Journal of Web Engineering, Vol. 15, No. 3&4 (2016) 361–396

ONTOLOGY-ASSISTED DISCOVERY OF HIERARCHICAL TOPIC CLUSTERS ON THE SOCIAL WEB

KRISTIAN SLABBEKOORN, TOMOYA NORO, TAKEHIRO TOKUDA
Department of Computer Science, Tokyo Institute of Technology
Meguro, Tokyo 152-8552, Japan
{*kt.slabbekoorn, noro.t.tech*}*@gmail.com, tokuda@cs.titech.ac.jp*

Received June 12, 2015
Revised February 22, 2016

Discovery and clustering of users by their topic of interest on the Social Web can help enhance various applications, such as user recommendation and expert finding. Traditional approaches, such as latent semantic analysis-based topic modeling or k-means document clustering, run into issues when content is sparse, the number of existing topics is unknown and/or we seek topics that are hierarchical in nature. In this paper, we propose a method for ontology-assisted topic clustering, in which we map Social Web user content to ontological classes to overcome sparsity. Using a novel ranking technique for calculating the topical similarity between individuals at different topic scopes, we construct graphs on which we apply a quasi-clique algorithm in order to find topic clusters at that scope, without having to pre-define a target number of topics. Our approach allows (1) the topic scope to be controlled in order to discover general or specific topics; (2) the automatic labeling of clusters with tags that are human and machine-understandable; and (3) graphs to be clustered recursively in order to generate a hierarchy of topics. The approach is evaluated against ground truths of Twitter users and the 20-newsgroups dataset, commonly used in document clustering research. We compare our approach to standard and Twitter-specific latent Dirichlet allocation (LDA), hierarchical LDA, and standard and hierarchical *k*-means clustering. Results show that our method outperforms regular LDA by up to 24.7%, Twitter-LDA by up to 11.9%, and *k*-means by up to 26.7% on Social Web content. It performs equivalently, depending on several factors, to these approaches on a dataset of traditional documents. Additionally, our method can discover the appropriate number and composition of topics at a given topic scope, whereas *k*-means clustering cannot account for differences in scope.

Keywords: Ontology, hierarchical clustering, topic modeling, community detection, Social Web
Communicated by: M. Gaedke & C. Bizer

1. Introduction

With the emergence and rapid popularization of various social media, communication is done less frequently in person, and increasingly more often via online social services such as Facebook and Twitter. Recent analyses show that US Web users spend on average 23 hours per week on email, text and social services; 87% of users log in to Facebook at least once per week, and 32% log in to Twitter at least once per week [14]. Inevitably, communities form on these media, bringing together friends and like-minded individuals. Restricting our scope to the Social Web, we can distinguish between connections between individuals based on (1) interpersonal ties, and (2) mutual interests. There are many applications that can benefit from the discovery of these types of connections, and from the clustering of individuals into social or interest-based groups. For example, discovering the structure of connections between people can provide social scientific insights into human behavior or assist

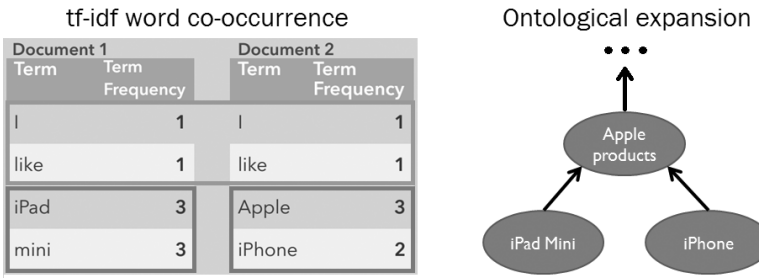

Fig. 1. A simple comparison of tf-idf word co-occurrence and ontological expansion. tf-idf is unable to capture the similarity between the documents on the left. We can expand both documents to the "Apple products" class when doing ontological expansion, on the right.

user recommendation based on similarity of interests, and expert finding, where we are looking for individuals with specific skills, mentioning unique types of terminology in his content.

The tie-based clustering of individuals (henceforth *users*) by their connections is a classic, interdisciplinary task within the social, natural and computer sciences, commonly known as *community detection*. There has been a significant amount of research into this notion [35][27]. A population is represented as a graph, where each vertex is a user and each edge ties two users together. Edges can be weighted in order to express strong or weak ties. Such a graph can then be clustered into one or more subgraphs (the communities). There exist many algorithms for clustering in graphs, such as modularity optimization [34] or clique percolation [12], which generally boil down to the principle of the *strength of weak ties* [13], where the goal is to identify weak, inter-community connections and remove or ignore them; what remains are the communities of interest. A distinguishing property of such community detection algorithms is that the number of communities, or clusters, do not need to be known beforehand. A downside to tie-based community detection is that content is not considered; we argue that on the modern Social Web, interpersonal ties matter less when we are purely interested in users with similar topics of interest.

When we are more concerned with content, we can turn to the related but distinct tasks of *topic modeling* and *document clustering* [48]. Here, we try to assign similar documents to the same semantic topic. Links between documents are not assumed to be present, and we must therefore analyze the content of the documents in order to determine conceptual similarity. This is most commonly achieved by leveraging word co-occurrence – if words that are relatively unique to the entire collection appear often together in the same subset of documents, we may assume that this subset shares some common topic. Since this topic is not expressed in the documents themselves, it is called *latent*; the most common class of topic modeling algorithms use a technique known as *latent semantic analysis* (LSA) to find these "hidden" topics [10]. Document clustering works in a similar way, where a k-means-based classifier is often used to divide documents into a pre-defined number of topics. We can map the tasks of topic modeling and document clustering onto the Social Web by considering each user's content as a document; the latent topics belonging to groups of users can then be regarded as user "interests". However, a problem with methods that rely on word co-occurrence is that in cases where not much text is available, or text is not necessarily grammatically accurate, these methods tend to perform poorly [19][50]. See figure 1 (left) for a simple example of this. Another limitation of most existing document clustering approaches is that the number of topics needs to be known beforehand, which is rarely the case when dealing with an ad hoc collection of users on the Social Web (such as a stream of users resulting from a keyword search on Twitter, for example).

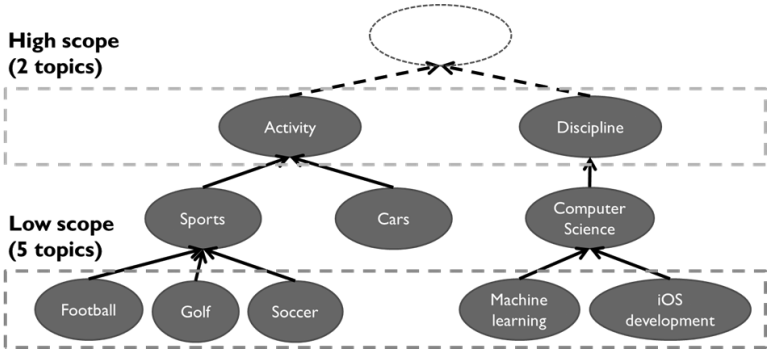

Fig. 2. Using *scoped topical similarity* clustering, we can manipulate the *topic scope* in order to cluster users at an arbitrary height of the topic hierarchy. In this example, we can cluster by two broad topics (high scope) or five specific topics (low scope).

In this paper, we take a hybrid approach: we use an ontology-assisted topic modeling technique to determine the topical similarity among a population of Social Web users, then use a quasi-clique community detection algorithm to cluster users by shared topics of interest – we assume the number of topics is not known, and infer this from the data.

Initially, we capture topic information by applying ontological expansion: we match named entities discovered in users' text content to an external knowledge base, DBpedia [3], and obtain full class hierarchies for three types of classes (DBpedia [1], YAGO [45], Schema.org [40]). See figure 1 (right). Based on these class hierarchies, we propose a weighting scheme that allows us to determine *trait vectors* (ontological classes weighted with a prominence score) that characterize individual users within a population, and capture users' dominant topics of interest. The scope of topics can be dynamically controlled with a *topic scope* mechanism – it depends on our application whether we want to detect users interested in "sports", or go a step down and distinguish the "American football" interest from the "soccer" interest, for example. This is an important distinction from other topic modeling approaches, which usually require the number of topics to be known in advance, and may not capture hierarchical topics.

In the second part, we construct an undirected *scoped topical similarity* (STS) graph of Social Web users, weighted based on the cosine similarity between users' trait vectors. This graph is clustered into topics of interest shared by groups of users using a variation of the clique-based Highly Connected Subgraph (HCS) algorithm [17], which finds communities by recursively splitting a graph along its minimum cut. We modify this algorithm by (1) considering edge weights when determining minimum cuts; (2) loosening constraints on what constitutes "highly-connectedness"; and (3) assigning lowly-connected users that would otherwise be dropped to clusters that match their traits best. Depending on how we set our topic scope, we obtain a high or low slice of the topic hierarchy. See figure 2 for an illustration of this. The flexibility of the approach enables two interesting side-effects: using the same algorithms, (1) we can compute cluster-based trait vectors that allow us to label clusters with appropriate tags that are both human-readable and machine-understandable, since they link directly into the Linked Data cloud [4]; and (2) we can cluster a graph divisively into increasingly more specific topics by recursively applying the described method on resulting topic clusters.

We evaluate the approach in three ways. We first investigate the quality of our STS weighting scheme, in terms of nDCG [22] rankings for two different seed users with known interests, and

compare the results to baselines such as tf-idf term weighting to show the advantages over word co-occurrence-based methods. We then evaluate scoped topic clustering based on (1) a manually constructed ground truth of 175 Twitter users, distributed over 4 topics and 11 sub-topics; and (2) a dataset commonly used in document clustering research (the 20-newsgroups dataset [38]). The topic clustering results are compared mainly to latent Dirichlet allocation (LDA) [7], the current gold standard for topic modeling; a state-of-the-art Twitter-centric version of LDA, called Twitter-LDA [50]; and k-means clustering [16], a classic document clustering algorithm. Lastly, we evaluate full hierarchical clustering, comparing to hLDA [6] and hierarchical k-means.

After a short re-iteration of the main contributions of this work, the remainder of the paper is organized as follows. In section 2, we outline past research into community detection and document clustering on the Social Web, their problems, and how our proposal differs from these works. Sections 3 (topic discovery and representation) and 4 (hierarchical clustering) will provide a detailed explanation of the two parts of our approach. In section 5, we outline our experimental setup and evaluate our work compared to baselines and the state-of-the-art. Finally, in section 6, we conclude the paper with a discussion of the results and directions for future work.

1.1. *Contributions*

We can define one main contribution and two sub-contributions for this work.

1. We introduce an ontology-assisted approach for the hierarchical clustering of Social Web users by topic of interest when the number of topics is not known in advance – we allow the discovery of topic clusters at a chosen *topic scope* that is independent of the underlying data. Using these techniques, we additionally describe:
   (a) a method for automatic labeling of clusters with human and machine-readable topic tags;
   (b) a method for divisive clustering of a user graph into a topic hierarchy.

2. Related work

Topic modeling, (ontology-based) document clustering and classification, and community detection are all classic, multi-disciplinary problems that have been the subject of significant research, leading to the development of many different methods and algorithms [5][27]. In this section we discuss some of the more significant works in each of these areas.

Topic modeling approaches In 2003, Blei et. al. [7] revolutionized the topic modeling field with the introduction of a pLSA-based generative probabilistic topic modeling technique called latent Dirichlet allocation (LDA). Since then, a great number of variations of and additions to LDA have been developed that address specific use cases. We briefly discuss two Twitter-centric approaches and a hierarchical approach.

Hoang et. al. [19] conducted an empirical study on different ways of performing topic modeling on Twitter tweets using the original LDA model and the author-topic model [41]. They find that topics learned from documents formed by aggregating tweets posted by the same users may help to significantly improve some user profiling tasks. The work by Zhao et. al. [50] proposes the TwitterLDA topic model for microblogging data. TwitterLDA is another variant of LDA, *hLDA*, which assumes there is only one common topic for all words in each tweet, and that each word in a tweet

is generated from either a background topic or the user's perceived topic. The authors demonstrate a significant improvement over standard LDA on Twitter data. In [6], Blei et. al. introduce a hierarchical version of LDA, using a nested Chinese restaurant process to generate infinitely deep trees in order to cluster documents at multiple levels of topic abstraction. We include these latter two works in our evaluation of a Twitter-based dataset. The main issue with all approaches that incorporate a variation of LDA is the inherent reliance on word co-occurrence between users or documents, and the need to pre-define the number of topics (number of hierarchical levels for hLDA) that exist in the data. This is especially problematic on the Social Web, where text content is sparse and noisy, leading to insufficient term overlap to make accurate predictions about their latent semantics, and where topics cannot be concretely pre-defined. We address the first concern by mapping text content to a subset of the roughly 300,000 hierarchical classes of an external ontology; all subsequent processing is done on these classes. This dimensionality reduction helps minimize sparsity and noise, while leaving enough classes to be able to model unique user interests. Secondly, we do not need to pre-define any number of topics: given a topic scope, we infer this from the data using clustering techniques.

Document clustering and classification approaches Approaches that leverage some form of document classification involve the a priori definition of a set of possible topics or categories, with the subsequent application of NLP and machine learning techniques to allocate each document or user to one of these categories. For example, Zhao et. al. [51] propose a topic-oriented community detection approach which combines both social objects clustering and link analysis, in order to identify more meaningful topical communities. They classify documents according to a number of pre-defined topics, taking into account interpersonal connections as well as content similarity. Tsur et. al. [47] present an algorithm for classifying Twitter tweets into pre-defined topics based on hashtags. While we agree with these works that topic-oriented communities are more meaningful than tie-oriented ones, these types of techniques are classification methods that pre-define a set of around eight to ten flat topics to assign users or documents to, which limits usefulness in scenarios where we are interested in specialized topics or hierarchical topics. Our approach is not one of classification, but of clustering: we construct topics dynamically in terms of classes that characterize groups of topically similar users.

An advanced streaming, machine learning-based method for matching individual tweets to a predefined ontology of 300 classes with high precision was introduced by Twitter's user modeling team [49], and is currently in active use as a part of Twitter Analytics.[a] Although this work focuses on individual tweet topic modeling, it offers some insights into extending this to user interest modeling based on tf-idf-style weighting of the discovered tweet topics and additional user behavior (retweet, reply, favorite, etc): this is similar to how we model user interests, however we aim to discover much more fine-grained interests by leveraging and defining them in terms of hundreds of thousands of classes instead. This allows us explicit control over the scope of interests and to cluster a larger collection of users by their shared interests.

(Ontology-assisted) Social Web approaches The emergence of the Social Web has given rise to a substantial body of research exploring differences between Social Web content and standard content, and developing specialized methods and variations on common techniques to deal with these differences. In [39], for example, the authors rebuild the NLP pipeline from start to end with microblogs in mind, reporting double the F-measure compared to the state-of-the-art. In [28], a graph-based method

[a] `https://analytics.twitter.com`

is introduced that conducts named entity normalization and recognition simultaneously, looking for pairs of words that share the same lemma and judging whether these words mention the same entity in a stochastic way. A general exploration into the difficulties of standard NLP on the Social Web is given in [11]. We will limit ourselves to discussing works that specifically combine the Social Web and semantic expansion to external ontologies (most commonly Wikipedia) into their approach.

In [32], Michelson et. al. present results on discovering Twitter users' topics of interest by examining entities mentioned in their tweets, and deriving a topic profile based on Wikipedia categories. Their findings corroborate our own results that using an external concept hierarchy can be a good indicator for categorizing general user interests, but we argue that Wikipedia categories are too disorganized to be useful, and choose to rely mostly on the more robust YAGO [45] taxonomy instead. In [2], Abel et. al. incorporate semantic expansion to an external ontology into their Twitter user modeling process for improved personalized news recommendations, showing significant improvements over simpler approaches such as relying on hashtags. Similarities between the above two works and ours end at the way user topic profiles are represented, however; they do not incorporate clustering of users using these profiles. We are not aware of research on the clustering of Social Web users specifically using ontological classes. However, significant research has been conducted into the clustering of regular documents with the assistance of ontologies. The most prominent work is by Hotho et. al. [20], who propose using WordNet [33] concepts to assist in document clustering, and show improved results. They disambiguate terms to concepts in a simple way and apply a k-means type clustering based on concept frequencies into a pre-defined number of topics. Unlike our approach, they do not exploit a tf-idf-style weighting of concepts and rely only on concept frequency to determine similarity, require topics to be pre-defined, and do not support hierarchical topics. We also argue that WordNet does not provide sufficient granularity for concepts: only relatively high-level topics can be obtained, whereas we allow even highly specific topics to be defined by considering not only classes, but at the lowest level also the Wikipedia pages themselves.

We have previously introduced a method in which we enhanced a Twitter recommendation system based on user follow relations with a post-processing step that leveraged an external ontology [43]. We reported positive results, but the approach is not applicable for comparing different users to find if they are similar. Our new work described in this paper essentially extends this previous post-processing step into a general approach for clustering a population of users by interest.

Community detection-based approaches To our knowledge, there has been little to no substantial research into hybrid topic modeling and graph-based community detection approaches for clustering. A notable exception is Lancichinetti et. al. [25], who develop a community detection-based method for document clustering applied on a graph expressing tf-idf-based similarities between documents, and show large improvements over the LDA gold standard in terms of accuracy and reproducibility. Our work can be regarded as a further exploration of this approach, substituting an ontology-based method for the topic modeling part – which we argue should work better with sparse content – and additionally allowing hierarchical clustering as well as automatic topic labeling.

3. Topic discovery and representation

In this section, we will detail our approach for the discovery of ontology-based, hierarchical clusters of user interests. Although our approach is general enough to be applied to any type of documents, it is designed to be especially useful in a noisy Social Web environment. We will focus on Twitter as a

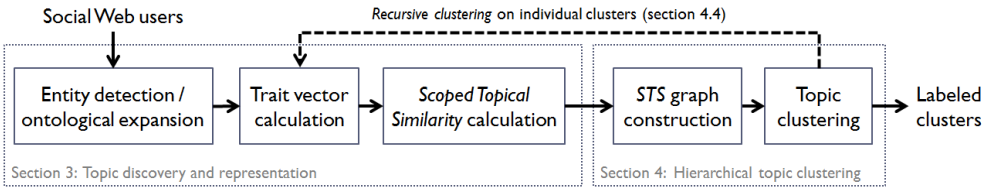

Fig. 3. The general flow of our approach. Sections 3 and 4 explain each step in detail.

use case for the majority of this paper. Realistically, user interests are dynamic and evolve over time, so an accurate topic model would need to incorporate the time dimension in some form. Dynamic topic detection is outside the scope of this paper: we choose to make the simplification of considering the latest 500 or so posts on a user's timeline to be representative of their current interests. Note that users do not necessarily have only one topic of interest: we consider "user interest" to be a distribution over topics, where a user can be interested in an undetermined number of topics in various degrees. Furthermore, we also posit that each topic is hierarchical in nature: that is, a user generally interested in "sports", might be specifically interested in "hockey" and "golf" to different degrees. We aim to develop a model that can accurately capture this type of information.

To differentiate Social Web-oriented clustering from traditional document clustering, we consider a collection of users U, as opposed to documents D. These notions are conceptually similar: each u_i in U contains some number of posts $\{p_1, p_2, ..., p_n\} \in P_i$, similar to how a document $d_i \in D$ would consist of lines or paragraphs, which are both in turn collections of words $\{w_1, w_2, ..., w_m\} \in W_i$. In order to keep our approach application-agnostic, we consider only raw text content – usually the users' text (micro-)posts, P_i.

Overall, we can divide the approach into two parts, comprising a combined five steps, as shown in figure 3. Sections 3.1 through 3.4 and section 4 will explain each step in detail.

3.1. *Detecting named entities*

The first step is to ontologically expand each user's posts by applying *named entity recognition* (NER) to link entities to ontology classes. We do this to mitigate the sparsity of raw text data in a Social Web context. In order to keep the scope of our research within bounds, we have decided not to devise our own approach for NER. Instead, we opt to use DBpedia Spotlight [31][8], a proven off-the-shelf entity recognizer that detects named entities in text and generates links to DBpedia [3]. See figure 4 for a visual explanation of how we use DBpedia Spotlight in our system.

We aim to derive per-user topic profiles, meaning we map whole users to some distribution of topics. To this end, we take a significant portion of a user's text content – for our Twitter use case, we take the most recent 500 tweets – and apply DBpedia Spotlight on this content. We make the observation that the more input data that we have, the better entity recognition imperfection will be mitigated. Recognition is only roughly 85% accurate on well-formed text [8], and this number is likely significantly lower in a Social Web context ([11] shows an F1 score of 28.3% on non-preprocessed "whole tweet" entity linking, but it is unclear how performance changes with microblog-specific preprocessing and term-level entity linking, which is what we are interested in). However, by the law of large numbers, entities should converge to the most dominantly present topics in spite of this ratio of error. Still, recognizing these difficulties of entity recognition on the Social Web, we pre-process the input in two ways to improve entity recognition performance for this use case.

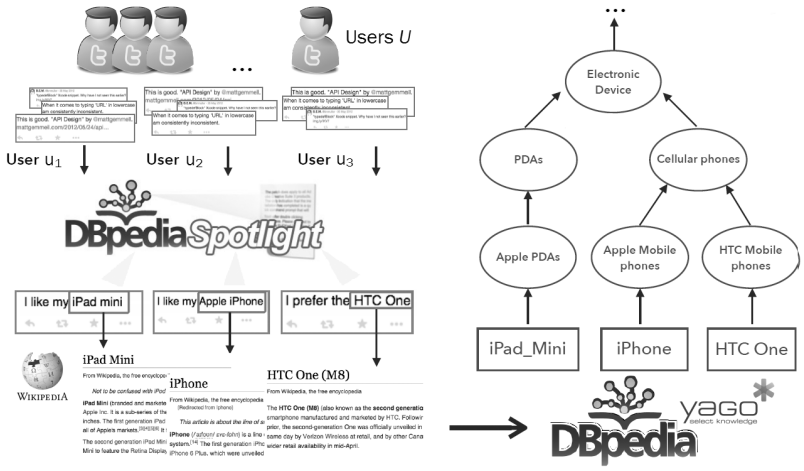

Fig. 4. Tweets are collected from the timelines of a collection of users U. DBpedia Spotlight is then applied on the text in these tweets to find named entities, which are linked to DBpedia entities and (a simplified representation of) their class hierarchy.

First, we pre-filter the raw text associated to tweets: we remove user mentions, urls, and the hashbangs from hashtags, as well as certain terms that are noise more often than not (e.g. "Twitter", and netslang such as "lol"). Since Spotlight already applies tokenization and other common NLP techniques to its input, any further text pre-processing is left to the entity recognizer. Second, since a lack of context in single tweets is a leading cause for poor NER performance [11], we concatenate several tweets together before attempting to apply NER (see section 3.1.1).

More formally, for each user $u_i \in U$, we apply Spotlight on all posts $\{p_1, p_2, ..., p_n\} \in P_i$ to obtain a set E of unique DBpedia entities $\{e_1, e_2, ..., e_m\} \in E_i$. For each entity, we keep track of its number of occurrence within P_i. In the simple example that is illustrated in figure 4, we have $E_1 = \{(\texttt{iPad\_Mini}, 1)\}$, $E_2 = \{(\texttt{iPhone}, 1)\}$ and $E_3 = \{(\texttt{HTC\_One}, 1)\}$.

It is important to note that this step is a dimensionality reduction from words W to entities E: while each post $p \in P$ may contain up to 140 characters (in the case of Twitter), with an empirically observed average of around 15 words for English tweets [36], typically only zero to two entities will be discovered in each p. This reduces the sparsity of the problem by roughly an order of magnitude.

3.1.1. *Concatenation window size*

Spotlight leverages word co-occurrence between source text and text from candidate Wikipedia entries for disambiguation, so supplying extra context information can significantly improve the accuracy of the entity recognition. For microblogs such as Twitter, multiple consecutive tweets are often about the same topic (consider conversations between users, or a message that does not fit in 140 characters), resulting in better accuracy for the recognizer if we have more of the same type of content to consider. We need to determine a *concatenation window size* for the number of individual posts $p \in P_i$ to concatenate before sending them to the recognizer. We set up a simple experiment in order to determine what window size values give best results; see section 5.3 for details.

3.1.2. *Spotlight parameter tuning*

DBpedia Spotlight has a number of parameters that can be adjusted for different results. Most importantly, we can set *support* and *confidence* values. Setting these parameters is essentially deciding on a trade-off between precision and recall. With the support parameter we can set a minimum number of in-links that we require a Wikipedia page to have, and with the confidence parameter we can filter out matches that have insufficient similarity between source and target texts. For our approach, we are interested in even the very specialized subjects that may have only a single in-link on Wikipedia, but also want to exclude orphaned pages that may have been abandoned. By this reasoning, we fix the support parameter to a value of 1. It is not clear however which confidence parameter will yield the best results, so we perform experiments to find a good value empirically. See section 5.3 for details.

3.2. *Ontological expansion*

DBpedia incorporates various ontologies and taxonomies, automatically or manually extracted from the underlying Wikipedia data and external sources. For our purposes, there are three ontologies that are useful in particular:

- **DBpedia ontology classes** [1]: these have been derived mainly from the info boxes on Wikipedia. They form a hierarchy of 685 classes.
- **Schema.org classes** [40]: these are originally semantic classes to mark up HTML pages in ways recognized by major search providers. They have been included for DBpedia resources through a mapping from the DBpedia ontology [18], with only minimal differences.
- **YAGO classes** [45]: this ontology is a cleaned version of the Wikipedia category hierarchy, with top-level classes represented by terms from WordNet [33]. It contains roughly 300,000 classes, and is the taxonomy we rely on the most, as it contains both broad and very specific types of classes.[b]

All three of these ontologies form tree-like hierarchies, but none are strict trees – that is, they are directed acyclic graphs, which means some classes may have multiple parents. They are all hierarchical in the sense that concepts get broader the closer one gets to the root, and narrower the closer one gets to the leaves.

Recall that we gathered a collection of entities E_i for each user u_i in the entity recognition step. Now, for each entity $e_j \in E_i$ we collect hierarchical class information. This means that we obtain the DBpedia, Schema.org and YAGO classes directly assigned to this entity, as well as the ancestor class(es) of these classes, all the way up to the root of the hierarchy. We keep count of the number of occurrence of each unique class as we did for entities. We end up with a bag of unique classes $\{c_1, c_2, ..., c_n\} \in C_i$ per user u_i, where each c is a tuple consisting of the class along with its number of occurrence. For our example in figure 2, we get:

$C_1 = \{(\texttt{Apple\_PDAs}, 1), (\texttt{PDAs}, 1), (\texttt{Electronic\_Device}, 1)\}$
$C_2 = \{(\texttt{Apple\_Mobile\_phones}, 1), (\texttt{Cellular\_phones}, 1), (\texttt{Electronic\_Device}, 1)\}$
$C_3 = \{(\texttt{HTC\_Mobile\_phones}, 1), (\texttt{Cellular\_phones}, 1), (\texttt{Electronic\_Device}, 1)\}$

[b] A simplified excerpt of this ontology can be seen on the right-hand side of figure 4, showing the beginning of the YAGO hierarchy for some example entities.

3.2.1. *Including entities as classes*

Wikipedia, and by extension DBpedia, are crowd-sourced data sources, and are therefore noisy: not all entities have classes associated to them, assigned classes may be incorrect, or the classes may not be sufficiently descriptive. To mitigate this, aside from including the three different class hierarchies for extra redundancy, we include the entities themselves as classes as well. This means that we count not only class occurrences, but also entity occurrences. From here on, we will consider $\text{cf}_{i,c}$ to be the *class frequency* map consisting of the union between the entities E_i and classes C_i:

$$\text{cf}_{i,c} = E_i \cup C_i = \text{ the number of times entity } e \text{ or class } c \text{ occurs for user } u_i. \tag{1}$$

Given the ontologies' hierarchical properties, highly specialized classes will tend to get lower cf values, and more general classes will get higher cf values. So while the entities and the classes of the three different ontologies have been combined, flattened and assumed independent from each other, their hierarchical properties remain intact.

3.3. *cf-iuf weighting and trait vectors*

We introduce a weighting strategy for classes that is based on tf-idf, a common strategy used in document processing [44]. On top of the class frequency map $\text{cf}_{i,c}$ described in the previous section, we now additionally calculate a *user frequency* map uf that records for every unique c how many users in U have at least one instance c:

$$\text{uf}_c = \text{ the number of users that have at least one instance of } c. \tag{2}$$

We can see that the higher the uf_c for some c, the more common a class is among the collection of users. Hence, as with traditional inverse document frequency, we take the inverse of this measure, so that more common classes get a lower weight:

$$\text{iuf}_c = \log \frac{|U|}{\text{uf}_c} \tag{3}$$

Combining equations 1 and 3 leads us to our *cf-iuf* weighting strategy for each u_i:

$$\text{cf-iuf}_{i,c} = \text{cf}_{i,c} \times \text{iuf}_c, \text{ where } c \in C_i. \tag{4}$$

We can calculate a cf-iuf weight for each unique class that occurs for each user. We use this weighting strategy as components for a *trait vector* $\mathbf{t}_i$, projecting traits that characterize a user's interests into a class-based vector space model:

$$\mathbf{t}_i = [w_1, w_2, ..., w_{|C_i|}]^T, \text{ where } w_k = \text{cf-iuf}_{i,c_k} \text{ and } c_k \in C_i. \tag{5}$$

In other words, each user has as many traits as he has unique classes, and these traits are weighted according to their overall uniqueness compared to other users. The traits with the highest weights characterize this user best within the full collection of users.

3.4. *Scoped topical similarity*

At this point, we can formally define a Social Web user as follows.

Definition 1 *A Social Web user u_i in a collection of users U is a 3-tuple*

$$u_i = (P, cf, \mathbf{t}), \quad u_i \in U, \tag{6}$$

where P is a collection of (micro-)posts; cf is a class frequency map, and $\mathbf{t}$ is a trait vector expressing the topical affinity of the user to each class.

We leverage these properties to group similar users by their common topics of interest. It is hard to define "topics of interest" concretely, for a number of reasons. First of all, the topic scope can be almost arbitrarily narrow or broad (consider e.g. *Premier league players* → *Football* → *Sports* → *Activity*): there is no "true" scope for the topics, as this depends completely on what we are interested in finding. A second reason is that there is ambiguity as to the delineation of topics, and how many and what kind of concepts can even be considered "topics". This is often called the "user's dilemma" [21]: each observer has different ideas about what constitutes a "topic" and how to draw the right delineation between different topics. We deal with these problems by (1) making the scope of topics sought controllable: different use cases require different topic allocations, so we allow the retrieval of topics at arbitrary levels of the latent topic hierarchy; and (2) inferring mutual topics of interest and their boundaries from the underlying data based on users' *scoped topical similarity*.

3.4.1. *Controlling topic scope*

As explained above, the scope of the topics we seek is application-specific, and there is no "right" or "wrong" scope or number of topics. Exploiting the hierarchical properties of the ontological classes that form the basis of our user trait vectors, we introduce a *topic scope* parameter, γ, into the equation for the cf-iuf weighting strategy. This allows us to forego the definition of a target number of topics for the clustering that is common to most existing topic modeling and document clustering techniques.

$$\text{cf-iuf}_{i,c} = \text{cf}_{i,c}^{1+\gamma} \times \text{iuf}_c^{1-\gamma}, \quad -1 \leq \gamma \leq 1, \quad \gamma \in \mathbf{R} \tag{7}$$

γ can take on any real value between -1 and 1. If we set γ closer to -1, we put more weight on iuf_c and less on $\text{cf}_{i,c}$, leading to a bias towards rare classes (a lower slice of the topic hierarchy), and vice versa. This behavior was depicted in figure 2.

The exact generality of classes that we obtain at a given, absolute value of γ depends on a number of other dependent variables of our approach, which we will introduce in the coming sections. Therefore, there is a need to *calibrate* γ to yield a certain topic scope at a certain value, and optimize the remaining variables around this calibration. We describe this calibration and optimization process in section 5.3.3.

3.4.2. *Inferring mutual interests: scoped topical similarity*

Given each user's trait vector $\mathbf{t}$, we can find which users are similar to each other at scope level γ by calculating the pair-wise cosine similarity between all users in U. We call this the *scoped topical similarity* (STS) function.

$$STS_\gamma(u_i, u_j) = \frac{\mathbf{t}_i \cdot \mathbf{t}_j}{|\mathbf{t}_i||\mathbf{t}_j|} \tag{8}$$

Since the cf-iuf weights for traits were normalized, it follows that $0 \leq STS_\gamma(u_i, u_j) \leq 1$. At any point, we can alter the topic scope γ and re-calculate the trait vectors $\mathbf{t}$ and the STS_γ between all users for different topic distributions. However, this is not enough the determine topic boundaries within a collection of users; the next section will detail how we cluster users into distinct topics.

4. Hierarchical topic clustering

In this section, we describe how we use graph-based techniques to perform the clustering of users into topics. The key characteristic of this type of approach is that we do not need to pre-define the number of clusters. First, we explain how we represent users and their similarity as a weighted graph (section 4.1), followed by a detailed description of the methods we developed for scoped clustering (4.2), topic labeling (4.3) and recursive clustering to construct topic hierarchies (4.4).

4.1. *Scoped topical similarity graph*

From our collection of users $u_i = (P, \mathrm{cf}, \mathbf{t}),\ u_i \in U$, we can construct a *scoped topical similarity graph* $G_\gamma = (V, E)$. G_γ is an undirected weighted graph where each vertex is a user and each edge connects two users, with their STS_γ as edge weight. Initially, we calculate edge weights between every pair of users to obtain a fully connected graph.

Such a graph is not an ideal form if we want to find topic clusters; it is hard to find clusters within a fully connected graph, and the number of edges $|E|$ will be exponential in $|V|$, leading to intractable computational complexity when dealing with large graphs. We solve this by first pruning the edges E, imposing a minimum STS_γ threshold τ. In other words, if two users are less than τ similar, we consider them too different to have any kind of interest in common. It is not immediately clear which value of τ would yield the best results. We include τ as one of three dependent variables that we optimize through hyperparameter optimization; see section 5.3.3 for details.

When visualized, the topological graph structure gives an intuitive idea of where clusters are located, but is not sufficient to say anything explicit about the clusters and their content. We can apply a clustering algorithm to isolate highly-connected clusters into distinct topics. As mentioned, we can expect to find a high or low number of clusters if the graph G_γ has a low or high value for γ, respectively. How many topics we end up with depends entirely on the underlying data.

4.2. *Finding highly-connected subgraphs*

Given the pruned STS_γ graph G_γ, we apply a custom version of the Highly Connected Subgraph (HCS) clique-based, strict partitioning clustering algorithm [17]. Originally, this algorithm works by dividing a graph into subgraphs that fulfill some minimum edge degree condition. First, a minimum cut is determined, along which the graph is cut. The resulting subgraphs are examined to see if they are *highly connected*; each vertex in the subgraph must have an edge degree that is at least greater than half the number of vertices in the subgraph. If a subgraph is not highly connected, the algorithm is repeated on this subgraph, assuming there are at least 2 vertices left. HCS allows outliers: singleton and twin vertices are not considered clusters, and therefore discarded. The final result is a clustered graph containing however many highly connected subgraphs were found.

For our purposes, we modify this algorithm in three ways. Firstly, the original algorithm was developed for unweighted graphs. Since we deal with a weighted graph, we apply a minimum cut[c]

[c] Strictly speaking, we seek the *maximum* cut, since a higher weight means a higher similarity between users, but for simplicity's

algorithm that takes weights into account. Secondly, the original full-clique constraint for determining highly-connectedness of a cluster is too strict for our purposes; we relax these constraints to obtain a *quasi-clique* algorithm. Lastly, we do not want to drop outliers from the result, thus also apply a *singleton adoption* heuristic as described in [17]. We explain these three modifications in brief.

Weighted minimum cut Since we have a weighted graph, G_γ, we use a modified version of Kruskal's minimum spanning tree algorithm [24]. First, we sort all edges $E \in G_\gamma$ by STS_γ in descending order. Then we start constructing the minimum spanning tree as per Kruskal's algorithm using the sorted edge list, but stop at the second-to-last step, at which point we have two trees that contain all of the vertices. The last step would connect the two trees to form the minimum spanning tree; since we constructed the trees in descending order of edge weight, it follows that the cut between the two trees represents the minimum cut. A proof of this can be found in [9].

Highly-connectedness constraint We rewrite the clique constraint as follows, parameterizing the numerator for the proportion of vertices that must be higher than the minimum vertex degree of a subgraph as α:

$$\min\{degrees(v_j) \;\mid\; 1 \leq j \leq |V_{sub}|,\; v_j \in V_{sub}\} \;>\; \frac{|V_{sub}|}{\alpha} \tag{9}$$

In the original algorithm, $\alpha = 2$; we experiment with different values for α. A larger α means a less strict lower bound requirement for a highly connected graph, hence we find larger subgraphs. We regard such highly connected subgraphs as *topic clusters*.

Singleton adoption Dropping lowly connected or isolated nodes is not desirable for our purposes; we want to allocate users to at least one topic cluster, even if their affinity to the topics this cluster represents is low. We apply a singleton adoption heuristic where for all remaining, unassigned users after one iteration of the algorithm, we recalculate the STS_γ between each unassigned user trait vector $\mathbf{t}_i$ and each cluster-based trait vector $\mathbf{t}_k$. The latter has been calculated from the union of all traits of all users assigned to that cluster, compared to those of other clusters. The user is assigned to the cluster with highest similarity. The next section explains this in more detail.

4.3. *Topic clusters and labeling*

An advantage of using ontology classes to model topics is that it allows us to reason about the semantic content of topic clusters. One important result is that we can obtain intuitive labels for the clusters by regarding them as single entities, and re-calculating STS_γ; this time, we calculate the similarity between *clusters* rather than *users*, and take class names of the top scoring cluster traits as labels. We formally define *topic clusters* as follows.

Definition 2 *A topic cluster T_k in a clustered graph G'_γ with K clusters is a 3-tuple*

$$T_k = (U_k, cf_k, \mathbf{t}_k), \quad T_k \subseteq G'_\gamma,\; 1 \leq k \leq K,\; U_k \subseteq U, \tag{10}$$

sake we maintain the common terminology.

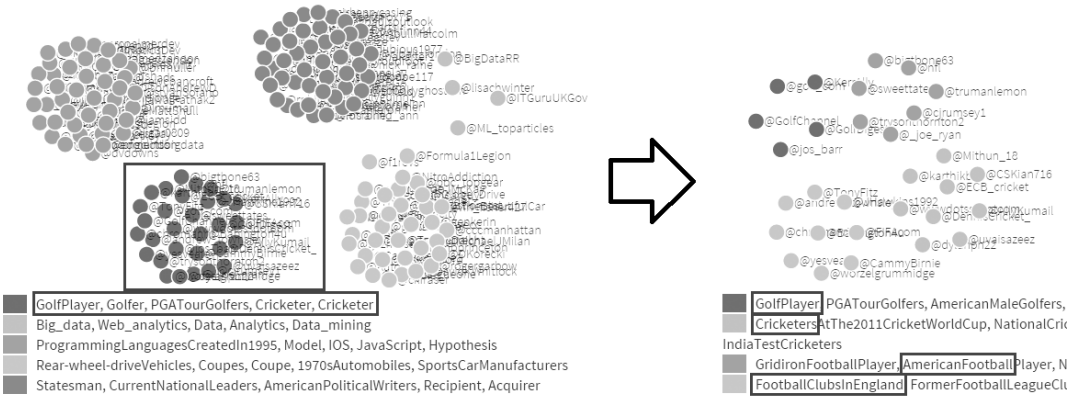

Fig. 5. An example of cluster labeling and STS_h. For each cluster k, the top-5 cluster-based traits from $\mathbf{t}_k$ are listed. A "Sports" cluster is isolated and further sub-divided into the individual sports that made up the cluster.

where U_k is the subset of users assigned to the k-th topic cluster; cf_k is the combined class frequency map for the cluster taken from U_k; and $\mathbf{t}_k$ is a trait vector that expresses cluster-based topical affinity.

In other words, we merge the class frequency maps for every user $u_i \in U_k$ in a cluster T_k to form a cluster-based class frequency map cf_k. From all cf_k, we calculate a *cluster frequency map* clf which records for each class in how many clusters this class occurs at least once (analogous to the user frequency map uf). We can then take the same steps as before to calculate cluster-based, cf-iclf-weighted trait vectors $\mathbf{t}_k$. Sorting these trait vectors by their topical affinity component in descending order provides us with the most characteristic traits for each cluster at the top of the list; we use the names of the top traits as topic cluster labels. See the left side of figure 5, for example, which shows the result of clustering a testset of 175 Twitter users into $K = 5$ dominant topics ($\gamma = 0.8$). For each cluster k, the top-5 cluster-based traits from $\mathbf{t}_k$ are listed.

4.4. *Hierarchical STS*

While we consider altering the topic scope γ and re-calculating trait vectors a form of hierarchical clustering (since we can obtain topics at an arbitrary height in the latent topic hierarchy – something we cannot with traditional approaches), our approach also allows divisive clustering to generate a full topic hierarchy in a top-down fashion. After one clustering iteration, we can apply the clustering once again on each resulting cluster separately. That is, we ignore all users that do not belong to the selected cluster, and re-evaluate the trait vectors of the remaining users. Since dominant traits of each user are now calculated only with regard to the other users originally in the same cluster, we can further divide users into more specific topics of interest within this more general topic of interest. We call this type of recursive topic clustering *hierarchical STS*, or STS_h, and it allows us to extract a full topic hierarchy, rather than just a slice, from a collection of users. An example of one iteration of recursive clustering is shown in figure 5, where we have a cluster of users that is about "Sports", as is evident from the topic labels. We can sub-divide this topic cluster into sub-clusters lower in the topic hierarchy. We find the cluster consists of "Golf", "Cricket", "American football" and "Football". A simple pseudo-code representation of the recursion we perform is shown in algorithm 1.

Algorithm 1. STS_h: Recursive topic clustering to obtain a full topic hierarchy.

```
1: procedure CLUSTERGRAPH(G(V,E)):
2:     V ← calculateTraitVectors(V, γ)
3:     for e in E do
4:         e.weight ← STS(V[e.i], V[e.j])
5:     subGraphs[] ← HCS(G)
6:     for subGraph in subGraphs do
7:         clusterGraph(subGraph)
```

5. Evaluation

In this section we describe the evaluation of our approach. First, we describe our target users and ground truths that we have prepared, in section 5.1. This is followed by an explanation of the evaluation metrics used and baselines we compare our approach to, in 5.2. In 5.3, we describe how we derive parameter settings and calibrate the topic scope by tuning the hyperparameter configuration of our approach. Sections 5.4 and 5.5 contain the experimental results of the STS_γ calculation, scoped clustering and hierarchical clustering. We end this section with a discussion of the results in 5.6.

5.1. *Ground truths*

There are two main aspects of our approach that we must evaluate: (1) the quality of the topical similarity calculation within a noisy Social Web context; and (2) the quality of the topic clustering, i.e. whether discovered topics are appropriate (content, number and size) for the selected scope or full hierarchy, and whether the topic labels are correct. We perform two different experiments, and therefore have two separate types of test sets: for (1), relevance maps, expressing relevance of a number of semi-random users to a known target user; and for (2), ground truths of users divided into topics and subtopics that we identified beforehand for Twitter, as well as an existing dataset of categorized newsgroup posts. For Twitter, we want human users that tweet actively and have enough content that we can process, so we only choose those users that tweet in English, have at least 500 tweets, and, excluding seed users, have between 100 to 10,000 followers and friends.

5.1.1. *User relevance maps*

We prepare two sets of 100 Twitter users each. Each set has one *seed user* with a clearly identifiable interest that was manually determined by the authors by checking their Twitter profiles. The remaining 99 users are collected semi-randomly from their immediate follower neighborhood. We gather direct followers, followers of those followers, and so on, in a breadth-first fashion until we have enough users that satisfy our constraints. We pick one seed user that tweets consistently about **iOS development** (@iOS_blog), and one seed user that tweets consistently about **cars** (@CCCManhattan). For the remaining 99 users of both sets, we manually determine relevance of the user's timeline to the seed user, assigning a score of 0, 1 or 2, as follows:

- **0:** This user is irrelevant; the user does not have tweets related to the topic
- **1:** This user is somewhere in between; in this category we include users that post some tweets related to the topic, but also a considerable number of unrelated tweets

Table 1. The 20 newsgroups dataset (20 topics over 6 subject matters).

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

2: This user is very relevant; most of the user's tweets are directly related to the topic

5.1.2. *Topic clustering ground truths*

We prepare a Social Web-based ground truth consisting of 175 Twitter users, containing 4 main topics that are further subdivided into 11 subtopics. Each user belongs to one main topic and one subtopic. We expect our method to be able to discover both clusterings, depending on the setting of the topic scope parameter γ. The topics are distributed as follows:

- **Computer science** (50 users): *iOS development* (20 users), *Web development* (15 users), *Data Science* (15 users)
- **Sports** (40 users): *Soccer* (10 users), *Football (American)* (10 users), *Golf* (10 users), *Cricket* (10 users)
- **Cars** (40 users)
- **Politics** (45 users): *US politics* (15 users), *UK politics* (15 users), *Australian politics* (15 users)

Note that for topic cars, we could not distinguish any clear further subtopics, therefore we expect the same result for main and subtopic. For each user, the authors made sure that the users were primarily interested in the subtopics defined above by manually checking their recent tweets. To make this process less laborious, some users were collected from follower neighborhoods of well-known seed users for each subtopic (e.g. *@BarackObama* for US politics, *@BBCTopGear* for cars, etc.). Others were found through the keyword search function on Twitter. The result is a well-defined ground truth that forms a good basis for comparison. The clusters have different sizes, which was a deliberate choice; it allows us to evaluate performance when dealing with asymmetric cluster sizes.

We prepare a second ground truth based on the 20 newsgroups dataset by Ken Lang [26], which is a commonly used dataset in the field of document clustering [38]. This dataset consists of over 20,000 posts across 20 newsgroups. These 20 newsgroups can be roughly divided into 6 main subject matters (see table 1). Due to performance limitations, we take a subset of 1800 randomly selected posts, evenly divided over the 20 newsgroups (90 posts per group). This dataset differs from Twitter data in that the text is generally well-structured and grammatically correct. However, posts can be short: in the extreme case, a post may consist of only a single line. Our approach requires an appropriate amount of context information (a portion of a user's timeline) for ontological expansion, which may not be available here, so we expect this to impact performance of our method. For a more even comparison given this limitation to our approach, we also prepare a second subset of all posts larger than 10.0 kilobytes in size (there are 239 such posts).

5.2. *Experimental methodology*

In this section, we describe our experimental methodology for this evaluation. In 5.2.1, we explain the metrics we use, and in 5.2.2 the baselines we compare to.

5.2.1. *Evaluation metrics*

To evaluate topical similarity given the relevance maps defined in section 5.1.1, we calculate the Discounted Cumulative Gain (DCG) according to the formula

$$\mathrm{DCG_k} = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2(i)}, \tag{11}$$

where k corresponds the top-k ranked users based on unscoped topical similarity STS_0, and rel_i is the relevance score assigned to the user at rank i. Since we want to evaluate for different top-k, we will obtain different $\mathrm{DCG_k}$ scores for different k, therefore we need to use the normalized version of the DCG, nDCG:

$$\mathrm{nDCG_k} = \frac{DCG_k}{IDCG_k}. \tag{12}$$

Here IDCG refers to the *Ideal* DCG, i.e. the maximum possible DCG until position k:

$$\mathrm{IDCG_k} = 2 + \sum_{i=2}^{k} \frac{2}{\log_2(i)}. \tag{13}$$

For topic clustering, since our method is already based on similarity, any cluster similarity-based internal evaluation would give biased results. Therefore, we apply an external clustering evaluation by representing our ground truths and cluster graphs as truth tables containing *true positives* (TP), *false positives* (FP), *true negatives* (TN), and *false negatives* (FN). Let $T_k(u_i)$ be the k-th topic cluster containing some user u_i. Then we can define the contingency table as in table 2. Using this table, we can calculate the *precision*, *recall* and F_1-*score*, as well as the Matthew's Correlation Coefficient (*MCC*) [29] for a topic clustering result over a collection of users U as follows.

$$Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN}, \quad F_1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{14}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{15}$$

Note that precision, recall and F_1 do not take into account true negatives. This makes it susceptible to bias, since the TN:FN ratio gets increasingly skewed towards more TNs the more clusters we have. The *MCC* accounts for this bias by giving FPs and FNs equal weight to TPs and TNs, taking on a value between -1 and 1, where a value of 0 means the result is no better than random; towards -1 increasingly worse than random; and towards 1 increasingly better than random (with 1 being a perfect score).

Additionally, we evaluate according to an entropy-based measure: *normalized mutual information* (*NMI*). This is a measure of the difference between the number and sizes of the ground truth clusters with the resulting clusters, without regard for their content:

Table 2. Determining true/false positives/negatives between the clustering result and the ground truth.

		Ground truth	
		$T_k(u_i) = T_k(u_j)$	$T_k(u_i) \neq T_k(u_j)$
Clustering result G'_γ	$T_k(u_i) = T_k(u_j)$	TP	FP
	$T_k(u_i) \neq T_k(u_j)$	FN	TN

$$NMI(X;Y) = \frac{\sum_{y\in Y}\sum_{x\in X} p(x,y)\,\log\left(\frac{p(x,y)}{p_1(x)p_2(y)}\right)}{\sqrt{\Big(-\sum_{x\in X} p(x)\,\log\,p(x)\Big)\Big(-\sum_{y\in Y} p(y)\,\log\,p(y)\Big)}} \tag{16}$$

Here, the X and Y variables are substituted with representations of the ground truth and clustering result, respectively. For details, refer to [46].

In order to evaluate full hierarchical clustering, we require some way to evaluate each level of the hierarchy as a whole rather than separately in order to get an accurate assessment of algorithm quality. We borrow the *hierarchical F-score* as introduced in [23], and further recommended as a generally usable measure to evaluate hierarchical classification algorithms in the survey presented in [42].

$$hP = \frac{\sum_i \mid \hat{P}_i \cap \hat{T}_i \mid}{\sum_i \mid \hat{P}_i \mid}, \quad hR = \frac{\sum_i \mid \hat{P}_i \cap \hat{T}_i \mid}{\sum_i \mid \hat{T}_i \mid}, \quad hF = \frac{2 \cdot hP \cdot hR}{hP + hR} \tag{17}$$

Here $\hat{P}_i$ is the set consisting of the most specific topic(s) predicted for user i and all their ancestor topics and $\hat{T}_i$ is the set consisting of the *true* most specific topic(s) of user i and their ancestor topics. Although designed for the evaluation of document classification (matching to an existing class hierarchy) rather than clustering (creating an entirely new class hierarchy), the measure is flexible enough to be adapted for clustering by regarding each topic in our two-level ground truths as the target classes we expect to see when clustering data hierarchically. This means that $\hat{T}_i$ will always have size 2. If a result cluster consists of more than one of the ground truth topics, we consider this a new class not in $\hat{T}_i$; we continue clustering hierarchically until cluster topics closely represent the topics defined in the lower layer of the ground truth in the result. To judge this, we look at the top terms identified per topic for hLDA, and the top labels generated for STS_h.

5.2.2. *Baseline approaches*

We compare our STS_γ similarity calculation to two baselines: (1) traditional tf-idf-based cosine similarity, and (2) a similarity calculation based on the simple taxonomical similarity of classes between users [43]. For (1), we take all words in all tweets per user (after basic filtering), calculate the document vectors with tf-idf component weights, and calculate the cosine similarity-based ranking compared to the seed users. For (2), taxonomical similarity s_{tax}, we apply the following formula between the seed user u_{seed} and each subject user u_i:

$$s_{tax} = \frac{\sum_{c_j \in \mathrm{cf}_{seed} \cap \mathrm{cf}_i} \min\{\mathrm{cf}_{seed}(c_j),\ \mathrm{cf}_i(c_j)\}}{\sum_{c_j \in \mathrm{cf}_{seed}} \mathrm{cf}_{seed}(c_j)}. \tag{18}$$

That is, we sum the class frequencies for all classes that u_{seed} and u_i have in common, and then divide that by the sum of all class frequencies of all classes in u_{seed}.

We evaluate four versions of our STS_γ-based topic clustering, each using different types of ontology information: Wikipedia resources only ($STS_{\gamma,res}$), DBpedia/Schema.org classes only ($STS_{\gamma,DBpedia}$), YAGO types only ($STS_{\gamma,YAGO}$) and all types combined (STS_γ). We compare all versions to four baselines:

1. *Random clustering*: we take the average of 100 random clusterings of the data.

2. *Latent Dirichlet allocation (LDA)*: we apply LDA on text content (after basic filtering). For Twitter, each user profile is considered one document. Since LDA requires us set a number of topics, we calculate the results for 2 to 24 topics. We cluster users into the topic to which LDA assigns them the highest probability. We use the state-of-the-art LDA implementation that is part of the MALLET package [30], with 800 iterations but otherwise default settings. We use 800 iterations since this finishes in roughly the same calculation time for 175 Twitter users as our implementation of STS_γ-clustering on our environment (around 67 seconds).

3. *Twitter-LDA* [50]: we apply the Twitter-optimized version of LDA as implemented and described in [37] for an evaluation compared to the state-of-the-art in Twitter topic modeling. We apply the same hyperparameter settings as regular LDA, and maintain 800 iterations.

4. *k-means clustering*: we cluster the STS_γ graph G_γ using standard k-means [16]. As with LDA, we calculate the results for the number of topics K ranging from 2 to 24 topics. For each K, we iterate 10 times and take the result for which the within-cluster sum of squares (WCSS) is maximum (since we maximize on cosine similarity):

$$\arg\max_{\mathbf{T}} \sum_{k=1}^{K} \sum_{\mathbf{t}\in T_k} \|\mathbf{t}-\mu_k\|^2 \tag{19}$$

   Here, $\mathbf{T}$ is the set of K topic clusters, and μ_k is the centroid vector of all trait vectors $\mathbf{t}_i$ of users $u_i \in U_k$ that belong to topic cluster $T_k \in \mathbf{T}$.

Lastly, we evaluate hierarchical clustering, where we apply our recursive topic clustering strategy (algorithm 1 described in section 4.4) to generate a full topic hierarchy. We compare STS_h against two baselines:

1. *Hierarchical latent Dirichlet allocation (hLDA)*: similar to the per-layer evaluation, we use the state-of-the-art hLDA implementation that is part of the MALLET package [30] to infer a topic hierarchy, with 800 iterations (we are not aware of any Twitter-specific hierarchical LDA algorithm). hLDA requires a hierarchy depth to be pre-determined; we set this to 3 (the first level of hLDA consists of the full data, so 3 levels matches our ground truth).

2. *Hierarchical k-means*: we cluster the data hierarchically using the k-means algorithm in a recursive fashion, with the expected number of topics at each level as value for K (for the Twitter dataset, $K = 4$ for the first level, and $K = 3$ or $K = 4$ depending on the number of subtopics expected). Since k-means works stochastically, we iterate 10 times and average the result.

5.3. *Parameter selection*

There are a large number of parameters to set for our approach, and it is unclear which settings will yield the best results. We want to derive optimal parameter settings that will provide good results on unseen data with high likelihood. To keep dimensionality of this optimization problem in check, we decide to tune the parameters for named entity recognition and for clustering separately.

5.3.1. *Named entity recognition parameter tuning*

For NER, key parameters we need to decide on are (1) the confidence value of the entity recognizer (DBpedia Spotlight); and (2) the size of the windows of concatenated tweets. We determine a good confidence value using a simple grid search optimization. We calculate the top-k nDCG rankings on both relevance map-based datasets in terms of their seed users, for $k \in \{3,5,10,25,50\}$ and for a range of values between 0 and 1 for the confidence parameter. For this initial experiment, we concatenate 10 tweets at a time before applying NER. The results are shown in figure 6.

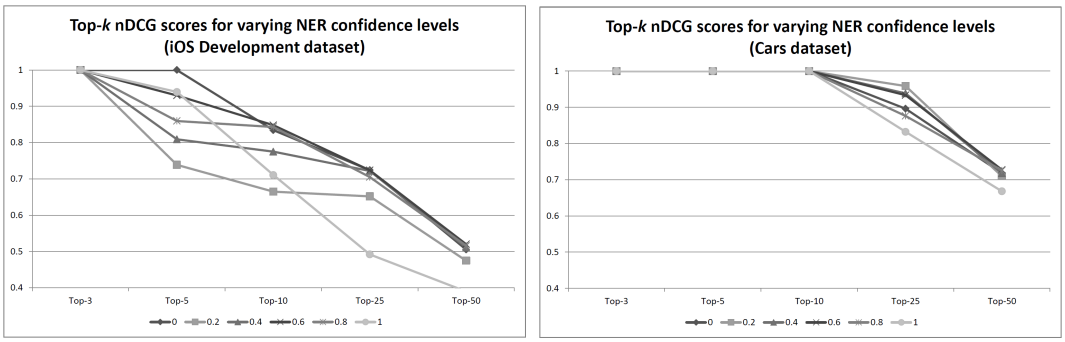

Fig. 6. nDCG scores for different NER confidence settings for the iOS development and cars datasets.

We see that the results are relatively close to each other. For top-50, which should be least affected by random noise, all settings perform roughly equal with the exception of the highest setting of 1. Given these results, we decide to keep the confidence value at 0 – aside from giving the best results on average (by a small margin), in situations where we have little content available the higher settings may no longer yield enough classes to determine topic clusters accurately.

Additionally, we must decide on window sizes for concatenating tweets (recall that the entity recognizer works by leveraging the surrounding context of source terms). We again experiment on both datasets, this time varying tweet window sizes between 3, 5, 10, 25 or 50 tweets. Results are shown in figure 7. Again, we see only minor differences in the results. We set the tweet window to 10 tweets, which performed best at larger sample sizes.

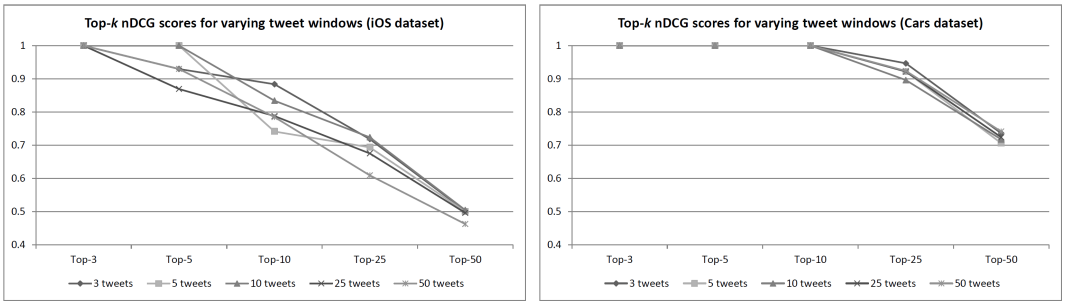

Fig. 7. nDCG scores for different tweet window sizes for the iOS development and cars datasets.

5.3.2. *Class frequency distribution and trait cut-off threshold* θ

The combined distribution of class frequencies over our ground truth of 175 users with 500 tweets each is plotted in figure 8. As is evident, class occurrences are skewed at the tail-end of the distribution, with almost 10% of all unique classes occurring 10 or less times. Classes with small occurrence numbers lead to negligible contributions to the cosine similarity, therefore consider it safe to prune the trait vectors up to a certain *trait cut-off threshold* θ to reduce the sparsity and dimensionality of data. Which value of θ would be best is not immediately clear. Along with the previously introduced τ and α, this variable is the last of three dependent variables that we set by hyperparameter optimization.

5.3.3. *Hyperparameter optimization and scope calibration*

We apply a hyperparameter optimization approach to calibrate θ, τ and α around a neutral topic scope $\gamma = 0$. This is necessary since all four variables are dependent on one another. Since our dataset is not that large, and we have only three dimensions, it is feasible to apply a grid search on the parameters – that is, check each combination exhaustively within a reasonable discrete range of possibilities. We can make surface plots to visually determine appropriate values. We test the following parameter ranges:

- *Trait cut-off threshold*: $\theta = \{0.0, 0.001, ..., 0.01\}$
- *Similarity threshold*: $\tau = \{0.0, 0.01, ..., 0.1\}$
- *Highly-connectedness parameter*: $\alpha = \{2, 3, 4, 5\}$

Next, we need a data set and some fitness function to optimize on. We choose to use our cluster ground truth of 175 users, testing the MCC score of the resulting cluster graphs compared to the 11 subtopics we defined in the ground truth. This means that our method gets calibrated to the extent that $\gamma = 0$ will detect topic clusters roughly corresponding to the scope expressed by these 11 subtopics. This choice is made for both intuitive and practical reasons. Intuitively, the subtopics represent a mid-level position in the hierarchy: given a topic such as Football, it is easy to come up with topics that are more general ("Sports" $\rightarrow$ "Activity") and more specific ("Premier League clubs" $\rightarrow$ "Arsenal F.C."). Practically, we need a concrete ground truth to perform the calibration, and our 11-topic testset is the most comprehensive we have available.

To avoid overfitting, we must incorporate some form of regularization, such as cross-validation. However, since our approach does not involve a training phase, we cannot split the data into a training set and test set as with traditional cross-validation. Therefore, we regularize using random-fold cross-validation in the following way:

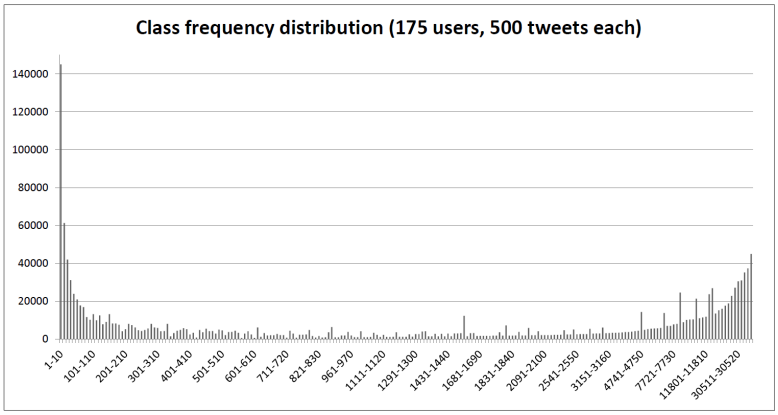

Fig. 8. The combined class frequency distribution for 175 Twitter users with 500 tweets each. The x-axis denotes buckets of size 10, and the y-axis the number of classes with a number of occurrence that falls within one of the buckets. There were 1,535,108 classes found in total.

1. For each parameter combination, do the following 50 times:
   (a) Take 7 out of 11 sub-topics from the ground truth uniformly at random
   (b) For each sub-topic, take between 50-100% of the users assigned to that topic uniformly at random
   (c) Apply the approach from start to end to these users and calculate the MCC of the resulting cluster graph G'_0 in comparison to the reduced, 7-topic ground truth
2. Take the average of the 50 iterations as the final MCC value

It is now important to pick a combination of values that has a high probability of yielding a good result when dealing with new, unseen data. In other words, we must take care to choose values so that the area around combinations of θ, τ and α is of consistently good quality. We find that the values that best fit this condition are $\theta = 0.003$, $\tau = 0.07$, and $\alpha = 4$. For illustration, in figure 9 we have plotted the resulting surface for all combinations of θ and τ given $\alpha = 4$, showing a consistently high MCC score around our selected parameters. These will be the values we will use in the coming experiments where we compare our approach to baselines and established approaches.

5.3.4. *Standard deviation and standard error of folds*

Regularizing the parameter selection only works if the cross-validation is adequate given the size of the testset. Since our testset of 175 users is rather small, it is not immediately clear if our random-fold cross-validation is reliable. By calculating the average standard deviation and margin of error of the sampling, we can obtain a statistical measure of the reliability of the hyperparameter optimization.

We calculated the standard deviation σ of 50 iterations for all parameter combinations as described above. There are $11 \times 11 \times 4 = 484$ such parameter combinations. The resulting distribution function of standard deviations is plotted in figure 10.

The standard error from these standard deviations can be calculated using 95% confidence interval for the average standard deviation, taking the square root of the mean of variances σ^2:

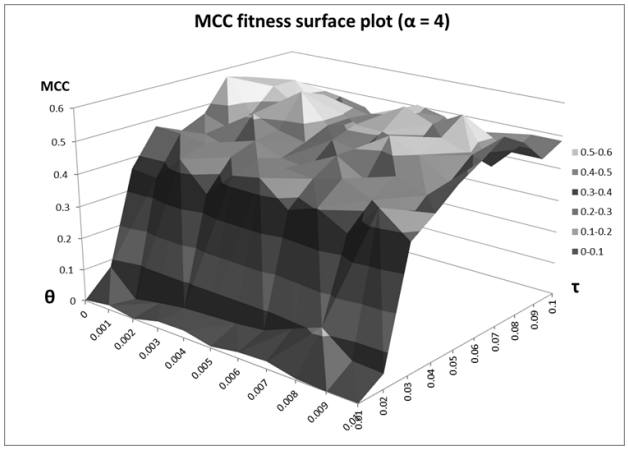

Fig. 9. Example MCC surface plot for varying values of θ and τ and $\alpha = 4$. Each MCC data point is the average over 20 iterations with semi-random cluster selections.

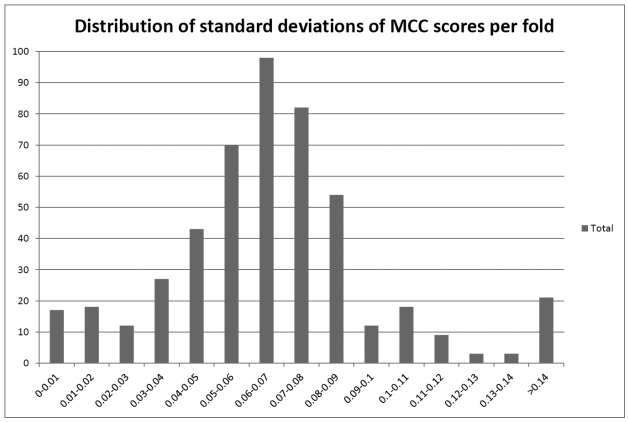

Fig. 10. Distribution of standard deviations for each fold of the cross-validation. To save space, a long tail of outliers on the right side of the graph have been aggregated as > 0.14.

$$\overline{\sigma} = \sqrt{\overline{\sigma^2}} = \sqrt{0.0067} \approx 0.08 \tag{20}$$

It follows that the 95% confidence interval is $[-0.08 \cdot 1.96, +0.08 \cdot 1.96]$. This result means that we can say with 95% confidence that the resulting mean MCC score for a given parameter combination lies within approximately -0.157 and 0.157 of the true mean MCC score. Since MCC ranges from -1 to 1, this is a standard error of around 7.8%. We judge this to be a reasonable enough value to conclude that the hyperparameter optimization is adequate.

5.4. *Experimental results: topical similarity*

We now evaluate the topical similarity portion of our approach in terms of two nDCG rankings to seed users with known interests (*iOS development* and *cars*), as explained in section 5.1. These evaluations will be similar to the entity recognition confidence and tweet window experiments, although this time we compare the quality of our STS_γ-based rankings to the two baseline approaches: (1) traditional

tf-idf-based cosine similarity, and (2) taxonomical similarity of classes between users [43].

We compare the top-k nDCG rankings, for $k \in \{3,5,10,15,20,25\}$. Since the baseline methods do not incorporate topic scope, we keep γ fixed to 0 for a fair comparison. The results are plotted in figure 11. A consistent improvement of roughly 15% on average over the second-best baseline approaches (taxonomical for iOS development, tf-idf for cars) can be observed. We leave further interpretation and discussion of these results for the discussion section (5.6).

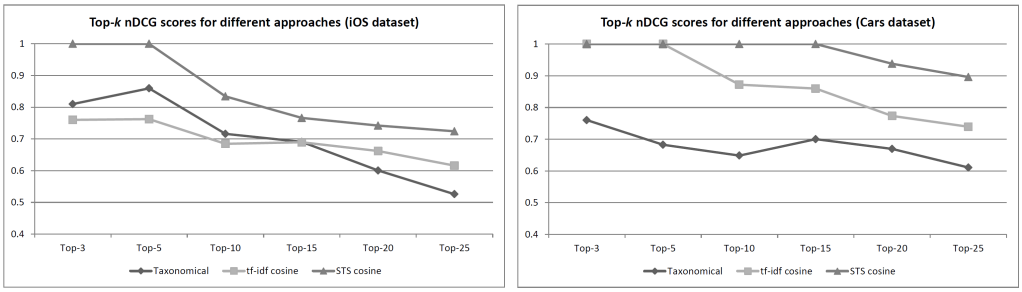

Fig. 11. nDCG scores for different similarity calculation approaches for the iOS development and cars datasets.

5.5. *Experimental results: scoped topic clustering and hierarchical clustering*

In this section we detail the results for scoped and hierarchical topic clustering. First, we show the scoped clustering results we obtained for the Twitter dataset, followed by the results for the newsgroups datasets. We conclude with an evaluation of hierarchical clustering.

5.5.1. *Twitter dataset*

We apply STS_γ-clustering for the Twitter ground truth, distinguishing between 4 topics and 11 subtopics that we aim to cluster the same set of 175 users in. The goal is to be able to cluster into the right 4 main topics *or* the right 11 sub-topics depending on our desired topic scope setting. The $P/R/F_1$, NMI and MCC scores are calculated for a range of topic scope parameter γ values between -1 and 1, in increments of 0.1. The results are compared to random clustering, LDA and Twitter-LDA-based clustering, and k-means clustering.

For STS_γ, for both the 4-topic and 11-topic testsets, a progression of the F-score, NMI and MCC metrics (lines) and the number of topics they yielded (columns) for different topic scopes (x-axis) is plotted in figure 12. The same metrics for LDA, Twitter-LDA and k-means, trying topic selections from 2 to 24, are plotted in figures 13, 14 and 15, respectively. Table 3 shows a summary of the best values with corresponding settings that were obtained for each method. This summary includes the best results for all 4 versions of our method: resources-only ($STS_{\gamma,res}$), DBpedia/Schema.org classes only ($STS_{\gamma,DBpedia}$), YAGO types only ($STS_{\gamma,YAGO}$) and all types combined (STS_γ. Tables 4 and 5 list topic labels discovered for the best results: terms with highest probability per topic for Twitter-LDA, and top class labels calculated as described in section 4.3 for STS_γ-clustering. k-means clustering does not provide topic labels.

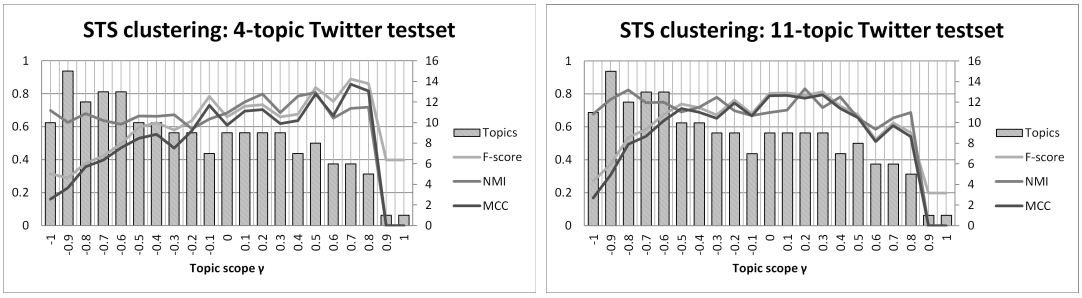

Fig. 12. Results for STS_γ-clustering on the Twitter ground truth.

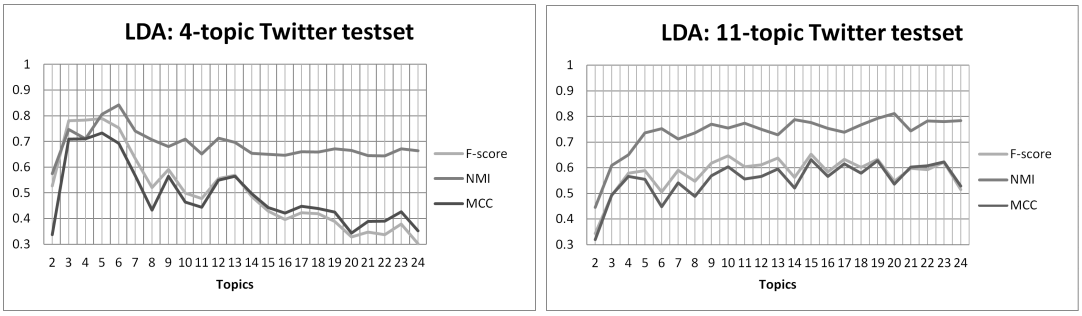

Fig. 13. Results for LDA-based clustering on the Twitter ground truth.

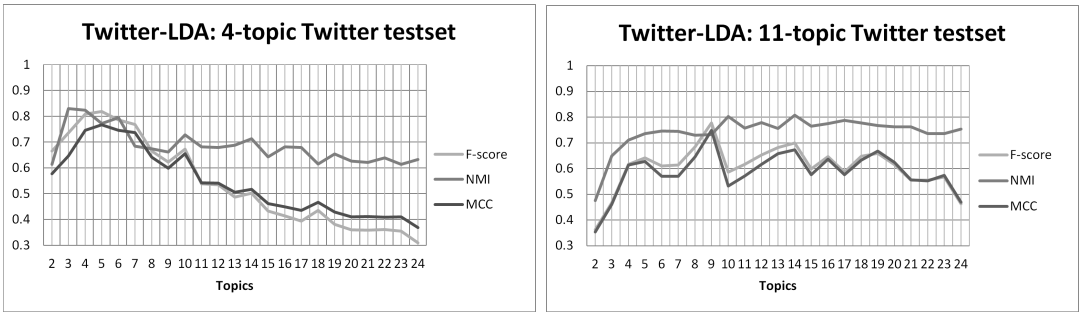

Fig. 14. Results for Twitter-LDA-based clustering on the Twitter ground truth.

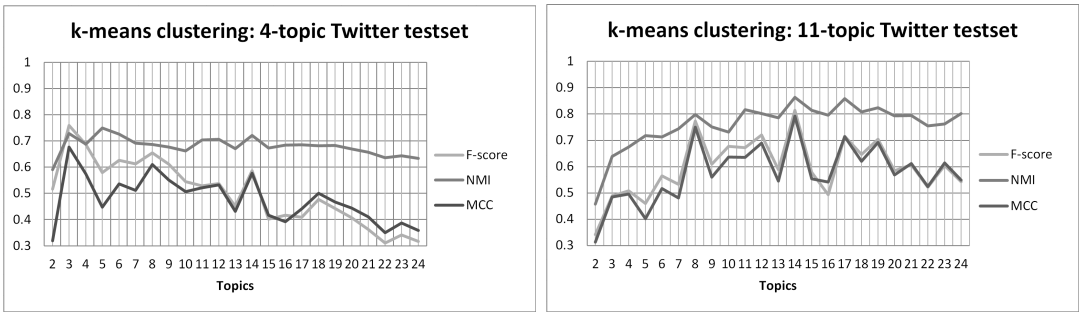

Fig. 15. Results for *k*-means clustering on the Twitter ground truth.

Table 3. Summary of optimal results for the random clustering baseline, LDA, Twitter-LDA, *k*-means and our STS_γ-clustering method with different ontology class selections.

175 Twitter users, 4-topic testset

	Precision	*Recall*	*F-score*	*NMI*	*MCC*	*Topics*
Random	0.248	0.250	0.249	0.885	-0.001	4
LDA	0.870	0.722	0.789	0.806	0.733	5
Twitter-LDA	0.877	0.767	0.819	0.772	0.767	5
k-means	0.663	0.887	0.759	0.729	0.677	3
$STS_{0.7}$	0.945	0.840	0.889	0.711	**0.858**	6
$STS_{0.3,res}$	0.972	0.633	0.767	0.787	0.736	9
$STS_{0.8,DBpedia}$	0.926	0.824	0.872	0.781	0.836	7
$STS_{0.6,YAGO}$	0.959	0.807	0.876	0.611	0.846	7

175 Twitter users, 11-topic testset

	Precision	*Recall*	*F-score*	*NMI*	*MCC*	*Topics*
Random	0.110	0.092	0.100	0.769	0.001	11
LDA	0.799	0.552	0.653	0.777	0.632	15
Twitter-LDA	0.736	0.822	0.777	0.732	0.749	9
k-means	0.819	0.810	0.814	0.864	*0.791*	14
STS_0	0.700	0.947	0.805	0.686	*0.788*	9
$STS_{0.35,res}$	0.791	0.794	0.793	0.768	0.767	6
$STS_{-0.1,DBpedia}$	0.484	0.938	0.639	0.677	0.621	8
$STS_{0.35,YAGO}$	0.655	0.927	0.768	0.735	0.748	9

Looking at the results in figure 12, we observe an expected progression in terms of the number of topics detected: the higher the topic scope, the less topic clusters we find. For the 4-topic testset, the best result in terms of the F-score and MCC is a scope of $\gamma = 0.7$ (6 topics found), and for the 11-topic set a scope of $\gamma = 0$ (9 topics found). For LDA, Twitter-LDA and *k*-means in figures 13, 14 and 15, the progression looks similar (although reversed). For LDA we get the best results for 5 topics and 15 topics respectively; for Twitter-LDA, for 5 and 9 topics; and for *k*-means, for 3 and 14 topics.

Regarding the different versions of STS_γ used, we see a somewhat expected result: using only DBpedia classes (of which there are few and they are generic) we do well at discovering the generic topics but does poor at specific topics; using only resources we see the opposite result. Using only YAGO classes and using all classes combined do well at both parts, with STS_γ outperforming $STS_{\gamma,YAGO}$ by a small margin.

Overall, our best-performing method, STS_γ, outperforms standard LDA by 17% for 4 topics and by 26% for 11 topics, and Twitter-LDA by 11.9% for 4 topics and by 5.2% for 11 topics. STS_γ outperforms *k*-means by 26.7% for the 4-topic testset, but performs roughly equal for the 11-topic testset. This conspicuous result can be explained by the fact that *k*-means does not take any topic hierarchy into account; we discuss this further in the discussion section (5.6). Lastly, the random baseline performs as expected, with the MCC score hovering around 0.

We now look at the discovered topics and their labels for the best results, taking the top terms for

Table 4. Top topic labels discovered for the best results for the 4-topic testset (top terms per topic for Twitter-LDA with 5 topics, top traits per cluster for STS_γ).

4-topic set	Twitter-LDA terms	$STS_{0.7}$ classes
Computer Science	1: swift, ios, app 2: data, bigdata, big	1: Big_data, EmergingTechnologies, Web_analytics 2: Model, ProgrammingLanguagesCreatedIn1995, IOS
Sports	golf, game, win	1: Cricketer, NationalCricketTeams, SoccerPlayer 2: GolfPlayer, PGATourGolfers, Golfer
Cars	car, f1, ferrari	CarManufacturer, SportsCarManufacturers, Coupes
Politics	auspol, people, obama	Statesman, NationalLeaders, PoliticiansFromSydney

Table 5. Top topic labels discovered for the best results for the 11-topic testset (top terms per topic for Twitter-LDA with 9 topics, top traits per cluster for STS_γ).

11-topic set	Twitter-LDA terms	STS_0 classes
iOS development	swift, ios, app	-
Web development	-	Model, RichInternetApplicationFrameworks, IOS
Data Science	data, bigdata, big	Big_data, EmergingTechnologies, Data
Soccer	-	FootballClubsInEngland, FIFAWorldCupPlayers
Football	good, game, day	AmericanFootballLeagueTeams, GridironFootballPlayer, AmericanFootballPlayer
Golf	golf, cup, rydercup	GolfPlayer, PGATourGolfers, AmericanMaleGolfers
Cricket	ausvind, cricket, india	-
Cars	car, f1, ferrari	CarManufacturer, Coupes, SportsCarManufacturers
US politics	obama, president, people	DemocraticPartyUSSenators, AmericanLegalScholars, HarvardLawSchoolAlumni
UK politics	labour, people, nhs	NationalistPartiesInTheUK, UK_Independence_Party, ConservativePartiesInTheUK
Australian politics	auspol, abbott, australia	PoliticiansFromSydney, AustralianPoliticians, AustralianRhodesScholars

Table 6. Summary of optimal results for the 1800 newsgroup article dataset.

1800 Newsgroup posts, 6-subject testset

	Precision	*Recall*	*F-score*	*NMI*	*MCC*	*Topics*
Random	0.190	0.167	0.177	0.684	0.000	6
LDA	0.289	0.786	0.423	0.506	0.262	2
Twitter-LDA	0.488	0.345	0.404	0.664	0.300	9
k-means	0.373	0.352	0.362	0.648	0.219	13
STS_{-1}	0.334	0.760	0.464	0.526	**0.321**	12

1800 Newsgroup posts, 20-topic testset

	Precision	*Recall*	*F-score*	*NMI*	*MCC*	*Topics*
Random	0.050	0.051	0.050	0.873	0.001	20
LDA	0.282	0.370	0.320	0.820	**0.283**	20
Twitter-LDA	0.239	0.254	0.246	0.833	0.206	25
k-means	0.167	0.259	0.203	0.801	0.156	27
$STS_{-0.6}$	0.099	0.862	0.177	0.504	0.198	18

Twitter-LDA and using our topic labeling technique described in section 4.3 to find the top classes for STS_γ. From table 4, we see that for the 4-topic set, the best Twitter-LDA result (5 topics) has wrongly split *Computer Science* into two sub-topics. For $STS_{0.7}$, *Computer Science* and *Sports* are erroneously split up into sub-topics. Upon inspection of the ontologies for the *Sports*-related topics and entities, we find that many entities simply do not link to a "Sports" class: in fact, there is no such class in the YAGO hierarchy. For *Soccer*, for example, common entities found in tweets are team names. Entities such as "Arsenal F.C." expand to "PremierLeagueClub" $\rightarrow$ "Club" $\rightarrow$ "Association" $\rightarrow$ ... , never reaching a class that could identify it with *Sports*. This is why we only see "CricketTeams", "SoccerPlayer", "Golfer", etc. in our result.

Looking at table 5 for the 11-topic testset, we see that for the best Twitter-LDA result (9 topics), the *Web development* and *Soccer* topics have not been properly identified. For STS_0, we fail to identify *Cricket*, and *iOS development* gets erroneously (although the two topics are highly related) combined with the *Web development* cluster, as is evident from the "iOS" label. Of note is that STS_γ, unlike LDA, correctly distinguishes between *Soccer* and *Football*.

Not listed are the top terms found for topics using standard LDA. These were generally similar to the terms Twitter-LDA found, but contained a number of extra "noise" topics, containing words such as "great", "today", "make". This noise was successfully filtered out by Twitter-LDA, explaining its better performance over standard LDA.

5.5.2. *Newsgroups dataset*

Next, we apply the same methods with the same settings to two configurations of the newsgroups dataset. We consider clustering in either the 6 subject matters or in the full 20 newsgroups (see table 1). First, we try the set of 1800 newsgroup posts consisting of 90 posts evenly and randomly taken from each newsgroup. The best STS_γ, LDA, Twitter-LDA and k-means results, including a comparison to the random baseline, are summarized in table 6. For space reasons, we omit the progression plots

Table 7. Summary of optimal results for the larger than 10.0 kb newsgroup article dataset.

Newsgroup posts larger than 10 kilobytes, 6-subject testset

	Precision	*Recall*	*F-score*	*NMI*	*MCC*	*Topics*
Random	0.213	0.164	0.186	0.756	0.000	6
LDA	0.683	0.319	0.434	0.718	0.387	12
Twitter-LDA	0.528	0.731	0.613	0.669	*0.497*	4
k-means	0.756	0.377	0.503	0.647	*0.457*	15
$STS_{0.3}$	0.669	0.467	0.550	0.744	*0.465*	12

Newsgroup posts larger than 10 kilobytes, 20-topic testset

	Precision	*Recall*	*F-score*	*NMI*	*MCC*	*Topics*
Random	0.092	0.048	0.063	0.778	0.000	20
LDA	0.683	0.497	0.576	0.761	0.548	17
Twitter-LDA	0.562	0.564	0.563	0.765	0.518	14
k-means	0.537	0.619	0.575	0.751	0.530	15
STS_0	0.512	0.688	0.587	0.773	0.546	15

for this dataset. We also omitted the clearly inferior versions of STS_γ for this evaluation, comparing only the variant using all types of ontology classes.

As expected, we see that results in terms of absolute F-score and MCC are significantly poorer compared to the Twitter dataset when we aim to detect all 20 topics, some of which we lack enough information about due to the brevity of posts. For the 6-subject test set, however, STS_γ still manages to outperform (Twitter-)LDA, but oddly enough it does so for lower topic scopes. STS_{-1} found 12 topics, while LDA and Twitter-LDA gave wildly different results at 2 and 9 topics respectively. *k*-means peaked at 13 topics, but with a significantly lower score. A likely reason for the poor performance of LDA is that the cluster sizes for the 6-subject testset are very uneven, ranging from 90 to 450 documents (see table 1): this is a known weakness of LDA, which is biased towards even-sized clusters. This is especially visible in the NMI scores for the 20-topic testset: both LDA variants score highly here due to all 20 clusters having roughly the right sizes (interestingly, the random clustering has a significantly higher NMI than even LDA: this makes sense, since we distribute nodes uniformly at random over 20 topics, and take an average of 100 iterations, leading to a very even distribution).

For 20-topics, standard LDA performs best, with a very significant improvement over the Twitter-specific version. STS_γ performs very poorly here. We interpret these results in more detail later.

Next, we try the set of 239 newsgroup posts larger than 10.0 kilobytes in size. The best results are summarized in table 7. This time, results in terms of the absolute F-score and MCC are higher, as expected; both for (Twitter-)LDA, but especially for STS_γ and *k*-means. Most methods performs roughly equal now for the both testsets, with the exception of standard LDA on the 6-subject testset. We note that we find the best result for the 20-topic testset with the topic scope parameter γ set to 0.0, yielding 15 topics; this is the same setting for which we had the best result for the Twitter 11-topic testset, which yielded 9 topics. $\gamma = 0.0$ yielded 15 topics for the 1800 newsgroup post set as well. This suggests that, using STS_γ-clustering, we do not have to know how many topics there are, nor does the size or content of the dataset matter; we can specify a desired topic scope, and automatically discover roughly the number of topics that exist at that scope.

Table 8. Summary of optimal results for hierarchical topic clustering.

175 Twitter users, 4 main and 11 sub-topics

	hP	*hR*	*hF*	*Levels*	*Topics per level*
hLDA	0.633	0.543	0.585	3	L1: 1, L2: 6, L3: 16
Hier. k-means	0.696	0.763	0.728	2	L1: 4, L2: 11
$STS_{0.7,recursive}$	0.877	0.837	0.857	3	L1: 6, L2: 8, L3: 2

5.5.3. *Hierarchical topic clustering*

Lastly, we will evaluate hierarchical clustering. We use the Twitter ground truth to try and cluster 175 Twitter users into a full hierarchy that contains both the 4 main topics and the 11 subtopics. For STS_γ, we use our recursive topic clustering algorithm (see algorithm 1), starting at the clustering that gave the best results for the 4-topic testset ($STS_{0.7}$). We compare against hierarchical LDA (hLDA) and hierarchical k-means, which we apply in the manner described in section 5.2.2. We also initialize hierarchical k-means to the best 4-topic clustering. hLDA requires no initialization; instead we fix the depth to 3 levels. We evaluate the results using hierarchical precision, recall and F-score as explained in section 5.2.1. The results are summarized in table 8. Table 9 gives an overview of the actual topic hierarchies created for hLDA and STS_h (k-means does not support topic labeling), detailing the location of each cluster within each hierarchy using a number-based notation.

We see that STS_h gives the most accurate results, followed by hierarchical k-means. We found that even with 800 iterations, hLDA was unable to converge to the appropriate number of topics, leading to a poor hF score. hLDA could not detect a single topic of the 4 topics in the top layer accurately: either the clusters in level 2 consisted of more than 1 of the topics (e.g. cluster 1 of level 2 in table 9 comprised both *cars* and *computer science*), or of the topics we expected in the lower half of the hierarchy.

5.5.4. *Statistical significance of results*

In [15], it is shown that given a sample of the full collection of users/documents (i.e. our ground truths), and p the true proportion of samples produced that is correct (which is unknown), n the size of the sample used to approximate p (the size of the ground truths: in our case 175 for Twitter users, and 1800 and 239 for each newsgroups set respectively) and $\hat{P}$ the approximation of p based on the ground truth, then this approximation lies in the interval

$$\hat{P} \in [p-\delta, p+\delta] \text{ where } \delta = \frac{1}{\sqrt{n}} \tag{21}$$

with 95% confidence. δ is thus the 95% confidence margin of error for the result. For example, if one result falls within this range of another result, then these two results do not differ sufficiently from each other in order to state with certainty that one is better than the other. We can apply this calculation on the resulting MCC scores for each approach to verify their statistical significance: we want to make sure that a good result according to the ground truth is sufficiently generalizable to a good result on larger, unseen data, and that an improvement of our method over the baselines is actually significant enough to draw conclusions about it.

For the Twitter users, we have $n = 175$, so the margin of error δ becomes $\frac{1}{\sqrt{175}} = 0.076$. Referring back the summary of results in table 3, we can conclude that for the 4-topic testset, our STS_γ-

Table 9. Clusters discovered at each level **L** for hierarchical clustering (top terms per topic for hLDA, top traits per cluster for STS_h). The numbers represent the level and position each cluster was located in the hierarchy.

L	hLDA topics	STS_h topic clusters
1	1: good, time, great	1: Big_data, EmergingTechnologies, Web_analytics 2: Model, ProgrammingLanguagesCreatedIn1995, IOS 3: Cricketer, NationalCricketTeams, SoccerPlayer 4: GolfPlayer, PGATourGolfers, Golfer 5: CarManufacturer, SportsCarManufacturers, Coupes 6: Statesman, NationalLeaders, PoliticiansFromSydney
2	1.1: car, cars, bmw 1.2: byu, game, coach 1.3: australia, today, minister 1.4: ballondor, live, fifa 1.5: data, learning, science 1.6: wt, pakvnz, pakistan	2.1: DistributedFileSystems, Apache_Hadoop, Run_batted_in 2.2: WebApplicationFrameworks, RichInternetApplicationFrameworks, Swift 3.1: NationalFootballLeagueTeams, GridironFootballPlayer, AmericanFootballPlayer 3.2: FootballClubsInEngland, PremierLeagueClubs, FootballClubsInLondon 3.3: CricketersAtThe2011CricketWorldCup, NationalCricketTeams, IndiaTestCricketers 6.1: RepublicanPartyStateGovernorsOfTheUnitedStates, UnitedStatesAirForceOfficers, Patient_Protection_and_Affordable_Care_Act 6.2: PoliticiansFromSydney, AustralianPoliticians, AustralianLeadersOfTheOpposition 6.3: NationalistPartiesInTheUnitedKingdom, UK_Independence_Party, ConservativePartiesInTheUnitedKingdom
3	1.1.1: swift, ios, app 1.1.2: vettel, webber, video 1.1.3: gsl, youtube, http 1.2.1: cowboys, bro, mfjs 1.3.1: auspol, abbott, manus 1.3.2: spotmyride, spotted, ferrari 1.4.1: golf, rydercup, year 1.4.2: league, goal, goals 1.5.1: obama, people, president 1.5.2: indyref, labour, scotland 1.5.3: car, ford, cars 1.5.4: data, bigdata, big 1.5.5: porsche, atlanta, photo 1.5.6: initializr, yelp, css 1.5.7: http, eurotour, sqcom 1.6.1: ausvind, india, cricket	2.2.1: CascadingStyleSheets, Sheet, WebDev 2.2.2: OS_X, Swift, Macintosh

clustering gives a significant improvement over the baselines – the smallest difference, between STS_γ and Twitter-LDA, is $0.858 - 0.767 = 0.091 > 0.076$. For the 11-topic testset, our method again provides a significant improvement over LDA, but the difference between STS_γ and Twitter-LDA lies within the margin of error ($0.039 < 0.067$): we cannot state that one method is better than the other with 95% confidence. Similarly, results for STS_γ and k-means can be considered equivalent, which is in line with our earlier observations, and subject of discussion in the next section.

For the main newsgroup dataset, $\delta = \frac{1}{\sqrt{1800}} = 0.024$. Due to the size of this testset, we obtain a much smaller margin of error compared to the set of Twitter users. Referring back to table 6, we see that all pair-wise results are statistically significant. Finally, for the set of newsgroup posts larger than 10kb, we obtain $\delta = \frac{1}{\sqrt{239}} = 0.065$. From table 7, we learn that for the 6-subject testset, the improvement of STS_γ over LDA is significant again, but we cannot make conclusions about the difference with Twitter-LDA and k-means clustering. For the 20-topic testset, we see that all results (excluding random) are essentially equivalent.

5.6. *Results discussion*

For Social Web content, our ontology-assisted topical similarity calculation and graph-based STS_γ-clustering results show a significant improvement over traditional tf-idf weighting, LSA-based topic modeling such as LDA and Twitter-LDA and common document clustering approaches such as k-means clustering. An important reason for the poor performance of tf-idf and LDA is the absence of overlapping terms due to the high dimensionality (and therefore high sparsity given the limited and noisy content on the Social Web) of the term vector space compared to our trait vector space. For k-means, despite using the same similarity calculation to construct graph G_γ as our STS_γ-clustering, γ was fixed to 0, which we found discovered appropriate topics at roughly the scope expressed by the 11-topic testset. However, even though we could force k-means to cluster the graph into 4 topics, it performed poorly in terms of F-score and MCC for the 4-topic testset. This is because k-means clustering does not take the existence of a latent topic hierarchy into account: the topology of the graph G_0 is shaped with a bias towards how many topics exist within the data at that particular topic scope. These results make clear the advantages of a hybrid approach – hierarchical topic modeling combined with graph-based community detection – compared to traditional methods.

For regular documents, we see that our ontology-assisted topic modeling approach yields results that are on par with LDA only when there is enough content available per document. It appears that ontologically expanding text content only works well when there is enough context information available – when the dataset contains many documents that sometimes consist of only one sentence, such as with the 20-newsgroups dataset, STS_γ-clustering gives poor results. This is a significant weakness of our approach. This is in line with results reported in [11]. Topic modeling approaches that iterate over the whole dataset when deriving a model, such as LDA, work better in these cases.

When there is enough content per document – as was the case for the dataset of newsgroup posts over 10.0 kb in size – ontology-assisted topical similarity as employed in STS_γ-clustering and k-means performs roughly equivalent to document-level word co-occurrence as employed in LDA. This is not a surprising result, since the newsgroup posts use more formal language and contain less noise than their Twitter counterparts, leading to more word overlap between documents.

Another reason for discrepancies is due to the (un-)evenness of clusters. Our clustering algorithm can deal with uneven clusters well, since clusters are determined solely based on the degree of similarity and number of similar users in the dataset. LDA is biased towards evenly sized topics, and it

Fig. 16. qualitative comparison between the LDA and k-means baselines and STS_γ-clustering.

	LDA	Twitter-LDA	k-means	STS_γ-clustering
Perf. on Social Web content	-	+	+-	++
Perf. on formal documents	++	+	+-	+
Pre-defined topics	Yes	Yes	Yes	No
Hierarchical topics	Yes	Yes	No	Yes
Topic labels	Top terms	Top terms	None	Top classes (machine-readable)
Time complexity	$O(nkvi)$	$O(nkvi)$	$O(nkcij)$	$O(n^2c + n^3 \log n)$

is very difficult for the method to detect topics of vastly different sizes, unless some form of iterated hyperparameter optimization is employed. In comparison to our approach, LDA gave the overall best result when dividing the 1800 newsgroup posts into 20 topics; this is the only dataset where all desired clusters are of even size (90 documents per cluster).

A important observation is that classes from our trait vectors originally form a concrete hierarchy; this is not the case for terms, where this hierarchy needs to be guessed from the terms available in the corpus. This hierarchical information is retained within the trait vectors we calculate, making it easier for the original class ontology to be reconstructed in terms of traits. A nice side effect of this is that for traits, we may simply use the most characteristic ontology classes for a cluster as labels; and these labels are machine-readable and directly linked to DBpedia and the rest of the Linked Data cloud. Using LDA, the best we can do is to pick the top terms per topic that actually occurred within the text content, which are often less descriptive and harder for machines to reason about.

In conclusion, we summarize the results of the evaluation by means of a qualitative comparison between our approach and the baselines, in table 16. This overview includes computational time complexity per method, which we will touch upon briefly. Due to STS_γ's reliance on a full connectivity graph based algorithm, its big-O complexity is in the $n^3 \log n$ order of magnitude, which is the highest of all the algorithms. This means that in its current form, the method does not scale well to large datasets. This is a significant limitation of the approach. A substitution of the current HCS-based algorithm with e.g. a density-based one such as ES clustering would improve scalability, but this is outside the scope of this paper.

6. Conclusion

In this paper, we have presented our work on ontology-assisted hierarchical clustering of Social Web users by their shared topics of interest. We have shown that we can bypass common limitations of the term vector space model by leveraging an external ontology to express user topic profiles in terms of trait vectors, and subsequently calculating scoped topical similarity, or STS_γ, between users; a measure that expresses the distinguishing characteristics of groups of users compared to the full collection of users at a certain level of topic generality.

We applied community detection techniques on a graph constructed from the STS_γ between users and showed that, by incorporating the concept of topic scope into our calculations, we can get results based on the desired scope rather than the desired number of topics – with the same topic scope values,

we managed to discover roughly the appropriate numbers of topics even for different data sources of different sizes. Furthermore, the approach could be used to generate human- and machine-readable labels for clusters, and to divisively cluster a group of users or documents in order to generate a full topic hierarchy.[d]

The experimental results presented showed an improvement of up to 14.7% over standard latent Dirichlet allocation, 11.9% over Twitter-LDA and up to 26.7% over k-means clustering on Social Web data. We also showed that we can correctly detect topics at different scopes by changing the topic scope parameter, whereas changing the number of topics for k-means clustering failed to detect the correct topics. Results on traditional documents were mixed, with results equivalent to or worse than the LDA state-of-the-art. For full hierarchical clustering, STS_h outperformed *hLDA* and a hierarchical version of k-means. Notably, we observe similar improvements over LDA using community detection-based clustering as was shown in Lancichinetti et. al. [25]. This is an encouraging result, suggesting that perhaps graph-based community detection methods for document clustering are a better fit for the task than probabilistic LSA-based methods, at least when the topics are not known in advance.

Future work Throughout this work we have assumed that users can be cleanly divided into disjoint topic clusters, but in the real world this is obviously not the case: users can be part of multiple shared-interest communities or not have any specific interest at all. We need to further develop and evaluate methods for overlapping community detection. One possible way is to replace the current HCS community detection algorithm with a clique percolation method [12] to discover overlapping cliques in a graph.

A significant untapped area of potential lies in multi-lingual topic clustering. Since the semantic classes we collect from DBpedia are language-agnostic, and DBpedia resources exist in multiple languages, it becomes trivial to link together equivalent entities from different languages and collect the same classes for both languages. This would allow us to detect similar content in different languages without requiring any knowledge about these languages.

The current graph connectivity-based clustering algorithm used does not scale to large datasets. Future work would experiment with more modern density-based clustering algorithms as well.

Lastly, we relied largely on YAGO, DBpedia and Schema.org classes to model user topics of interest. In reality, these class hierarchies are still quite limited in what they can express, as many entities simply do not have classes in DBpedia, or they are not sufficiently connected to the parent classes we are looking for (such as most sports-related entities not actually being connected to a "Sports" class). Other than these types of super- and subsumption relations, there are a great many other relations available in DBpedia (e.g. "team of" linking players to teams, "field of" linking academic disciplines to famous researchers, etc.). How to make good use of these relations is another promising area to explore.

References

1. Dbpedia wiki: The dbpedia ontology (2014). *http://wiki.dbpedia.org/Ontology2014*, Retrieved on April 14 2015.
2. F. Abel, Q. Gao, G.-J. Houben, and K. Tao. Analyzing user modeling on twitter for personalized news recommendations. In *User Modeling, Adaption and Personalization*, pages 1–12. Springer, 2011.

[d] The source code used to obtain the results described in this paper, including an interactive visualizer for hierarchical clustering, can be found at `https://github.com/ktslabbie/TwinterestExplorer`

3. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, chapter 52, pages 722–735. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2007.
4. C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far, 2009.
5. D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.
6. D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.
7. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
8. J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM, 2013.
9. S. Dasgupta, C. Papadimitriou, and U. Vazirani. Algorithms–chapter 5, 2006.
10. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.
11. L. Derczynski, D. Maynard, N. Aswani, and K. Bontcheva. Microblog-genre noise and impact on semantic annotation accuracy. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 21–30. ACM, 2013.
12. I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005.
13. M. S. Granovetter. The strength of weak ties. *American journal of sociology*, pages 1360–1380, 1973.
14. T. B. Group. Social usage involves more platforms, more often. *www.emarketer.com/Article/Social-Usage-Involves-More-Platforms-More-Often/1010019*, Retrieved on February 19 2013.
15. W. V. Hage, A. Isaac, and Z. Aleksovski. Sample evaluation of ontology-matching systems. In *Fifth Int. Workshop on Evaluation of Ontologies and Ontology-based Tools, ISWC 2007*.
16. J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.
17. E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information processing letters*, 76(4):175–181, 2000.
18. M. Hausenblas and R. Cyganiak. Schema.rdfs.org. *http://schema.rdfs.org/*, Retrieved on April 20 2015.
19. T.-A. Hoang and E.-P. Lim. On joint modeling of topical communities and personal interest in microblogs. In *Social Informatics*, pages 1–16. Springer, 2014.
20. A. Hotho, S. Staab, and G. Stumme. Ontologies improve text document clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 541–544. IEEE, 2003.
21. A. K. Jain, R. C. Dubes, et al. *Algorithms for clustering data*, volume 6. Prentice hall Englewood Cliffs, 1988.
22. K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.
23. S. Kiritchenko, F. Famili, S. Matwin, and R. Nock. Learning and evaluation in the presence of class hierarchies: Application to text categorization. 2006.
24. J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.
25. A. Lancichinetti, M. I. Sirer, J. X. Wang, D. Acuna, K. Körding, and L. A. N. Amaral. High-reproducibility and high-accuracy method for automated topic classification. *Physical Review X*, 5(1):011007, 2015.
26. K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, pages 331–339, 1995.
27. J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.
28. X. Liu, M. Zhou, F. Wei, Z. Fu, and X. Zhou. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 526–535. Association for Computational Linguistics, 2012.

29. B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.
30. A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.
31. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proc. of the 7th Intl. Conference on Semantic Systems*, 2011.
32. M. Michelson and S. A. Macskassy. Discovering users' topics of interest on twitter: a first look. In *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, pages 73–80. ACM, 2010.
33. G. A. Miller. WordNet: a lexical database for English. *Commun. ACM*, 38(11):39–41, 1995.
34. M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.
35. S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012.
36. O. U. Press. Rt this: Oup dictionary team monitors twitterer's tweets. http://blog.oup.com/2009/06/oxford-twitter/, 2009.
37. M. Qiu, F. Zhu, and J. Jiang. It is not just what we say, but how we say them: Lda-based behavior-topic model. SIAM.
38. J. Rennie. The 20 newsgroups data set. *http://qwone.com/ jason/20Newsgroups/*, Retrieved on April 2 2015.
39. A. Ritter, S. Clark, O. Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.
40. J. Ronallo. Html5 microdata and schema. org. *Code4Lib Journal*, 16, 2012.
41. M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press, 2004.
42. C. N. Silla Jr and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.
43. K. Slabbekoorn, T. Noro, and T. Tokuda. Towards twitter user recommendation based on user relations and taxonomical analysis. In *23nd European-Japanese Conference on Information Modelling and Knowledge Bases (EJC), 2013*, 2013.
44. K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.
45. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of WWW'07*, pages 697–706, 2007.
46. L. Tang and H. Liu. Community detection and mining in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2(1):1–137, 2010.
47. O. Tsur, A. Littman, and A. Rappoport. Efficient clustering of short messages into general domains. In *Proceedings of the 7th International Conference on Weblogs and Social Media*, 2013.
48. P. Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing & Management*, 24(5):577–597, 1988.
49. S.-H. Yang, A. Kolcz, A. Schlaikjer, and P. Gupta. Large-scale high-precision topic modeling on twitter. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1907–1916. ACM, 2014.
50. W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval*, pages 338–349. Springer, 2011.
51. Z. Zhao, S. Feng, Q. Wang, J. Z. Huang, G. J. Williams, and J. Fan. Topic oriented community detection through social objects and link analysis in social networks. *Knowledge-Based Systems*, 26:164–173, 2012.