Document Clustering with Evolved Single Word Search Queries

HIRSCH, Laurence <http://orcid.org/0000-0002-3589-9816>, DI NUOVO, Alessandro <http://orcid.org/0000-0003-2677-2650> and PRASANNA, Haddela

Document Clustering with Evolved Single Word Search Queries

Laurence Hirsch
Sheffield Hallam University
Sheffield, UK
0000-0002-3589-9816

Alessandro Di Nuovo
Sheffield Hallam Univeristy
Sheffield, UK
0000-0003-2677-2650

Prasanna Haddela
Sri Lanka Institute of Information Technology
Colombo, Sri Lanka
0000-0002-6969-8772

***Abstract*— We present a novel, hybrid approach for clustering text databases. We use a genetic algorithm to generate and evolve a set of single word search queries in Apache Lucene format. Clusters are formed as the set of documents matching a search query. The queries are optimized to maximize the number of documents returned and to minimize the overlap between clusters (documents returned by more than one query in a set). Optionally, the number of clusters can be specified in advance, which will normally result in an improvement in performance. Not all documents in a collection are returned by any of the search queries in a set, so once the search query evolution is completed a second stage is performed whereby a KNN algorithm is applied to assign all unassigned documents to their nearest cluster. We describe the method and compare effectiveness with other well-known existing systems on 8 different text datasets. We note that search query format has the qualitative benefits of being interpretable and providing an explanation of cluster construction.**

Keywords— document clustering, search query, genetic algorithm, Lucene

I. Introduction

Document clustering is considered one of the most important problems in unsupervised learning. Clustering algorithms group a collection of documents into subsets or clusters to enable users to explore, organise, summarise and visualise large volumes of text. Documents within a cluster should be similar to each other (cohesion) whilst documents in different clusters should be dissimilar (separation) [1].

For automated clustering, documents are traditionally represented by a multi-dimensional feature vector where each dimension corresponds to a weighted value of a term within the document collection [2]. Various similarity or distance measures have been proposed and are a central component of most text clustering algorithms. Using such a method it is often difficult for a human to understand how the clustering is performed. Indeed, we should note that there has been criticism of the black box nature of many successful machine learning models due to a lack of transparency, and little or no information about how algorithms arrive at their predictions [3]. Work has been done on alternative models which recognise word order such as using lexical chains to preserve the semantic relationships between words, for example by using WordNet [4]. Efforts have also been made to generate a set of human interpretable rules from 'black box' systems such as support vector machines as in [5]

In their seminal work Manning et al [6] assert the cluster hypothesis:

"*if there is a document from a cluster that is relevant to a search request, then it is likely that other documents from the same cluster are also relevant*." [6]

Following this hypothesis, we have developed a system, called eSQ (evolved Search Queries), which uses a Genetic Algorithm (GA) for evolving a set of search queries (requests). Each query is a single word and a cluster is defined as the set of documents which contain that word.

The overall objective of eSQ is to develop an effective clustering system with a natural fit for information retrieval needs and with the following desirable characteristics:

1. easily interpreted by a human.
2. modifiable by a human.
3. provide a causal explanation of cluster construction.

Many algorithms have been proposed to achieve document clustering. We compare the eSQ system with several widely used alternatives which we briefly summarise here.

Although originating in the 1950s K-means is still one of the most popular clustering algorithms and together with its variants is used in a wide variety of modern day clustering algorithms (e.g. [10]). K-means begins with *k* arbitrary centres, typically chosen uniformly at random from the data points (often actual documents in the collection). Each point is then assigned to the nearest centre, and each centre is recomputed as the centre of mass of all points assigned to it. These two steps are repeated until the process stabilizes.

K-means has been shown to be highly sensitive to the initial centres and much of the research effort in document clustering has been focused on optimizing this aspect. K-means++ is an enhanced version of the k-means algorithm and uses a randomised seeding technique which is a specific way of selecting initial centres [7] [8]. Farthest first traversal is another alternative initialization method used in k-means algorithm [8]. Expectation-Maximization Clustering is a parametric probability distribution and the entire data set is a mixture of these distributions. Therefore, it is possible to cluster the dataset using a finite mixture density model of k probability distributions and identify clusters. The Expectation-

978-1-7281-8393-0/21/$31.00 ©2021 Crown

Maximization (EM) algorithm can find the best fitting parameters using iterative refinement algorithm [9]. With Agglomerative hierarchical clustering, from the beginning each data point is a cluster. Next, two nearest points are joined repeatedly until it forms a single cluster [1].

The genetic algorithm (GA) is a stochastic global optimisation technique which mimics the process of Darwinian evolution such that the search for solutions is guided by the principles of selection and heredity [10]. GAs have proved to be an effective computational method, especially in situations where the search space is un-characterized (mathematically), not fully understood, or highly dimensional [11]. Text clustering is a problem that fits these characteristics, and GAs have been used in this area, mostly as a means of optimizing the allocation of cluster centres [12-15]. Typically, each chromosome specifies a combination of centres, which represent the candidate solution to the clustering problem.

GAs have also been used to generate rules for text classification [16-19] and clustering [20], which have the advantage of being transparent and explainable. The eSQ system presented here has a novel fitness test based entirely on the count of unique query hits. Furthermore, this is the first time that a query-based clustering system has been combined with a second stage where a KNN algorithm is used to assign documents which do not contain any of the search query words to their nearest cluster.

Most text clustering systems require that the user provide the number of expected clusters (k) in advance. However, k is often not known. We also present a solution to the harder problem of discovering k as well as defining clusters.

II. Materials

A. Apache Lucene

Apache Lucene (https://lucene.apache.org/) [21] is a high-performance full-text indexing and searching software library written in Java. Evolutionary computation is notoriously resource-intensive, and we find that using Lucene to store and query the document collections is a significant boost to the performance of the system. Lucene also includes implementations of various classifiers.

B. Document Collections

Different clustering algorithms can produce divergent results when compared to each other on different datasets. We, therefore, ran our experiments on 8 different datasets selected from 4 document collections containing very different types of document. We have tried to match the datasets as closely as possible in terms of size and categories, to those reported previously when evaluating text clustering systems. Each dataset is labelled in bold.

1) CrisisLex

An increasing number of short texts are being generated. It has been noted that in a short text environment the situation is complicated by sparsity and high dimensionality, meaning that the vector space model and classic text clustering methods may not work so well [22]. CrisisLex.org is a repository of crisis-related social media data and tools [23]. The 'CrisisLexT6' collection2 contains tweets collected in 2012-13 in different crisis situations. We use 500 of the tweets from each of the three of the sets: Colorado wildfires, Boston bombings and Queensland floods (**Crisis3**).

2) Classic 4

The Classic 4 dataset contains 7095 documents, where each document belongs to one of the four distinct collections: CACM (titles and abstracts from the journal Communications of the Association of Computing Machinery), CISI (information retrieval papers), CRANFIELD (aeronautical system papers), MEDLINE (medical journals). We take the commonly used approach (e.g. [24]) and randomly select 500 documents from each category (**Classic4**).

3) 20 Newsgroups

In the 20 Newsgroups collection [25] documents are messages posted to Usenet newsgroups, and the categories are the newsgroups themselves. The data on this set is considered particularly noisy and as might be expected does include complications such as duplicate entries and cross postings. We create three datasets from this collection by randomly selecting 100 documents from each of the categories. **NG3** is created from: rec.sport.hockey, sci.space and soc.religion.christian. **NG5** is from: comp.os.ms-windows.misc, misc.forsale, rec.sport.hockey, sci.space, soc.religion.christian. **NG6** is from: comp.graphics, rec.sport.hockey, sci.crypt, sci,space, soc.religion.christian, talk.politics .gun as used in [14].

4) Reuters-21578

Reuters-21578 news collection contains news articles collected from the Reuters newswire in 1987. We create three datasets using 100 documents from each category. **R4** contains documents from crude, earn, grain, money-fx. **R5** contains documents from: coffee, crude, interest, sugar, trade. **R6** contains documents from acq, crude, earn, grain, money-fx and ship as used in [15]

III. Method

We use a GA to specify a set of search queries where the set size k is equal to the number of clusters. A simplified example based on the problem of clustering documents in the Newsgroup 5 (NG5) dataset is used to assist the explanation.

Step 1: pre-processing

Before we start evolving queries, all the text is placed in lower case and a small stop set is used to remove common words with little semantic weight. For each dataset, a Lucene index is constructed from the collection of documents, and each document labelled (using Lucene fields) according to its pre-set category. Of course, the GA has no access to the category label which is only used to evaluate the effectiveness of the clustering once all the stages have completed.

Step 2: create a wordlist

In the second step, we create an ordered list of significant words which is used by the GA for building queries. To construct the list, the TF*IDF (term frequency * inverse document frequency) [2] value for each term in the collection is calculated. TF-IDF (often used in term weighting) is used to identify terms that are concentrated in particular documents and may therefore be of more significance in a collection. TF is the

number of occurrences of a term in a document and IDF is the inverse of the number of documents in which the term occurs. For each term in the index, we determine TF*IDF values occurring in each document as indicated in the groovy style pseudo code below, where *terms* is the set of terms and *documents* is the set of documents in the index.

```
Map<Term, Double> tMap = [:]
terms.each {term ->
  double tfidfTotal = 0
  documents.each {doc ->
    tfidfTotal += tf * idf
  }
  tMap.put(term, tfidifTotal)
}
TermList tList= tMap.sort{it.value}.keySet
```

Terms are sorted by their overall TF*IDF value and the top 100 words are selected for use in query building. This step is only required once for each index before the start of the evolution, after which the list is fixed. The integer index is simply the words place in the TF*IDF ordering. In the example shown in Table I the length of the list is only 8.

TABLE I. EXAMPLE LIST OF WORDS FOR QUERY CONSTRUCTION

0	1	2	3	4	5	6	7
space	nasa	god	windows	hockey	file	sale	game

Step 3: create generation 0

Table II shows a sample chromosome from the population of generation 0. Chromosomes have an integer representation where the values can be in the range [0 .. 100] (the maximum size of the wordlist). In the example below each gene defines a single word search query (SQ) and each search query defines the cluster as those documents which contain that word.

TABLE II. CREATING SINGLE WORD SEARCH QUERIES (SQ).

	SQ0	**SQ1**	**SQ2**	**SQ3**	**SQ4**
Chromosome	0	4	5	1	7
Query Word	space	hockey	file	nasa	game

In the example above, the number of clusters (k) is determined from the known number of categories before the start of the evolution. However, we can optionally add another gene in the chromosome to set a value for k (the number of clusters) where k can be in the inclusive range [2 .. 9] (8 possible cluster sizes).

Step 4: fire each query in the set.

In our example, five search queries are generated for the NG5 dataset. For each individual in the population fire each of the search queries in its set and determine its fitness by examining the clusters of documents returned by the queries and counting the number of documents returned which occur in only 1 cluster (see fitness calculation below).

Step 5: Apply genetic operators to create a new generation.

Step 6: Repeat steps 4-5 for 300 generations (termination criteria) and select the individual with the highest fitness.

Step 7: KNN

The selected individual at the end of a run will produce a set of single word search queries and a cluster will be the set of documents containing a search query word. However, many documents in the collection may not contain any of the query words. We have found that we can increase effectiveness by applying a final stage to assign the unassigned documents to their nearest search query defined clusters. We use the Lucene implementation of the K-Nearest Neighbour (KNN) algorithm to add each unassigned document to its closest cluster.

Once a run is complete, we can measure the final F1 value of the expanded clusters with reference to the original category labels. A GA contains many random elements, so we therefore repeat each run 11 times.

A. Fitness Calculation

Text clustering aims to return sets of documents which are related to each other but not to documents in other clusters. We have created and tested two fitness functions that aim to partition a document collection into clusters by generating a set of search queries. The first fitness function is for the case where the desired number of clusters (k) is known in advance. In the second case, the GA will attempt to determine the optimal value for k.

When calculating fitness from a set of queries generated by a chromosome, we define *uniqueHits* as the count of documents returned by exactly one query in the set of queries and *totalHits* as the count of documents returned by any query in the set.

Let Q be a set of queries, let D be a set of documents. Let $M \subseteq Q\ X\ D$ be the set of pairs (q, d) where query $q \epsilon Q$ matches document $d \epsilon D$

$$uniqueHits: |\{d \epsilon D : q \epsilon Q, \exists!(q,d) \epsilon M|\quad (1)$$

$$totalHits: |\{d \epsilon D : q \epsilon Q, \exists q\ (q,d) \epsilon M\}|\quad (2)$$

For the case where k is known in advance, we have found that *uniqueHits* is a good fitness measure where the higher the value (the size of the set of documents returned by only one query) the better the fitness. However, we have noticed that in the case where k is set in the chromosome the GA often produces solutions with too many categories with respect to the labelled collections. Introducing a slight penalty as in the second fitness test (below) improved effectiveness. We have found a suitable penalty value to be 0.04 (see results section below).

$$uniqueHits * (1 - (k * penalty)) \quad (3)$$

Below we show Groovy style pseudo code to calculate *uniqueHits* for a set of queries generated from a chromosome. Note that in Apache Lucene 'SHOULD' behaves like an OR and 'MUST_NOT' is used to indicate that the term MUST NOT occur if a query is to return a document (https://lucene.apache.org/core/8_4_1/core/org/apache/lucene/search/package-summary.html#query).

```
Set queries = [q0..qk]
int uniqueHits = 0
queries.each {q ->
  BooleanQuery uniqueQ= new BooleanQuery()
  uniqueQ.add(q, SHOULD)
  Set otherQueries = queries.minus(q)

  otherQueries.each {otherQ ->
   uniqueQ.add(otherQ, MUST_NOT)
  }

  uniqueHits += search(uniqueQ).size
}
```

B. Parameters

We used a fixed set of standard GA parameters in all our experiments which are summarised in Table III. We use an island model with 4 subpopulations to increase diversity and exchange 3 individuals every 50 generations.

TABLE III. GA PARAMETERS

Parameter	Value
Selection Type	Tournament
Tournament Size	5
Subpopulations	4
Population Size	512
Generations	300
Crossover Probability	0.8
Mutation Probability	0.1
Elitism	Best 2 individuals

C. Effectiveness Measures

Effectiveness is determined by referring to the original category labels from the document collection. The relevant category is determined by computing the most frequent category occurring in the cluster. We then compute precision (p), which is defined as the portion of relevant instances that are retrieved, and recall (r), which is the share of relevant instances retrieved by the algorithm. Finally, the main effectiveness measure we use is the F1 score (sometimes known as Pairwise F-Measure (PFM)) [1] to measure the overall accuracy. The F1 measure has the advantage of giving equal weight to precision and recall and is given by:

$$F1 = \frac{2pr}{p+r} \qquad (4)$$

In our example shown in Table II above Search Query 0 (SQ0) will return all documents in the dataset which contain the word '*space'*. On its own this is likely to produce a reasonable cluster as measured by F1 since it will retrieve documents mostly in the space category. However, looking at all five clusters, both the fitness of the individual and the final average F1 across all categories is likely to be low since we also have a cluster based on the word '*nasa'* which overlaps the '*space'* cluster to a significant degree. We also have a cluster based on the word '*hockey'* and another based on the word '*game'* which will both retrieve documents mainly from the hockey category, therefore making the unique hits count low. Also, none of the search queries in the set are likely to return documents mainly from the christian category.

In the experiments, we consider the case of the GA discovering the number of clusters k, thus, we also calculate the cluster count error as a measure of how effective the GA is in selecting k. The cluster count error is the absolute value of the number of clusters generated by the GA minus the number of categories in the original data set.

IV. RESULTS AND DISCUSSION

A. Penalty for more clusters

In the situation where k is not known in advance, the number of clusters produced by the GA is typically higher than the number of categories existing in the original collection. This higher fragmentation leads to weaker results and we found effectiveness could be improved by introducing a small penalty into the fitness test based on the number of clusters:

$$fitness = uniqueHits * (1 - (penalty * k)) \qquad (5)$$

We investigated various values for the penalty. The effect of a range of sizes for the penalty is evaluated both in terms of the average F1 Score (Fig. 1) and the cluster count error (Fig. 2). Results of both analyses indicate that 0.04 is a good value for the penalty across the datasets we have investigated here.

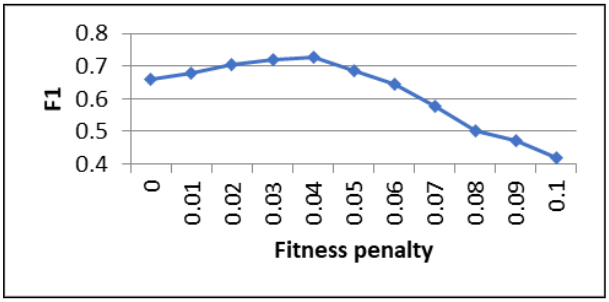

Fig. 1: Average F1 for varying fitness penalty values.

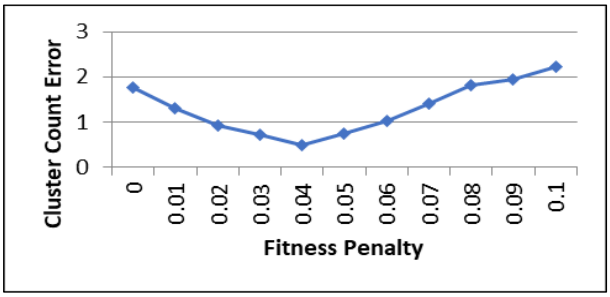

Fig. 2: Average Cluster Count Error with varying fitness penalty values

Following these results, we selected a penalty of 0.04 in the experiments described in the effectiveness section below for the case where k is not known in advance.

B. *Interpretability*

One of the important, qualitative advantages of the search query method is that the queries which define the clusters are human readable. As examples of this feature, Table IV and Table V show queries produced by the best individual at the end of a run for the Crisis3 and NG5 datasets. Queries will return documents which contain the query words. The category is identified as the most frequently occurring among the documents returned by the query.

TABLE IV. CRISIS 3 QUERY SET (F1 FOR QUERY SET: 0.74)

Category	Search Query	F1
Boston bombing	boston	0.74
Colorado wildfires	colorado	0.88
Queensland floods	bigwet	0.61

TABLE V. NG5 QUERY SET (F1 FOR QUERY SET: 0.68)

Category	Search Query	F1
soc.religion.christian	god	0.65
comp.os.ms-windows.misc	windows	0.78
rec.sport.hockey	nhl	0.58
sci.space	space	0.70
misc.forsale	sale	0.71

The GA has found the query words which are the distinguishing features of the categories. The fitness test promotes a set of query words which return the maximum number of documents occurring in only one cluster. Commonly the original category label (or a variant) is the query word. As well as acting as labels for the categories, the query words can give the analyst additional useful information. For example, regarding the Queensland floods the generated search query indicates that '*bigwet'* is an important term used by people tweeting in the crisis. In fact, '*bigwet'* became the informal name of the disaster. Similarly, '*nhl'* refers to the 'National Hockey League' which is the topic of most posts in the NG5 rec.sport.hockey category

C. *Cluster Expansion using KNN*

The clusters produced by the eSQ system have the characteristic of high precision but low recall with respect to the original labelled categories. On average, across all the datasets eSQ achieves precision of 0.87 and recall of 0.55. The recall is so low mainly because many of the documents are not returned by any query. We have found it useful to use a KNN classifier to assign each of these documents to their nearest cluster. In other words, after the GA driven clustering stage we add a second classification stage in which the document clusters returned by the search queries are used as a training set for the KNN classifier. Fig. 3 shows the NG3 collection where an X represents a document which has been assigned to a category. For example, cluster A shows all the documents which contain the word 'god'. Y indicates a document which does not contain any of the search query words ('*god'*, '*hockey'* or '*nasa'*) and are therefore not included in any cluster. The arrows represent the process whereby the Lucene implementation of KNN assigns a document Y to its nearest cluster. We use a Euclidean distance measure with a K value of 10.

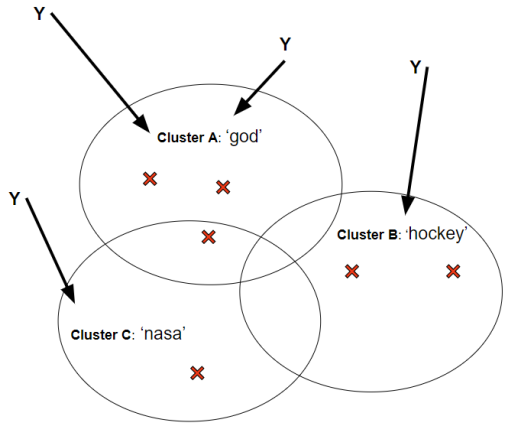

Fig. 3: KNN cluster expansion stage

To demonstrate the effectiveness of the proposed technique over the standard exploration, we compared the result before (eSQ) and after (eSQ+KNN) the application of KNN. For every index the KNN stage improves effectiveness with an average 0.167 improvement to the F1 scores after the KNN stage.

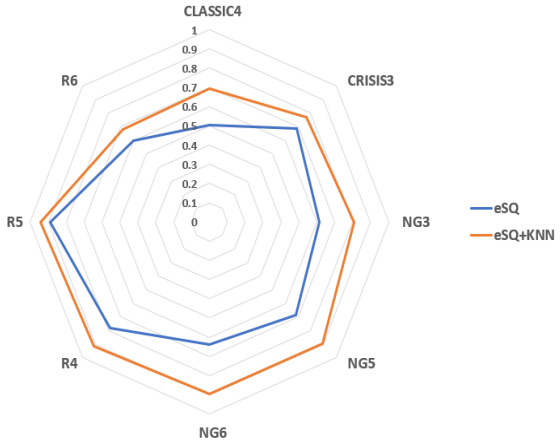

Fig. 4: F1 for eSQ and eSQ+KNN

D. *Effectiveness Comparison*

As discussed above we present two versions of the evolved Search Query (eSQ) here. In the first case the algorithm is given the required number of clusters in advance (eSQ-k-predefined) and in the second case the number of clusters must be discovered (eSQ-k-discovered).

Table VI presents the results of the eSQ and eSQ+KNN explorations, with and without a predefined number of clusters. Average and F1 scores [27] are reported as well as the standard deviation. F1 is the average over 11 runs. In bold are significantly better results (t-test p-value < 0.05). We consider eSQ+KNN k-predefined the direct comparison for the standard clustering algorithms, which all require the number of clusters to be fixed and given in advance. However, eSQ+KNN k-discovered must determine the number of clusters as well as perform the clustering and should be considered a harder problem.

TABLE VI. FINAL RESULTS OF THE GA (BEFORE KNN) AND ESQ EXPLORATIONS, THE LATTER WITH AND WITHOUT A PREDEFINED NUMBER OF CLUSTERS K. IT PRESENTS ALSO THE DIFFERENCES (DIFF COLUMN) AND THEIR P-VALUE (STUDENT T-TEST). SIGNIFICANT DIFFERENCES (P<0.05) ARE IN BOLD

Collection	eSQ+KNN k-predefined		eSQ+KNN k-discovered		Predefined minus Discovered		eSQ k-predefined		eSQ+KNN minus eSQ k-predefined	
	F1 average	*std*	*F1 average*	*std*	*Diff*	*p*	*F1 average*	*std*	*diff*	*p*
CLASSIC4	0.822	0.000	0.563	<0.001	**0.259**	<0.001	0.554	0.000	**0.267**	<0.001
CRISIS3	0.862	0.000	0.677	<0.001	**0.185**	<0.001	0.743	<0.001	**0.119**	<0.001
NG3	0.915	<0.001	0.702	0.000	**0.213**	<0.001	0.671	0.000	**0.244**	<0.001
NG5	0.895	0.000	0.895	0.000	0.000	1.000	0.685	0.000	**0.211**	<0.001
NG6	0.895	0.007	0.896	0.007	-0.001	0.682	0.637	<0.001	**0.257**	<0.001
R4	0.913	<0.001	0.913	<0.001	0.000	1.000	0.784	0.000	**0.130**	<0.001
R5	0.941	0.004	0.942	0.004	-0.001	0.400	0.887	0.000	**0.054**	<0.001
R6	0.672	<0.001	0.689	0.003	**-0.017**	<0.001	0.619	<0.001	**0.053**	<0.001
AVERAGE	0.864		0.785		0.080		0.697		0.167	

TABLE VII. REFERENCE RESULTS WITH VARIOUS CLUSTERING ALGORITHMS (BENCHMARKS). K WAS PREDEFINED IN ALL CASES. BEST RESULTS ARE IN BOLD

Collection	farthest first		kmeans++		kmeans		EM		HieraClus	
	F1	*std*	*F1*	*std*	*F1*	*std*	*F1*	*std*	*F1*	*std*
CLASSIC4	0.780	0.000	0.742	0.100	0.750	0.100	**0.916**	0.036	0.347	0.000
CRISIS3	0.760	0.000	0.748	0.128	0.711	0.126	**0.776**	0.014	0.677	0.000
NG3	0.588	0.000	0.926	0.010	**0.935**	0.016	0.916	0.005	0.866	0.000
NG5	**0.729**	0.000	0.607	0.041	0.645	0.033	0.670	0.014	0.596	0.000
NG6	**0.818**	0.000	0.725	0.120	0.641	0.122	0.599	0.060	0.628	0.000
R4	0.758	0.000	0.763	0.143	**0.771**	0.117	0.693	0.048	0.687	0.000
R5	0.658	0.000	0.679	0.117	0.596	0.080	0.717	0.053	**0.775**	0.000
R6	0.651	0.000	**0.660**	0.045	0.653	0.101	0.602	0.049	0.606	0.000

TABLE VIII. PAIRWISE DIFFERENCES (DIFF) OF THE MEANS BETWEEN ESQ+KNN K-PREDEFINED AND THE BENCHMARKS, INCLUDING P-VALUE OF THE STUDENT T-TEST (2 TAILS). SIGNIFICANT DIFFERENCES (P<0.05) ARE IN BOLD

Collection	*farthest first*		*kmeans++*		*kmeans*		*EM*		*HieraClus*	
	diff	*p*	*diff*	*p*	*diff*	*p*	*diff*	*p*	*diff*	*p*
CLASSIC4	**0.042**	<0.001	**0.079**	0.033	**0.072**	0.049	**-0.094**	<0.001	**0.475**	<0.001
CRISIS3	**0.102**	<0.001	**0.113**	0.021	**0.151**	0.004	**0.085**	<0.001	**0.185**	<0.001
NG3	**0.327**	<0.001	**-0.011**	0.007	**-0.020**	0.004	-0.002	0.369	**0.049**	<0.001
NG5	**0.166**	<0.001	**0.288**	<0.001	**0.251**	<0.001	**0.225**	<0.001	**0.299**	<0.001
NG6	**0.077**	<0.001	**0.170**	0.002	**0.253**	<0.001	**0.296**	<0.001	**0.267**	<0.001
R4	**0.155**	<0.001	**0.150**	0.009	**0.142**	0.004	**0.221**	<0.001	**0.226**	<0.001
R5	**0.283**	<0.001	**0.262**	<0.001	**0.345**	<0.001	**0.224**	<0.001	**0.166**	<0.001
R6	**0.021**	<0.001	0.012	0.421	0.019	0.568	**0.070**	0.002	**0.066**	<0.001

Table VI shows that predefining the number of clusters makes the GA exploration easier and the results are significantly better on average. However, we can see that the eSQ+KNN k-discovered is as good as the predefined version in the databases with more than 4 clusters. In particular, the average F1 of the eSQ+KNN k-discovered is significantly the best for the database R6. This suggests that the eSQ+KNN k-discovered has the potential to achieve better results in the exploration, even without the additional information of the number of clusters, but further investigation is needed in order to identify the best combination of fitness function and parameters to guide the exploration toward the maximum value more frequently.

We also give results using 5 widely used alternative strategies for document clustering namely farthest first, k-means, kmeans++, Expectation Maximization (EM) and Hierarchical Clustering [1]. We should note that although some of these techniques were developed many years ago they are still used regularly in a broad range of applications (e.g. [28][29]). Table VII presents the reference results of the clustering algorithms considered as a benchmark to evaluate the eSQ+KNN performance. Average F1 scores and the standard deviation is provided for each one. Note that in some cases, where there is not a random initialisation, the standard deviation is, of course, 0, while in the other cases there is variance due to the random initialisation. Each of the algorithms in Table VII are given the required number of clusters in advance. Table VII shows that each algorithm considered achieves the best average F1 score at least for one database. Best F1 averages for each database are in bold - where algorithms have a random initialisation F1 is the average of 11 runs.

Table VIII presents the paired comparison between the eSQ+KNN k-predefined and the benchmark clustering algorithms, in practice values in Table VIII are calculated subtracting the values in Table VII to the values in Table VI. Positive values mean that eSQ+KNN k-predefined is better. Table VIII gives the average difference and the p-value obtained with the Student t-test.

We remark that the comparisons are made among all algorithms that use a predefined number of clusters for their optimisations.

In comparison with other clustering algorithms, the eSQ+KNN k-predefined consistently shows a better overall performance than EM and the Hierarchical Clustering. Indeed, the eSQ+KNN k-predefined achieves a statistically significant higher average F1 score than the Hierarchical Clustering in each case. The eSQ+KNN k-predefined outperforms the EM algorithm in 6 collections where positive differences are statistically significant (CRISIS3, NG5, NG6, R4, R5, R6), vice versa, the EM is significantly better than the GA in only one case (CLASSIC4), they are the same for NG3. Similarly, the eSQ+KNN k-predefined significantly outperforms the farthest first and the Hierarchical clustering algorithms in all collections. With respect to k-means and k-means++, eSQ+KNN k-predefined is the best in 6 collections, while its F1 score is significantly lower in only one collection, NG3, they are the same for R6. Looking at the overall results, the performance of the eSQ+KNN k-predefined is always the best in 5 collections CRISIS3, NG5, NG6, R4, R5, and it achieves always a higher F1 score, but not always statistically significant, in the collection R6

V. CONCLUSIONS

In this article, we presented a hybrid approach (eSQ+KNN) for clustering text databases by evolving a set of single word search queries to create the initial clusters then by applying a KNN algorithm to assign all unassigned documents to their nearest cluster.

The experimental results with 8 data sets show that the proposed eSQ+KNN algorithm clearly outperforms two of the

classic clustering algorithms (farthest first and Hierarchical) in all collections considered, while it achieves better results than EM, kmeans and kmeans++ algorithm in all except 2 collections (1 is significantly lower and 1 is not statistically different).

We have shown that eSQ can create human readable clustering queries without compromising effectiveness. Unlike other methods, the search query format is not reliant on a similarity measure between documents. Search queries can act as labels for each cluster and give a human-readable explanation for cluster creation, meaning that there is no need to extract a label as a final stage [30]. Clustering can be framed as a search for the cluster centres, which can be points in a multi-dimensional space or actual documents in the space. Here we use simple, human understandable search queries as cluster centres.

In future work, we will explore the possibility of using the evolved search queries as a way of clustering images or other media. We also aim to examine the effectiveness of multi-word search queries, other query types such as AND, NOT and advanced query types as used in [19].

REFERENCES

[1] C. C. Aggarwal, and C. Zhai, "A survey of text clustering algorithms," in Mining text data, Boston, MA, Springer, 2012, pp. 77-128

[2] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing and Management, vol. 24, no. 5, pp. 513-523, 1988.

[3] W. Samek, T. Wiegand and K. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," ITU Journal: ICT Discoveries, Special Issue The Impact of AI on Communication Networks and Services, vol. 1, pp. 1-10, 2017.

[4] T. Wei, Y. Lu, H. Chang, Q. Zhou and X. Bao, "A semantic approach for text clustering using WordNet and lexical chains.," Expert Systems with Applications,, vol. 42, no. 4, pp. 2264-2275., 2015.

[5] N. Allahverdi , H. Kahramanli and M. Koklu , "Rule extraction from linear support vector machines," in Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005.

[6] C. D. Manning, R. Raghavan and H. Schultze, Introduction to Infromation Retrieval, Cambridge University Press, 2008

[7] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding.," in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007.

[8] S. Dasgupta and P. M. Long, "Performance guarantees for hierarchical clustering.," Journal of Computer and System Sciences, vol. 70, no. 4, pp. 555-569., 2005.

[9] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm.," Journal of the royal statistical society, pp. 1-38, 1997.

[10] J. H. Holland, "Genetic Algorithms," Scientific American, vol. 267, no. 1, pp. 66-72, 1992.

[11] E. R. Hruschka, R. J. Campello and A. A. Freitas, "A survey of evolutionary algorithms for clustering," IEEE Transactions on Systems, Man, and Cybernetics, vol. 39, no. 2, pp. 133-155, 2009.

[12] A. Abraham, S. Das and A. Konar, "Document Clustering using differential evolution," in IEEE congress on evolutionary computation, 2006.

[13] A. G. Di Nuovo, and V. Catania, "An evolutionary fuzzy c-means approach for clustering of bio-informatics databases.," in Proceedigns of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2008.

[14] W. Song, Y. Qiao, S. C. Park and X. Qian, "A hybrid evolutionary computation approach with its application for optimizing text document clustering," Expert Systems with Applications, vol. 42, no. 5, pp. 2517-2524, 2015.

[15] D. Mustafi, A. Mustafi, and G. Sahoo. "A novel approach to text clustering using genetic algorithm based on the nearest neighbour heuristic". International Journal of Computers and Applications, 1-13, 2020

[16] C. Clack, J. Farringdon, P. Lidwell and T. Yu, "Autonomous document classification for business," in Proceedings of the first international conference on Autonomous agents, 1997.

[17] L. Hirsch, "Evolved Apache Lucene SpanFirst queries are good text classifiers," in Evolutionary Computation (CEC), IEEE Congress on Evolutionary Computation., Barcelona, 2010.

[18] A. Pietramala , V. Policicchio , P. Rullo and I. Sidhu, "A Genetic Algorithm for Text Classification Rule Induction," in Proc. European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD '08), 2008.

[19] L. Hirsch and T. Brunsdon, "A comparison of Lucene search queries evolved as text classifiers," Applied Artificial Intelligence, vol. 32, no. 7, pp. 768-784., 2018.

[20] L. Hirsch and A. Di Nuovo, "Document Clustering with Evolved Search Queries," in Evolutionary Computation (CEC), IEEE Congress on., Donostia - San Sebastián, 2017

[21] K Koitzsch, Advanced Search Techniques with Hadoop, Lucene, and Solr. In Pro Hadoop Data Analytics (pp. 91-136). Apress, Berkeley, CA, 2017.

[22] C. Jia, M. B. Carson, X. Wang and J. Yu, "Concept decompositions for short text clustering by identifying word communities," Pattern Recognition, vol. 76, pp. 691-703, 2018.

[23] A. Olteanu, S. Vieweg and C. Castillo, "What to expect when the unexpected happens: Social media communications across crises," in Proceedings of the 18th ACM conference on computer supported cooperative work & social computing, 2015.

[24] K. k. Bharti and P. K. Singh, "Hybrid dimension reduction by integrating feature selection with feature extraction method for text clusterin," Expert Systems with Applications, vol. 42, no. 6, pp. 3105-3114, 2015.

[25] K. Lang, "Newsweeder: Learning to filter netnews," in Proceedings of the Twelfth International Conference on Machine Learning., 1995.

[26] L. Hirsch and A. Di Nuovo, "Document Clustering with Evolved Search Queries," in Evolutionary Computation (CEC), IEEE Congress on., Donostia - San Sebastián, 2017.

[27] C. Goutte and E. Gaussier, "A probabilistic Interpretation of Precision, Recall and F-score, with Implication for Evaluation," in Advances in Information Retrieval, vol. 3408, D. E. Losada and J. M. Fernandez-Luna, Eds., Berlin Heidelberg New York, Springer, 2005, pp. 345-359.

[28] Nguyen, H., Bui, X. N., Tran, Q. H., & Mai, N. L. (2019). A new soft computing model for estimating and controlling blast-produced ground vibration based on hierarchical K-means clustering and cubist algorithms. Applied Soft Computing, 77, 376-386.

[29] Devi, R. D. H., Bai, A., & Nagarajan, N. (2020). A novel hybrid approach for diagnosing diabetes mellitus using farthest first and support vector machine algorithms. Obesity Medicine, 17, 100152.

[30] G. Fung, S. Sandilya and R. B. Rao, "Rule extraction from linear support vector machines," in Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005