Evaluating Top-K Approximate Patterns via Text Clustering

Claudio Lucchese[1(✉)], Salvatore Orlando[1,2], and Raffaele Perego[1]

[1] ISTI-CNR, Pisa, Italy
claudio.lucchese@isti.cnr.it
[2] DAIS - Università Ca' Foscari Venezia, Venice, Italy

Abstract. This work investigates how approximate binary patterns can be objectively evaluated by using as a proxy measure the quality achieved by a text clustering algorithm, where the document features are derived from such patterns. Specifically, we exploit approximate patterns within the well-known FIHC (Frequent Itemset-based Hierarchical Clustering) algorithm, which was originally designed to employ exact frequent itemsets to achieve a concise and informative representation of text data. We analyze different state-of-the-art algorithms for approximate pattern mining, in particular we measure their ability in extracting patterns that well characterize the document topics in terms of the quality of clustering obtained by FIHC. Extensive and reproducible experiments, conducted on publicly available text corpora, show that approximate itemsets provide a better representation than exact ones.

1 Introduction

Clustering is one of the most studied fields in text mining. Text data are characterized by high dimensionality, and this greatly influences the scalability and the effectiveness of clustering algorithms. One of the most common approaches to reduce dimensionality is to exploit stemming and removal of stop words, and, finally, use a vector representation of documents, where the presence of each term in a vector is weighted by using, for example, TF-IDF. However, stemming and removal of stop words only alleviate the curse of dimensionality, due to the large vocabulary of terms mentioned in a typical document corpus. A technique pursued by seminal papers (see [1,12]) is to exploit an algorithm for mining Frequent Itemset Mining (FIM) to extract a reduced set of more meaningful features, with the aim of shrinking the vectorial representation of documents. In particular, in this paper we focus on the FIHC algorithm (Frequent Itemset-based Hierarchical Clustering), proposed by Fung et al. in their influential paper [3].

The idea of FIHC is to first mine the frequent itemsets, namely *word-sets*, that co-occur in the corpus with a frequency not less than a pre-determined *minimum support* threshold, thus leading to a set of top-k most frequent word-sets. FIHC exploits the extracted frequent word-set to determine an initial clustering of documents, based on the documents that contain or overlap those patterns. The collection of mined word-sets naturally identifies a vocabulary of frequent

S. Madria and T. Hara (Eds.): DaWaK 2016, LNCS 9829, pp. 114–127, 2016.
DOI: 10.1007/978-3-319-43946-4_8

terms, which determines the dimensionality of the vectors used to represent each document. This representation is then used to recursively merge the initial clusters until the desired number of clusters is obtained.

The quality of the top-k word-sets exploited by FIHC impacts on the quality of the resulting clustering. For instance, changing the minimum support threshold has the effect of changing the set of top-k most frequent word-sets, and thus impacts on the feature space into which documents are mapped, as well as on the seed clusters initially identified by each word-sets. Specifically, the smaller the minimum support, the larger the set of the word-sets extracted. In fact, a large support threshold may result in a reduced set of word-sets that occur in many documents and thus having a loose discriminative power. On the other hand, a small threshold may result in too many patterns and, consequently, a large vocabulary of words used in the vectorial representation of documents, again suffering from the curse of dimensionality during clustering.

We propose to use the clustering quality as a *proxy measure* to quantitatively evaluate the quality of different kinds of patterns that are used to feed the FIHC algorithm. In particular, we study *approximate patterns* that identify itemsets that are approximately included in the corresponding sets of transactions [7,9,13]. This means that, given an approximate pattern, some *false positives* are allowed, i.e., some of the items included in the patterns may not occur in a few transactions supporting the pattern. While the *exact* patterns are commonly ranked according to the popular concept of frequency in the collection, the alternative *approximate* patterns we study in this paper are ranked according to different definitions of importance.

To limit the number of patterns used to model documents and identity the initial clusters, for both the exact and approximate cases we select the top-k one. Whereas for frequent ones these top-k patterns are simply the most frequent ones, an approximate pattern algorithm aims at discovering the set of k patterns that best *describes/models*, the input dataset. State-of-the-art algorithms differ in the formalization of the above concept of *dataset description*. For instance, in [9] the goodness of the description is given by the number of occurrences in the dataset incorrectly modeled by the extracted patterns, while shorter and concise patterns are promoted in [7,13]. The goodness of a description is measured with some cost function, and the top-k mining task is casted into an optimization of such cost. In most of such formulations, the problem is proved to be NP-hard, and greedy strategies are therefore adopted. At each iteration, the pattern that best optimizes the given *cost function* is added to the solution. This is repeated until k patterns have been found or until it is not possible to improve the cost function.

In this paper we study the quality of document clustering achieved by exploiting the approximate top-k patterns extracted by three state-of-the-art algorithms: Asso [9], Hyper+ [13] and PaNDa$^+$ [8], where the cost functions adopted by Asso [9] and Hyper+ [13] share important aspects that can be generalized into a unique formulation. The PaNDa$^+$ framework can be plugged with such generalized formulation, which makes it possible to greedily mine approximate patterns according to several cost functions, including the ones proposed in [7,10]. PaNDa$^+$ also allows to include maximum noise constraints [2].

Concerning the evaluation methodology of these patterns, we adopt the quality of the clustering obtained by FIHC as a proxy of the quality of the patterns extracted by the various algorithms. Specifically, we use the aforementioned mining algorithms to extract top-k approximate patterns sets, which are then used to fed FIHC in order to cluster the input documents. Several commonly used "external" measures are used to evaluate the goodness of the pattern-based clusters with respect to the true classes of the documents. Moreover, we also compared such methods with a couple of baselines, i.e., K-MEANS and a version of FIHC exploiting classical frequent word-sets (exact patterns).

The main contribution of this paper is an extensive evaluation of approximate patterns. Our investigation shows that approximate patterns provide a better representation of the given dataset than exact patterns, and that PANDA$^+$ generates patterns of better quality than other state-of-the-art algorithms. In other words, our experiments shows that PANDA$^+$ seems to be able to better capture the patterns/features characterizing the most salient topics being discussed in the given corpus of documents.

The rest of the paper is organized as follows. Section 2 discusses exact and approximate pattern mining, and briefly introduces some algorithms for top-k approximate pattern miming. Section 3 discusses the clustering algorithm FIHC, and the possible exploitation within FIHC of either frequent or approximate patterns. In Sect. 4 we describe the experimental setting and the quality of document clustering identified by the various versions of FIHC and the baselines. Finally, Sect. 5 draws some concluding remarks.

2 Approximate and Exact Patterns

The *binary representation* of a transactional dataset, indeed a multi-set of itemset where each itemset is a subset of a given collection of items $\mathcal{I}$, is convenient to introduce pattern mining extracted from textual datasets. A *transactional dataset* of N transactions and M items – which is analogous to representing a corpus of N documents with a vocabulary of M terms as a collections of "sets of words", thus ignoring the positions and the number of occurrences of each term in a document – can be represented by a *binary* matrix $\mathcal{D} \in \{0,1\}^{N \times M}$, where $\mathcal{D}(i,j) = 1$ if the j^{th} item occurs in the i^{th} transaction, and $\mathcal{D}(i,j) = 0$ otherwise.

An *pattern* P is thus identified by a set of items, along with the set of transactions where the items occur. In terms of text documents, P is a word-set occurring in a given set of documents. We represent these two sets as binary vectors $P = \langle P_I, P_T \rangle$, where $P_I \in \{0,1\}^M$ and $P_T \in \{0,1\}^N$ are the *indicator vectors* of two subsets of items and transactions, respectively. The outer product $P_T \cdot P_I^\mathsf{T} \in \{0,1\}^{N \times M}$ identifies a sub-matrix of $\mathcal{D}$. These patterns are also called *hyper-rectangles* [13]: each pattern can be visualized as a rectangle if we properly reorder rows (transactions) and columns (items) to make them contiguous.

If a pattern is *exact*, the sub-matrix only covers 1-bits in $\mathcal{D}$, where $\|P_I\|$ is the *length* of the pattern and $\|P_T\|$ is its *support*, with $\|\cdot\|$ being the L^1-norm (or

Hamming norm) that simply counts the number of 1 bits in each binary vector. Conversely, in case a pattern is approximate, it only approximately covers 1-bits in $\mathcal{D}$ (*true positives*), but it may also cover a few 0-bits too (*false positives*). Still we have that $\|P_I\|$ is the *length* of the patten, and $\|P_T\|$ is its *approximate support*.

2.1 Exact Closed Patterns

Let $\Pi^\sigma = \{P_1, \ldots, P_{|\Pi^\sigma|}\}$ be a set of exact frequent patterns, where σ is the *minimum support ratio*. These patterns may *overlap*, since they may share items or transactions. Therefore, $\forall P \in \Pi^\sigma$, $P = \langle P_I, P_T \rangle$, we have that $\frac{\|P_T\|}{N} \geq \sigma$, where N is the number of documents in the corpus, represented as $\mathcal{D}$.

In this paper, we exploit the popular concept of *closed* frequent patterns, by removing from Π^σ some redundant patterns, since this also prevents the creation of redundant initial seed clusters used by FIHC. Specifically, a pattern $P_i \in \Pi^\sigma$, $P_i = \langle P_I^i, P_T^i \rangle$, is said closed *iff* $\nexists P_j \in \Pi^\sigma$, $P_j = \langle P_I^j, P_T^j \rangle$, such that $set(P_I^j) \subset set(P_I^i)$[1] and $P_T^i = P_T^j$. In other words, we maintain in Π^σ only the frequent itemsets such that there is no other super-itemset occurring in exactly the same set of transactions.

The number of frequent closed item sets may be orders of magnitudes smaller than all the frequent ones, still providing the same information: frequent itemsets can be in fact derived from closed ones. We denote by $\widehat{\Pi}^\sigma$, where $\widehat{\Pi}^\sigma \subseteq \Pi^\sigma$, the set of closed patterns given a minimum support σ. Several frequent closed itemsets mining algorithm [5,14] can be used to mine $\mathcal{D}$.

Since we need to limit the set of (closed) patterns to the top-k most frequent ones, we first select the largest σ_k such that:

$$\sigma_k = \underset{\sigma}{\operatorname{argmax}} |\widehat{\Pi}^\sigma| \geq k \tag{1}$$

and then select the top-k in $\widehat{\Pi}^{\sigma_k}$, denoted by $\widehat{\Pi}_k^{\sigma_k}$, where the patterns in $\widehat{\Pi}_{\sigma_k}$ are first sorted in decreasing order of support (and then of pattern length). This minimum support σ_k used to identify $\widehat{\Pi}_k^{\sigma_k}$, is then employed by FIHC within specific similarity measures.

In this work, we thus exploit such top-k closed frequent itemsets as a baseline of the possible pattern-based features used to model the text documents that feed the FIHC algorithm.

2.2 Approximate Patterns

Let $\Pi = \{P_1, \ldots, P_{|\Pi|}\}$ be a set of approximate overlapping patterns that aim at best describing/modelling the input dataset $\mathcal{D}$. This means that Π *approximately cover* the 1's in dataset $\mathcal{D}$, except for some noisy item occurrences, identified by matrix $\mathcal{N} \in \{0,1\}^{N \times M}$:

[1] $set(\cdot)$ takes an indicator vector and returns the corresponding subset.

$$\mathcal{N} \;=\; \bigvee_{P \in \Pi} (P_T \cdot P_I^{\mathsf{T}}) \quad \veebar\; \mathcal{D}. \tag{2}$$

where $\vee$ and $\veebar$ are respectively the element-wise *logical or* and *xor* operators. Note that some 1-bits in $\mathcal{D}$ may not be covered by any pattern in Π (*false negatives*).

Indeed, our formulation of noise (matrix $\mathcal{N}$) models both *false positives* and *false negatives*. If an occurrence $\mathcal{D}(i,j)$ corresponds to either a false positive or a false negative, we have that $\mathcal{N}(i,j) = 1$.

We define the top-k approximate pattern discovery problem as an optimization one, where the goal is to minimize a given cost function $J(\Pi_k, \mathcal{D})$:

$$\overline{\Pi}_k = \operatorname*{argmin}_{\Pi_k} J(\Pi_k, \mathcal{D}) \tag{3}$$

A general formulation of the cost function J is the following:

$$J(\Pi_k, \mathcal{D}) \;=\; \gamma_{\mathcal{N}}(\mathcal{N}) \;+\; \rho \cdot \sum_{P \in \Pi_k} \gamma_P(P) \tag{4}$$

where $\mathcal{N}$ is the noise matrix defined by Eq. 2, $\gamma_{\mathcal{N}}$ and γ_P are user defined functions measuring the cost of encoding noise and pattern descriptions, respectively. Constant $\rho \geq 0$ works as a regularization factor weighting the relative importance of the patterns cost. It is worth noting that such cost J is directly proportional to the complexity of the pattern set and the amount of noise, respectively.

The various algorithms for top-k patterns greedily optimize a specialization of the function of Eq. 4. In addition, they exploit some specific *parameters*, whose purpose is to make the pattern set Π_k subject to particular *constraints*, with the aim of (1) reducing the algorithm search space, or (2) possibly avoiding that the greedy generation of patterns brings to local minima. As an example of the former type of parameters, we mention the frequency of the pattern. Whereas, for the latter type of parameters, an example is the amount of false positives we can tolerate in each pattern. Table 1 summarizes the specialization of the generalized cost function.

In the following, we briefly discuss some state-of-the-art algorithms for top-k approximate pattern mining, in turn used to select a significant set of features modelling documents to be clustered by FIHC.

Asso [9] is a greedy algorithm that minimizes function J_A in Table 1, which only measures the amount of noise in describing the input data matrix $\mathcal{D}$. Note that this noise, namely $\gamma_{\mathcal{N}}(\mathcal{N}) = \|\mathcal{N}\|$, is measured as the L^1-norm $\|\mathcal{N}\|$ (or Hamming norm), which simply counts the number of 1 bits in matrix $\mathcal{N}$. Indeed, Asso aims at finding a solution for the *Boolean matrix decomposition problem*, thus identifying two low-dimensional factor binary matrices of rank k, such that their *Boolean product* approximates $\mathcal{D}$. The authors of Asso called this matrix decomposition problem the Discrete Basis Problem (DBP). It can be shown that the DBP problem is equivalent to the approximate top-k pattern mining problem when optimizing J_A. Asso works as follows. First, it creates a set of candidate item sets by extending each item with every other item having correlation grater

Table 1. Objective functions for Top-k Pattern Discovery Problem.

Cost function	Specialization	Description
$J_A(\Pi_k, \mathcal{D})$	$\gamma_{\mathcal{N}}(\mathcal{N}) = \|\mathcal{N}\|$	Minimize noise [9]
	$\gamma_P(P) = 0$	
	$\rho = 0$	
$J_H(\Pi_k, \mathcal{D})$	$\gamma_{\mathcal{N}}(\mathcal{N}) = 0$	Minimize pattern set complexity [13]
	$\gamma_P(P) = \|P_T\| + \|P_I\|$	
	$\rho = 1$	
$J_P(\Pi_k, \mathcal{D})$	$\gamma_{\mathcal{N}}(\mathcal{N}) = \|\mathcal{N}\|$	Minimize noise and pattern set complexity [6,7]
	$\gamma_P = \|P_T\| + \|P_I\|$	
	$\rho = 1$	
$J_P^{\overline{\rho}}(\Pi_k, \mathcal{D})$	$\gamma_{\mathcal{N}}(\mathcal{N}) = \|\mathcal{N}\|$	Extend J_P to leverage the trade-off between noise and pattern set complexity
	$\gamma_P(P) = \|P_T\| + \|P_I\|$	
	$\rho = \overline{\rho}$	
$J_E(\Pi, \mathcal{D})$	$\gamma_{\mathcal{N}}(\mathcal{N}) = \mathsf{enc}(\mathcal{N})$	Minimize the encoding length [11] of the pattern model according to [10]
	$\gamma_P(P) = \mathsf{enc}(P)$	
	$\rho = 1$	

than a given parameter τ. Then Asso iteratively selects a pattern from the candidate set by greedily minimizing the J_A.

Hyper+ [13] is a two-phase algorithm aiming at minimizing function J_H in Table 1, which only considers the pattern set complexity. Specifically, the complexity of each pattern in $P \in \Pi_k$ is measured by $\gamma_P(P) = \|P_T\| + \|P_I\|$. In the first phase, the algorithm aims to cover in the best way all the items occurring in $\mathcal{D}$, with neither false negatives nor positives, and thus without any noise. The rationale is to promote the simplest description of the whole input data $\mathcal{D}$, without any constraint on the amount k of patterns. For this first phase Hyper+ uses a collection of frequent item sets, for a given minimum support parameter σ. In the second phase, pairs of patterns previously extracted are recursively merged as long as a new collection of approximate patters can be obtained without generating an amount of *false positive* occurrences larger than a given budget β. Finally, since the pattern set produced by Hyper+ is ordered (from most to least important), we can simply select Π_k as the top-listed k patterns, as done by the algorithm authors in Sect. 7.4 of [13]. Note that this also introduces false negatives, corresponding to all the occurrences $\mathcal{D}(i, j) = 1$ in the dataset that remain uncovered after selecting only the top-k patterns.

Finally, we considered PaNDa$^+$, a pattern mining framework [8] that can be plugged in with all the cost functions in Table 1, including the last three functions J_P, $J_P^{\overline{\rho}}$, and J_E, which can fully leverage the trade-off between patterns

description cost and noise cost. In particular, J_E, originally proposed in [10], realizes the MDL principle [11]. The regularities in $\mathcal{D}$, corresponding to the discovered approximate patterns Π_k, are used to *lossless compress* the whole $\mathcal{D}$, expressed as pattern model and noise, as in Eq. 2. Hence, the best pattern set Π_k is the one that induces the smallest encoding of $\mathcal{D}$, namely J_E. PANDA$^+$ adopts a greedy strategy by exploiting a two-stage heuristics to iteratively select each pattern: (a) discover a noise-less pattern that covers the yet uncovered 1-bits of $\mathcal{D}$, and (b) extend it to form a good approximate pattern, thus allowing some false positives to occur within the pattern. Finally, in order to avoid the greedy search strategy accepting too noisy patterns, PANDA$^+$ supports two maximum noise thresholds $\epsilon_r, \epsilon_c \in [0, 1]$, inspired by [2], aimed at bounding the maximum amount of noise along the *rows* and *columns* of each pattern.

3 Frequent Itemset-Based Hierarchical Clustering

In this section we discuss the FIHC framework [3] for document clustering. FIHC implements 3 steps:

1. frequent item sets are mined and transformed in a set of initial grouping of transactions;
2. these groups are refined to produce a partitional clustering;
3. these clusters are recursively merged until the desired number of clusters is obtained.

In the first step, FIHC transforms the input corpus of documents in a binary representation suitable for frequent pattern mining algorithms, where the vector dimensions state the presence/absence of a vocabulary term in a document. According to the framework of Sect. 2, each mined frequent (closed) pattern $P \in \widehat{\Pi}_k^{\sigma_k}$, $P = \langle P_I, P_T \rangle$, trivially identifies a *group* of documents – i.e., the dataset documents corresponding to P_T that support the word-set identified by P_I. Since a transaction may support several frequent itemsets, by construction these document groups may overlap. We focus on closed frequent itemsets [5, 14] as they are a succinct representation of *all* the frequent itemsets, avoid redundancies by definition: this is because they are maximal with respect to the set of supporting transactions, and therefore there are no two closed itemsets supported by the same set of transactions.

We call *candidate* seed clusters the resulting groups of transactions supporting the the various frequent (closed) itemsets extracted, used in the subsequent step of FIHC.

In the second step, FIHC enforces a *partitional* clustering, where each transaction is assigned to only one of the *candidate* clusters. To this end, FIHC uses a function $\mathsf{Score}(\cdot)$, which measures how well a given transaction fits within a candidate cluster. Each transaction is then assigned to the best fitting cluster. The $\mathsf{Score}(\cdot)$ function is defined in terms of the items' *global frequency* and *local frequency*.

Given an item $x \in \mathcal{I}$, the global frequent $\phi_{\mathcal{D}}(x)$ is the ratio $supp_{\mathcal{D}}(x)/|\mathcal{D}|$ that considers all the transactions in the input dataset $\mathcal{D}$ supporting item x, while the local frequency $\phi_{\mathcal{C}}(x)$ is the ratio $supp_{\mathcal{C}}(x)/|\mathcal{C}|$ limited to transactions associated with a given cluster $\mathcal{C}$ under consideration.

On the basis of the extracted pattern set $\widehat{\Pi}_k^{\sigma_k}$, we first prune from $\mathcal{I}$ the infrequent items, thus obtaining $\mathcal{I}' = \{x \in \mathcal{I} \mid \phi_{\mathcal{D}}(x) \geq \sigma_k\}$.

Besides the minimum *global* frequency threshold σ_k, the same used to extract the top-k frequent (closed) patterns, FIHC also defines a minimum *local* frequency threshold σ_{loc}, used to identify the *set of locally frequency items* of cluster $\mathcal{C}$, defined by $LF_{\mathcal{C}} = \{x \in \mathcal{I}' \mid \phi_{\mathcal{C}}(x) \geq \sigma_{loc}\}$.

It is worth recalling that in order to compute $\phi_{\mathcal{C}}(x)$ and $\phi_{\mathcal{D}}(x)$, in this phase we ignore possible multiple occurrences of term x in each transaction/document. Indeed, FIHC combines this concept of global/local frequency with a typical word *weighting* scheme, where the term frequency is instead taken into account. Given a transaction t, which simply represents the presence/absence of the various words in a document, the algorithms builds an associated vector $\overrightarrow{\omega}_t$, where $\omega_t(x)$ weights the *importance* of term x in the original document, measured by the usual $TF \cdot IDF$ statistics. The matching of a transaction t to a cluster $\mathcal{C}$ is thus defined as a function of such weight vector, that only consider the items in the pruned set $\mathcal{I}'$:

$$\mathsf{Score}(\mathcal{C} \leftarrow \overrightarrow{\omega}_t) = \sum_{x \in \mathcal{I}', x \in t, x \in LF_{\mathcal{C}}} \omega_t(x) \cdot \phi_{\mathcal{C}}(x) - \sum_{x \in \mathcal{I}', x \in t, x \notin LF_{\mathcal{C}}} \omega_t(x) \cdot \phi_{\mathcal{D}}(x) \quad (5)$$

where the first term of the function *rewards* cluster $\mathcal{C}$ if word x is locally frequent in $\mathcal{C}$, whereas the second term penalizes the same cluster for all the items of t that not locally frequent. The last term encapsulates the concept of dissimilarity into the score.

Intuitively, a cluster $\mathcal{C}$ is *good* for t if there are relatively many items in t that appear in many other transactions assigned to $\mathcal{C}$, and this happens when t is similar to these transactions because they share many common frequent items. Finally, terms with larger $TF \cdot IDF$ values have a larger impact.

According to such scoring function, each transaction in the dataset is associated with one and only one of the *candidate* clusters identified in the previous step. At the end of this second stage, a *partitional clustering* is thus attained, where only a few of the original candidate clusters survived by attracting other transactions.

During the third and last phase, FIHC merges similar pairs of clusters, using an ad-hoc similarity measure. Merging is performed recursively until the desired number of final clusters is reached. Indeed, the *inter-cluster similarity* is defined on top of the above $\mathsf{Score}(\cdot)$ function. Given two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$, all the transactions in the latter are combined and matched to the former. Let $\omega_{\mathcal{C}_j}$ be this combined weight vector, obtained by summing up the weight vectors of all the transactions in $\mathcal{C}_j$, i.e. $\overrightarrow{\omega}_{\mathcal{C}_j} = \sum_{t \in \mathcal{C}_j} \overrightarrow{\omega}_t$. Thus, the following cluster similarity is defined:

$$\mathsf{Sim}(\mathcal{C}_i \leftarrow \mathcal{C}_j) = \frac{\mathsf{Score}\left(\mathcal{C}_i \leftarrow \overrightarrow{\omega}_{\mathcal{C}_j}\right)}{\Omega} + 1 \tag{6}$$

where Ω in a normalization factor. The whole similarity computed in Eq. 6 is asymmetric and normalized between 0 and 2. It is finally made symmetric by taking the geometric mean of $\mathsf{Sim}(\mathcal{C}_i \leftarrow \mathcal{C}_j)$ and $\mathsf{Sim}(\mathcal{C}_j \leftarrow \mathcal{C}_i)$:

$$\mathsf{Inter\_Sim}(\mathcal{C}_i \leftrightarrow \mathcal{C}_j) = \sqrt{\mathsf{Sim}(\mathcal{C}_i \leftarrow \mathcal{C}_j) \cdot \mathsf{Sim}(\mathcal{C}_j \leftarrow \mathcal{C}_i)} \tag{7}$$

At each step of the recursive merging, the two most similar clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ are replaced by a new cluster $\mathcal{C}_{ij} = \mathcal{C}_i \cup \mathcal{C}_j$.

3.1 Exploiting Approximate Patterns in FIHC

The FIHC framework can be easily adapted to produce a clustering starting from the approximate patterns extracted by algorithms such as Asso, Hyper+, and PaNDa$^+$. Indeed, they return patterns of the form $P = \langle P_I, P_T \rangle$, where each pattern identifies not only a set of items (namely vector P_I), but also a set of related transactions (namely vector P_T). Therefore, these patterns are analogous to those returned by a frequent (or frequent closed) itemset mining algorithm. The only difference is that, due to noise, the item set identified by P_I may be only *approximatively* supported by the set of transactions corresponding to P_T.

In order to apply the score function in Eq. 5, we need to redefine the concept of globally frequent item, since both PaNDa$^+$ and Asso do not use any frequency threshold to extract the patterns.

The original FIHC uses the minimum support threshold σ_k, which works as a sort of *a priori* filter, by limiting the selected features to only the frequent items and disregarding the infrequent ones. Conversely, the patterns returned by PaNDa$^+$ or Asso are not derived on the basis of any frequency threshold, even if both the algorithms extract significant patterns, even if the single items occurring in each approximate pattern are likely to be well supported in the dataset.

In analogy with frequent itemsets, where the single items that occur in any patterns are globally frequent by definition, we consider all the items occurring in the various approximate patterns $P \in \overline{\Pi}_k$ as the ones to be included in the pruned set of items $\mathcal{I}'$, $\mathcal{I}' \subseteq \mathcal{I}$. Thus, in order to permit FIHC to exploit an approximate pattern set, we need to replace the concept of global frequency of an item by the concept of occurrence of the same items in the pattern set.

We argue that a high quality pattern set should boost the quality of the generated clustering by FIHC. This is confirmed by our experimental results, where the clustering quality obtained by using top-k approximate patterns are better than using exact frequent (closed) ones.

4 Experimental Evaluation of Approximate Patterns

We compared the quality of the approximate patterns extracted by PaNDa$^+$, Asso, and Hyper+ by using as a proxy the quality of the clustering obtained

by FIHC, which in turn uses such top-k patterns as described in Sect. 3. We run our experiments on four categorized text collections (R52 and R8 of Reuters 21578, WebKB [2], and Classic-4 [3]). The main characteristics of the datasets used are reported in Table 2. As expected, these datasets have a very large vocabulary with up to 19,241 distinct terms/items. The binary representation of those datasets, after class labels removal, was used to extract patterns. The number L of the class labels varies from 4 to 52.

Table 2. Datasets.

Dataset	L	M	N	avg. doc. len
Classic-4	4	5896	7094	34.8
R8	8	17387	7674	40.1
R52	52	19241	9100	42.1
WebKB	4	7770	4199	77.2

During the cluster generation step, the usual $TF \cdot IDF$ scoring was adopted to instantiate $\overrightarrow{\omega}_t$, and $\sigma_{loc} = 0.25$ was used. We forced FIHC to produce a number of clusters equal to L. Even if the goal of this work is to evaluate different solutions for pattern-based clustering, we also reported as a reference the results obtained with the K-Means clustering algorithm, by still setting parameter K of K-Means equal to the number L of classes in the datasets. Finally, cosine similarity was used to compare documents. This baseline is used only to make sure that the generated clustering is of good quality. All the pattern-based algorithms evaluated perform better than K-Means.

The quality of the clusters generated by each algorithm was evaluated with 5 different measures: Jaccard index, Rand index, Fowlkes and Mallows index (F-M), Conditional Entropy (the conditional entropy H_K of the class variable given the cluster variable), and average F-measure (denoted F_1) [4]. For each measure the higher the better, but for the conditional entropy H_K where the opposite holds. The quality measures reflect the matching of the generated clusters with respect to the true documents' classification.

Tables 3,4 report the results of the experiments conducted on the four text categorization collections.

In order to evaluate the benefit of approximate patterns over exact frequent patterns, we also investigated the clustering quality obtained by FIHC with the 50 and 100 most frequent closed item sets. As shown in Table 3, closed patterns provide a good improvement over K-Means. The best F_1 is achieved when 100 patterns are extracted, with an improvement of 13% over K-Means, and similarly for all other measures. This validates the hypothesis of pattern-based text clustering algorithms, according to which frequent patterns provide a better feature space than raw terms.

[2] http://www.cs.umb.edu/~smimarog/textmining/datasets/index.html.
[3] http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/.

Table 3. Pattern-based clustering evaluation. Best results are highlighted in boldface.

Algorithm	# Patt.	Dataset	$F_1 \uparrow$	Rand↑	Jaccard↑	F-M↑	$H_K \downarrow$	Avg.Len.	Avg.Supp.
K-Means	L	Classic-4	0.397	0.525	0.214	0.358	1.668	–	–
		R52	0.394	0.687	0.271	0.428	2.630	–	–
		R8	0.523	0.464	0.377	0.596	1.843	–	–
		WebKB	0.508	0.617	0.287	0.453	1.364	–	–
		avg.	*0.455*	*0.573*	*0.288*	*0.459*	*1.876*	–	–
Closed	50	Classic-4	0.470	0.633	0.250	0.400	1.461	1.000	852.220
		R52	0.407	0.769	0.177	0.360	1.894	1.860	2977.580
		R8	0.624	0.671	0.384	0.555	1.404	1.940	2675.980
		WebKB	0.432	0.331	0.277	0.506	1.880	2.020	1828.500
		avg.	*0.483*	*0.601*	*0.272*	*0.455*	*1.660*	*1.705*	*2083.570*
Closed	100	Classic-4	0.472	0.585	0.249	0.401	1.539	1.100	660.890
		R52	0.495	0.819	0.355	0.557	1.888	2.240	2442.170
		R8	0.648	0.692	0.423	0.596	1.364	2.360	2215.390
		WebKB	0.435	0.318	0.281	0.516	1.879	2.400	1534.810
		avg.	*0.512*	*0.603*	***0.327***	***0.517***	*1.668*	*2.025*	*1713.315*
Asso	L	Classic-4	0.452	0.628	0.217	0.357	1.537	1.000	1456.250
		R52	0.300	0.761	0.098	0.272	1.574	4.692	976.385
		R8	0.446	0.680	0.222	0.401	1.258	6.500	1656.875
		WebKB	0.436	0.627	0.200	0.333	1.700	9.000	1142.750
		avg.	*0.409*	***0.674***	*0.184*	*0.341*	***1.517***	*5.298*	*1308.065*
Asso	50	Classic-4	0.519	0.633	0.256	0.407	1.406	1.040	844.300
		R52	0.287	0.762	0.106	0.283	1.669	4.640	995.920
		R8	0.693	0.762	0.454	0.630	1.116	5.040	800.980
		WebKB	–	–	–	–	–	–	–
		avg.	–	–	–	–	–	–	–
Hyper+	L	Classic-4	0.452	0.628	0.217	0.357	1.537	1.000	1456.250
		R52	0.352	0.749	0.117	0.264	1.953	4.558	132.404
		R8	0.368	0.599	0.156	0.283	1.667	6.375	236.000
		WebKB	0.410	0.422	0.248	0.433	1.831	7.500	185.500
		avg.	*0.396*	*0.599*	*0.185*	*0.335*	*1.747*	*4.858*	*502.538*
Hyper+	50	Classic-4	0.480	0.596	0.255	0.409	1.509	1.040	854.700
		R52	0.357	0.749	0.118	0.265	1.962	4.580	136.660
		R8	0.668	0.733	0.404	0.581	1.191	4.840	116.940
		WebKB	0.436	0.313	0.283	0.520	1.883	5.940	70.100
		avg.	*0.485*	*0.598*	*0.265*	*0.444*	*1.636*	*4.100*	*294.600*
Hyper+	100	Classic-4	0.511	0.675	0.271	0.427	1.345	1.010	656.930
		R52	0.480	0.803	0.313	0.511	1.955	3.930	86.190
		R8	0.639	0.665	0.376	0.547	1.315	4.180	75.160
		WebKB	0.437	0.305	0.284	0.525	1.884	5.460	48.960
		avg.	***0.517***	*0.612*	*0.311*	*0.503*	*1.625*	*3.645*	*216.810*

For all the *approximate* pattern mining algorithms, we evaluated the clusters generated by feeding FIHC with L, 50, or 100 patterns.

The Asso algorithm has a minimum correlation parameter τ which determines the initial patterns candidate set. We reported results of $\tau = 0.6$, for which

Table 4. Pattern-based clustering evaluation. Best results are highlighted in boldface.

Algorithm	# Patt.	Dataset	$F_1 \uparrow$	Rand↑	Jaccard↑	F-M↑	$H_K \downarrow$	Avg.Len.	Avg.Supp.
PANDA$^+$ ($\epsilon = 0.75$)	L	Classic-4	0.439	0.621	0.215	0.354	1.637	3.250	401.000
		R52	0.347	0.771	0.133	0.334	1.653	6.712	578.692
		R8	0.479	0.658	0.214	0.377	1.354	6.250	1468.250
		WebKB	0.361	0.557	0.192	0.324	1.886	14.500	1261.500
		avg.	*0.406*	*0.652*	*0.188*	*0.347*	*1.632*	*7.678*	*927.361*
PANDA$^+$ ($\epsilon = 1.00$)	L	Classic-4	0.471	0.639	0.228	0.373	1.528	3.750	356.500
		R52	0.314	0.765	0.115	0.299	1.730	5.962	558.942
		R8	0.529	0.697	0.266	0.452	1.297	5.250	1676.000
		WebKB	0.351	0.576	0.179	0.305	1.885	22.750	1111.000
		avg.	*0.416*	***0.669***	*0.197*	*0.357*	*1.610*	*9.428*	*925.611*
PANDA$^+$ ($\epsilon = 0.75$)	50	Classic-4	0.498	0.633	0.238	0.384	1.436	2.560	193.120
		R52	0.352	0.769	0.126	0.320	1.624	7.380	566.920
		R8	0.672	0.756	0.435	0.614	1.172	8.220	457.320
		WebKB	0.433	0.331	0.279	0.509	1.886	33.100	325.940
		avg.	*0.489*	*0.622*	*0.269*	*0.457*	*1.530*	*12.815*	*385.825*
PANDA$^+$ ($\epsilon = 1.00$)	50	Classic-4	0.468	0.573	0.242	0.393	1.525	2.320	209.020
		R52	0.320	0.768	0.125	0.316	1.746	12.120	561.400
		R8	0.643	0.698	0.421	0.593	1.277	9.700	486.200
		WebKB	0.426	0.372	0.268	0.479	1.885	11.120	362.580
		avg.	*0.464*	*0.603*	*0.264*	*0.445*	*1.608*	*8.815*	*404.800*
PANDA$^+$ ($\epsilon = 0.75$)	100	Classic-4	0.510	0.647	0.251	0.402	1.444	2.710	158.645
		R52	0.554	0.827	0.376	0.581	1.642	5.700	372.340
		R8	0.704	0.769	0.467	0.642	1.055	6.140	326.360
		WebKB	0.435	0.320	0.282	0.517	1.886	14.190	252.520
		avg.	***0.551***	*0.641*	***0.344***	***0.535***	***1.507***	*7.185*	*277.466*
PANDA$^+$ ($\epsilon = 1.00$)	100	Classic-4	0.490	0.644	0.239	0.387	1.420	3.910	151.110
		R52	0.564	0.826	0.378	0.580	1.649	5.000	374.710
		R8	0.645	0.702	0.420	0.592	1.280	6.790	320.990
		WebKB	0.432	0.337	0.277	0.504	1.885	21.180	229.990
		avg.	*0.533*	*0.627*	*0.329*	*0.516*	*1.558*	*9.220*	*269.200*

we observed the best average results after fine-tuning in the range $[0.5, 1.0]$. We always tested the best performing variant of the algorithm which is named Asso + *iter* in the original paper. Unfortunately, we were not able to include all Asso results, since this algorithm was not able to process the four datasets (we stopped the execution after 15 h). We highlight that Asso is however able to provide good performance on the datasets with a limited number of classes. The results on the other datasets are not as high quality as those obtained by PANDA$^+$.

To get the best performance of Hyper+, we used a minimum support threshold of $\sigma = 10\,\%$ and we fine-tuned its β parameter on every single dataset by choosing the best β in the set $\{1\,\%, 10\,\%\}$. The results obtained with only L patterns are poorer than the K-Means baseline, and 50 Hyper+ patterns do not improve over the most frequent 50 closed item sets. However, some improvement is visible with 100 Hyper+ patterns. Both F_1 and Rand index exhibit some improvement over closed item sets, and an improvement over K-Means of 14% and 26% respectively.

Finally, we report quality of PaNDa$^+$ patternsin Table 4. We tested several settings for PaNDa$^+$, and we achieved the best results with the J_P cost function and varying the noise tolerance, namely $\epsilon = \epsilon_r = \epsilon_c$. For the sake of space, we report only results for $\epsilon \in \{0.75, 1.0\}$. Even in this case, L patterns are insufficient to achieve results at least as good as K-Means, and 50 patterns provide similar results as the other algorithms tested. The best results are observed with the top-100 patterns extracted. In this case, PaNDa$^+$ patterns are significantly better, achieving an improvement over the K-Means baseline in terms of F_1 and Rand index of 21% and 41% respectively. In fact, PaNDa$^+$ patterns with $\epsilon = 0.75$ provide a better clustering with all of the measures adopted. We thus highlight, that imposing noise constraints $\epsilon < 1$ generally provides better patterns.

Tables 3,4 also report the average length and support of the patterns extracted by the various algorithms (see the last two columns). As expected, the most frequent closed itemsets are also very short, with at most 2.4 items on average. Hyper+ is better able to group together related items, mining slightly longer patterns up to an average length of 5.4 for the WebKB dataset. Unlike all other algorithms, PaNDa$^+$ provides much larger patterns, e.g., of length 14.19 for WebKB in the best setting. We conclude that PaNDa$^+$ is more effective in detecting items correlations, even in presence of noise, thus providing longer and more relevant patterns which are successfully exploited in the clustering step.

5 Conclusion

This paper analyzes the performance of approximate binary patterns for supporting the clustering of high-dimensionality text data within the FIHC framework. The result of reproducible experiments conducted on publicly available datasets, show that the FIHC algorithm fed with approximate patterns outperforms the same algorithm using *exact* closed frequent patterns. Moreover, we show that the approximate patterns extracted by PaNDa$^+$ performs better than other state-of-the-art algorithms in detecting, even in presence of noise, correlations among items/words, thus providing more relevant knowledge to exploit in the subsequent FIHC clustering phase. From our tests, one of the motivation is the higher quality of the patterns extracted by PaNDa$^+$, which are longer than the ones mined by the other methods. These patterns are fundamental for FIHC, which exploits them for the initial document clustering which is then refined in the following steps of the algorithm.

Acknowledgments. This work was partially supported by the EC H2020 Program INFRAIA-1-2014-2015 *SoBigData: Social Mining & Big Data Ecosystem* (654024).

References

1. Beil, F., Ester, M., Xiaowei, X.: Frequent term-based text clustering. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 436–442. ACM (2002)

2. Cheng, H., Yu, P.S., Han, J.: Ac-close: Efficiently mining approximate closed itemsets by core pattern recovery. In: Sixth International Conference on Data Mining, 2006, ICDM 2006, pp. 839–844. IEEE (2006)
3. Fung, Benjamin C. M Wang, K., Ester, M.: Hierarchical document clustering using frequent itemsets. In: Proceedings of SIAM International Conference on Data Mining (SDM), pp. 59–70 (2003)
4. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Dubes (1988)
5. Lucchese, C., Orlando, S., Perego, R.: Fast and memory efficient mining of frequent closed itemsets. IEEE Trans. Knowl. Data Eng. **18**, 21–36 (2006)
6. Lucchese, C., Orlando, S., Perego, R.: A generative pattern model for mining binary datasets. In: Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1109–1110. ACM (2010)
7. Lucchese, C., Orlando, S., Perego, R.: Mining top-k patterns from binary datasets in presence of noise. In: Proceedings of SIAM International Conference on Data Mining (SDM), pp. 165–176. SIAM (2010)
8. Lucchese, C., Orlando, S., Perego, R.: A unifying framework for mining approximate top-k binary patterns. IEEE Trans. Knowl. Data Eng. **26**, 2900–2913 (2014)
9. Miettinen, P., Mielikainen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE Trans. Knowl. Data Eng. **20**(10), 1348–1362 (2008)
10. Miettinen, P., Vreeken, J.: Model order selection for boolean matrix factorization. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 51–59 (2011)
11. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 465–471 (1978)
12. Wang, K., Chu, X., Liu, B.: Clustering transactions using large items. In: International Conference on Information and Knowledge Management, CIKM-99, pp. 483–490 (1999)
13. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F.: Summarizing transactional databases with overlapped hyperrectangles. Data Min. Knowl. Discov. **23**(2), 215–251 (2011)
14. Zaki, M.J., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Trans. Knowl. Data Eng. **17**(4), 462–478 (2005)