FP7-ICT Strategic Targeted Research Project (STREP) TrendMiner (No. 287863)

Large-scale, Cross-lingual Trend Mining and Summarisation of Real-time Media Streams

D3.2.1 Clustering models for discovery of regional and demographic variation

Daniel Preoţiuc-Pietro, University of Sheffield
Dr. Sina Samangooei, University of Southampton
Dr. Vasileios Lampos, University of Sheffield
Dr. Trevor Cohn, University of Sheffield
Dr. Nicholas Gibbins, University of Southampton
Prof. Mahesan Niranjan, University of Southampton

Abstract.
FP7-ICT Strategic Targeted Research Project (STREP) ICT-2011-287863 TrendMiner Deliverable D3.2.1 (WP3)

Keyword list: clustering, spectral clustering, DBSCAN, MediaEval, word co-occurrence, Pointwise Mutual Information

Project	TrendMiner No. 287863
Delivery Date	November 4, 2013
Contractual Date	October 31, 2013
Nature	Other
Reviewed By	Paul Ringler, Thiery Declerk
Web links	`https://github.com/danielpreotiuc/trendminer-clustering`
Dissemination	PU

TrendMiner Consortium

This document is part of the TrendMiner research project (No. 287863), partially funded by the FP7-ICT Programme.

DFKI GmbH
Language Technology Lab
Stuhlsatzenhausweg 3
D-66123 Saarbrcken
Germany
Contact person: Thierry Declerck
E-mail: declerck@dfki.de

University of Southampton
Southampton SO17 1BJ
UK
Contact person: Mahensan Niranjan
E-mail: mn@ecs.soton.ac.uk

Internet Memory Research
45 ter rue de la Rvolution
F-93100 Montreuil
France
Contact person: France Lafarges
E-mail: contact@internetmemory.org

Eurokleis S.R.L.
Via Giorgio Baglivi, 3
Roma RM
00161 Italy
Contact person: Francesco Bellini
E-mail: info@eurokleis.com

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Contact person: Kalina Bontcheva
E-mail: K.Bontcheva@dcs.shef.ac.uk

Ontotext AD
Polygraphia Office Center fl.4,
47A Tsarigradsko Shosse,
Sofia 1504, Bulgaria
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Sora Ogris and Hofinger GmbH
Bennogasse 8/2/16
AT-1080 Wien
Austria
Contact person: Christoph Hofinger
E-mail: ch@sora.at

Hardik Fintrade Pvt Ltd.
227, Shree Ram Cloth Market,
Opposite Manilal Mansion,
Revdi Bazar, Ahmedabad 380002
India
Contact person: Suresh Aswani
E-mail: m.aswani@hardikgroup.com

Executive Summary

This deliverable describes the first experiments for clustering in social media streams. Our main goal in these experiments was the constructions of scalable clustering algorithms capable of dealing with extremely large datasets as well as streaming contexts. In particular, two algorithms, spectral clustering and DBSCAN, are described. The experiments are conducted on two different scenarios. First is clustering multi-modal objects into events whilst second is clustering words in a Twitter corpora with Austrian data. The former was submitted to MediaEval 2013 Multimedia Social Event Detection benchmark at which it achieved the best results submitted in 2013, whilst the later represents a study of word clustering in the context of a Trend Miner use case.

Contents

Chapter 1

Introduction

Clustering aims to discover underlying groups of data by inferring a general representation to charactise the data in terms of a few key topics or events. For example, clustering is often used as a means for browsing and searching unstructured document collections. Using the same motivation, in this deliverable we aim to use clustering on Social Media data to facilitate access to large volumes of data. These clusters can also be used as features in Machine Learning models. This hints at our future applications where both clusters and predictions can be learnt together in order to uncover causes of variation e.g. demographic or regional.

This document presents initial work on research and software development for Task 3.3 on clustering for discovery of regional and demographic variation. We present two algorithms and systems that group social media items based on shared features. The main idea is to represent the clustering problem in the form of a similarity matrix and perform clustering using the graph described by this matrix.

The document contains the following work:

1. A presentation of 2 clustering algorithms which can be used to identify topics/events given social media input and their magnitude (Chapter 2).

2. An incremental strategy that can be applied with any clustering technique which, by making certain assumptions about the nature of the social media stream, can efficiently cluster large datasets.

3. A system that identifies events and assigns individual items to these events. This system can integrate multi-modal features which can be text, images, geo-coordinates and timestamps. This system was evaluated as part of the Social Event Detection (SED) challenge from MediaEval 2013 with data from the Flickr social network (Section 3.1). The technique we present achieved the best results at the SED2013 challenge competing against benchmarks submitted by 12 other teams.

4. A system that identifies clusters of co-occurring terms in social media texts (e.g.

tweets). These can be roughly assigned to topics of discussion or events in social media. We consider real world events as use cases to show the practicality of this method for end-user analysis. This system can be extended in the future to incorporate geo-spatial, temporal or any other type of information (i.e. demographics) by using the methods developed in the previous subsection (Section 3.2).

Finally, all the software is released as part of a REST web service for integration in further applications at `https://github.com/danielpreotiuc/trendminer-clustering`.

1.1 Relevance to TrendMiner

TrendMiner processes large volumes of social media items. Given the massive volume, it is impossible to manually analyse or even visualise these data by an end-user. Clustering words into topics means that collections of data become easier to analyse. The clusters can also represent features that can be integrated in further applications, like prediction algorithms, while also providing better interpretability e.g. for discovering regional or demographic variation as planned in the work plan of the consortium agreement.

1.1.1 Relevance to project objectives

The software released with this deliverable offers methods of discovering underlying topics tailored for Social Media datasets. The software is deployed as a REST web service that can provide the most important topics in a batch of data and assign social media items to topics.

1.1.2 Relation to other workpackages

Clusters identified in collections of Social Media items can be used as features for training Machine Learning models for prediction from WP3. They are also useful for using as features by the summarisation algorithms from WP4 and for display and driving the filtering in the user interface in WP5. Clustering results will help the political and financial use cases (in WP6 and WP7) to discover key players and events and track trends caused by events over time.

Chapter 2

Clustering

In this section we present the clustering algorithms we use in our experiments. We present 2 methods, Spectral clustering and DBSCAN, both of which operate on the similarity matrix between objects. Both algorithms are computationally efficient and have the desirable property of working well on large high dimensional data. Furthermore we present an incremental clustering technique which can be applied to any clustering technique if applied to a stream.

2.1 Spectral clustering

Spectral clustering (Shi and Malik, 2000; Ng et al., 2001) is a clustering algorithm that has been shown to achieve state-of-the-art performance for a range of tasks, from image segmentation (Shi and Malik, 2000) to community detection (Smyth and White, 2005). This method treats the clustering problem as one of graph partitioning on the similarity graph between objects. The algorithm projects the objects via Singular Value Decomposition into a reduced space which aims for maximal separation of clusters. This way, spectral clustering is useful when data dimensionality is high. It is also particularly useful when the clusters can't be discovered using a spherical metric, as is the case, for example, with k-means. The algorithm is also appealing because it is supported by spectral graph theory (Chung, 1997).

The algorithm works as follows: we start with the similarity matrix W, the diagonal matrix D with elements the row sums of W and the graph laplacian L. A few choices of laplacians exist depending on the partitioning problem we aim to solve. For example, the $L_{rw} = I - D^{-1}W$ laplacian, is used to find a clustering such that a random walk in the graph rarely changes cluster memberships. The solution to this optimisation problem is NP-hard. The algorithm solves a relaxed version, by eliminating the constraint of Z to have a particular structure:

$$Z_{RW} = \underset{Z}{\operatorname{argmin}} \operatorname{Tr}(Z^T L Z) \text{ s. t. } Z^T Z = I \tag{2.1}$$

The solution is the matrix with the k smallest eigenvectors, ranked by their eigenvalues and ignoring zeros. The objects are clustered with a standard clustering algorithm (i.e. k-means) in this reduced space.

A good clustering relies on a good similarity matrix which best reflects the connections between objects. Sparsity of this matrix is another desirable property as the eigensolver can produce better and faster results with sparser matrices. Most of the work in applying spectral clustering is performed to build the optimal similarity graph. For further details about spectral clustering please consult (Von Luxburg, 2007).

2.2 DBSCAN

The Density-based spatial clustering of applications with noise (DBSCAN) is a clustering algorithm initially proposed by Ester et al. (1996). It has always enjoyed a great deal of interest and has been one of the the most cited clustering algorithms in the literature. More recently it has been used extensively to handle large-scale data problems over numbers of data points which other clustering methods fail to handle. This is because DBSCAN is a one pass algorithm that requires only a function which can return the data items which count as being "within a neighbourhood" of a given starting data item. Such an operation can be implemented efficiently given appropriate indexing structure and therefore DBSCAN enjoys $O(n\log(n))$ complexity as compared to k-means which ostensibly has an $O(nk)$ complexity, but the number of iterations is unknown, or the $O(n^3)$ time complexity of the eigendecomposition required for spectral clustering (here n is the number of items to be clustered, e.g., words, tweets or Flickr images).

These efficiency benefits aside, DBSCAN: (*1*) successfully finds clusters of arbitrary shape (as compared to the Gaussian distributions implicit in k-means); (*2*) does not require an explicit statement of the number of clusters expected; (*3*) can report that a given data item is more likely noise rather than a member of any cluster.

The basic DBSCAN algorithm defines the *ε-neighbourhood* of a data point p as a number of points within ε of p or:

$$N_\varepsilon(p) \quad = \quad \{q \in \mathcal{D} \mid dist(p,q) \leq \varepsilon\} \tag{2.2}$$

The *ε-neighbourhood* contains the points within some neighbourhood. A point could be seen as being densely surrounded, and therefore within a cluster if $|N_\varepsilon(p)| \geq minPts$, where $minPts$ is the second parameter of DBSCAN. Two points p and q are *directly* density-reachable if p is densely surrounded and $q \in N_\varepsilon(p)$, and are said to be *generally* density-reachable if there is a chain of points between p and q such that each point in the chain is *directly* density reachable from the previous. It is important to realise that the *generally density reachable* property is not symmetric. If one were to start at q, a point on the edge of the cluster, p would not be density reachable because q itself would not be considered to be *directly* density reachable to any point (being on the edge of a cluster).

However by starting at p, which is part of the cluster, one could find q. Therefore the final definition is that of *density connected* points p and q which are density reachable to some shared point o. From here we define a cluster in DBSCAN as a collection of points which are all mutually density connected to each other.

Algorithm 1 The DBSCAN algorithm

```
function DBSCAN($\mathcal{D}$,$\varepsilon$,$minPts$)
    for all $p \in \mathcal{D}$ do
        $N_\varepsilon$ = REGIONQUERY($p$)
        if $|N_\varepsilon| \leq minPts$ then
            mark $p$ as noise
        else
            $C$ = next cluster
            EXPANDCLUSTER($p$,$C$,$N_\varepsilon$,$\varepsilon$,$minPts$)
        end if
    end for
end function
function EXPANDCLUSTER($p$,$C$,$N_\varepsilon$,$\varepsilon$,$minPts$)
    add $p$ to $C$
    for all $P' \in N_\varepsilon$ do
        $N'_\varepsilon$ = REGIONQUERY($P'$)
        if $N'_\varepsilon \geq minPts$ extend $N_\varepsilon$
        add $P'$ to $C$ if it isn't already clustered
    end for
end function
```

Given this definition of a DBSCAN cluster, the DBSCAN algorithm can be seen in Algorithm 1. Of particular note in this algorithm is that DBSCAN actually only needs some notion of *regionhood* and a count of points in a region which is dubbed dense enough to be said to be clustered. In applications where data points can be represented spatially, this $regionQuery$ can be defined in terms of Euclidean distance. However in other applications, take for example a graph for which only the weight between edges is known, the $regionQuery$ could be just as easily defined, but not necessarily have any direct geometric interpretation. With this in mind we implement two distinct implementations of DBSCAN, both of which take advantage of this core algorithm, but define regionhood in slightly different ways suited for a particular application. The first implementation called the *NearestNeighbourDBSCAN* builds an efficient spatial index and gathers the neighbourhood of any given point in a spatial sense. We show this implementation working in place of k-means as the final stage of a spectral clustering algorithm. The other called the *SparseMatrixDBSCAN* works against a sparse similarity matrix, defining neighbourhoods as points which have a similarity above a threshold. We show this technique working in isolation against a carefully constructed sparse similarity matrix.

We have developed an open source implementation of the DBSCAN algorithm[1]

2.3 Incremental Clustering

In general, clustering algorithms have scaling concerns, especially when dealing with data that is generated in an incremental, streaming fashion. Spectral clustering requires the eigendecomposition of the laplacian of the similarity matrix, a calculation which quickly becomes intractable beyond $100,000$ items. The sparse matrix DBSCAN is a relatively efficient algorithm and can easily cluster the number of items in the $100,000$ region. However, even DBSCAN has limits in terms of performance, its unmodified form requiring the entire space's similarity matrix to be held in memory, this could eventually become intractable to hold.

In response to challenges of scale that both DBSCAN and spectral clustering methods face given enough data, we were inspired to investigate a general, incremental clustering technique which takes advantage of the streaming nature of social media data. If we assume not only that the social media data appears incrementally, but that this arrival order implies some inherent relatedness and potential cluster membership. Further, we could make the assumption that, within a large enough window, a single event is likely to cause many social media items which are related to it, but after some time (or a few windows) no more items will ever be created which are related to that event.

With these assumptions, our incremental algorithm is implemented as follows. A window size is selected and social media items are read in order to fill that window size. A round of clustering is performed against this window. The stream is then read again, allowing the window to grow, and we perform the clustering again. After this second clustering we might notice that certain clusters and their members are stable across window growth steps, which is to say, a cluster has identical members before and after the window growth. If stable within one window growth (again assuming window size is large enough to likely capture relatively complete events) this stability is taken as a signal that a given cluster will not acquire new members in future window growths. This requirement for exact cluster matching between windows could be applied in a relaxed form where stability might be defined as clusters which are not exact, but have a high overlap between window growth steps.

Regardless, once a cluster is defined as stable, those items which belong to that cluster are removed and not involved in the clustering step after the next window size increase. In this way, the effective number of items to be clustered in any given round will increase as items arrive but will also decrease as clusters become stable. By implementing and applying this technique we can successfully apply expensive clustering algorithms to large datasets. An algorithm with similar goals to this approach as been explored recently by Mayurathan et al. (2013).

[1] `https://github.com/sinjax/trendminer-java`

Chapter 3

Experiments

In this section we describe experiments on clustering using the algorithms presented in the previous section. We present two different scenarios, both that use Social Media data.

The first consists of work that aims to cluster multi-modal (i.e. photos, text, time) Flickr items into events. As an entry to the Social Event Detection competition, a part of MediaEval 2013, this work allows us to benchmark our clustering algorithms in a Social Media context against other state of the art techniques. Also, this work explores the integration of multi-modal information in a clustering context, thus addressing a major concern when the clustering social media content is attempted.

The second scenario consists of work using a dataset directly relevant to TrendMiner's objectives. We analyse tweets coming from Austrian users that talk about current affairs and extract clusters which represent topics discussed in the corpus. We present various ways of using the clusters in an end-user scenario.

3.1 MediaEval

In their June 2013 WWDC keynote, Apple announced a new photo collection feature for their iOS mobile operating system. With the evocative tag-line "Life is full of special moments. So is your photo library", Apple noted the importance of clustering social streams by their belonging to some real life event. This, along with the plethora of mobile and desktop applications which offer some degree of event detection in user photo streams, demonstrates that detecting events in multimedia streams has real practical utility for users. If such detection and clustering of events within private collections is useful, detecting events and clustering items in a multi-user social media context must also have practical benefits. Within this context, challenges of scale and feature error inherent in non-curated collections must be addressed to create meaningful groupings of social media artifacts which afford the user with the ability better understand, consume and contextualise social streams.

In this section we present an approach to achieving clustering of social media artifacts towards addressing task 1 in the Social Event Detection (SED) challenge of the MediaEval 2013 multimedia evaluation (Troncy et al., 2013). Task 1 asks that a collection of Flickr photos be organised into events such that events are defined as "events that are planned by people, attended by people and the media illustrating the events are captured by people". The Flickr items in this task contain metadata beyond the content of the image itself. Namely, the Flickr photos are guaranteed to include accurate: Flickr IDs, user IDs and *timeposted*. The photos may also contain in varying degrees of accuracy location information, time taken according to the camera, and textual information including title, tags and a description.

3.1.1 Methodology

The overarching strategy of our technique is the construction of a square sparse similarity matrix whose elements represent the similarity of two items of social media. Once this object is created, two clustering algorithms are applied. However, the naive creation of such a matrix for any number of items beyond a trivial number is a time consuming $O(n^2)$ operation. Therefore, the first stage of our process was the efficient construction of such an affinity matrix.

Given the SED2013 training set we know for $\sim 300,000$ objects there exist $\sim 14,000$ clusters. The average number of items per cluster in the training set is 20. From this information, we know that the similarity between most objects must be 0 if similarity is a reasonable indication of cluster membership. However, naively inducing sparsity after feature extraction and element comparison has been performed is an approach without merit. In doing so, the expensive operation has already been performed, and the matrix elements are only forced to 0 after this fact. Instead we aim to inherently limit the number of feature comparison operations to be performed. To do so we construct a Lucene [1] index of the items to be clustered. The Flickr images are indexed using their metadata, each meta data component is given a field in the Lucene index. Then, for each item in the dataset we construct a custom Lucene query, receiving an artificially limited number of documents. Only then we extract features and compare distances, using only the documents returned by this query. Once the work is done to construct this Lucene index, this operation has a complexity of $O(n)$ which allows a much faster construction of the similarity matrix.

Once documents are filtered using Lucene, the similarity matrix is constructed. The items being clustered are inherently multi-modal. These modalities include time information (both posted and taken), geographic information, textual information (tags, descriptions and titles) as well as the visual information of the Flickr photos themselves. Any of these modalities might serve as a strong signal of cluster membership. Photos taken in the same place, or at the same time, or containing similar text might all serve as strong indication of these photos being of the same event. However on their own the features might

[1] http://lucene.apache.org/core/

also serve to confuse unrelated events, for example, two events happening on a Friday, but one in Nottingham and one in London. Therefore, the first stage in the construction of a unified similarity matrix is a separate similarity matrix for each of these features. The second step then becomes the combination of the similarity matrices. Inspired by Reuter and Cimiano (2012) we use a logarithmic distance function for our two time features. We also use a logarithmic distance function for our geographic Haversine based distance function. Fundamentally, this forces distances and times beyond a certain distance to count as being infinitely far, or as having 0 similarity. For the textual features we use the TF-IDF score with the IDF statistics calculated against the entire corpus of Flickr objects. We also experimented with SIFT based visual features for image feature similarity matrix construction. However, we found this feature only made F1 scores worse in the training set and the visual features were completely ignored in all submitted runs against the test set. If any given feature is missing or empty for either object represented by a particular cell in the similarity matrix, for the purpose of the sparse similarity matrix it is treated as being ‘not present’ rather than 0 similarity. The distinction here is important for how these similarity matrices are combined.

Once the per-modality similarity matrices were constructed, we experimented with various feature-fusion techniques to combine them as discussed more completely by Kittler et al. (1998). Combination schemes including: *product*, *max*, *min*, and *average* were tested, average was shown to work well. Different feature weightings were also investigated. To calculate the combined similarity w_{ij} of the i^{th} image with the j^{th} image, the feature affinities w_{fij} for all features $f \in F_{ij}$ were used where F_{ij} is all the features which had values for both images i and j, i.e. those features ‘present’ in the similarity matrix:

$$w_{ij} = \sum_{f}^{F} p_f w_{fij}, \quad \sum_{f}^{F} p_f = 1 \tag{3.1}$$

where p_f is the weight of a given feature f. The final similarity matrix produced by this process was used by the clustering techniques below.

Event Clustering

The exact number of events in the corpus was unknown and hard to estimate. We explored clustering techniques which work without setting the number of clusters k explicitly. An algorithm which does not have this limitation is the DBSCAN algorithm. In our first approach to clustering these events we use the *SparseMatrixDBSCAN* as described in Section 2.2. Secondly, we explored more sophisticated spectral clustering techniques which interprets the similarity matrix as weights of edges on a graph and uses graph theoretic techniques described in 2.1 to automatically estimate the number of clusters present in the graph. The data items are projected into a metric space which ensures better separation between the clusters. At this stage spectral clustering requires another spatial clustering algorithm to cluster the, now better separated, data items. We used the

NearestNeighbourDBSCAN described in Section 2.2 to cluster data items in the spectral space.

Spectral clustering could not be used in the training or test sets of the MediaEval challenge due to the volume of data. This problem was in fact the inspiration for the incremental clustering algorithm described in section 2.3. By applying our incremental approach we were able to successfully apply the spectral clustering algorithm to the large training set of $300,000$ items unlike Petkos et al. (2012) who also applied a spectral clustering technique to event detection task, but on a comparatively small sample size.

3.1.2 Experiments and Results

The social event detection task for MediaEval 2013 worked against a dataset of ~ 480000 Flickr images. Of these, $\sim 300,000$ where provided as a training set in the months before the competition and $\sim 180,000$ were to be used as the test set for the actual competition. The task outlined was the grouping of the Flickr images into events. For the training set a ground truth was provided and it was therefore useful for exploring and identifying the various parameters of the algorithms used. However, beyond this the training set could not be used directly to 'learn' events as there was explicitly no overlap of events between the training and test set. This is primarily due to the events, which had very few images to be grouped together.

The results of the experiment measure NMI, F1 and Random Base F1 against a ground truth of crowd sourced event detection. The normalised mutual information (NMI) score measures in some normalised way how much our knowledge of the true clustering is increased by our knowledge of the proposed clustering. The F1 score is the geometric mean of the recall and precision of the proposed clustering against the ground truth. The interesting score proposed for MediaEval is the Random Base F1. This score is calculated as the F1 score minus the F1 score attained if the items in the proposed clusterings were kept in the same proportion (i.e. cluster sizes were maintained) but cluster membership was completely randomised. This score aims to address the issues of the high F1 score which is attainable if very many clusters are proposed. This effect is dampened because in such a proposal if cluster membership is randomised an equally high F1 score can be attained. Thus if the Random Base F1 score remains high, the quality of the proposed clustering can be more highly trusted.

For the runs submitted to the challenge against the training set, we first had to explore and set the various parameters of our approach against the training set. Our first parameter was the weightings with which the feature similarity matrices were combined. We performed a search across the weighting simplex and found that for the training set the best F1 score was achieved when the features were weighted as: $p_{timetaken} = 3, p_{timeposted} = 0$, $p_{geolocation} = 1$, $p_{description} = 1$, $p_{tags} = 3$, $p_{title} = 0$. However, we noticed that the F1 achieved by the top feature weightings on the simplex were very similar. Therefore, to avoid over-fitting we selected the top 1000 F1 ranked selections on the weightings sim-

	DBSCAN best-weight	**Spectral best-weight**	**DBSCAN average-weight**	**Spectral average-weight**
F1	0.9454	0.9114	0.9461	0.9024
NMI	0.9851	0.9765	0.9852	0.9737
F1 (Div)	0.9350	0.8819	0.9358	0.8663
Random Base F1	0.0589	0.0580	0.0597	0.0569
Div F1	0.8865	0.8534	0.8864	0.8455

Table 3.1: Results from our four runs.

plex and got the weightings average in the training set. This resulted in the features weighted as: $p_{timetaken} = 2.1$, $p_{timeposted} = 1.8$, $p_{geolocation} = 1.4$, $p_{description} = 0.7$, $p_{tags} = 1.7$, $p_{title} = 0.3$. We performed a similar line search to find the optimal values for DBSCAN's parameters ($eps = 0.45$, $minpts = 3$) and spectral clustering's parameters ($eigengap = 0.8$, $eps = -0.6$). The 4 runs we submitted for the MediaEval 2013 SED Task 1 were therefore: DBSCAN with the best weighting, spectral clustering with the best, DBSCAN with the average weighting and spectral clustering with the average. All these runs were performed using our incremental clustering technique.

Several interesting conclusions can be drawn from these feature weightings. Firstly, we note that for the purposes of event detection, the time that a given Flickr image is taken is quite high in importance. We also noted that on average, weightings on the simplex performed well when *timeposted* was weighted highly but geolocation was weighted low, or vice versa. This can be understood if we note that timeposted is measured centrally on the Flickr servers as images are uploaded, and that most people will post images at a similar time of day across the world. Therefore *timeposted* is likely to be a proxy for geolocation. We also notice that the title and description features are far less important than the tags. In many cases in Flickr the titles default to the auto-generated title from the camera software (e.g. IMG_00001 or DSC0002 etc.) and the descriptions are often empty. Tags will often contain the event name and in this way serve to draw together Flickr images belonging to the same event.

An overall summary of the results from the runs can be seen in Table 3.1. The F1 scores and NMI achieved by this technique were the best submitted to the SED MediaEval task in 2013.

3.2 Austrian tweets

In this section we present a system that identifies clusters of words based on word co-occurrence patterns. The co-occurrence statistics are computed over a large corpora of social media texts. They are then used as a similarity matrix in a spectral clustering algorithm which assigns each word to a unique cluster. The clusters will contain words that occur in the same context. Importance of a word in a cluster can be associated with its centrality score. Using these scores we can then weight each word and tweet and find

No.tweets	No.unique tweets	Avg.Tokens/tweet	Avg.Vocab Tokens/tweet
2,282,006	1,799,933	14.66	9.25

Table 3.2: Dataset statistics for the Austria1Year Dataset.

the topics of interest in a tweet or batch.

3.2.1 Data

The dataset for our experiments consists of tweets collected from a set of Austrian users selected by SORA for their importance in discussion of current affairs in Austria and world. Collection was performed by a method that monitors the entire stream of the selected set of users in order to access 100% of the data they produce.[2] We selected the data from 1 September 2012 to 31 August 2013 and called this dataset Austria1Year.

The data is preprocessed by tokenising using the Trendminer pipeline (Preoţiuc-Pietro et al., 2012). A deduplication of content is then performed in order to remove retweets and copied tweets. This is a very important step before computing co-occurrences, as the co-occurrence statistics for words in a very popular tweet can be artificially skewed. We consider as duplicates all retweets and all the tweets that have the same first 5 unprotected tokens (see (Preoţiuc-Pietro et al., 2012)). The vocabulary is created by words that occur more than 10 times in our dataset and that are not in a list of English, German and Twitter stopwords. In total the vocabulary size consists of 48,946 words. Dataset statistics are presented in Table 3.2.

This dataset aims to present a snapshot of the topics talked about in a country over a broad time interval. It is also particularly interesting that most of its contents is written in German. This is helpful for establishing the generality of the methods we describe.

3.2.2 Method

We use spectral clustering in order to discover groups of words that co-occur in the same tweet in our dataset. The underlying graph laplacian is built in our case on the word-by-word similarity matrix. The measure of similarity is given by the co-occurrence inside a tweet between the two words as given by the Normalised Pointwise Mutual Information (NPMI) computed over the entire corpus:

$$\text{NPMI}(w1, w2) = -\log P(w1, w2) \cdot \log \frac{P(w1, w2)}{P(w1) \cdot P(w2)} \tag{3.2}$$

The NPMI is the normalised version (in the $[-1, 1]$ interval) of Pointwise Mutual

[2] `https://github.com/danielpreotiuc/twitter-collection-utils`

Very good	Okay	Unclear	Irrelevant
121	33	30	16

Table 3.3: Coherence judgement results.

Information. This is an information theoretic measure that indicates which words co-occur in the same context. A value of 1 means that the words occur only together, while a value of 0 means that they occur as often as would be expected by chance.

We then apply spectral clustering, we use the random walk graph laplacian with mutual 30 k-nearest-neighbours in order to exploit sparsity. For NPMI values we use a threshold of 0.1 as values below we consider insignificant. We only keep the largest connected component of the resulting graph. The number of clusters needs to be specified in advance. We present results using 200 clusters as this value was subjectively assessed to give a good trade-off between cluster coherence and specificity. Once assigned to clusters, we can identify the most important words in each cluster by computing the centrality of each word with the following formula:

$$C_w(c) = \frac{\sum_{x \in c} \text{sim}(w, x)}{|c| - 1} \tag{3.3}$$

where w is the target word and c its cluster. The centrality score for a word is non zero only for the cluster in which it belongs.

3.2.3 Experiments

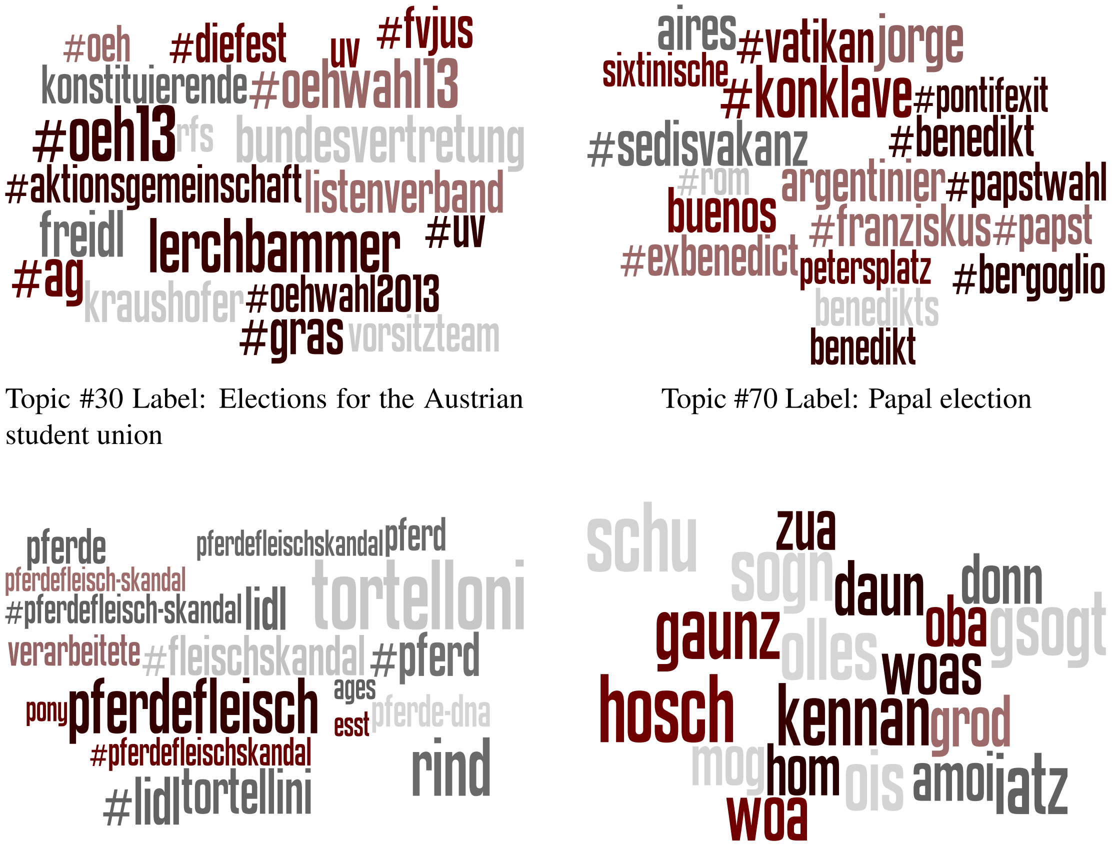
To start, we will present some sample clusters. Every cluster can be presented by a list of its most important t words in terms of importance. SORA have provided us a label and a manual assessment of coherence for these clusters. Methods for automatic cluster labelling have been proposed using text (Lau et al., 2011) or images (Aletras and Stevenson, 2013b) and can be integrated in the future. We present sample clusters by their top 20 words as word clouds is Figure 3.1. All the clusters and their labels are made available together with the open source computer code.[3]

We have also asked SORA to subjectively asses the coherence of each cluster because of their expertise and domain knowledge. The coherence scores were judged as being, from lowest to highest coherence: ‘Unclear’, ‘Okay’, ‘Very good’. We also have a special marker ‘Irrelevant’ for slang or foreign words which are expected to co-occur but are not useful for use cases. Results are shown in Table 3.3

We see that the results are very good with around 60% of the clusters being judged as ‘Very good’ and only 15% with ‘Unclear’ meaning.

We now focus on a specific use case and show how we can use the clustering results

[3] https://github.com/danielpreotiuc/trendminer-clustering



Topic #30 Label: Elections for the Austrian student union

Topic #70 Label: Papal election

Topic #92 Label: Horse meat scandal

Topic #196 Label: Southern German/Austrian slang words

Figure 3.1: 4 Sample clusters from the Austria1Year corpus. Size is proportional to word importance. The topics are manually labelled. Colours are only for clarity purposes.

to discover related tweets and judge the magnitude of an event in a specific time interval. To exemplify, we use topic #70 which was labelled as 'Papal election'. This refers to the papal conclave for electing the new Pope of the Catholic Church after Pope Benedict's resignation which started on 12 March 2013 and finished on 13 March 2013 with the election of the Argentinian Franciscan archbishop of Buenos Aires, Jorge Mario Bergoglio as Pope Francis I.

For each tweet we compute its most likely cluster using the following formula:

$$\operatorname*{argmax}_{c} s_t(c) = \frac{\sum_{t \in W} \mathrm{C}_t(c)}{\sqrt{|W|}} \tag{3.4}$$

where c is the cluster, W is the set of all tokens in a tweet and $C_t(c)$ the centrality score as defined in Equation 3.3.

From a total of 8403 of tweets authored on 13 March 2013, 1561 were assigned to

Tweet text	Score	Time
Wettquote Bergoglio war 34.00 - das wär's gewesen	0.0312	19:29:11
'Christoph, du steigst nicht auf diesen Balkon!' - Mama Schönborn hängt bestimmt schon am Telefon #HabemusPapam	0.0255	18:40:45
Und die Welt feiert ein Ofenrohr. Schwarzer Rauch im Vatikan. Hier schon mal vorab erklärt was noch kommen könnte ;) [URL]	0.0229	10:42:22
'Gib uns heute unser Brot wie wir vergeben den Schuldnern' #orf #fail #habemuspapam #papst	0.0224	19:27:25
#Papst Frantschisku!	0.0216	19:13:32

Table 3.4: Most important tweets on 13 March 2013 for the topic 'Papal election'.

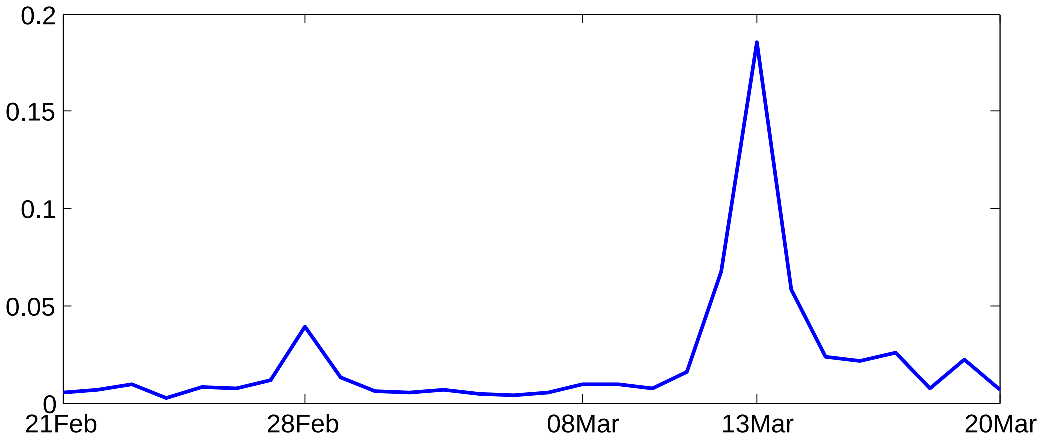


Figure 3.2: Magnitude of topic 'Papal election' over 1 month.

the 'papal election' topic. The most relevant tweets as determined by our score $s_t(c)$ are presented in Table 3.4. Indeed, all the tweets are relevant to our subject. Note that the $\sqrt{|W|}$ term is used instead of $|W|$ in the score, such that we favour longer tweets over shorter ones.

The variation in the magnitude of a topic over time is also instructive. Given a time interval, we can compute the magnitude as the proportion of tweets assigned to a topic relative to the total number of tweets. The magnitude for the 'Papal election' topic over a month computed over daily time intervals is presented in Figure 3.2.

We see that there are two peaks in magnitude of the topic. The most important one is centred around 13 March 2013, the day when the new Pope was elected. The second peak, with lower magnitude, is centred around 28 February 2013, the date when the previous Pope, Benedict XVI officially ended his papacy.

Chapter 4

Conclusions and Future Work

In this section we outline the topics of this deliverable and map out the direction of future work.

4.1 Conclusions

In this deliverable we have presented first experiments using clustering to group words and items in order to discover topics/events. We have presented two clustering algorithms that use the notion of similarity between items to be clustered. The first set of experiments was performed in order to establish a benchmark for clustering algorithms and show how we can cluster items with multiple modalities. The techniques used for these experiments achieved the best results in the MediaEval 2013 SED task. The second set of experiments showed how clustering can be used to uncover events and aid use cases.

The experiments have shown very promising results for clustering tailored to Social Media items. An experiment with human judging topical coherence showed very promising results, with only 15% of clusters having unclear meaning. We have also exemplified using the 'Papal election' topic how we can conduct a use case analysis using clusters of words.

The clustering methods presented in this deliverable show clear potential for practical application in the use cases of TrendMiner: clusters signify topics being talked about in Social Media and the changes in magnitude associated with them are trends that potential users are interested in. If a cluster is detected by the algorithm and salient changes occur in time (e.g. clear increases/decreases in comparison to earlier development), this information may be of great interest to users who need timely information about the activities in streaming Social Media, such as Twitter, both in the political and the financial use case. A concrete application could be a list of 'Top 5 Topics currently being discussed', generated by measuring the magnitude of individual clusters over a new set of messages.

Finally, we have delivered our clustering code as open source. Methods that use the clustering for analysis, such as ranking of topics in a collection of Social Media items, is delivered as a REST web service. We presented the functionality of this tool and the practicality of it being used by human users.

4.2 Future Work

On the subject of the clustering method in Section 3.2, we can imagine the model improved in multiple ways. First, we need to identify relevant and coherent clusters. There are methods of automatically assessing the coherence of clusters (Aletras and Stevenson, 2013a), which can be incorporated in a future model. Relevance to the use cases can be incorporated also by filtering out clusters that don't refer for example to politicians or don't contain named entities or nouns. For this, we can make use of features coming from WP2.

Automatic labelling of topics can also be accomplished in automatic ways, by using texts or images, replacing the current need for manually crafted labels. Our data items provide much richer information that goes beyond text. We can use the incremental clustering technique described in Section 2.3 or temporal spectral clustering (Chi et al., 2009) in order to add temporal smoothness constraints for event tracking in a temporal clustering model. The experiments from Section 3.1 shows how we can combine further metadata into our model.

A new potential avenue for research is represented by graphical models and Bayesian non-parametrics. Graphical models like LDA (Blei et al., 2003) and DMR (Mimno and McCallum, 2012), capture latent groups of words (called *topics*) that are assumed to generate the data. They essentially perform a soft clustering with each word belonging to every topic with a certain probability. Future experiments will first analyse use of other modalities, like geography or demographics, and then attempt to incorporate them in order to discover clusters and user attributes jointly.

Another line of exploration will be the link with the work carried out also in WP3 on prediction models for trends. The clustering results can be incorporated as features into prediction models in order to find topics that are relevant to the prediction task and identify signal in presence of noise, the main motivation of our previous work (Lampos et al., 2013). In the same way, we expect that the cluster features could be used as features for the summarisation work in WP4.

Appendix A

Clustering Web Service

This appendix provides a reference for the web service provided to access clustering results. The code and futher details can be found online[1].

To start the webservice, the dependent libraries must first be installed and the web service script to be run:

```
easy_install cherrypy
easy_install numpy
python wsscl.py cluster.file cluster.label.file
```

cluster.file is the file with the cluster assignments formated `word cluster.id importance` for each line.

cluster.label.file is the file with cluster coherence scores and labels formated `coherence label` for each line.

Then, the web service can be accesed using a tool like *curl*. There are 3 endpoints, all returning JSON objects:

```
GET cluster?c=cid&t=topw
```

returns the cluster with cid with the following proprieties: its label, coherence score and a list of the top topw words and their importance in the cluster.

```
GET words?w=word
```

returns the id of the cluster of the given word.

```
POST topics?t=notop&coh=c
```

receives a tweet file with 1 JSON/line and returns the top notop topics in the file and their magnitude score. Only topics with coherence greater or equal to c will be considered.

[1] https://github.com/danielpreotiuc/trendminer-clustering

We now show a few sample requests and their response:

```
curl -X GET  http://localhost:8080/cluster?c=70&t=5
```

gives the response:

```
{   "words": {
        "bergoglio": 0.0176,
        "#konklave": 0.01649,
        "argentinier": 0.01347,
        "#benedikt": 0.01303,
        "jorge": 0.01543,
        "#franziskus": 0.01338,
        "aires": 0.01422,
        "buenos": 0.01423,
        "#sedisvakanz": 0.01444,
        "#vatikan": 0.01385
        },
    "label": "Papal election",
    "coherence": "3"
}
```

This POST request:

```
cat mar13 | curl -i -k -H "Content-Type: application/json"
   -H "Accept: application/json" -X POST --data-binary @-
   http://localhost:8080/topics?t=5
```

generates the response:

```
HTTP/1.1 200 OK
Date: Tue, 01 Oct 2013 12:56:48 GMT
Content-Length: 282
Content-Type: application/json
Allow: POST
Server: CherryPy/3.2.4
```

```
{   "10": 0.04494563578393450 5,
    "195": 0.27725676241507263,
    "22": 0.03325237085901127 5,
    "70": 0.09150630579943611 1,
    "103": 0.07452533641801242 8
}
```

Bibliography

Aletras, N. and Stevenson, M. (2013a). Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics*, IWCS '13, pages 13–22.

Aletras, N. and Stevenson, M. (2013b). Representing topics using images. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '13, pages 158–167.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. L. (2009). On evolutionary spectral clustering. *ACM Trans. Knowl. Discov. Data*, 3(4):17:1–17:30.

Chung, F. (1997). *Spectral Graph Theory*, volume 92. American Mathematical Society.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*, KDD '96, pages 226–231.

Kittler, J., Hatef, M., Duin, R. P. W., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern analysis and Machine Intelligence*, 20:226–239.

Lampos, V., Preotiuc-Pietro, D., and Cohn, T. (2013). A user-centric model of voting intention from Social Media. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13.

Lau, J. H., Grieser, K., Newman, D., and Baldwin, T. (2011). Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, NAACL-HLT '11, pages 1536–1545.

Mayurathan, B., Pinidiyaarachchi, U., and Niranjan, M. (2013). Compact Codebook Design For Visual Scene Recognition By Sequential Input Space Carving. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*.

Mimno, D. M. and McCallum, A. (2012). Topic models conditioned on arbitrary features with dirichlet-multinomial regression. *CoRR*, abs/1206.3278.

Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14(2):849–856.

Petkos, G., Papadopoulos, S., and Kompatsiaris, Y. (2012). Social event detection using multimodal clustering and integrating supervisory signals. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, ICMR '12, pages 23:1–23:8.

Preoţiuc-Pietro, D., Samangooei, S., Cohn, T., Gibbins, N., and Niranjan, M. (2012). Trendminer: An Architecture for Real Time Analysis of Social Media Text. In *6th International AAAI Conference on Weblogs and Social Media*, pages 38–42. AAAI Press.

Reuter, T. and Cimiano, P. (2012). Event-based classification of social media streams. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, ICMR '12, pages 22:1–22:8.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905.

Smyth, S. and White, S. (2005). A spectral clustering approach to finding communities in graphs. In *Proceedings of the 5th SIAM International Conference on Data Mining*, pages 76–84.

Troncy, R., Mezaris, V., Reuter, T., Geva, S., and Papadopoulos, S. (2013). Social event detection: Mediaeval benchmarking initiative for multimedia evaluation. `http://www.multimediaeval.org/mediaeval2013/sed2013/`.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.