |l|}. \tag{13}$$

Entropy measures how 'good' a cluster is in terms of homogeneity. The higher the homogeneity of a cluster, the lower the entropy is, and vice versa. For every cluster m the probability $p_{l,m}$ that a member of cluster m belongs to class l is computed. The entropy is then calculated using the standard equation

$$E_m = -\sum_l p_{l,m} \cdot \log(p_{l,m}), \tag{14}$$

where the sum is taken over all classes l. The total entropy for the final cluster set C is calculated as the sum of entropies of

each cluster by taking into account the size of each cluster:

$$E_C = -\sum_{m=1}^{|C|}\left(\frac{N_m}{N}\cdot E_m\right), \quad (15)$$

where $|C|$ is the number of final clusters N_m is the number of documents in cluster m and N is the total number of documents.

5.3. Experimental results

For each document of each dataset, we follow the procedure described in Fig. 1 and represent it using Wikipedia knowledge. For each Wikipedia concept that we map, we use Equation (8) to compute its weight in the document and keep a global hash with its Keyphraseness value.

We experimented with various values for the α, β and γ parameters of Equation (8) in order to define the effect of WFreq, LinkRank, OrderRank and ConceptSim on document representation. In our experiments, we varied each parameter (α, β and γ) from 0 to 1 (with step 0.1 and by taking into account that the four weighting parameters must sum up to 1) in order to find the effect of each feature. By this procedure, we found the optimal values (by checking the full grid of possible parameter values) which produced the best clustering results in terms of F-measure value. ConceptSim and RankLink have the biggest effect on document representation, whereas OrderRank has the smallest. The optimal values for the parameters of Equation (8) are shown in Table 4.

After completing the concept vector space document representation and the concept weight computation, we proceed with the clustering procedure as described in Section 4. We select the initial clusters by setting the *minimum keyphraseness threshold* (MinKeyph) and the *minimum global frequency* threshold (MinFreq) to values aiming to create clusters with descriptive labels. This is achieved through a large enough Keyphraseness threshold (which wipes out many general concepts) and through a relatively low *frequency* threshold (depending on the documents available and given the fact that concepts are generally not simple words). Experiments show that a value for MinKeyph around 0.5 always yields good results in different datasets, provided that there are at least a few hundreds of documents available.

TABLE 4. Optimal values.

Optimal values		
Parameter		Value
Wfreq	α	0.2
LinkRank	β	0.4
OrderRank	γ	0.1
ConceptSim	$1-\alpha-\beta-\gamma$	0.3
	MinFreq	$0.01 \div 0.06$
	MinKeyph	0.5

To find the optimal value for MinFreq, we have to capture the effect of dataset size on the algorithm results, and so we ran the following experiments on datasets with varying size of documents : On 20 datasets (resampled from the 20-NG dataset) containing a random number of documents (<5000), we checked for the optimum value for MinFreq (by varying its value and checking which one produced the best F-measure). The value range of optimal MinFreq was [0.025, 0.07] and the average value was 0.045 (0.013 SD). Then, we performed the same process on 20 datasets (again resampled from the 20-NG dataset) containing a random number of documents (>5000). This time the value range of optimal MinFreq was [0.005, 0.035] and the average value was 0.02 (0.009 SD). Thus, MinFreq should be set between 0.03 and 0.06 for datasets with <5000 documents, otherwise MinFreq should be set between 0.01 and 0.03. All optimal parameters are shown in Table 4 (for MinFreq the union of the two value ranges is presented, i.e. [0.01, 0.06]). Figures 5 and 6 depict the F-measure and entropy values of CHC, respectively, with respect to the selected MinFreq value for the three initial datasets (MinKeyph was set to the default value 0.5 for these experiments).

The clustering results in comparison with those of HAC and k-NN, for the 20-NG and Reuters datasets, are shown in Table 5. The improvements shown were achieved using the parameters of Table 4. For the HAC method, the UPGMA variant was implemented, while in the k-NN method k was set to 8 and the similarity threshold to 0.25.

We also studied the runtime of our algorithm in comparison with that of the k-NN method (the HAC performance is much inferior and so it is not included in this experiment). For this experiment we used the 20-NG dataset (it has the largest number of documents among the three datasets used) and the Brown corpus (it has the documents with the most words). The results of this experiment are presented in Figs 7 and 8. The proposed CHC method greatly improves runtime over the baseline k-NN method in both datasets, mainly due to the transition from the large and sparse BOW vector space to the dimensionally reduced and semantically richer concept vector space. Both methods perform the same way in smaller datasets, but when the number of documents increases, CHC performs better. The improvement in terms of execution time is greater in the 20-NG dataset due to the fact that the Brown corpus contains word-rich documents, and thus many Wikipedia mapped concepts.

5.4. Complexity analysis

Our method involves two major phases: transition from the BOW vector space to the concept vector space and the clustering process (CHC). The process of matching document words to Wikipedia concepts can be carried out on the fly, which is important due to the rapid changes applied to Wikipedia daily, although the same process can be applied using any Wikipedia

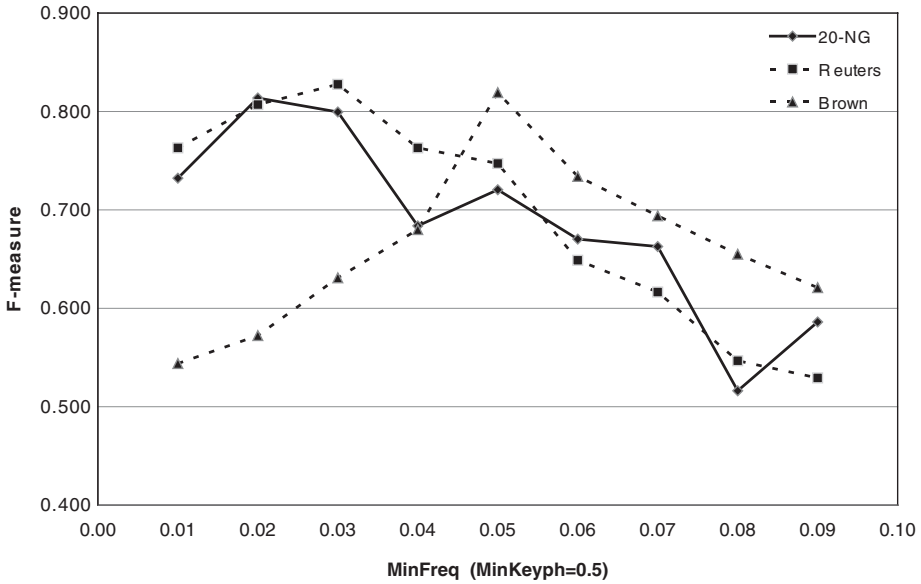

FIGURE 5. Sensitivity of F-measure to MinFreq.

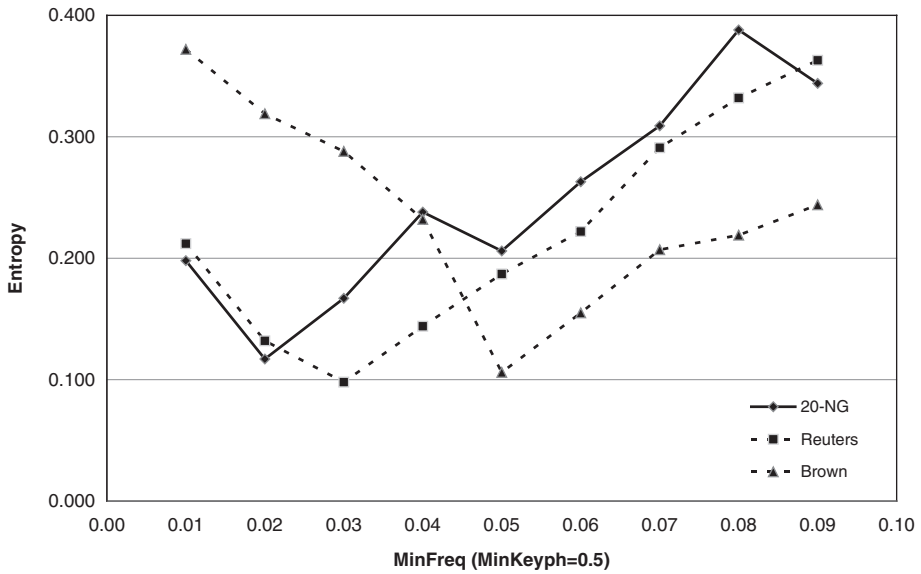

FIGURE 6. Sensitivity of entropy to MinFreq.

TABLE 5. Experimental results.

Dataset	Algorithm	F-measure	Improvement	Entropy	Improvement
Reuters	HAC	0.452	80.09%	0.351	−66.67%
	k-NN	0.671	21.31%	0.297	−60.61%
	CHC	**0.814**		**0.117**	
20-NG	HAC	0.521	58.92%	0.339	−71.09%
	k-NN	0.737	12.35%	0.248	−60.48%
	CHC	**0.828**		**0.098**	
Brown	HAC	0.582	40.72%	0.387	−72.61%
	k-NN	0.717	14.23%	0.262	−59.54%
	CHC	**0.819**		**0.106**	

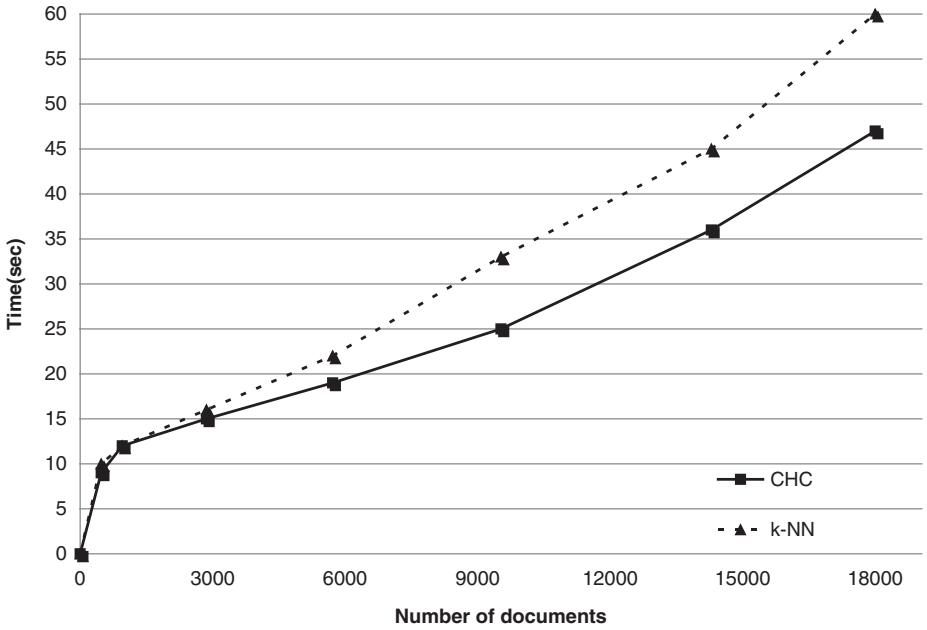

FIGURE 7. Performance of the CHC method using the 20-NG dataset.

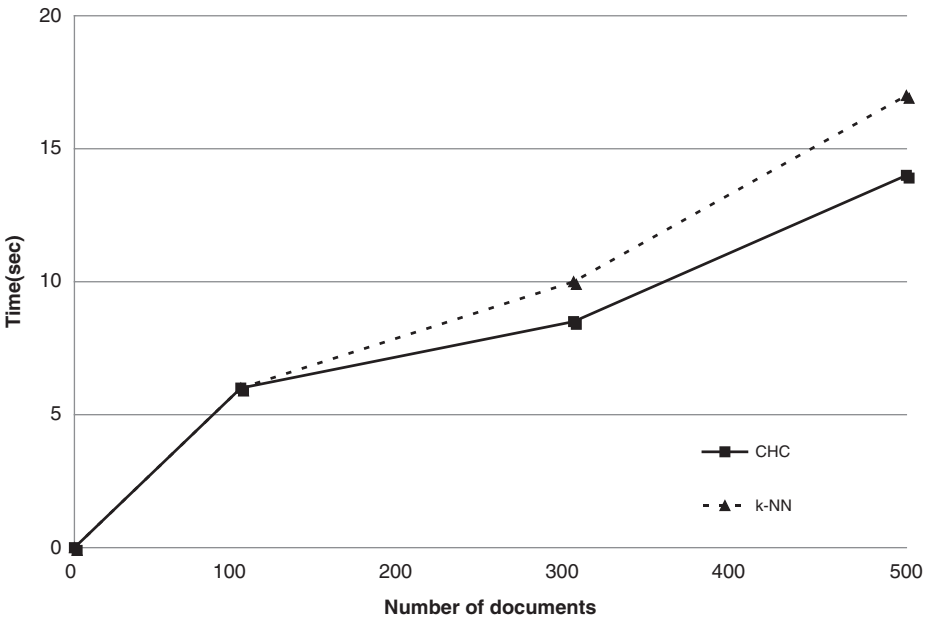

FIGURE 8. Performance of the CHC method using the Brown corpus.

dump available for downloading and local exploitation. At the same time, the disambiguation process (around 20% of each document's concepts are ambiguous) concludes the concept extraction. The next step is to compute the weights of concepts in each document (according to Equation (8)), which is carried out off-line given the fact that during concept matching, the features described in Fig. 2 are stored locally.

It must be pointed out that the document concept vector is not so large and sparse as the term vector (BOW model). For example in the 20-NG dataset, the BOW vector space's dimension is 61 174, whereas the concept space's dimension is only 22 510.

The initial clustering process requires the document vectors to be scanned twice, once for finding the global important concepts and once for disjoining the clusters, but keeping in mind that due to the dimensionality reduction this process is faster than it would be using the classic BOW model (CHC needs about half of BOW time).

As far as the tree construction is concerned, the process of finding a parent for a $k - cluster\ C$ at the $k - 1$ level requires to examine at most k clusters since C's potential parents have labels which are subsets of C's label (Section 4.3). This process requires the computation of specific cluster similarities (using Equations (10) and (11)) but k is usually very small, so the execution time remains linear. Finally, child pruning requires only one scan of clusters, and sibling merging is applied only at clusters of the highest level of the tree (level 1).

6. CONCLUSIONS: FUTURE WORK

In this paper, we proposed a novel method for CHC of documents using knowledge extracted from Wikipedia. The proposed method exploits Wikipedia textual content, link and category structure in order to create a rich and compact document representation which is built real-time using the Wikipedia API. A document clustering process extends the idea of frequent items by using Wikipedia knowledge for assigning cluster labels that are descriptive and significant to the examined corpus. The clustering approach is hierarchical, thus a tree structure is constructed. Experiments on three literature datasets (20-NG, Reuters, Brown) show that the proposed technique is faster and more accurate than the baseline approach.

We are currently investigating ways to improve the proposed clustering technique. As far as the clustering method is concerned, the accuracy of the similarity function between documents and clusters can be improved through the introduction of different strategies. In addition, the idea of 'cooperation', introduced by self-organizing Maps is examined as a technique, which will improve the quality of clustering. Another interesting direction is to apply the same concept-based representation model to other applications based on document similarity measurements such as text classification and IR tasks.

REFERENCES

[1] Salton, G., Wong, A. and Yang, C.S. (1975) A vector space model for automatic indexing. *Commun. ACM*, **18**, 613–620.

[2] Pullwitt, D. (2002) Integrating contextual information to enhance som-based text document clustering. *Neural Netw.*, **15**, 1099–1106.

[3] Breaux, T.D. and Reed, J.W. (2005) Using Ontology in Hierarchical Information Clustering. *HICSS '05: Proc. 38th Annual Hawaii Int. Conf. System Sciences (HICSS'05)—Track 4*, Washington, DC, USA, pp. 111–112. IEEE Computer Society.

[4] Sedding, J. and Kazakov, D. (2004) Wordnet-based Text Document Clustering. *ROMAND '04: Proc. 3rd Workshop RObust Methods in Analysis of Natural Language Data*, Morristown, NJ, USA, pp. 104–113. Association for Computational Linguistics.

[5] Li, Y., Luk, W.P.R., Ho, K.S.E. and Chung, F.L.K. (2007) Improving Weak Ad-Hoc Queries Using Wikipedia Asexternal Corpus. *SIGIR '07: Proc. 30th Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, New York, NY, USA, pp. 797–798. ACM.

[6] Gabrilovich, E. and Markovitch, S. (2006) Overcoming the Brittleness Bottleneck Using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. *AAAI'06: Proc. 21st National Conf. Artificial Intelligence*, Boston, MA, USA, pp. 1301–1306. AAAI Press.

[7] Wang, P., Hu, J., Zeng, H.-J. and Chen, Z. (2009) Using wikipedia knowledge to improve text classification. *Knowl. Inf. Syst.*, **19**, 265–281.

[8] Hu, X., Zhang, X., Lu, C., Park, E.K. and Zhou, X. (2009) Exploiting Wikipedia as External Knowledge for Document Clustering. *KDD '09: Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 389–396. ACM.

[9] Bloehdorn, S., Cimiano, P. and Hotho, A. (2006) Learning Ontologies to Improve Text Clustering and Classification. *From Data and Information Analysis to Knowledge Engineering: Proc. 29th Annual Conf. German Classification Society (GfKl 2005), March 9–11, Magdeburg, Germany*, Studies in Classification, Data Analysis, and Knowledge Organization **30**, pp. 334–341. Springer, Berlin–Heidelberg, Germany.

[10] Gabrilovich, E. and Markovitch, S. (2007) Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. *IJCAI'07: Proc. 20th Int. joint Conference Artificial intelligence*, San Francisco, CA, USA, pp. 1606–1611. Morgan Kaufmann Publishers Inc.

[11] Banerjee, S., Ramanathan, K. and Gupta, A. (2007) Clustering Short Texts Using Wikipedia. *SIGIR '07: Proc. 30th Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, New York, NY, USA, pp. 787–788. ACM.

[12] Wang, P. and Domeniconi, C. (2008) Building Semantic Kernels for Text Classification Using Wikipedia. *KDD '08: Proc. 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 713–721. ACM.

[13] Hu, J., Fang, L., Cao, Y., Zeng, H.-J., Li, H., Yang, Q. and Chen, Z. (2008) Enhancing Text Clustering by Leveraging Wikipedia Semantics. *SIGIR '08: Proc. 31st Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, New York, NY, USA, pp. 179–186. ACM.

[14] Fung, B.C.M., Wang, K. and Ester, M. (2003) Hierarchical Document Clustering using Frequent Itemsets. *Proc. 3rd SIAM Int. Conf. Data Mining (SDM)*, San Francisco, CA, USA, May, pp. 59–70. SIAM.

[15] Wang, B.B., Mckay, R.I.B., Abbass, H.A. and Barlow, M. (2003) A Comparative Study for Domain Ontology Guided Feature Extraction. *ACSC '03: Proc. 26th Australasian Computer Science Conf.*, Darlinghurst, Australia, pp. 69–78. Australian Computer Society, Inc.

[16] Schmid, H. (1994) Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proc. Int. Conf. New Methods in Language Processing*, Manchester, UK. Routledge.

[17] Marcus, M.P., Marcinkiewicz, M.A. and Santorini, B. (1993) Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, **19**, 313–330.

[18] Lang, K. (1995) Newsweeder: Learning to Filter Netnews. *Proc. Int. Conf. Machine Learning*, Tahoe City, CA, USA. Morgan Kaufmann.

[19] Xue, X.-B. and Zhou, Z.-H. (2009) Distributional features for text categorization. *IEEE Trans. Knowl. Data Eng.*, **21**, 428–442.

[20] Mihalcea, R. and Csomai, A. (2007) Wikify!: Linking Documents to Encyclopedic Knowledge. *CIKM '07: Proc. 16th ACM Conf. Information and Knowledge Management*, New York, NY, USA, pp. 233–242. ACM.

[21] Stumme, G., Taouil, R., Bastide, Y., Pasquier, N. and Lakhal, L. (2002) Computing iceberg concept lattices with titanic. *Data Knowl. Eng.*, **42**, 189–222.

[22] Hotho, A. and Stumme, G. (2002) Conceptual Clustering of Text Clusters. *Proc. FGML Workshop*, Hanover, Germany, pp. 37–45. Special Interest Group of German Informatics Society (FGML—Fachgruppe Maschinelles Lernen der GI e.V.).

[23] Steinbach, M., Karypis, G. and Kumar, V. (2000) A Comparison of Document Clustering Techniques. In Grobelnik, M., Mladenic, D. and Milic-Frayling, N. (eds) *KDD-2000 Workshop on Text Mining*, Boston, MA, USA, pp. 109–111. AAAI Press.

[24] Karypis, G. (2002) Cluto Clustering Toolkit. Technical Report, 02-017.

[25] Carnegie Group, I. and Reuters, L. (1997) *Reuters-21578 Text Categorization Test Collection*. Carnegie Group, Inc. and Reuters, Ltd, Academic Press.

[26] Francis, W.N. and Kucera, H. (1964) *A Standard Corpus of Present-Day Edited American English, for use with Digital Computers (Brown)*. Academic Press, Providence, RI.