A Graph-based Approach to Topic Clustering for Online Comments to News

Ahmet Aker[1], Emina Kurtic[1], Balamurali A R[2], Monica Paramita[1], Emma Barker[1], Mark Hepple[1], and Rob Gaizauskas[1]

[1] University of Sheffield
ahmet.aker@, e.kurtic@, m.paramita@, e.barker@, m.r.hepple@, r.gaizauskas @sheffield.ac.uk
[2] LIF-CNRS Marseille, France
balamurali.ar@lif.univ-mrs.fr

Abstract. This paper investigates graph-based approaches to labeled topic clustering of reader comments in online news. For graph-based clustering we propose a linear regression model of similarity between the graph nodes (comments) based on similarity features and weights trained using automatically derived training data. To label the clusters our graph-based approach makes use of DBPedia to abstract topics extracted from the clusters. We evaluate the clustering approach against gold standard data created by human annotators and compare its results against LDA – currently reported as the best method for the news comment clustering task. Evaluation of cluster labelling is set up as a retrieval task, where human annotators are asked to identify the best cluster given a cluster label. Our clustering approach significantly outperforms the LDA baseline and our evaluation of abstract cluster labels shows that graph-based approaches are a promising method of creating labeled clusters of news comments, although we still find cases where the automatically generated abstractive labels are insufficient to allow humans to correctly associate a label with its cluster.

1 Introduction

Online news outlets attract large volumes of comments every day. *The Huffington Post*, for example, received an estimated 140,000 comments in a 3 day period[3], while *The Guardian* has reported receiving 25,000 to 40,000 comments per day[4]. These figures suggest that online commenting forums are important for readers as a means to share their opinions on recent news. The resulting vast number of comments and information they contain makes them relevant to multiple stakeholders in the media business. All user groups involved in online commenting on news would profit from easier access to the multiple topics discussed within a large set of comments. For example, commenter posters would be able to gain a quick overview of topics already discussed and insert their contributions at a relevant place in the discussion. Journalists who wrote the news article would have access to multiple conversation topics that their article has triggered and would be able to engage with their readers in a more focused way. Editors would be able to monitor the topics that are most interesting to readers, comment forum moderators' work would be easier and marketers could use conversations grouped around topics for developing personalized marketing strategies.

[3] http://goo.gl/3f8Hqu
[4] http://www.theguardian.com/commentisfree/2014/aug/10/readers-editor-online-abuse-women-issues

In most current on-line commenting forums, comments are grouped into threads – micro-conversations within the larger set of comments on an article, initiated and expanded by users, occasionally with some intervention of moderators. However, threads do not correspond to topics. As in all freely developing conversations, threads tend to drift away from the topic first introduced and often end up addressing multiple topics. Furthermore, comments addressing a particular topic may occur in multiple threads or on their own. Thus, in the current thread-based setup there is no easy way for readers to get access to all comments pertaining to a particular topic. Such topic-based clusters would be highly useful, allowing users to get an overview of the conversation and to home in on parts of particular interest to themselves, particularly if good-quality and coherent labels were associated with the clusters, permitting them to quickly understand what the comments within a particular cluster were about.

In this paper we introduce a way to automatically generate end-user friendly topic clusters of reader comments to online news articles. We propose graph-based methods to address two tasks: (1) to group reader comments into topic clusters; and (2) to label the clusters for the topic(s) they represent.

These tasks present us with several challenges that our methods need to address. For instance: (1) the number of topics discussed in a conversation about a news article is always unknown; (2) a single reader comment can be multi-topical itself and therefore one comment can belong to different topic clusters; (3) comments that implement a conversational action like jokes, sarcastic remarks or short support items (e.g. ‘Great’) are typically difficult to assign to a topic cluster according to their contents; (4) assigning cluster labels is not an easy task as a cluster of comments can represent a topic along multiple dimensions of granularity and expression.

Related work has reported Latent Dirichlet Allocation (LDA) [4] as the best-performing approach for reader comment clustering. However, LDA has the limitation that it requires prior knowledge of the number of topics, which cannot be set in our domain as news articles vary in numbers of comments and topics they address. In this paper we investigate the Markov Clustering Algorithm (MCL) [20], a graph-based approach that is able to determine the number of clusters dynamically from the data. We adapt it to clustering of reader comments and show that it significantly outperforms LDA.

The cluster labels are generated from the MCL clusters. From each cluster we extract topic terms using LDA trained over reader comments, which are then used to create a concept-based graph using DBPedia.[5] Using graph centrality, abstract labels are created for topics, which in turn are projected onto comment clusters.

The paper is organised as follows. Section 2 reviews relevant previous work. In Section 3 we describe the dataset we work with, which we downloaded from *The Guardian* online news portal. Section 4 discusses our clustering and cluster labelling approaches. The experimental setup for the evaluation of the proposed methods on Guardian data is reported in Section 5. The results are reported in section 6 and discussed in Section 7. In Section 8 we conclude the paper and outline directions for future work.

2 Related work

2.1 Comment clustering

Clustering of user comments has mostly been addressed in the context of automatic comment summarization, where LDA [4] has emerged as the best-performing clustering approach. For instance, Khabiri et al. [10] work on summarization of YouTube video comments. Prior to the summarization step, comments are grouped into clusters using K-Means clustering and LDA. Although the nature of LDA allows soft clustering[6], the

[5] http://wiki.dbpedia.org/

[6] Soft clustering methods allow one data item to be assigned to multiple clusters.

authors convert LDA output to hard-clusters[7] by assigning a comment C to the most likely topic, i.e. the topic t_r that maximizes $P(C|t_r) * P(t_r)$, where r is the topic/cluster index. The authors claim that LDA is superior to the K-Means approach.

Another comment clustering approach is reported by Llewellyn et al. [14], who apply LDA and K-Means, as well as simple metrics such as the cosine measure to cluster the comments of a single news article. The authors report LDA to be the best performing system. Ma et al. [15] also report an LDA approach to clustering news comments, where cluster information is used to generate comment summaries. Their evaluation happens at the level of the final output, i.e. the summary built using the clusters, and clustering is not evaluated in its own right. Since the summaries are generated using only the three clusters with the most comments, it is not clear how the investigated methods perform on clustering only. A similar study is reported in Liu et al. [13], who treat clustering as a prior step to comment summarisation and do not directly assess the quality of clusters.

Graph-based clustering has not been considered so far for user comments. However, it has been applied to clustering snippets resulting from Web search [19]. Each result snippet is annotated using TAGME[8]. TAGME identifies short phrases or topics in the snippet and links them to a pertinent Wikipedia page. The nodes in the graph are the topics. Edges between nodes denote how the nodes are related to each other within Wikipedia. A detailed survey of similar approaches is provided by Carpineto et al. [5].

The good performance of graph-based methods on snippet clustering indicates that it may be well-suited for the comment clustering task. In addition, graph-based clustering allows us to build clusters without prior knowledge of the number of topics required – a feature needed for our task of comment clustering and one which LDA lacks.

2.2 Cluster labelling

Labelling clusters can be seen as analogous to topic labelling. Most of the existing topic labelling procedures are extractive, i.e. they depend on extracting topic labels from within a text [12, 11]. For example, given a set of topic words representing the cluster, one word can be chosen as a label based on topic coherence [17]. However, using extractive labeling it is not possible to obtain collective terms or concepts as cluster labels. For instance, it is not possible to obtain *color* as the label for topic terms *red, blue, green, yellow*. This can be achieved using abstractive labeling methods. For labeling clusters in this work, we modify the graph-based topic labeling algorithm described in Hulpus et al. [8]. A DBPedia concept graph is created and the center of the graph is selected as the label. We modify the way graph is created to increase the fine-grainedness of labels as compared to the original work.

3 Data

3.1 Training data

For the graph-based clustering approach we present here, a regression model of similarity between graph nodes needs to be trained. Our training data was derived from a set of 3,362 online news articles and their associated comments downloaded from *The Guardian* online news portal over a period of two months (June-July 2014). *The Guardian* provides a specific RSS feed URL for each broad topic area, e.g. business, politics, etc. We manually collected RSS feeds for the topics: politics, health, sport, education, business, society, media, science, the-northerner, law, world-news, Scotland-news, money and environment. Using an in-house tool we visited the news published

[7] I.e. one comment can be assigned to only one cluster.

[8] http://tagme.di.unipi.it/

through the RSS feeds every 30 minutes, downloaded the article content and also recorded the news URL. Every recorded news URL was re-visited after a week (the time we found sufficient for an article to attract comments) to obtain its comments. Articles had between 1 and 6,223 associated comments, averaging 425.95 comments per article.

We build positive and negative training instances consisting of comment-comment pairs deemed to be topically similar/dissimilar from *The Guardian* data. To construct positive pairs we assume that if two or more comments associated with the same news article quote the same sentence in that article, then they are on the same topic and thus belong to the same topic cluster; i.e., positive pairs consist of comments that quote the same article sentence. When computing comment-comment similarity, if quotes are left in the comments, the similarity metric may be biased by the exact match as found in the quotes and may not be sensitive enough to capture similarity in comments that do not contain exactly matching quotes. For this reason, we expect that clustering results will be better if quotes are removed from the comments before computing similarity. To test this assumption we created two sets of training data. In the first set positive instances are comment pairs where the quote is left in the comments. In the second set positive instances are pairs of comments where we removed the shared quotes from the comments. For both training sets we set the topical similarity measure for each positive instance to be

$$quoteScore = len(quote_{C_1}) + len(quote_{C_2})/2 * len(sentence) \quad (1)$$

as the outcome. $len(X)$ returns the length of X in words and $quote_{C_i}$ is the segment of comment C_i quoted from $sentence$ in the original article. When computing the $quoteScore$ we make sure that the quoted sentence has at least 10 words. We add comment pairs to the positive training data whose $quoteScore$ values are $>= 0.5$ – a value we obtained empirically.

The negative instances are created by pairing randomly selected comments from two different articles from *The Guardian*. They are used to present the linear regression algorithm with the instances of comment pairs that are not on the same topic or are only weakly topically related. The topical similarity measure for each such pair was set to 0.We have in total 14,700 positive pairs and the same number of negative instances.

3.2 Testing data

For testing, clusters generated by human annotators are used as a gold standard data set. This data set was derived from 18 Guardian articles and associated comments distinct from those included in our training set. These articles and the first 100 comments associated with them served as the basis for a set of human-authored reference summaries of reader comments, created for the purpose of evaluating automatic comment summarization techniques. The procedure the summary authors were instructed to follow yielded comment clusters as a by-product: to facilitate the challenging task of writing summaries of 100 reader comments annotators were instructed to first group or cluster comments on the same topic and then to write the summary drawing on their clusters. We captured the clusters created in this process as well as the final summaries. At least two reference summaries plus related clusters were created for each article-comment set in the test set. The news article topics covered by this data are politics, sport, health, environment, business, scotland-news and science. On average, each annotator identified 8.97 clusters per news article.

4 Methods

4.1 Graph-based clustering

Our graph-based clustering approach is based on the Markov Cluster Algorithm (MCL) [20] shown in Algorithm 1. The nodes (V) in the graph $G(V, E, W)$ are the comments. Edges (E) are created between the nodes and have associated weights (W). Each comment is potentially connected to every other comment using an undirected edge. An edge is present if the associated weight is greater than 0. Such a graph may be represented as a square matrix M of order $|V|$, whose rows and columns correspond to nodes in the graph and whose cell values $m_{i,j}$, where $m_{i,j} > 0$, indicate the presence of an edge of weight $m_{i,j}$ between nodes V_i and V_j. Following the recommendation in [20] we link all nodes to themselves with $m_{i,i} = 1$. Other edge weights are computed based on comment-comment similarity features described in the next section below.

Once such a graph is constructed, MCL repeats steps 11-13 in the Algorithm until the maximum number of iterations $iter$ is reached[9]. First in step 11 the matrix is normalized and transformed to a column stohastic matrix, next expanded (step 12) and finally inflated (step 13). The expansion operator is responsible for allowing flow to connect different regions of the graph. The inflation operator is responsible for both strengthening and weakening this flow. These two operations are controlled by two parameters, the power $p -> 2$ results in too few clusters – and the inflation parameter $r - >= 2$ results in too many clusters. After some experimentation we set p to *2* and r to *1.5*, as these resulted in a good balance between too many and too few clusters.

Algorithm 1 MCL Algorithm

```
Require: a set of comments $\mathbf{C} = \{C_1 \ldots C_n\}$, a square matrix $M$ of order $n$, power parameter
    $p$, inflation parameter $r$, number of iterations $iter$, comment similarity measure $Sim\_Score$,
    minimum similarity parameter
 1: for all $m_{i,j}$ do
 2:     if $i = j$ then
 3:         $m_{i,j} = 1$
 4:     else if $Sim\_Score(C_i, C_j) \geq Min\_Sim$ then
 5:         $m_{i,j} = Sim\_Score$
 6:     else
 7:         $m_{i,j} = 0$
 8:     end if
 9: end for
10: repeat
11:     Normalize $M$ $(m_{i,j} = m_{i,j} / \sum_{k=1}^{n} m_{k,j})$
12:     Expansion: Raise $M$ to the $p^{th}$ power
13:     Inflation: $m_{ij} = (m_{ij})^r$
14: until current iteration $> iter$
15: Extract clusters from the final matrix
```

Once MCL terminates, the clusters are read off the rows of the final matrix (step 15 in Algorithm 1). For each row i in the matrix the comments in columns j are added to cluster i if the cell value $M_{i,j} > 0$ (the rows for items that belong to the same cluster will each redundantly specify that cluster). In this setting the MCL algorithm performs hard clustering, i.e. assigns each comment to exactly one cluster.

[9] MCL runs a predefined number of iterations. We ran MCL with 5000 iterations.

4.2 Weighting edges between comments

To weight an edge between two comments C_1 and C_2 we use the features below. When computing these features, except the $NE_{overlap}$ feature, we use terms to represent a comment instead of words, since we have found that terms are more suitable for computing similarity between short texts than single words [1]. Terms are noun phrase-like word sequences of up to 4 words. Terms are extracted using the TWSC tool [16], which uses POS-tag grammars to recognize terms.

- **Cosine raw count:** Cosine similarity [18] is the cosine of the angle between the vector representations of C_1 and C_2 :

$$cosine(C_1, C_2) = \frac{V(C_1) \cdot V(C_2)}{|V(C_1)| * |V(C_2)|} \tag{2}$$

  where $V(.)$ is the vector holding the raw frequency counts of terms from the comment.
- **Cosine TF*IDF:** Similar to the first cosine feature but this time we use the tf*idf measure for each term instead of the raw frequency counts. The idf values are obtained from the training data described in Section 3.1.
- **Cosine modified:** Liu et al. [13] argue that short texts can be regarded as similar when they have already a predefined D terms in common. We have adopted their modified cosine feature:

$$cosine_{modified}(C_1, C_2) = \begin{cases} \frac{V(C_1) \cdot V(C_2)}{D}, & \text{if } V(C_1) \cdot V(C_2) \leq D \\ 1, & \text{otherwise} \end{cases} \tag{3}$$

  We have set D experimentally to 5.
- **Dice:**

$$dice(C_1, C_2) = \frac{2 * |I(C_1, C_2)|}{|C_1| + |C_2|} \tag{4}$$

  where $I(C_1, C_2)$ is the intersection between the set of terms in the comments C_1 and C_2.
- **Jaccard:**

$$jaccard(C_1, C_2) = \frac{|I(C_1, C_2)|}{|U(C_1, C_2)|} \tag{5}$$

  where $U(C_1, C_2)$ is the union of sets of terms in the comments.
- **NE overlap:**

$$NE_{overlap}(C_1, C_2) = \frac{|I(C_1, C_2)|}{|U(C_1, C_2)|} \tag{6}$$

  where $I(C_1, C_2)$ is the intersection and $U(C_1, C_2)$ is the union set between the unique named entities (NEs) in the comments. We use the OpenNLP tools[10] to extract NEs.
- **Same thread**: Returns 1 if both C_1 and C_2 are within the same thread otherwise 0.
- **Reply relationship**: If C_1 replies to C_2 (or vise versa) this feature returns 1 otherwise 0. Reply relationship is transitive, so that the reply is not necessarily direct, instead it holds: $reply(C_x, C_y) \wedge reply(C_y, C_z) \Rightarrow reply(C_x, C_z)$

We use a weighted linear combination of the above features to compute comment-comment similarity:

$$Sim\_Score(C_1, C_2) = \sum_{i=1}^{n} feature_i(C_1, C_2) * weight_i \tag{7}$$

To obtain the weights we train a linear regression[11] model using training data derived from news articles and comments as described in Section 3.1 above. The target value for

[10] https://opennlp.apache.org/

[11] We used Weka (http://www.cs.waikato.ac.nz/ml/weka/) implementation of linear regression.

positive instances is the value of $quoteScore$ from equation 1 and for negative instances is 0.

We create an edge within the graph between comments C_i and C_j with weight $w_{i,j} = Sim\_Score(C_i, C_j)$ if Sim_Score is above 0.3, a minimum similarity threshold value set experimentally.

4.3 Graph-based cluster labelling

We aim to create abstractive cluster labels since abstractive labels can be more meaningful and can capture a more holistic view of a comment cluster than words or phrases extracted from it. We adopt the graph-based topic labelling algorithm of Hulpus et al. [8], which uses DBPedia [3], and modify it for comment cluster labelling.

Our use of the Hulpus et al. method proceeds as follows. An LDA model, trained on a large collection of Guardian news articles, plus their associated comments, was used to assign 5 (most-probable) topics to each cluster.[12] A separate label is created for each such topic, by using the top 10 words of the topic (according to the LDA model) to look up corresponding DBPedia concepts.[13] The individual concept graphs so-identified are then expanded using a restricted set of DBPedia relations,[14] and the resulting graphs merged, using the DBPedia merge operation. Finally, the central node of the merged graph is identified, providing the label for the topic.[15] The intuition is that the label thus obtained should encompasses all the abstract concepts that the topic represents.[16] Thus, for example, a DBPedia concept set such as {*Atom, Energy, Electron, Quantum, Orbit, Particle*} might yield a label such as *Theoretical Physics*.

5 Experiments

We compare our graph-based clustering approach against LDA which has been established as a successful method for comment clustering when compared to alternative methods (see Section 2). We use two different LDA models: *LDA1* and *LDA2*[17]. The LDA1 model is trained on the entire data set described in Section 3.1. In this model we treat the news article and its associated comments as a single document. This training data set is large and contains a variety of topics. When we require the clustering method to identify a small number of topics, we expect these to be very general, so that the resulting comment clusters are less homogeneous than they would be if only comments of a single article are considered when training the LDA model, as do Llewellyn et al. [14].

Therefore we also train a second LDA model (LDA2), which replicates the setting reported in Llewellyn et al. [14]. For each test article we train a separate LDA2 model on its comments. In training we include the entire comment set for each article in the

[12] The number of topics (k) to assign was determined empirically, i.e. we varied $2<k<10$, and chose k=5 based on the clarity of the labels generated.

[13] We take the most-common sense. The 10 word limit is to reduce noise. Less than 10 DBPedia concepts may be identified, as not all topic words have an identically-titled DBPedia concept.

[14] To limit noise, we reduce the relation set c.f. Hulpus et al. to include only *skos:broader, skos:broaderOf, rdfs:subClassOf, rdfs*. Graph expansion is limited to two hops.

[15] Several graph-centrality metrics were explored: *betweeness_centrality*, *load_centrality*, *degree_centrality*, *closeness_centrality*, of which the last was used for the results reported here.

[16] Hulpus et al. [8] merge together the graphs of multiple topics, so as to derive a single label to encompass them. We have found it preferable to provide a separate label for each topic, i.e. so the overall label for a cluster comprises 5 label terms for the individual topics.

[17] We use the LDA implementation from `http://jgibblda.sourceforge.net/`.

training data, i.e. both the first 100 comments that are clustered and summarised by human annotators, as well as the remaining comments not included in the gold standard. In building LDA2 we treated each comment in the set as separate document.

LDA requires a predetermined number of topics. We set the number of topics to 9 since the average number of clusters within the gold standard data is 8.97. We use 9 topics within both LDA1 and LDA2. Similar to Llewellyn et al. [14] we also set the α and β parameters to 5 and 0.01 respectively for both models.

Once the models are generated they are applied to the test comments for which we have gold standard clusters. LDA distributes the comments over the pre-determined number of topics using probability scores. Each topic score is the probability that the given comment was generated by that topic. Like [14] we select the most probable topic/cluster for each comment. Implemented in this way, the LDA model performs hard clustering.

For evaluation the automatic clusters are compared to the gold standard clusters described in Section 3.2. Amigo et al. [2] discuss several metrics to evaluate automatic clusters against the gold standard data. However, these metrics are tailored for hard clustering. Although our graph-based approach and baseline LDA models perform hard clustering, the gold standard data contains soft clusters. Therefore, the evaluation metric needs to be suitable for soft-clustering. In this setting hard clusters are regarded as a special case of possible soft clusters and will likely be punished by the soft-clustering evaluation method. We use fuzzy BCubed Precision, Recall and F-Measure metrics reported in [9] and [7]. According to the analysis of formal constraints that a cluster evaluation metric needs to fulfill [2], fuzzy BCubed metrics are superior to Purity, Inverse Purity, Mutual Information, Rand Index, etc. as they fulfill all the formal cluster constraints: *cluster homogeneity*, *completeness*, *rag bag* and *clusters size versus quantity*. The fuzzy metrics are also applicable to hard clustering.

To evaluate the association of comment clusters with labels created by the cluster labelling algorithm, we create an annotation task by randomly selecting 22 comment clusters along with their system generated labels. In the annotation bench for each comment cluster label, three random clusters are chosen along with the comment cluster for which the system generated the label. Three annotators (A, B, C) are chosen for this task. Annotators are provided with a cluster label and asked to choose the comment cluster that best describes the label from a list of four comment clusters. As the comment clusters are chosen at random, the label can correspond to more than one comment clusters. The annotators are free to choose more than one instance for the label, provided it abstracts the semantics of the cluster in some form.

In some instances, the comment label can be too generic or even very abstract. It can happen that a label does not correspond to any of the comment clusters. In such cases, the annotators are asked not to select any clusters. These instances are marked NA (not assigned) by the annotation bench. Inter-annotator agreement is measured using Fleiss Kappa metric [6]. We report overall agreement as well as agreement between all pairs of annotators. The output of the cluster labelling algorithm is then evaluated with the annotated set using standard classification metrics.

6 Results

Clustering results are shown in Table 1. A two-tailed paired t-test was performed for a pairwise comparison of the fuzzy Bcubed metrics across all four automatic systems and human-to-human setting.

Metric	Human1-Human2	graph-Human	graph-Human-quotesRemoved	LDA1-Human	LDA2-Human
Fuzzy B^{3}Precision	0.41	0.29	0.30	0.25	0.23
Fuzzy B^{3}Recall	0.44	0.30	0.33	0.29	0.17
Fuzzy B^{3}FMeasure	0.40	0.29	0.31	0.24	0.18

Table 1. Cluster evaluation results. The scores shown are macro averaged. For all systems the metrics are computed relative to the average scores over Human1 and Human2. *graphHuman* indicates the setting where similarity model for graph-based approach is trained with quotes included in the comments (see Section 4.1).

Annotators	**A-B**	**B-C**	**C-A**	**Overall**
Agreement	0.76	0.45	0.64	0.61

Table 2. Annotator agreement (Fleiss Kappa) for comment labelling over 22 comment clusters

Firstly, we observe that human-to-human clusters are significantly better than each of the automatic approaches in all evaluation metrics[18]. Furthermore, we cannot retain our hypothesis that the graph-based approach trained on the training data with quotes removed performs better than the one that is trained on data with quotes intact.[19] Although the results in the *quotes removed* condition are better for all metrics, none of the differences is statistically significant. We use the better performing model (graph without quotes) for comparisons with other automatic methods.

Secondly, the LDA1 baseline performs significantly better than the re-implementation of previous work, LDA2, in all metrics. This indicates that training LDA model on the larger data set is superior to training it on a small set of articles and their comments, despite the generality of topics that arises from compressing topics from all articles into 9 topic clusters for LDA1.

Finally, the *quotes removed* graph-based approach (column 4 in Table 1) significantly outperforms the better performing LDA1 baseline in all metrics. This indicates that graph-based method is superior to LDA, which has been identified as best performing method in several previous studies (cf. Section 2). In addition, clustering comments using graph-based methods removes the need for prior knowledge about the number of topics - a property of the news comment domain that cannot be considered by LDA topic modelling.

Tables 2 and 3 present results from the evaluation of the automatically generated comment cluster labels. Table 2 shows the agreement between pairs of annotators and overall, as measured by Fleiss' Kappa on the decision: given the label, which cluster does it describe best. Overall there is a *substantial agreement* of $\kappa = 0.61$ between the three annotators. The annotator pair B-C, however, achieves only *moderate agreement* of $\kappa = 0.45$, suggesting that some annotators make idiosyncratic choices when assigning more generic abstractive labels to clusters.

Table 3 shows the evaluation scores for the automatically generated labels, given as precision, recall and F scores results, along with the percentage of labels not assigned (NA) to any cluster. Overall, annotators failed to assign labels to any cluster in 40.9% of cases. In the remaining cases, where annotators *did* assign the labels to clusters, this

[18] The difference in these results is significant at the Bonferroni corrected level of significance of $p < 0.0125$, adjusted for 4-way comparison between the human-to-human and all automatic conditions.

[19] We apply both models on comments regardless whether they contain quotes or not. However, in case of *graph-Human-quotesRemoved* before it is applied on the testing data we make sure that the comments containing quotes are also quotes free.

was done with fairly high precision (0.8), and so as to achieve an overall average recall of 0.5, suggesting that meaningful labels had been created.

Annotator	**Precision**	**Recall**	**F-score**	**NA%**
A	0.78	0.32	0.45	59.1 (13/22)
B	1.00	0.45	0.62	54.5 (12/22)
C	0.62	0.73	0.67	9.1 (2/22)
mean	0.80	0.50	0.58	40.9

Table 3. Evaluation results of the cluster labeling system for each of the 3 annotators. NA corresponds to the number of labels not assigned.

7 Discussion

The comment clustering results demonstrate that graph-based clustering is able to outperform the current state-of-the-art method LDA as implemented in previous work at the task of clustering reader's comments to online news into topics.

In addition to the quantitative study reported above we also performed a qualitative analysis of the results of the graph-based clustering approach. That analysis reveals that disagreements in human and automatic assignment of comments to clusters are frequently due to the current approach ignoring largely conversational structure and treating each comment as an independent document. Commenting forums, however, are conversations and as such they exhibit internal structuring where two comments are functionally related to each other, so that the first pair part (FPP) makes relevant the second pair part (SPP). In our automatic clusters we frequently found answers, questions, responses to compliments and other stand-alone FPPs or SPPs that were unrelated to the rest of an otherwise homogeneous cluster. For example, the comment *"No, just describing another right wing asshole."* is found as the only odd comment in otherwise homogeneous cluster of comments about journalistic standards in political reporting. Its FPP *"Wait, are you describing Hillary Clinton?"* is assigned to a different cluster about US politician careers. We assume that our feature *reply relationship* was not sufficiently weighted to account for this, so that we need to consider alternative ways of training, which can help identify conversational functional pairs.

A further source of clustering disagreements is the fact that humans cluster both according to content and to the conversational action a comment performs, while the current system only clusters according to a comment's content. Therefore, humans have clusters labelled *jokes*, *personal attacks to commenters or empty sarcasm*, *support*, etc., in addition to the clusters with content labels. A few comments have been clustered by the annotators along both dimensions, content and action, and can be found in multiple clusters (soft clustering). Our graph-based method reported in this work produces hard clusters and is as such comparable with the relevant previous work. However, we have not addressed the soft-clustering requirement of the domain and gold standard data, which has most likely been partly reflected in the difference between human and automatic clustering results. When implementing soft-clustering in future one way to proceed would be to add automatic recognition of a comment's conversational action, which would make graph based clustering more human-like and therefore more directly comparable to the gold standard data we have.

Our evaluation of cluster labelling reveals that even though the labelling system has acceptable precision, recall is rather low, due, in large part, to the high number of NA labels. We qualitatively analysed those instances that were NA for more than one annotator. Barring three instances, where the system generated labels like *concepts in Metaphysics, Chemical elements, Water* with no obvious connection to the underlying cluster content, labels generated by the system describe the cluster in a meaningful way. However, in some cases annotators failed to observe the connection between the comment cluster and the label. This may be due to the fact that users expect a different level of granularity – either more general or more specific – for labeling. For instance, a comment talking about a *dry, arid room* can have a label like *laconium* but users may prefer having a label that corresponds to dryness. This is very subjective and poses a problem for abstractive labelling techniques in general.

The expansion of a graph using DBPedia relations encompasses related concepts. However, this expansion can also include abstract labels like *Construction, organs, monetary economics, Articles_containing_video_clips* etc. This happens due to merging of sub-graphs representing concepts too close to the abstract concepts. In these cases, the most common abstract node may get selected as the label. These nodes can be detrimental to the quality of the labels. This can be prevented by controlled expansion using more filtered DBPedia relations and a controlled merging.

8 Conclusion

We have presented graph-based approaches for the task of assigning reader comments on online news into labeled topic clusters. Our graph-based method is a novel approach to comment clustering, and we demonstrate that it is superior to LDA topic modeling – currently the best performing approach as reported in previous work. We model the similarity between graph nodes (comments) as a linear combination of different similarity features and train the linear regression model on an automatically generated training set consisting of comments containing article quotations.

For cluster labeling we implement a graph-based algorithm that uses DBPedia concepts to produce abstractive labels that generalise over the content of clusters in a meaningful way, thus enhancing readability and relevance of the labels. User evaluation results indicate that there is a scope for improvement, although in general the automatic approach produces meaningful labels as judged by human annotators.

Our future work will address soft-clustering, improve feature weighting and investigate new features to better model conversational structure and dialogue pragmatics of the comments. Furthermore, we aim to create better training data. Currently, the quote-based approach for obtaining positive training instances yields few comment pairs that are in some reply structure – a comment replying to a previous comment is unlikely to quote the same sentence in the article and thus comment-pairs where one comment replies to the other are not taken to the training data. Due to this our regression model does not give much weight to the reply feature even though this feature is very likely to suggest comments are in the same topical structure. Finally, we also aim to improve the current DBPedia-based labeling approach, as well as explore alternative approaches to abstractive labeling to make cluster labels more appropriate.

Acknowledgements

The research leading to these results has received funding from the EU - Seventh Framework Program (FP7/2007-2013) under grant agreement n610916 SENSEI.

References

1. Aker, A., Kurtic, E., Hepple, M., Gaizauskas, R., Di Fabbrizio, G.: Comment-to-article linking in the online news domain. In: Proceedings of MultiLing, SigDial 2015 (2015)

2. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. Information retrieval 12(4), 461–486 (2009)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. Springer (2007)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research 3, 993–1022 (2003)
5. Carpineto, C., Osiński, S., Romano, G., Weiss, D.: A survey of web clustering engines. ACM Computing Surveys (CSUR) 41(3), 17 (2009)
6. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological bulletin 76(5), 378 (1971)
7. Hüllermeier, E., Rifqi, M., Henzgen, S., Senge, R.: Comparing fuzzy partitions: A generalization of the rand index and related measures. Fuzzy Systems, IEEE Transactions on 20(3), 546–556 (2012)
8. Hulpus, I., Hayes, C., Karnstedt, M., Greene, D.: Unsupervised graph-based topic labelling using dbpedia. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. pp. 465–474. WSDM '13, ACM, New York, NY, USA (2013), `http://doi.acm.org/10.1145/2433396.2433454`
9. Jurgens, D., Klapaftis, I.: Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In: Second joint conference on lexical and computational semantics (* SEM). vol. 2, pp. 290–299 (2013)
10. Khabiri, E., Caverlee, J., Hsu, C.F.: Summarizing user-contributed comments. In: ICWSM (2011)
11. Lau, J.H., Grieser, K., Newman, D., Baldwin, T.: Automatic labelling of topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. pp. 1536–1545. Association for Computational Linguistics (2011)
12. Lau, J.H., Newman, D., Karimi, S., Baldwin, T.: Best topic word selection for topic labelling. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. pp. 605–613. Association for Computational Linguistics (2010)
13. Liu, C., Tseng, C., Chen, M.: Incrests: Towards real-time incremental short text summarization on comment streams from social network services (2015)
14. Llewellyn, C., Grover, C., Oberlander, J.: Summarizing newspaper comments. In: Eighth International AAAI Conference on Weblogs and Social Media (2014)
15. Ma, Z., Sun, A., Yuan, Q., Cong, G.: Topic-driven reader comments summarization. In: Proceedings of the 21st ACM international conference on Information and knowledge management. pp. 265–274. ACM (2012)
16. Pinnis, M., Ljubešić, N., Ştefănescu, D., Skadiņa, I., Tadić, M., Gornostay, T.: Term extraction, tagging, and mapping tools for under-resourced languages. In: Proceedings of the 10th Conference on Terminology and Knowledge Engineering (TKE 2012), June. pp. 20–21 (2012)
17. Röder, M., Both, A., Hinneburg, A.: Exploring the space of topic coherence measures. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. pp. 399–408. ACM (2015)
18. Salton, G., Lesk, E., M.: Computer evaluation of indexing and text processing. In: Journal of the ACM. vol. 15, pp. 8–36. ACM Press, New York, NY, USA (1968)
19. Scaiella, U., Ferragina, P., Marino, A., Ciaramita, M.: Topical clustering of search results. In: Proceedings of the fifth ACM international conference on Web search and data mining. pp. 223–232. ACM (2012)
20. Van Dongen, S.M.: Graph clustering by flow simulation (2001)