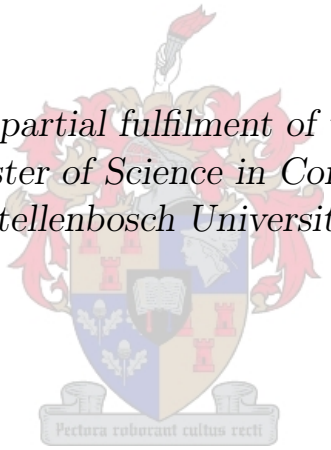
Constructing Topic-based Twitter Lists

by

Francois de Villiers



Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Computer Science at Stellenbosch University

Computer Science Division,
Department of Mathematical Sciences,
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Supervisors:

Dr M.R. Hoffmann Dr R.S. Kroon

March 2013

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: February 17, 2013
.....

Abstract

Constructing Topic-based Twitter Lists

P.F. de Villiers

Computer Science Division,
Department of Mathematical Sciences,
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Thesis: M.Sc. Computer Science

March 2013

The amount of information that users of social networks consume on a daily basis is steadily increasing. The resulting information overload is usually associated with a loss of control over the management of information sources, leaving users feeling overwhelmed.

To address this problem, social networks have introduced tools with which users can organise the people in their networks. However, these tools do not integrate any automated processing. Twitter has lists that can be used to organise people in the network into topic-based groups. This feature is a powerful organisation tool that has two main obstacles to widespread user adoption: the initial setup time and continual curation.

In this thesis, we investigate the problem of constructing topic-based Twitter lists. We identify two subproblems, an unsupervised and supervised task, that need to be considered when tackling this problem. These subproblems correspond to a clustering and classification approach that we evaluate on Twitter data sets.

The clustering approach is evaluated using multiple representation techniques, similarity measures and clustering algorithms. We show that it is

possible to incorporate a Twitter user's social graph data into the clustering approach to find topic-based clusters. The classification approach is implemented, from a statistical relational learning perspective, with kLog. We show that kLog can use a user's tweet content and social graph data to perform accurate topic-based classification. We conclude that it is feasible to construct useful topic-based Twitter lists with either approach.

Uittreksel

Die bou van onderwerp-gerigte Twitter lyste

("Constructing Topic-based Twitter Lists")

P.F. de Villiers

Afdeling Rekenaarwetenskap,
Departement van Wiskundige Wetenskappe,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: M.Sc. Rekenaarwetenskap

Maart 2013

Die stroom van inligting wat sosiale-netwerk gebruikers op 'n daaglikse basis verwerk, is aan die groei. Vir baie gebruikers, skep hierdie oordosis inligting 'n gevoel dat hulle beheer oor hul inligtingsbronne verloor.

As 'n oplossing, het sosiale-netwerke meganismes geïmplementeer waarmee gebruikers die inligting in hul netwerk kan bestuur. Hierdie meganismes is nie selfwerkend nie, maar kort toevoer van die gebruiker. Twitter het lyste geïmplementeer waarmee gebruikers ander mense in hul sosiale-netwerk kan groepeer. Lyste is 'n kragtige organiserings meganisme, maar tog vind grootskaal gebruik daarvan nie plaas nie. Gebruikers voel dat die opstelling te veel tyd in beslag neem en die onderhoud daarvan te veel moeite is.

Hierdie tesis ondersoek die probleem om onderwerp-gerigte Twitter lyste te skep. Ons identisifeer twee subprobleme wat aangepak word deur 'n nie-toesig en 'n toesighoudende metode. Hierdie twee metodes hou verband met trosvorming en klassifikasie onderskeidelik. Ons evalueer beide die trosvorming en klassifikasie op twee Twitter datastelle. Die trosvorming metode word geëvalueer deur te kyk na verskillende voorstellingstegnieke, eendersheid maatstawwe en trosvorming algoritmes.

Ons wys dat dit moontlik is om 'n gebruiker se Twitter netwerkdata in te sluit om onderwerp-gerigte groeperinge te vind. Die klassifikasie benadering word geïmplementeer met kLog, vanuit 'n statistiese relasionele leertoerie perspektief. Ons wys dat akkurate onderwerp-gerigte klassifikasie resultate verkry kan word met behulp van gebruikers se tweet-inhoud en sosiale-netwerk data. In beide gevalle wys ons dat dit moontlik is om onderwerp-gerigte Twitter lyste, met goeie resultate, te bou.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

- my study leaders, Dr M.R. Hoffmann and Dr R.S. Kroon, for their mentorship and support throughout this thesis;
- MIH for their financial assistance;
- the MIH Media lab management for providing an excellent working environment;
- my friends in the Media lab who made the time spent working on this thesis worth it;
- and finally, my family for their much needed support.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AMI	Adjusted mutual information
ANID	Adjusted normalised information distance
AP	Affinity propagation
API	Application programming interface
KL	Kullback-Leibler
LDA	Latent Dirichlet allocation
MI	Mutual information
NID	Normalised information distance
NLTK	Natural language toolkit
NSPDK	Neighbourhood subgraph pairwise distance kernel
RBF	Radial basis function
SRL	Statistical relational learning
SVM	Support vector machine
TF-IDF	Term frequency-inverse document frequency

Chapter

1

Introduction

The demands of users have changed since the advent of social networks. In the beginning, users were satisfied with being instantly connected to friends, family and colleagues. As the social circles of each user expanded, so did the number of messages, images and links that each user consumed daily (Borgs *et al.*, 2010).

Social networks, such as Facebook, Twitter and Google+, have recently introduced mechanisms to enable users to manage this vast flow of information. For example, Twitter introduced lists, which allow a user to categorise users and label them. However, initial setup cost and continual curation are common problems for a user creating these lists; a user of any system with many connections will find it tedious to sort through all his connections and organise them according to topics. The whole process could be simplified if the user could be provided with a good initial configuration.

1.1 Motivation

The rapid growth of social networks has introduced new problems, arguably the most prominent of these being information overload (Borgs *et al.*, 2010). There is no single generally accepted definition of information overload. Bawden and Robinson (2009) state that the term is used in referring to a state of affairs where an individual's efficiency in using information in their work is hampered by the amount of relevant and useful information available to them.

Information in a user's network is typically presented as a time-line of events. As a result, if a user has many information sources in his network, not all the relevant information will be displayed. To alleviate this problem of information overload, social networks provide tools with which users can organise their information sources into manageable groups. The tools to create these manageable groups are known as lists in Twitter, circles in Google+ and smart lists in Facebook. At the time of writing, these tools are all manually managed by the user, in other words each grouping action is performed by the user. Two problems, initial setup costs and continual curation, are possible factors that influence the adoption rate of these tools. A user of any system with many connections will find it tedious to sort through all his/her connections and organise them according to topics.

In the past, machine learning techniques have performed well in the task of automatic data organisation (Witten and Frank, 2005). Therefore, we pose the question: "Can machine learning techniques be used to automate group creation procedures in social networks?" We focus our efforts on Twitter.

Twitter introduced lists in 2009, which allow a Twitter user to create curated groups containing other users in the ecosystem. The manual list creation process consists of identifying a set of related users based on a particular attribute, and iteratively adding and removing users based on the defined attribute.

1.2 Objectives

This work's principal objective is to provide a detailed investigation into the problem of constructing topic-based Twitter lists. The principal objective can be split into two subproblems (a) constructing lists for users who have not created lists and (b) constructing lists for users who have created a limited number of lists. As such, we investigate unsupervised and supervised list construction approaches.

1.3 Contributions of this work

This thesis makes the following contributions:

- We introduce a clustering approach based on tweet content that can construct topic-based Twitter lists (De Villiers *et al.*, 2012).

- For this clustering approach, we perform a comparative study of similarity measures on short text documents.
- We evaluate two clustering algorithms, k-means (Tan *et al.*, 2006) and affinity propagation (Frey and Dueck, 2007), over a range of clusters sizes on multiple short text document data sets.
- We show how a Twitter user's social graph information can be used in a clustering approach.
- We define a kLog model (Frasconi *et al.*, 2012) that effectively incorporates graph content from Twitter and achieves good classification results when constructing topic-based lists.
- Using kLog, we perform a comparative study on different graph formulations. This provides insight into how the shape of the graph affects the classification results.

Chapter

2

Literature Review

This chapter presents related work to provide the necessary background for the work in the rest of this thesis. We introduce two approaches that form a general framework in which the two subproblems of constructing topic-based Twitter lists can be solved. The two approaches, unsupervised and supervised, are examined in the context of previous work. Finally, we review the studies that examine critical aspects of the topic-based Twitter lists construction process.

2.1 Overview

In Chapter 1, we discussed the problem of information overload, and proposed the automated construction of topic-based Twitter lists to alleviate it. To adequately address this topic, it is necessary to consider two types of users: those who have already used the lists tool to create lists, and those who have not yet created any lists. Therefore, the available information consists of partially labelled and unlabelled data. A natural solution to these two subproblems are clustering and classification respectively.

The clustering process is an example of unsupervised classification (Xu and Wunsch, 2005), and is applied to unlabelled data (Everitt *et al.*, 2011; Jain and Dubes, 1988). The goal of clustering is to find natural hidden structure (Arbib, 2002) in data, which is used to separate the data set into discrete partitions (Cherkassky and Mulier, 2007). Xu and Wunsch (2005) describe the clustering approach as a 5-step process.

The first step consists of gathering the data to cluster and, depending on the domain, can be either an easy or a difficult task. The second step is feature selection or extraction; feature selection involves choosing distinguishing features from a set of candidates (Jain *et al.*, 1999), while feature extraction applies a transformation to generate useful features from those selected (Jain *et al.*, 2000). The third step consists of selecting a clustering algorithm, and based on the clustering algorithm an appropriate proximity measure. Cluster validation, the fourth step, is applied after the proposed clusters have been found and usually involves any of three categories of testing criteria. These three categories are referred to as external indices, internal indices and relative indices (Xu and Wunsch, 2005). External indices evaluate the clustering result on some pre-specified structure; internal indices evaluate the clustering structure on the original data; and relative indices evaluate the comparison of different clustering solutions (Gordon, 1999). The final step in the clustering process is interpretation or visualisation of the results, providing users with meaningful insights from the original data (Xu and Wunsch, 2005).

In contrast to the clustering approach, classification is a supervised task (Kotsiantis *et al.*, 2007). The goal of supervised classification is to learn a function or set of rules from examples in a training set, thus creating a classifier that can be used to label new examples (Dutton and Conroy, 1997; Nguyen and Armitage, 2008). Kotsiantis *et al.* (2007) describe a 5-step process for applying supervised machine learning to real-world problems.

The first step consists of three parts, namely identifying the problem, gathering the required labelled data, and performing pre-processing. The data can either be selectively gathered using only the attributes and features that are most informative (Kotsiantis *et al.*, 2007) or all available data can be captured. In most cases, the data set contains noise and missing feature values, which can require significant pre-processing (Zhang *et al.*, 2003). The data pre-processing task consists of removing noise (Hodge and Austin, 2004), feature selection (Liu and Motoda, 1998) and feature transformation (Markovitch and Rosenstein, 2002). Applying noise removal and feature selection techniques reduce the dimensionality of the data and allows for more efficient implementations of supervised learning algorithms. The feature transformation step can possibly improve the accuracy of the classifier by constructing new features from the basic feature set (Markovitch and Rosenstein, 2002).

The second step is the selection of a training set, and the corresponding classifier evaluation techniques that influence it. To effectively evaluate the performance of a classifier, careful consideration should be given to the selection of a training and test set. Three popular approaches are two-thirds split, cross-validation and leave-one-out validation. In the first case, two-thirds of the data set is used for training and the rest for testing the classifier (Kotsiantis *et al.*, 2007). Cross-validation splits the data set into mutually exclusive subsets of approximately equal size. Training is then conducted on the union of all these subsets except one which is used for testing; this is repeated until each set has been used for testing (Kohavi, 1995). Leave-one-out validation is a special case of cross-validation where each subset consists of a single data point (Kohavi, 1995).

The third step of supervised learning is the selection of a learning algorithm, which depends on the task under consideration. This is followed by the fourth step, training and testing; training consists of executing the supervised learning algorithm to obtain a classifier that is evaluated in the testing step. In the fifth and final step, the results are interpreted with techniques that are effective in representing and describing the different performance aspects of the classifier.

The description of the clustering and classification process corresponds to the two subproblems, which we identified as (a) unlabelled and (b) partially labelled data. In the rest of this chapter, we discuss additional background related to the elements of each process.

2.2 Clustering

Clustering has been applied to a variety of problems (Markman, 2011; Kyriakopoulou and Kalamboukis, 2006; Millar *et al.*, 2009) and data types (Jain *et al.*, 1999). In the document clustering domain, we see that most studies evaluated clustering solutions on full-length text documents (Steinbach *et al.*, 2000; Huang, 2008). Only recently has the application of clustering to short text documents been evaluated (Rangrej *et al.*, 2011).

The studies on full-length text documents have all comprehensively evaluated various aspects of the clustering approach: Zhang *et al.* (2011) evaluated the document processing and feature representation step, Huang (2008) compared the performance of different similarity measures, and Singh *et al.* (2011)

compared different clustering algorithms. These studies provide valuable insights into the selection of methods and techniques when clustering full-length text documents. However, these insights are not always directly transferable to short text documents (Perez-Tellez *et al.*, 2011; Pinto *et al.*, 2011). It is therefore important to consider the different aspects of the clustering approach for short text documents.

2.2.1 User representation

The main content produced by a Twitter user is the user's tweets. Tweets thus form important data from which a user's topic-based expertise can be extracted. A user's topic-based expertise is the set of topics that the user tweets about. Studies have shown that it is difficult to extract information from a single tweet (Xu and Oard, 2011; Jin *et al.*, 2011; Hong and Davison, 2010), and as a result some authors have recommended techniques that augment tweets with external information (Jin *et al.*, 2011).

Hong and Davison (2010) followed a different approach to Jin *et al.* (2011): instead of augmenting a user's tweets with external information, they rather aggregate a user's tweets into a single document. This document is then used to generate a feature vector. They found that a simple aggregation of user tweets in a single document shows good performance in both the classification and clustering tasks. Their evidence suggests that the length of the document is the biggest contributing factor to the success of the feature representation techniques discussed in this section.

2.2.2 Feature vector representation

Well-known feature vector representation techniques for short text documents are term frequency-inverse document frequency (TF-IDF) (Salton and McGill, 1983) and latent Dirichlet allocation (LDA) (Blei *et al.*, 2003). LDA is a topic model related to the well-known probabilistic latent semantic analysis (Hofmann, 2001) technique with the addition of a dirichlet prior.

A number of studies have compared TF-IDF and LDA on different tasks, such as predicting popular tweets (Hong and Davison, 2010), user recommendations (Pennacchiotti and Gurumurthy, 2011) and tweet recommendations (Ramage *et al.*, 2010). These studies found that LDA tends to outperform

TF-IDF. However, this has not been verified for the clustering task on short text documents.

2.2.3 Similarities

In Section 2.1, we reviewed the clustering process and noted the third step of selecting a clustering algorithm and proximity/similarity measure. Lin (1998) defines the similarity between two objects as the ratio of information contained in the common features to the information contained in the total set of all features between two objects. A number of studies (Huang, 2008; Sandhya *et al.*, 2008; Subhashini and Kumar, 2010) have evaluated different similarity measures on full-length text documents. The measures compared in these studies, were the Euclidean distance, cosine similarity, Pearson correlation coefficient, Kullback-Leibler divergence and Jaccard coefficient. The studies showed similar performance between these measures, except in some cases where the Euclidean distance performed notably worse (Sandhya *et al.*, 2008). In terms of performance on short text documents, a comprehensive study has not been performed. Rangrej *et al.* (2011) performed a preliminary study comparing multiple clustering algorithms and two similarity measures, the Euclidean distance and cosine similarity. The results indicate similar performance to full-length text documents.

2.2.4 Clustering algorithms

Clustering algorithms can generally be considered to be of one of two types (Zhao *et al.*, 2005), namely hierarchical or partitional. Hierarchical clustering solutions provide a view of data at different levels of abstraction, which is ideal for interactive exploration and visualisation (Zhao *et al.*, 2005). Partitional clustering approaches separate data into different non-overlapping subsets (Frey and Dueck, 2007).

Larsen and Aone (1999) claim that partitional clustering algorithms are well suited for clustering large document data sets due to their relatively low computational requirements. k-Means, a well-known partitional clustering algorithm, has been successfully applied to tasks on both full-length (Amine *et al.*, 2010) and short text documents (Rangrej *et al.*, 2011). The algorithm performs well on both types of documents. Affinity propagation (AP) (Frey and

Dueck, 2007), a new clustering algorithm that performs well in the document clustering domain, has recently been introduced. AP is also a partitional clustering algorithm, and a recent study (Rangrej *et al.*, 2011) on short text documents has shown that it outperforms k-means. Frey and Dueck (2007) recommend AP for the clustering task when a large number of natural clusters exist. However, in the case of relatively few clusters, k-means may be better. An evaluation of k-means and AP over a range of cluster sizes on short text documents has to the best of our knowledge not yet been performed.

2.3 Classification

Statistical learning is the study of inference, making predictions or constructing models from data in a statistical framework (Bousquet *et al.*, 2004). In contrast to statistical learning is relational learning, which is the study of machine learning and data mining where knowledge is represented in relational form or first-order logic form (De Raedt, 2008). Statistical relational learning attempts to incorporate elements of these two approaches to represent, reason, and learn in domains with complex relational and rich probabilistic structure (Getoor and Taskar, 2007).

Classification has been applied to a range of tasks in the document domain, including recommendations (Chen *et al.*, 2010) and topic-based expertise predictions (Ghosh *et al.*, 2012). Earlier classification studies on Twitter have mostly consisted of analysing tweet content (Chen *et al.*, 2010; Hong and Davison, 2010) of a Twitter user to perform classification tasks as mentioned above. Recent studies have started to incorporate the user's social graph information (Ghosh *et al.*, 2012; Wagner *et al.*, 2012), which can be used as a strong topic-based expertise indicator for a user. However, incorporating this graph structure is not a trivial task. The above studies that incorporate social graph information do not explicitly model the connections between users, but use them to retrieve label information that is combined with tweet content to represent a user.

As a result, by using only the label information and not explicitly modelling the graph connections, important data that could be used to construct user features are discarded. A possible solution to the above problem is the use of graph-based classification (Ketkar *et al.*, 2009). This approach models a user according to his explicit social graph connections. A learning algorithm is

then applied to each user's individual graph. This learning algorithm builds a classifier, which assigns each user to a category. Graph-based classification techniques are good at modelling explicit graph structure, but to include the contents of a user's tweets in the process is not a trivial task (Ketkar *et al.*, 2009).

Recently, a statistical relational learning system, kLog (Frasconi *et al.*, 2012), has been introduced that can model explicit social graph structure and incorporate content as properties for nodes in the graph model. It has been shown that kLog can be applied to a wide variety of supervised classification tasks (Verbeke *et al.*, 2012*a*,*b*), and achieves good results in text-intensive domains such as WebKB (Frasconi *et al.*, 2012). We provide an in-depth discussion of the kLog system in Chapter 4.

2.4 Twitter lists

The problem of user topic discovery (Wagner *et al.*, 2012; Michelson and Macskassy, 2010) and automatic list construction (Greene *et al.*, 2011) has gained popularity. Early approaches to topic discovery used a knowledge base that defined topic categories, and used trigger words in tweets to perform topic assignments (Bernstein *et al.*, 2010; Michelson and Macskassy, 2010).

Wagner *et al.* (2012) evaluated which features of a Twitter user best encodes that user's topic-based expertise. They evaluated tweets, retweets, profile information and user list memberships; the results indicated that list memberships are the best indicator of a user's topic-based expertise. Furthermore, Ghosh *et al.* (2012) introduced techniques to extract metadata from user list memberships, which they then used to infer a user's topic-based expertise. Both of these studies found evidence that list memberships are a good indicator of a user's topic-based expertise.

Related to the task of finding a user's topic-based expertise is the automatic construction of topic-based Twitter lists. This involves extracting the topic-based expertise of users and combining those who share the same expertise to form topic-based Twitter lists. This task has had limited coverage in the literature. One of the first works (Greene *et al.*, 2011) attempted to support the curation process of Twitter lists by recommending users to add to a list. A graph-based approach was used to model friendships, mentions, retweets

and list memberships. The approach evaluated the graph edges to perform recommendations and did not incorporate tweet content.

Greene *et al.* (2012) extended this work by constructing topic-based communities using list membership information. In the study, a community detection algorithm that detects communities based on the overlap of list memberships was used. Their communities are defined based on a stability measure of the community. A user is assigned to a community based on the number of lists in that community that the user is a member of. The results were evaluated over 18 categories of 499 users, using precision, recall and the F1-measure (Chapter 5). They achieved mixed results: four categories showed good performance with a F1 score greater than 0.8, while 6 categories achieved less than 0.5 for the F1 score.

2.5 Conclusion

This chapter, presented the literature relevant to our work. We introduced an unsupervised and supervised learning framework, followed by a discussion of the clustering and classification approaches. Each important part of the clustering approach was discussed based on the work in that area. For the classification task we introduced kLog, a statistical relational learning system, which allow us to model a user based on tweet content as well as social graph information. We also discussed the literature related to topic-discovery and list creation on Twitter, including the results of a recent study on topic-based communities.

Chapter

3

Document clustering approach

This chapter introduces our clustering approach for application to data collected from Twitter. We discuss the concept of short text documents as well as how to use such documents collected from Twitter to represent a Twitter user.

Various important aspects of the clustering approach are considered, including feature vector representation, and the choice of a similarity measure and clustering algorithm. The feature vector representation discussion introduces TF-IDF and LDA. Followed by a discussion of five document similarity measures, before introducing the k-means and affinity propagation clustering algorithms.

3.1 Document types

Clustering algorithms have mostly been applied to full-length text documents. However, with the recently increased popularity of Twitter, there have been an increasing number of applications to short text documents. A *tweet*[1] consists of a maximum of 140 characters and may contain *mentions*, *hashtags* and *hyperlinks*. A mention occurs when user A refers to user B in a tweet by including the username of B preceded by an "@". Hashtags usually relate to the topic of a tweet and consist of a term or concatenation of terms starting with a "#". A hashtag is used when users discuss an event or topic on Twitter.

[1] A short text document on Twitter is referred to as a tweet.

It simplifies the process of finding tweets related to an event or topic, since one can search Twitter for tweets containing a specific hashtag.

Figure 3.1: A screenshot of a tweet containing a hyperlink, mentions and hashtags.

Figure 3.1 shows a tweet containing both mentions (@grantimahari and @scully313) as well as hashtags (#FireFly and #LEGO). Hong and Davison (2010) showed that although a tweet is short in length, it may still convey rich meanings. The same study showed that representing a user by a single document consisting of a collection of tweets is a good representation for topic discovery and clustering performance.

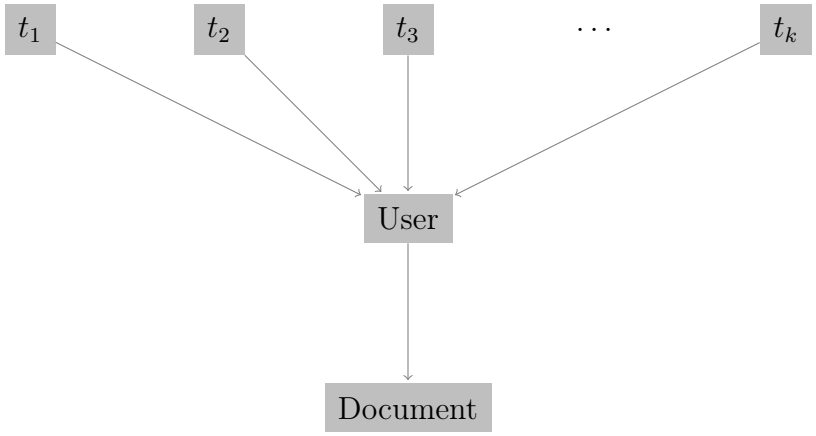


Figure 3.2: An illustration of how tweets are collected and combined to form a text document representing a Twitter user.

Figure 3.2 shows the process used to represent a Twitter user as a single document. Some number k of tweets are retrieved for a user, which are then concatenated to form a single document. A document created for a Twitter user in this way is referred to as a *user document* in this study.

A further possibility is to use this document creation approach using other information from Twitter. Using the list memberships of a Twitter user, we can retrieve the name or description of those Twitter lists and concatenate the information to form a *user document*. The retrieved list membership

information replaces the k tweets in Figure 3.2. We discuss this approach in Section 6.6.

3.2 Representation

We now discuss the representation of a user document in terms of a document corpus $D = \{d_1, d_2, \ldots, d_n\}$. Each d_i in the corpus is a Twitter user document as described in Section 3.1.

3.2.1 Vector space model

A popular way to represent documents is the bag-of-words model (Salton and McGill, 1983). In this approach, a document is represented by the terms that appear in it, disregarding the order of appearance of these terms. The terms in a document may occur multiple times throughout the document, and thus the importance of each term can further be estimated by calculating the term frequency. If $T = \{t_1, t_2, \ldots, t_m\}$ is the set of distinct terms in the corpus, a document $d \in D$ will then be represented by an m-dimensional vector $t_d = (\mathrm{tf}(d, t_1), \mathrm{tf}(d, t_2), \ldots, \mathrm{tf}(d, t_m))$, where $\mathrm{tf}(d, t)$ is the number of occurrences of t in document d.

Representing a document using term frequencies effectively reduces the dimensionality of the document. This representation implicitly assumes that terms that occur frequently are more important, which is not always the case. For example, if the terms "a" and "the" frequently appear in a document they will be considered important because of their high frequency count. The terms "a" and "the" do not generally convey any discriminatory information about a document, due to their frequent occurrence in other documents in the corpus.

To counteract this, the popular TF-IDF weighting scheme is used (Hong and Davison, 2010; Rangrej *et al.*, 2011; Huang, 2008). The TF-IDF scheme assumes that terms which appear frequently in a small number of documents, but relatively infrequently in others, tend to be much more relevant for discriminating between documents in the corpus.

To calculate the TF-IDF score, the term frequency (TF) is adjusted based on the inverse document frequency (IDF):

$$\mathrm{tfidf}(d, t) = \mathrm{tf}(d, t) \times \log\left(\frac{|D|}{\mathrm{df}(t)}\right). \tag{3.2.1}$$

Here $\mathrm{df}(t)$ is the number of documents in which term t appears. The inverse document frequency, $\log\left(\frac{|D|}{\mathrm{df}(t)}\right)$, serves as a weighting factor applied to the term frequency, $\mathrm{tf}(d,t)$. As a result, a term occurring in few documents in the corpus but with a high frequency in those documents will have a higher relative importance in the corpus.

In this study, we use the TF-IDF as one document representation.

3.2.2 Topic models

The other representation we use is a topic model. A topic model (Blei and Lafferty, 2009) represents a document as a vector of topic proportions, with each topic modelled as a distribution over terms. It is natural to assume that a document collection exhibits a variety of topics, because a collection tends to be heterogeneous with a number of central ideas and themes.

The set of topics derived from a set of documents D can be used to answer questions about the similarity of terms and documents: two terms are similar to the extent that they appear in the same topic, and two documents are similar to the extent that the same topics appear in those documents.

Latent Dirichlet allocation (LDA) is a generative probabilistic model, which represents documents as random mixtures over latent topics (Blei *et al.*, 2003). A topic is defined as a distribution over a fixed vocabulary of terms. Each document in a collection then consists of different proportions of a set of Q topics.

LDA is related to the well-known probabilistic latent semantic analysis (pLSA) (Hofmann, 2001) technique with the addition of a Dirichlet prior. Blei and Lafferty (2009) notes that pLSA is incomplete in that it provides no probabilistic model at the level of documents. pLSA represents each document as a list of numbers, the topic proportions, and there is no generative probabilistic model for these numbers. In LDA, a document's topic distribution is assumed to have a Dirichlet prior.

The generative model for LDA is shown in Figure 3.3 using plate notation (Blei *et al.*, 2003) for graphical models. Each topic q is represented by a multinomial distribution with parameter vector β_q over the terms in the vocabulary. All these β_q are governed by a Dirichlet prior with parameter η. A document d is represented by a multinomial distribution, with parameter vector, θ_d, over topics. These parameter vectors (θ_d) are governed by a Dirichlet

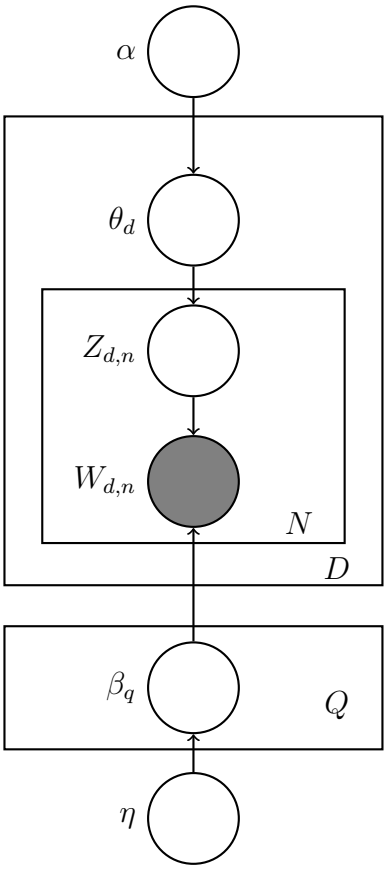


Figure 3.3: A graphical model representation of latent Dirichlet allocation (LDA). Nodes denote random variables; edges denote dependence between random variables. Shaded nodes are observed random variables; unshaded nodes denote latent random variables. The rectangular boxes are "plate notation", which denote replication.

prior with parameter α. Furthermore, each of the N terms in each document is generated by sampling a latent topic, $Z_{d,n}$, from the document's multinomial distribution (θ_d), and then the term $W_{d,n}$ is drawn from topic's multinomial distribution ($\beta_{Z_{d,n}}$).

As stated previously, each document d_i is represented by the parameter vector of a multinomial distribution over topics. We can view this representation as a vector of topic proportions present in the document. The similarity between documents d_i and d_j can thus be calculated by using the topic proportions in the documents.

In this thesis, we use the LDA library provided by Hoffman *et al.* (2010).

3.3 Calculating document similarities

In this section, we highlight the similarity measures selected for this study on the basis of previous work in the field. Let $D = \{d_1, d_2, \ldots, d_n\}$ be the document corpus and let $V = \{v_1, v_2, \ldots, v_m\}$ be the vocabulary of D. Each document d_i is represented by a vector of term scores (when processed with TF-IDF) or topic proportions (when processed with LDA).

3.3.1 Euclidean distance

The Euclidean distance between documents d_i and d_j is calculated as

$$\delta(d_i, d_j) = \sqrt{\sum_{k=1}^{m}(d_{ik} - d_{jk})^2}, \tag{3.3.1}$$

where m is the length of the document vector. It has a lower bound of 0 and is unbounded from above. The Euclidean distance has of course been used in a wide variety of studies. Notable studies in the clustering domain that evaluate the Euclidean distance as a document similarity measure are Huang (2008) and Sandhya *et al.* (2008).

3.3.2 Cosine similarity

The cosine similarity has been applied to document clustering (Subhashini and Kumar, 2010), short text clustering (Rangrej *et al.*, 2011) and user recommendations (Pennacchiotti and Gurumurthy, 2011). The cosine similarity represents the angle between two given vectors. If the value is 0, the two vectors are orthogonal. If the value is 1, the two vectors have the same direction. It is a bounded similarity measure with a lower bound of 0 and upper bound of 1. The cosine similarity between document vectors d_i and d_j is

$$\cos(d_i, d_j) = \frac{\langle d_i, d_j \rangle}{||d_i|| \cdot ||d_j||}. \tag{3.3.2}$$

3.3.3 Pearson correlation coefficient

The Pearson correlation coefficient has been applied to a variety of domains, including document recommendations (Zheng *et al.*, 2011) and document clustering (Torres *et al.*, 2009). It measures the strength and direction of

the linear relationship between two variables. The value is in the range of -1 to 1, where 0 is no relation, 1 is the strongest positive correlation and -1 is the strongest negative correlation. The Pearson correlation coefficient for documents d_i and d_j is

$$r = \frac{\text{cov}(d_i, d_j)}{\sigma_{d_i} \cdot \sigma_{d_j}}. \tag{3.3.3}$$

We calculate the Pearson correlation coefficient as

$$r = \frac{\sum_{k=1}^{m} d_{ik} d_{jk} - \frac{\sum_{k=1}^{m} d_{ik} \sum_{k=1}^{m} d_{jk}}{m}}{\sqrt{\sum_{k=1}^{m} d_{ik}^2 - \frac{\left(\sum_{k=1}^{m} d_{ik}\right)^2}{m}} \sqrt{\sum_{k=1}^{m} d_{jk}^2 - \frac{\left(\sum_{k=1}^{m} d_{jk}\right)^2}{m}}}. \tag{3.3.4}$$

The Pearson's distance (Fulekar, 2009) is defined as $\delta = 1 - r$, which bounds this distance in the $[0, 2]$ range. Huang (2008) extended this to

$$\delta = \begin{cases} 1 - r & \text{if } r \geq 0 \\ -r & \text{if } r < 0, \end{cases} \tag{3.3.5}$$

which has a lower bound of 0 and a upper bound of 1. We use this formulation because it performs well in the document clustering domain (Huang, 2008).

3.3.4 Averaged Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence is used to evaluate the difference between two probability distributions. Given two discrete distributions P and Q, the KL divergence from distribution P to distribution Q is defined as

$$\delta_{KL}(P||Q) = \sum P(i) \log \frac{P(i)}{Q(i)}. \tag{3.3.6}$$

For LDA, the document vectors represent discrete distributions. For TF-IDF, one can normalise the document vectors to obtain a distribution. One can then write the divergence of one document from another as

$$\delta_{KL}(d_i||d_j) = \sum_{k=1}^{m} \left[d_{ik} \times \log \frac{d_{ik}}{d_{jk}} \right], \tag{3.3.7}$$

where d_{ik} and d_{jk} are the values of the k'th component in documents i and j respectively. The KL divergence does not satisfy the symmetry requirement of a true metric, since in general $\delta_{KL}(P||Q) \neq \delta_{KL}(Q||P)$. The KL divergence has mostly been used in its symmetric form for topic models (Landauer, 2007),

and averaged form for document clustering (Huang, 2008). The averaged form is

$$\delta_{AvgKL}(d_i||d_j) = \sum_{k=1}^{m} \pi_1(k) \times \left(d_{ik} \times \log \frac{d_{ik}}{w_k} \right) + \pi_2(k) \times \left(d_{jk} \times \log \frac{d_{jk}}{w_k} \right), \quad (3.3.8)$$

where $\pi_1(k) = \frac{d_{ik}}{d_{ik}+d_{jk}}$, $\pi_2(k) = \frac{d_{jk}}{d_{ik}+d_{jk}}$ and $w_k = \pi_1(k)d_{ik} + \pi_2(k)d_{jk}$. Furthermore, we let $0 \times \log 0 = 0$.

The Kullback-Leibler divergence from P to Q is finite when the support of Q is contained in the support of P. For LDA, a document's topic distribution is governed by a Dirichlet prior and thus all topics are present with non-zero proportions for both P and Q. However, TF-IDF vectors may not share the same set of terms. Using the averaged form the contribution of those terms present in one document but not the other to the calculated distance is 0.

In the special case when two documents do not share any terms, the average KL divergence evaluates to 0. In this case, we modify the average KL divergence to set the distance to a large value. This large value indicates that the two documents are dissimilar, since the average KL divergence has a lower bound of 0 when two distributions are the same. The value is chosen as equal to the largest distance, found with the average KL divergence, between documents in the set.[2]

3.3.5 Extended Jaccard coefficient

The Jaccard coefficient measures the similarity of two sets by computing the ratio of the number of shared elements of the two sets to the total number of elements (Ye, 2004). Strehl *et al.* (2000) extended this definition to vectors with real-valued components. The extended Jaccard coefficient is defined as

$$\mathrm{EJ}(d_i, d_j) = \frac{\langle d_i, d_j \rangle}{||d_i||^2 + ||d_j||^2 - \langle d_i, d_j \rangle}, \quad (3.3.9)$$

[2]In retrospect, this choice of using the TF-IDF in combination with the average Kullback-Leibler divergence is poor. Bigi (2003) proposes a model in which terms from a vocabulary that do not appear in a document are given an epsilon probability. This assignment of probabilities to terms present in the vocabulary, but not in the document, seems to be a better way to deal with situations leading to infinite Kullback-Leibler divergence.

which reduces to the Jaccard coefficient when sets are represented as binary vectors. This similarity measure has a lower bound of 0 and an upper bound of 1. It is 0 when the two document vectors are orthogonal, and 1 when they are identical.

3.4 Clustering documents

This section describes two clustering algorithms we investigated, namely k-means and affinity propagation. Both of these are partitional clustering algorithms; objects are thus categorised into a number of non-overlapping subsets. The categorisation occurs around cluster prototypes such that each object is in some sense more similar to its cluster prototype than the other cluster prototypes (Tan *et al.*, 2006).

3.4.1 k-means

The k-means algorithm (Tan *et al.*, 2006) is initialised with k user-selected initial points. By assigning each data point to its nearest initial point, k clusters are formed. For each of the k clusters, the centroid of the cluster points is computed, which replaces the k user-selected initial points. In k-means, the centroids are the cluster prototypes. The process of assigning each point to its nearest centroid is repeated until the centroid of each clusters stays the same for consecutive iterations or some maximum number of iterations is reached.

An important consideration for k-means is how the initial starting points are selected. We opted for the k-means++ method (Arthur and Vassilvitskii, 2007), which has been shown to perform much better than the random seeding of k-means. In k-means++, the first centre is sampled uniformly at random from the data points after which new centres are repeatedly chosen with probability proportional to the distance from the nearest existing centre.

3.4.2 Affinity propagation

AP (Frey and Dueck, 2007) treats all data points as potential exemplars, exchanging suitability messages until the most suitable exemplars are found

and clusters are formed. The resulting exemplars are the medoids[3] of the generated clusters, and serve as cluster prototypes.

The AP procedure takes as input an $N \times N$ matrix S of document similarities and exchanges two types of "messages" between data points: *responsibilities* and *availabilities*. A responsibility, $r(i,k)$, is sent from data point i to a candidate exemplar k and reflects the evidence that k is suitable as an exemplar for data point i. An availability, $a(i,k)$, is sent from a candidate exemplar k to data point i and is the evidence indicating how appropriate it is for data point i to choose k as its exemplar.

The responsibilities are computed as

$$r(i,k) \leftarrow S_{ik} - \max_{k' \text{ s.t. } k' \neq k} \{a(i,k') + S_{ik'}\}, \tag{3.4.1}$$

where S_{ik} is the similarity of data point i and j. The availabilities are calculated as

$$a(i,k) \leftarrow \min \Big\{0, r(k,k) + \sum_{i' \text{ s.t. } i' \notin \{i,k\}} \max\{0, r(i',k)\}\Big\}, \tag{3.4.2}$$

for $i \neq k$. The self-availability $a(k,k)$ is calculated as

$$a(k,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, r(i',k)\}, \tag{3.4.3}$$

and reflects the accumulated evidence that data point k is an exemplar. This evidence is thus based on the positive responsibilities sent to candidate exemplar k from other data points.

The responsibilities and availabilities can be combined in order to identify the exemplars. The value of k that maximises $a(i,k) + r(i,k)$ identifies the data point that is the exemplar for point i.

AP allows us to set a self-preference value, p_i, for each data point. A data point with a higher self-preference value has a greater likelihood to be selected as a cluster exemplar. If the self-preference value is set equal to the same value, p, for all the data points, the number of clusters can be influenced by p. A large p in relation to the magnitude of the similarities will lead to many clusters while a smaller p will lead to fewer.

The algorithm described above can be terminated either after a fixed number of iterations, when changes to the messages fall below a threshold, or if the

[3]The medoid of a group of points is a multi-dimension generalisation of the median.

local decisions stay constant for some number of iterations. In order to avoid numerical oscillations that can arise in some circumstances, each message can be set to λ times its value from the previous iteration plus $1 - \lambda$ times its prescribed update value (Frey and Dueck, 2007), as described in Equation 3.4.1 – 3.4.3. The damping factor λ is usually set between 0.5 and 1.

3.5 Conclusion

In this chapter, we introduced our document clustering approach, which consists of creating a Twitter user document, converting to a vector representation and applying clustering algorithms using an underlying similarity measure.

We represented Twitter users by defining a user document as a concatenation of a Twitter user's tweets. We discussed two representation techniques, TF-IDF and LDA, which transform the user document to a suitable format for use with various similarity measures. The measures used to calculate the document similarities are Euclidean distance, cosine similarity, Pearson correlation coefficient, averaged Kullback-Leibler divergence and extended Jaccard coefficient. Finally, we introduced the k-means and AP clustering algorithms, which will be applied to the document similarities.

Chapter

4

Document classification with kLog

In Chapter 2, we discussed the different user content available on Twitter that can be used to represent a Twitter user. We indicated that in the current approaches it is simple to represent users by their tweet content, but difficult to include their social graph structure in that representation. A solution to this problem lies in statistical relational learning (SRL), where it is easier to incorporate the explicit social graph structure and tweet content to represent a user. This chapter introduces kLog (Frasconi *et al.*, 2012), a supervised learning system from the SRL field. kLog is a logical and relational language for kernel-based learning, which we will later use to construct topic-based Twitter lists.

This chapter is structured as follows: We discuss the importance of social graph information in Section 4.1. In Section 4.2, we introduce each component of the kLog system in context of a small example problem; the example problem is that of predicting the category of a document. In Section 4.3, we discuss the formulation of our interpretations, which is followed (Section 4.4) by a discussion of the kLog models we will use to train a classifier for constructing topic-based Twitter lists.

4.1 Social graph

Chapter 3 sets out our approach to clustering user documents constructed from only the users' tweet content. Twitter provides a large amount of extra information, such as friends, followers and list memberships. This extra information details connections between users and can thus be used to construct a social graph for each user. It is natural to ask whether we can extract features from this social graph information to better differentiate users when forming topic-based clusters or classifying users into topic-based lists.

A Twitter user's social graph provides valuable information, but the process of incorporating this information for use in a clustering or classification task is non-trivial. We use the kLog system, which readily provides us with tools to incorporate graph structure as well as tweet content to construct a feature vector for each user. The next section introduces and discusses each component of the kLog system.

4.2 kLog

An SRL system uses a model of some complex relational structure when performing inference to obtain answers to questions posed to the system. Frasconi *et al.* (2012) define such a SRL system, kLog, as a kernel-based approach to learning that employs features derived from a grounded entity-relationship (E-R) diagram.

Figure 4.1, shows the important parts of the kLog system. As input, the kLog system receives a problem with a corresponding data set, which consists of information with complex relational structure. This data set is described with a set of *signatures*, which captures the logical and relation structure. These signatures correspond to an E-R diagram that shows the entities and possible relationships between those entities that exist in the data.

Next, the information in the data set is converted into a graph format in a manner governed by signatures specified in the E-R diagram. kLog refers to this step as *graphicalisation*. Features are extracted from the resulting graphicalisations by defining a *graph kernel*. These features are then used as input to a statistical learning algorithm, which trains a classifier that can be used to answer the original query.

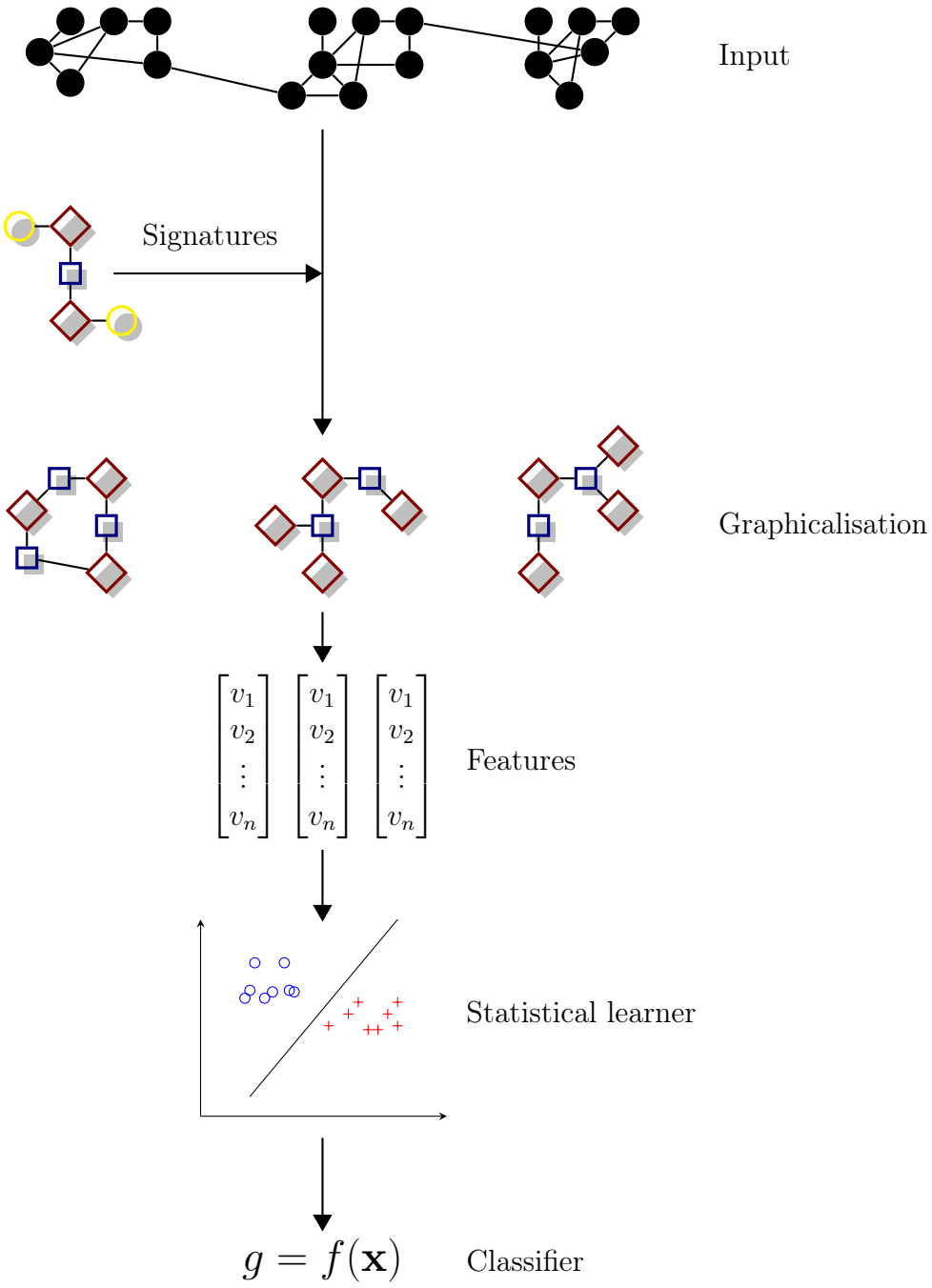


Figure 4.1: A graphical depiction of the important parts in the kLog system.

A kLog program is written in the Prolog programming language. Prolog is a logical programming language whose program logic is represented with rules and facts that are defined by clauses. In Prolog, a rule clause is specified as `HEAD :- BODY.` and states that `HEAD` is true if `BODY` is true. Furthermore, a fact is a clause without a `BODY`.

Code Listing 4.1, shows a Prolog script that defines a fact on the first line followed by a rule.

```
man(Tom).
human(X) :- man(X).
```

Code Listing 4.1: An example Prolog script that illustrates a rule and a fact.

If we load this script in Prolog, we can perform a query. For example, asking if `Tom` is a `human`, with `?- human(Tom)`, will return a true result.

kLog extends Prolog with a domain declaration and set of keywords. The domain declaration consists of the keywords `begin_domain` and `end_domain` as well as one or more signature declarations. A signature consists of a header followed by zero or more clauses.

For a concrete definition and specification of the kLog system as well as possible extensions, we refer the reader to Frasconi *et al.* (2012). In the following subsections, we discuss the important components of the kLog system in the context of a document classification task. In this task, a document is an entity that contains words, and is associated with one of several categories.

4.2.1 Relational model

The signatures that we briefly discussed in the previous section are similar to the conceptual E-R data model. This is the highest level E-R model that describes the data set with the least detail.

Chen (1976) describes an E-R diagram as consisting of three elements, namely *entities*, *relationships* and *properties*. An entity is an object, which can be distinctly identified. Furthermore, a relationship is an association between entities. Properties are assigned to entities or relationships and represent the information that all entities or relationships of the same type have in common.

kLog relaxes the definition of a relationship to a more general association among entities and properties. This general association is referred to as a

relation. kLog defines two types of relations, E-relations and R-relations: E-relations are similar to entities while R-relations are similar to relationships.

To address the task of predicting the category to which a document belongs, it is necessary to define the task as a logical and relational problem. In this example task, we assume that there is a training data set of multiple documents.

The signatures and E-R diagram that describe the data model for this task are shown in Code Listing 4.2 and Figure 4.2 respectively.

```
begin_domain
  signature document(doc_id::self)::extensional.
  signature category(doc_id::document,
                     cat::property)::extensional.
  signature has(doc_id::document,
                word::property)::extensional.
  signature link(doc_id_one::document,
                 doc_id_two::document)::intensional.
  link(D1, D2) :-
    category(D1, _C), category(D2, _C), not(D1=D2).
end_domain
```

Code Listing 4.2: An example of `extensional` and `intensional` signatures that models the toy data set.

The documents in the data set are each represented by a `Document` entity, which is associated with a `category` and `has` relationship. A `category` relationship has a `cat` property, which is the name of the `category`. Furthermore, the `has` relationship has a `word` property, which indicates the word in the document that the property represents. In Code Listing 4.2 the `link` signature is `intensional`, and uses Prolog rule clauses to define a `link` relationship between two documents with similar categories. The `extensional` signatures expect input in the form of kLog *interpretations*.

To describe an interpretation, it is necessary to introduce concepts from first-order logic (Getoor and Taskar, 2007). In first-order logic a ground term is a term that does not contain any free variables. Furthermore, a ground atom is an atomic formula whose argument terms are all ground terms. An interpretation is specified by a set of ground atoms that are true and all atoms not in the interpretation are assumed to be false.

Frasconi *et al.* (2012) describe a kLog interpretation in database terminology

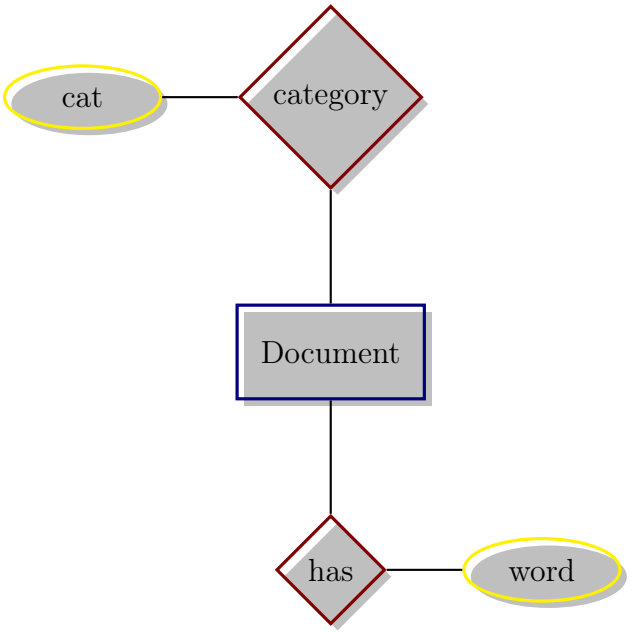


Figure 4.2: The E-R diagram that models the structure in the toy data set.

as corresponding to one instance of a relational database describing one possible world. kLog learns from multiple interpretations.

Let us specify the data set, for our example task, as consisting of three documents. These documents are shown in Table 4.1.

Identifier	Category	Words
doc1	music	acoustic, piano, song
doc2	entertain	sound, bass, song
doc3	music	piano, show, disc

Table 4.1: The documents in the toy data set.

We define these three documents as part of the `documents` interpretation. The resulting interpretation is shown in Code Listing 4.3. This interpretation corresponds to the extensional signatures, which we defined in Code Listing 4.2.

```
interpretation(documents, document(doc1)).
interpretation(documents, category(doc1, music)).
interpretation(documents, has(doc1, acoustic)).
interpretation(documents, has(doc1, piano)).
interpretation(documents, has(doc1, song)).
interpretation(documents, document(doc2)).
```

```
interpretation(documents, category(doc2, entertain)).
interpretation(documents, has(doc2, sound)).
interpretation(documents, has(doc2, bass)).
interpretation(documents, has(doc2, song)).
interpretation(documents, document(doc3)).
interpretation(documents, category(doc3, music)).
interpretation(documents, has(doc3, piano)).
interpretation(documents, has(doc3, show)).
interpretation(documents, has(doc3, song)).
```

Code Listing 4.3: An interpretation defined for three documents in the example data set.

In this subsection, we introduced the kLog relation model. We described signatures, and how they relate to a conceptual E-R data model. Furthermore, we described the logical and relational structure of an example data set with a set of signatures and showed the corresponding E-R diagram. We discussed interpretations in kLog, and defined a `documents` interpretation.

In the next subsection, we use the interpretation in Code Listing 4.3 and the signatures in Code Listing 4.2 to perform *graphicalisation*.

4.2.2 Graphicalisation

Formally, the graphicalisation process proceeds as follows (Frasconi *et al.*, 2012). Given an interpretation z, construct a bipartite graph $G_z([V_z, F_z], E_z)$ with vertex set $[V_z, F_z]$ and edge set E_z: each vertex in V_z corresponds to a ground atom of an E-relation and each vertex in F_z corresponds to a ground atom of a R-relation. Vertices are labelled by the name of the ground atom, followed by the list of properties. The identifiers in a ground atom do not appear in the label but they identify the vertices.

An edge between vertex u and v is in the edge set ($uv \in E_z$) if and only if $u \in V_z$ and $v \in F_z$, and $\text{ids}(u) \subset \text{ids}(v)$. The $\text{ids}(u)$ and $\text{ids}(v)$ are the identifiers in the E-relation represented by vertex u and R-relation represented by vertex v respectively.

Figure 4.3 shows the graphicalisation of the interpretation in Code Listing 4.3. This graphicalisation consists of three documents, of which two are connected by a pair of `link` vertices. These `link` vertices are created by the `intensional` signature in Code Listing 4.2.

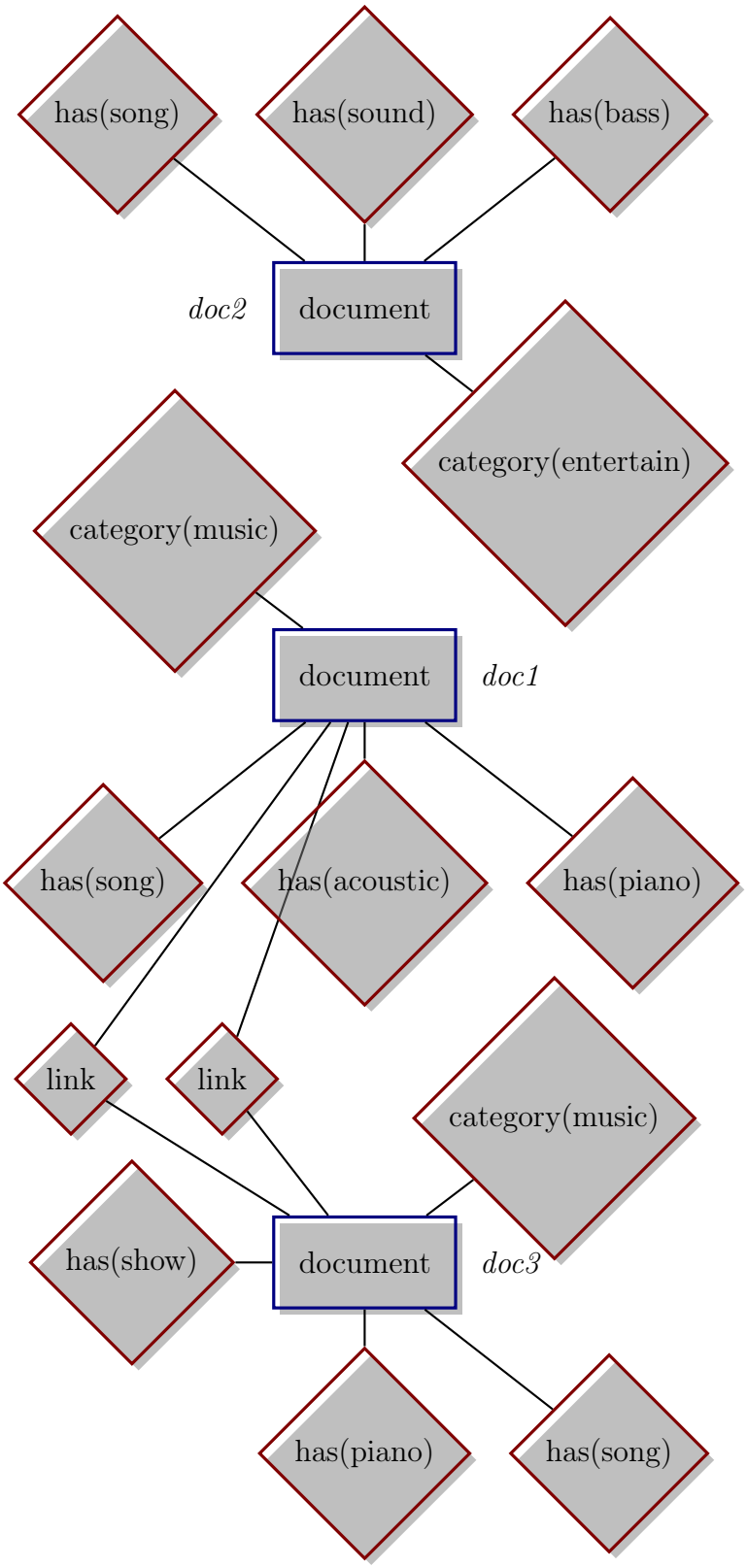


Figure 4.3: The graphicalisation of the example interpretation.

4.2.3 Graph kernels

The previous section discussed graphicalisation, the mapping of an interpretation z into an undirected labelled graph G_z. In this section the application of graph kernels to graphicalisations is discussed.

Features in kLog are generated by means of a graph kernel that calculates the similarity between two graphicalised interpretations. kLog uses an extension of the Neighbourhood Subgraph Pairwise Distance Kernel (NSPDK) (Costa and De Grave, 2010). The NSPDK is a decomposition kernel, which decomposes a graph into all pairs of neighbourhood subgraphs of small radii at increasing distances from each other (Costa and De Grave, 2010).

Extraction of subgraphs from a graph G for comparison is governed by three parameters: the set of *kernel points*, a *radius* (r) and a *distance* (d). Kernel points are entities or relationships on which the subgraphs used for comparisons are centered. They are also referred to as the root vertices of these subgraphs. The radius r describes the size of the subgraphs, by specifying which entities or relationships around a kernel point should be included in the subgraphs. Finally, the distance d determines how far apart from each other the kernel points are.

Let U be the set of kernel points in graph G. Furthermore, denote the vertex set of G as $V(G)$. Let $N_r^u(G)$ be a neighbourhood subgraph in G, rooted on vertex $u \in U$ such that $d_G^\star(u,x) \leq r$ where $x \in V(G)$. The distance $d_G^\star(u,x)$ is the length of the shortest path between u and x. We define the neighbourhood-pair relation as $R_{r,d}(G) = \{(N_r^u(G), N_r^v(G)) : d_G^\star(u,v) = d\}$. This relation identifies all pairs of r-radius neighbourhood subgraphs whose roots $u, v \in U$ are at distance d from each other in a given graph G.

The kernel $\kappa_{r,d}(G,G')$ between graphs G and G' is then

$$\kappa_{r,d}(G,G') = \sum_{\substack{(A_u,B_v)\in R_{r,d}(G)\\(A'_{u'},B'_{v'})\in R_{r,d}(G')}} \kappa\left((A_u,B_v),(A'_{u'},B'_{v'})\right), \tag{4.2.1}$$

where the general structure of κ is

$$\begin{aligned}\kappa\left((A_u,B_v),(A'_{u'},B'_{v'})\right) = \kappa_{\text{root}}\left((A_u,B_v),(A'_{u'},B'_{v'})\right) \times \\ \kappa_{\text{subgraph}}\left((A_u,B_v),(A'_{u'},B'_{v'})\right).\end{aligned} \tag{4.2.2}$$

Here κ_{root} ensures that only neighbourhood subgraphs centered on the same type of vertex pairs will be compared. The second factor, κ_{subgraph}, quantifies

the similarity between two pairs of subgraphs. We discuss κ_{subgraph} later in this section in the context of our running example problem.

The comparison performed by κ_{root} is

$$\kappa_{\text{root}}\left((A_u, B_v), (A'_{u'}, B'_{v'})\right) = \mathbf{1}_{l(u)=l(u')}\mathbf{1}_{l(v)=l(v')}, \tag{4.2.3}$$

where $l(u)$ is the label of the root vertex of subgraph A and $\mathbf{1}$ denotes the indicator function. The indicator function is defined as

$$\mathbf{1}_P = \begin{cases} 1 & P \text{ is true} \\ 0 & \text{otherwise.} \end{cases} \tag{4.2.4}$$

If we assume that κ_{subgraph} is a valid kernel, the NSPDK is defined as

$$K_{r^*,d^*}(G, G') = \sum_{r=0}^{r^*} \sum_{d=0}^{d^*} \kappa_{r,d}(G, G'), \tag{4.2.5}$$

where r^* and d^* are user-specified upper bounds on the radius and distance parameters respectively.

We now return to our example problem to discuss the application of the NSPDK to graphicalisations. Figure 4.3 shows the graphicalisation of the interpretation in Code Listing 4.3. To effectively illustrate the NSPDK, we simplify this interpretation to consist of only one document. Furthermore, it is necessary to create another interpretation because the kernel is applied to two graphicalisations.

Figure 4.4 represents the result of graphicalising two interpretations, both consisting of only a single document. The graphs, G and G', each consists of a single `document` entity, a `category` relation and multiple `has` relations. The `document` entity in each graph is the kernel point. The task is to predict the property of a category vertex. Therefore, these vertices are removed from the graphicalisations. We indicate this in Figure 4.4 by representing those vertices with dotted borders.

When examining the graphs in Figure 4.4, we see that the only sensible radius r and distance d values are $r \in \{0, 1\}$ and $d = 0$ respectively. A distance $d > 0$ between entities will not exist, since each graph consists of only one entity. Furthermore, there are no vertices at $r > 1$ from the kernel point of each graph.

From Equation 4.2.5, we thus use $r^* = 1$ and $d^* = 0$ to obtain

$$K_{r^*,d^*}(G, G') = \kappa_{0,0}(G, G') + \kappa_{1,0}(G, G'). \tag{4.2.6}$$

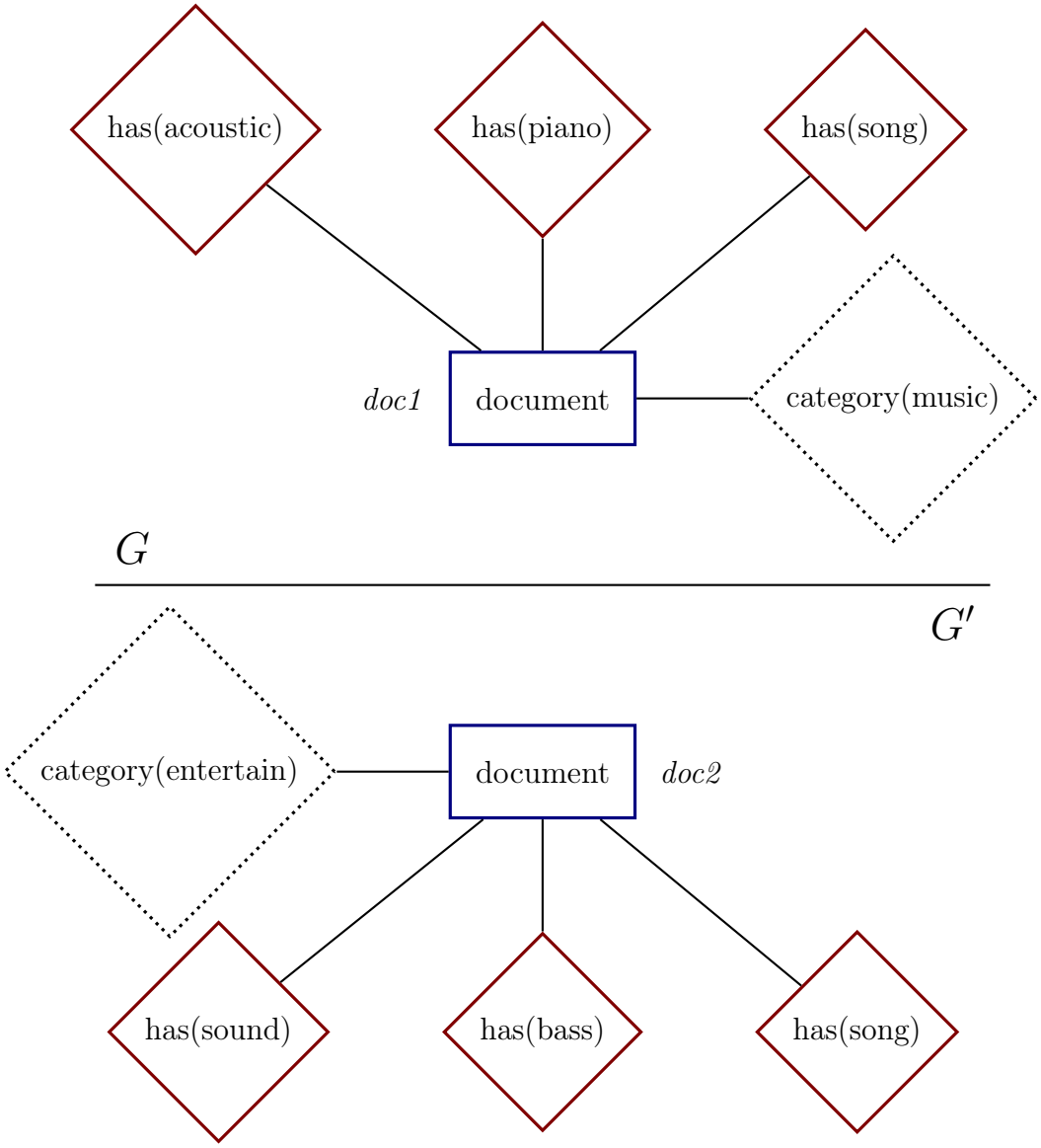


Figure 4.4: Two graphicalisations generated from two interpretations, each containing a single document.

We will consider the terms in Equation 4.2.6 separately. To calculate $\kappa_{0,0}(G, G')$ and $\kappa_{1,0}(G, G')$ from Equation 4.2.1, it is necessary to define the relations $R_{0,0}$ and $R_{1,0}$ for each graph interpretation.

The neighbourhood subgraph pairs for graph G and G' are shown in Figure 4.5. An interesting result, which is due to $d = 0$, is that the two subgraphs in a neighbourhood pair are identical. Figure 4.5 shows the neighbourhood subgraph pairs for relations $R_{0,0}$ and $R_{1,0}$ with the grey shaded circles and the grey shaded rectangles respectively.

Finally, with the relations defined for each combination of r and d, it is possible to evaluate κ_{subgraph} as used in Equation 4.2.2. One implementation of κ_{subgraph}, is testing for an exact match between the subgraphs. This is a graph isomorphism problem, which can be formalised as

$$\kappa_{\text{subgraph}}\left((A_u, B_v), (A'_{u'}, B'_{v'})\right) = \mathbf{1}_{A_u \cong A'_{u'}} \mathbf{1}_{B_v \cong B'_{v'}}, \tag{4.2.7}$$

where $\cong$ indicates graph isomorphism.

Applying Equation 4.2.7 in our example, we see that $\kappa_{0,0}(G, G')$ evaluates to 1 since both graphs have a subgraph rooted on the `document` entity without any connected vertices. Based on the vertex labels, it is clear for $\kappa_{1,0}(G, G')$ that the subgraphs in G and G' are not isomorphic and as a result $\kappa_{1,0}(G, G')$ evaluates to 0. Substituting[1] these values into Equation 4.2.5 yields

$$\begin{aligned} K_{1,0}(G, G') &= \kappa_{0,0}(G, G') + \kappa_{1,0}(G, G') \\ &= 1 + 0 \\ &= 1. \end{aligned} \tag{4.2.8}$$

In some cases it is valuable to calculate the kernel between two graphs based on partial rather than exact matches between the properties of vertex labels. Recall that in Subsection 4.2.2 we discussed the labelling of a vertex. We formalise this labelling as $r(c_1, c_2, \ldots, c_m)$, where r is the name of the ground atom/signature and c_i for $i = \{1, \ldots, m\}$ are the list of its properties.

The soft match for kernel $\kappa_{r,d}(G, G')$ is defined as

$$\kappa_{\text{subgraph}}((A_u, B_v), (A'_{u'}, B'_{v'})) = \sum_{\substack{z \in V(A_u) \cup V(B_v) \\ z' \in V(A'_{u'}) \cup V(B'_{v'})}} \mathbf{1}_{l(z)=l(z')} \kappa_{\text{tuple}}(z, z'), \tag{4.2.9}$$

[1] We do not show κ_{root}, because the set of kernel points consists of only one type of vertex. Therefore, κ_{root} will in this case always be true.

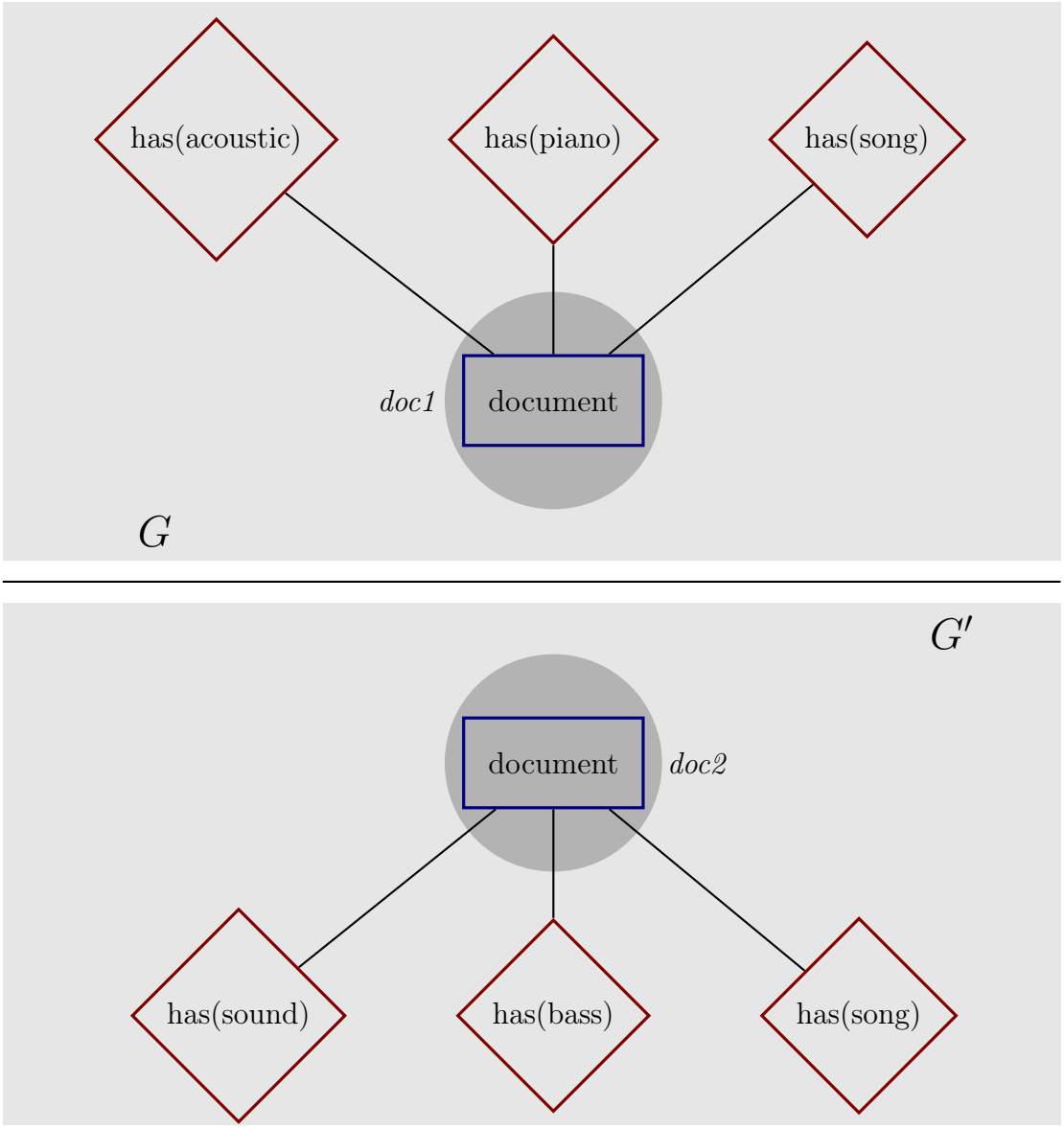


Figure 4.5: The graphicalisations of two interpretations, where the two subgraphs are indicated by the shaded circle and shaded rectangle respectively.

where $l(z)$ returns the signature name for a vertex z. Now the problem is defined by a soft match between vertex properties, such that

$$\kappa_{\text{tuple}}(z, z') = \begin{cases} \sum_d \mathbf{1}_{\text{prop}_d(z)=\text{prop}_d(z')} & \text{if } d > 0 \\ 1 & \text{otherwise,} \end{cases} \tag{4.2.10}$$

where $\text{prop}_d(v)$ returns the property value c_d.

Intuitively this states that the soft match of two subgraphs is the total number of matching properties of vertices with the same signature name. In our example $\kappa_{\text{tuple}}(z, z')$ reduces to either 1 or 0 since each vertex z only has one property. Using the soft match kernel, substituting in Equation 4.2.6, yields

$$\begin{aligned} K_{1,0}(G, G') &= \kappa_{0,0}(G, G') + \kappa_{1,0}(G, G') \\ &= 1 + 2 \\ &= 3. \end{aligned} \tag{4.2.11}$$

For the $\kappa_{0,0}(G, G')$ case, the result is the same as the hard match kernel due to matching the two document vertices. For $\kappa_{1,0}(G, G')$ there are 4 possible matches: the vertices to compare are {`document`, `has(acoustic)`, `has(piano)`, `has(song)`} in G and {`document`, `has(sound)`, `has(bass)`, `has(song)`} in G'. We see that two matches exist, namely the `document` vertices and `song` property.

In the first case there is only one possible match and in the second there are four, as a result the second case makes a larger contribution to the kernel. To equally weight each case, the kernel $\kappa_{r,d}$ is normalised as

$$\hat{\kappa}_{r,d}(G, G') = \frac{\kappa_{r,d}(G, G')}{\sqrt{\kappa_{r,d}(G, G)\kappa_{r,d}(G', G')}}. \tag{4.2.12}$$

The normalised NSPDK is then

$$\begin{aligned} \hat{K}_{1,0}(G, G') &= \hat{\kappa}_{0,0}(G, G') + \hat{\kappa}_{1,0}(G, G') \\ &= \frac{1}{\sqrt{1 \times 1}} + \frac{2}{\sqrt{4 \times 4}} \\ &= 1.5. \end{aligned} \tag{4.2.13}$$

This example illustrated the application of the hard match discrete kernel, and the soft match discrete kernel to graphicalisations. These kernels can only be applied to relations with discrete properties. The soft match kernel can be

extended to handle mixtures of discrete and real-valued properties. The soft match kernel for mixed discrete and real tuples is

$$\kappa_{\text{tuple}}(z, z') = \begin{cases} \sum_d \mathbf{1}_{\text{prop}_d(z)=\text{prop}_d(z')} + \sum_c \text{prop}_c(z) \cdot \text{prop}_c(z') & \text{if } d > 0 \text{ or } c > 0 \\ 1 & \text{otherwise,} \end{cases} \tag{4.2.14}$$

where the index d is defined over all the discrete properties and the index c is defined over all the continuous properties of vertex v.

In this section, we discussed the application of the NSPDK on two graphicalisations. Next, we discuss the feature extraction and statistical learning in kLog.

4.2.4 Feature extraction and statistical learner

kLog uses the NSPDK discussed in the previous section to extract feature vectors that are used as input to a statistical learning algorithm. An interpretation $z = (x, y)$ is mapped into a feature vector $\phi(z) = \phi(x, y)$, which allows the application of supervised learning algorithms in a feature space $\mathcal{F}$. In context of the NSPDK this $\phi(z)$ is defined implicitly via the kernel function such that $K(z, z') = \langle \phi(z), \phi(z') \rangle$ (Frasconi *et al.*, 2012).

Therefore, a feature vector $\phi(z)$ is used as input to a supervised learning algorithm. Currently, kLog uses LibSVM (Chang and Lin, 2011) to train a classifier on these feature vectors, which is used to make predictions.

4.3 Interpretations

We introduced kLog in the previous section, and showed how different formulations of an interpretation affects the graphicalisations. For the task of constructing topic-based Twitter lists, we consider two different formulations of an interpretation.

4.3.1 Many formulation

A simple formulation of an interpretation is a single Twitter user. In this case each interpretation consists of only one Twitter user. The graphicalisation of such an interpretation leads to a special graph shape, referred to as a *star*[2].

[2]In graph theory a star S_k is a complete bipartite graph $K_{1,k}$.

A star limits the NSPDK to a radius $r = 1$ and distance $d = 0$. As a result, we expect that the effectiveness of the NSPDK to extract features from such a graph will be diminished. Furthermore, based on this formulation of an interpretation it is not possible to model the explicit links between Twitter users. As such, only a limited amount of the information available for a Twitter user can be included in these models.

4.3.2 Five formulation

A different formulation of an interpretation to consider is the partitioning of a data set into two or more partitions such that a number of Twitter users from each category appear in a partition. Therefore, each partition can be used as an interpretation.

One possible formulation of these new interpretations is a partition of Twitter users into five groups, where each group represents an interpretation. Each group contains a set of Twitter users from each category in a data set, proportional to the number of Twitter users in that category. Even though the interpretations are in a sense forced, it is still a reasonable formulation. For the prediction task it is natural to have representatives of each category to train on.

This formulation allows us to effectively use the information available about Twitter users to model the explicit connections between them.

4.4 Representation

For the automatic construction of topic-based Twitter lists with kLog, it is necessary to define kLog models of Twitter users and their connections. The goal of these models is to represent users on Twitter such that a good classifier for constructing topic-based Twitter lists can be trained from the models. In the following subsections, we introduce a number of kLog models. Each model represents a Twitter user as a *document*.

In the previous section, we described two formulations of an interpretation. In the following models, we use the many interpretations formulation for those models that consist of only one document entity and the five interpretations

formulation for those that model the connections between multiple document entities.

4.4.1 Words

The simplest kLog model that we consider models only the words in a Twitter user's tweets. Figure 4.6 shows the E-R diagram representing a Twitter user as a `Document` entity associated with his/her tweet content.

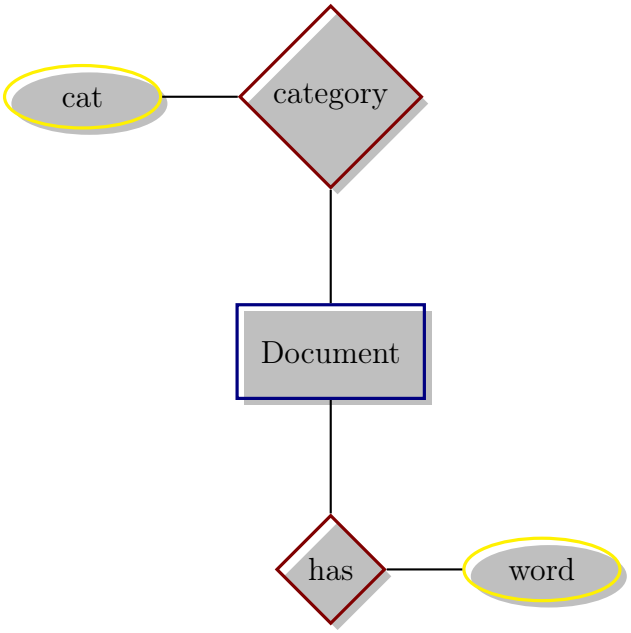


Figure 4.6: The E-R diagram for the words model.

An important aspect to consider is the specification of the relation properties. We consider three options, namely

- multiple occurrences of a word leads to multiple `has` relations;
- multiple occurrences of a word leads to a single `has` relation; and
- including the word's TF-IDF score as a property in the `has` relation[3] .

The first two possibilities can affect the performance, since the soft match kernel counts the frequency of vertices with the same label when generating

[3]It is possible to include the real-valued TF-IDF property, because kLog provides a soft match kernel for discrete and real-valued properties (Equation 4.2.14).

the feature vectors. As a result, words with multiple occurrences will have a higher frequency compared to a word with only one occurrence.

The last option adds the information obtained from the calculated TF-IDF to the system and may add valuable information to the generated feature vectors. This extra property is added to the `has` relation as shown in Figure 4.7.

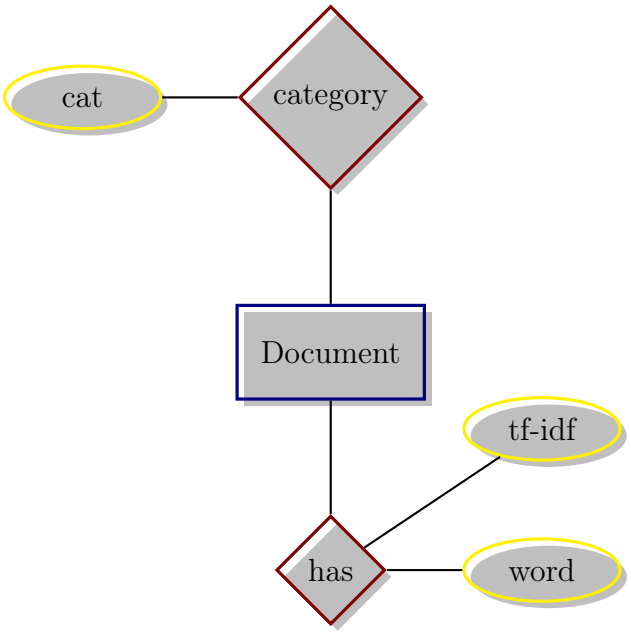


Figure 4.7: The E-R diagram for the words model including the TF-IDF real-valued property in the `has` relation.

We distinguish between these three variants of the words model by referring to them as the *multiple-word model*, the *one-word model* and the *tfidf-word model* respectively.

4.4.2 Topics

We can also define topic models that are similar to the previously defined word models. We have already defined the process of extracting topics from a document using LDA in Chapter 3.

Every document in a corpus exhibits the same set of topics, but with different magnitudes. Therefore, we can define a `has` relation with a `topic` and `value` property as shown in Figure 4.8. Furthermore, it is also possible to define a simple `has` relation with only the `topic` property by only including

the topics for a user document with magnitude greater than a pre-specified threshold as shown in Figure 4.9. This threshold is set for each user document as the mean of its topic proportions.

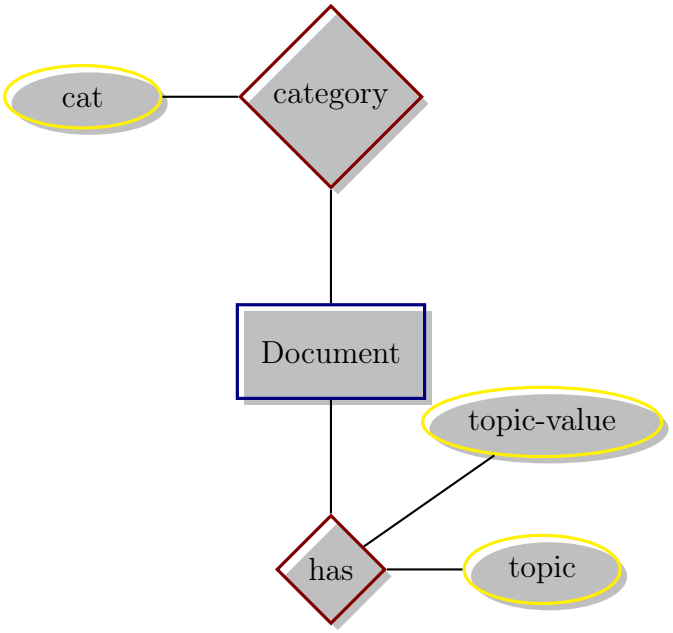


Figure 4.8: The E-R diagram for the topics model using topic proportions as found by LDA.

We refer to these two variants of the topics model as the *lda-topic model* and the *threshold-topic model*.

4.4.3 Social graph information

So far, we introduced kLog models to represent the content of a user document only. However, social networks contain a rich set of extra information that can be used to characterise users according to their topic-based expertise. This rich set of extra information includes the social graph information available for each user.

We consider two sources for this social graph information, namely Twitter user friendship and list membership information. We define a friendship on Twitter when two users follow each other. The list membership information comprises information on which Twitter lists users are members of, excluding the lists we are trying to predict.

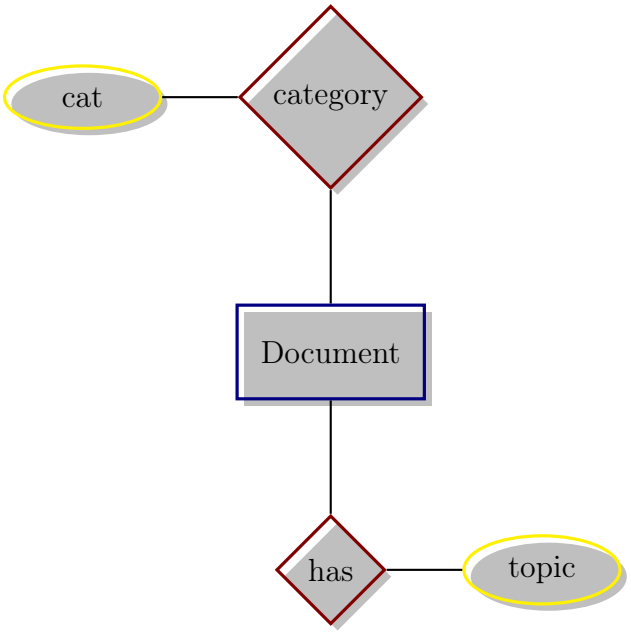


Figure 4.9: The E-R diagram for the topics model.

Figure 4.10 shows the E-R diagram that represents the friendship and list membership information respectively.

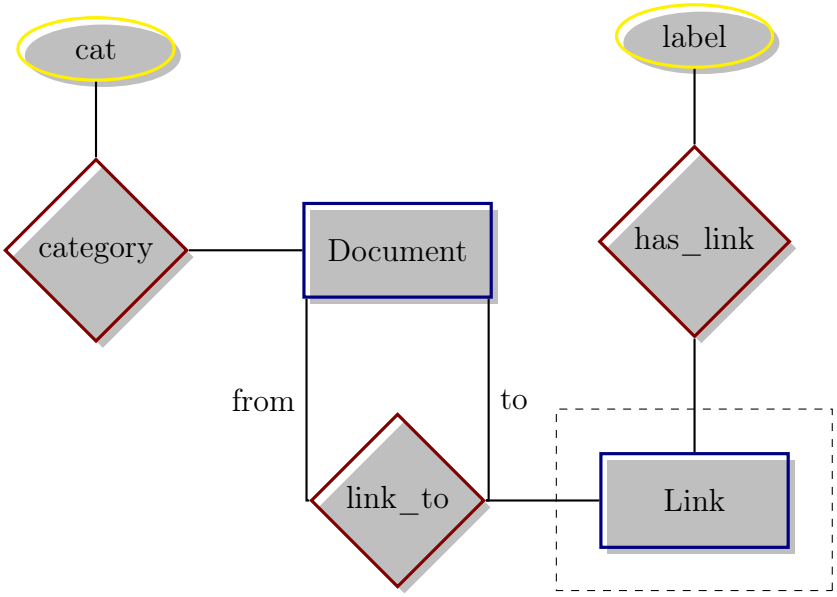


Figure 4.10: The E-R diagram for the social graph model, which introduces the `link_to` and `has_link` relationships.

The E-R diagram illustrates the additions that can be made when infor-

mation is incorporated that connects document entities. We introduce the `link_to` and `has_link` relationship. The `link_to` relationship connects two user documents based on their friendship or list memberships. The diagram also contains a `Link` entity, which is an intensional relationship, created if there exists a `link_to` between two user documents. This entity is shown in the E-R diagram with a dashed rectangle to illustrate the `has_link` relationship. Two document entities are thus connected by a `Link` entity if a corresponding `link_to` extensional signature exists.

The `label` property of the `has_link` relation is the identifier of a user document for friendships, and a list identifier for the list memberships. This is the first model that we use with the five interpretations formulation, we refer to these type of models as *complex*.

A different model, similar to the word and topic models, is shown in Figure 4.11. This model removes the `link_to` relation and `Link` entity, and connects the `has_link` relation directly to a document. This differs from the previous model by not explicitly modelling the connections between user documents. To differentiate from the model in Figure 4.10, we refer to these type of models as *simple*.

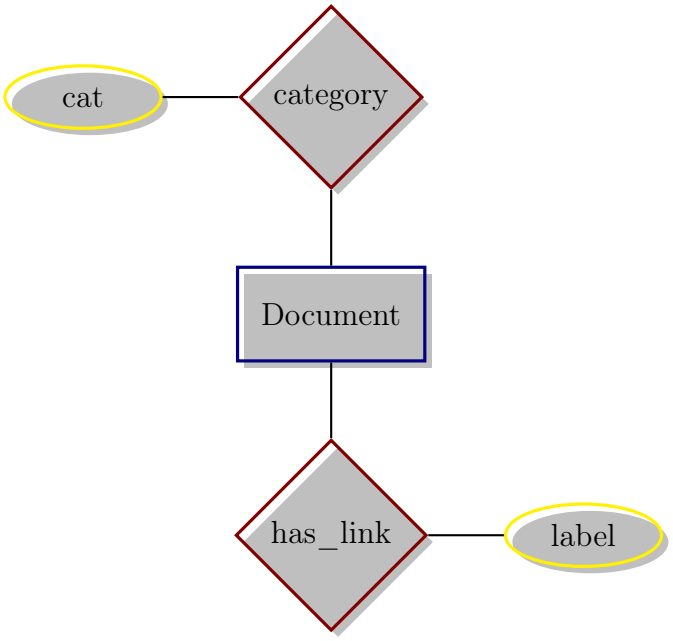


Figure 4.11: The E-R diagram for the social graph model that discards the link structure and uses the labels as attributes.

Based on the two information sources and two models, the four variants of the social graph model are: the *complex-friendship model*, the *complex-membership model*, the *simple-friendship model* and the *simple-membership model*.

4.4.4 Words and social graph information

In Chapter 6, we evaluate the performance of the various models defined in the previous subsections. The results show that in terms of tweet content the multiple-word model performs the best. Furthermore, in terms of social graph information the membership models outperform the friendship models.

In this subsection, we combine the multiple-word model with the complex-membership model and simple-membership model respectively. The multiple-word model is included in the complex-membership model, by adding the `has` relation to the `Document` entity. The resulting E-R diagram is shown in Figure 4.12.

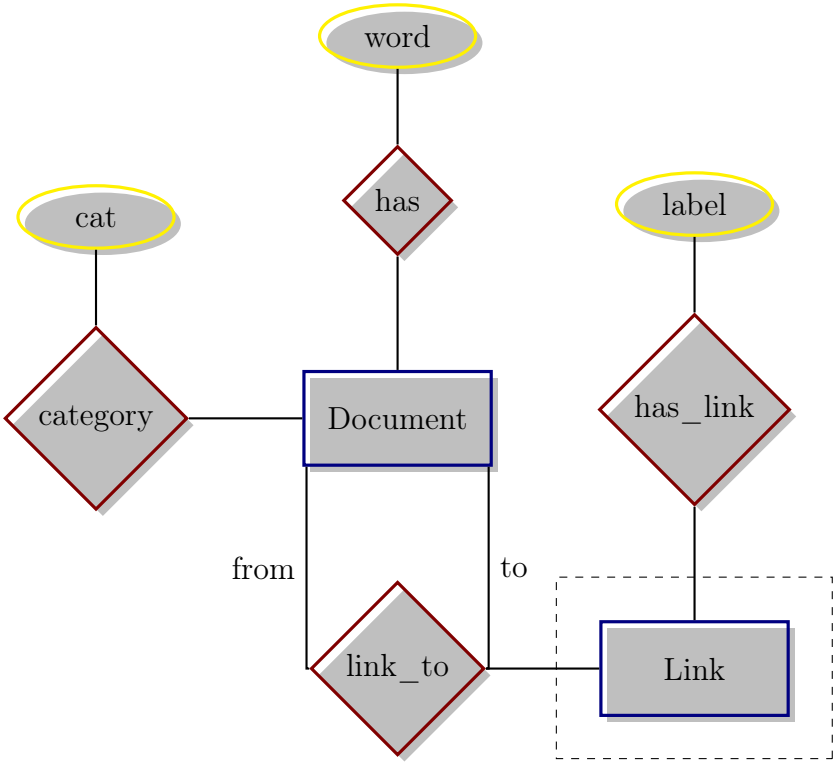


Figure 4.12: The complex E-R diagram for the combined multiple-word model and complex-membership model.

We also include the multiple-word model in the simple-membership model, resulting in the model shown in Figure 4.13.

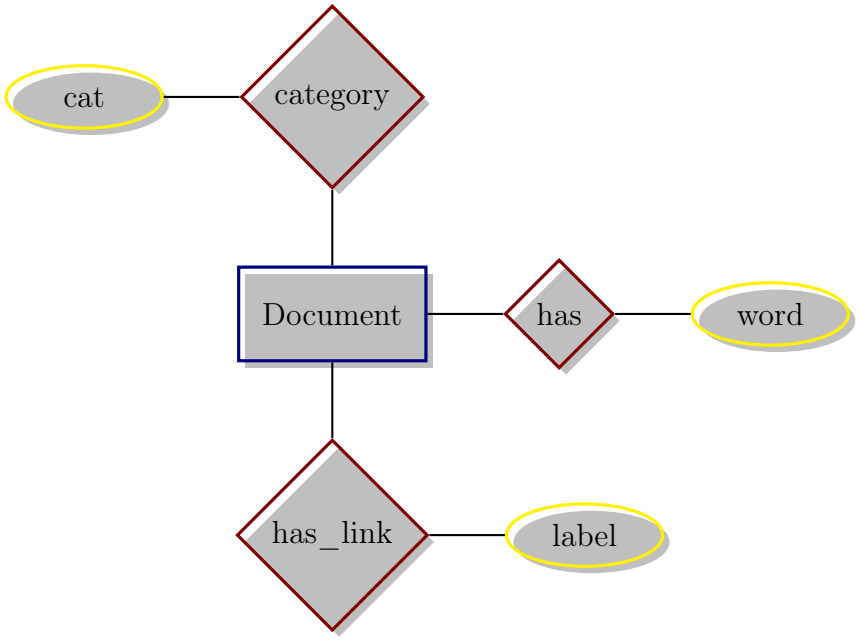


Figure 4.13: The simple E-R diagram for the combined multiple-word model and simple-membership model.

We refer to these two combined models as the *complex-membership-word model* and *simple-membership-word model* respectively.

4.5 Conclusion

This chapter introduced kLog as an approach to solving the classification task of constructing topic-based Twitter lists. We discussed the limitation of only analysing the content of tweets for discriminating between topic-based categories, and suggested that the information contained in the social graph of Twitter users could be used to improve the classification and clustering result. Based on these considerations, we presented various kLog interpretations and models for constructing topic-based Twitter lists.

Chapter

5

Data sets and evaluation

This chapter introduces two Twitter data sets, and the measures used to evaluate our clustering and classification approaches. To construct topic-based Twitter lists, a data set of users and relevant user information is needed. We introduce two data sets consisting of multiple Twitter lists, each in turn containing multiple users. For each data set we discuss the collection and processing of Twitter user information. We also highlight the different attributes of each data set. We follow the data set discussion by a description of the different evaluation measures.

5.1 Data sets

5.1.1 Initial collection

To empirically train our clustering and classification approaches, we need data sets consisting of multiple Twitter lists, with each list containing multiple users. Using such data sets provides a standard to which the performance of the various clustering and classification approaches can be compared.

Twitter provides new users with a number of categories, containing multiple users who are experts on the category topic or are active in that field. We use these categories and the users in them to collect a data set, which we shall call the *suggested lists* data set.

Our second data set is constructed from Twitter lists. Twitter lists provide a user with a means to group other users in the network. A list is managed

by a single user, who decides which users should be included in or removed from the list. A user may be a member of many lists, as well as the curator of many lists. We use Listorious (Listorious, 2012) as a source of well-maintained topic-based Twitter lists. Listorious is a Twitter search engine that keeps a ranking of the most popular lists on Twitter based on the number of followers. The top 140 of these lists are ranked on the Listorious website. To construct the second data set, which we call the *subscribed lists* data set, we select 14 lists based on ranking and the topic definition. The goal is to select lists that are not ambiguous in terms of their topic definition, and with minimal overlap between users.

The suggested list data set is constructed using the Twitter API, which provides endpoints[1] for retrieving the categories and users in those categories. Therefore, we retrieve each category profile, which consists of a name, a slug and a size field. A slug is a unique identifier, which is used to retrieve the users in that category. The category and all the users in the category are stored; each user is represented by a user profile, containing a unique user identifier.

The subscribed list data set is constructed by saving the list identifiers collected from Listorious. These list identifiers are used to retrieve list profiles, and all the user profiles of the Twitter users who are members of these lists.

Next, the tweets of the users in each data set are retrieved. The number of tweets retrieved for each user is limited to 100. Furthermore, for each user we retrieve their first 100 list memberships.

In summary, each data set consists of a collection of categories or list profiles. For each category/list, a profile of the Twitter users who are in that category/list are stored. These user profiles store the user's 100 latest tweets, the people that the user follows and the user's first 100 list memberships.

5.1.2 Processing

After retrieving the user data as described in Subsection 5.1.1, a number of checks are performed. Users are removed from a data set if:

- they are part of multiple lists/categories;
- their account is protected;

[1]Endpoints are HTTP URLs provided for interacting with Twitter.

- their account is not specified as English; or
- they have tweeted less than a hundred times.

A Twitter user is represented by a user document, which corresponds to a concatenation of the user's 100 latest tweets into a single document. To generate the user document, we tokenize the 100 tweets into an array of word tokens. A token is removed if it is part of the English stop words list (NLTK) (Bird *et al.*, 2009) or if its length is shorter than 3 characters. Furthermore, punctuation characters are removed from each token in the word array, and the resulting token is stemmed using the Porter stemmer[2] (Porter, 2006).

Finally, the frequency of each token in the data set is calculated and the 2000 most frequent tokens are stored as that data set's vocabulary. An intersection of the vocabulary, stored for each data set, and each user's word tokens are taken with the resulting array stored as the user document.

Next, friendships are calculated for each user. A friendship is defined when user A follows user B and vice versa. Furthermore, we remove a list membership from a user's profile if that list is one of those collected from Listorious.

5.1.3 Data set attributes

Table 5.1 shows information about the composition of each data set after Subsection 5.1.2. The subscribed lists data set contains more users than the suggested lists data set, but in fewer categories.

	Suggested lists	Subscribed lists
# Users	1737	2800
# Lists	26	14
Largest	184	834
Smallest	23	14
Average	66.8	200

Table 5.1: A summary of the attributes of the suggested lists and subscribed lists data sets.

We also include a bar chart for each data set in Figure 5.1 and Figure 5.2. By comparing these two charts, we see that users in the suggested lists data set are distributed more evenly over its categories than those in the subscribed lists

[2]Stemming is the removal of morphological and inflexional endings from words.

data set. We expect that this will impact the clustering result, since finding clusters in data is more challenging when these clusters differ widely in shape, size and density (Ertoz *et al.*, 2003).

Furthermore, the two data sets differ in how the underlying data is generated. The suggested lists data set consists of the Twitter-generated categories. Twitter introduced these categories in a blog post (Elman, 2010), in which they stated that algorithms are used to create them. In contrast, the lists in the subscribed lists data set are all created by Twitter users. We discuss the impact of this on our experiment results in Chapter 6.

5.2 Evaluation

This section presents the evaluation metrics used in this thesis.

5.2.1 Clustering

The measures used to evaluate the quality of a clustering are important since every clustering algorithm will identify clusters in a data set even if that data set has no natural clustering structure. In order to compare two clusterings, we can build a contingency table that summarises the overlap between the clusterings. This contingency table is used extensively in pair-counting-based measures (Hubert and Arabie, 1985).

Let S be a set of N data points, where $U = \{U_1, U_2, \ldots, U_R\}$ and $V = \{V_1, V_2, \ldots, V_C\}$ are clusterings of S, in other words partitions of S into disjoint subsets. The overlap between U and V can then be summarised in an $R \times C$ contingency table $M = [n_{ij}]_{j=1,\ldots,C}^{i=1,\ldots,R}$, where n_{ij} is the number of objects shared between subsets U_i and V_j. Table 5.2 illustrates the contingency table, where the clustering U represents the correct categorisation for the data set. The clustering U is thus the target clustering and we refer to each category in the data set as a *class* when discussing the evaluation techniques. The clustering V would then typically be the result of the k-means or affinity propagation algorithm.

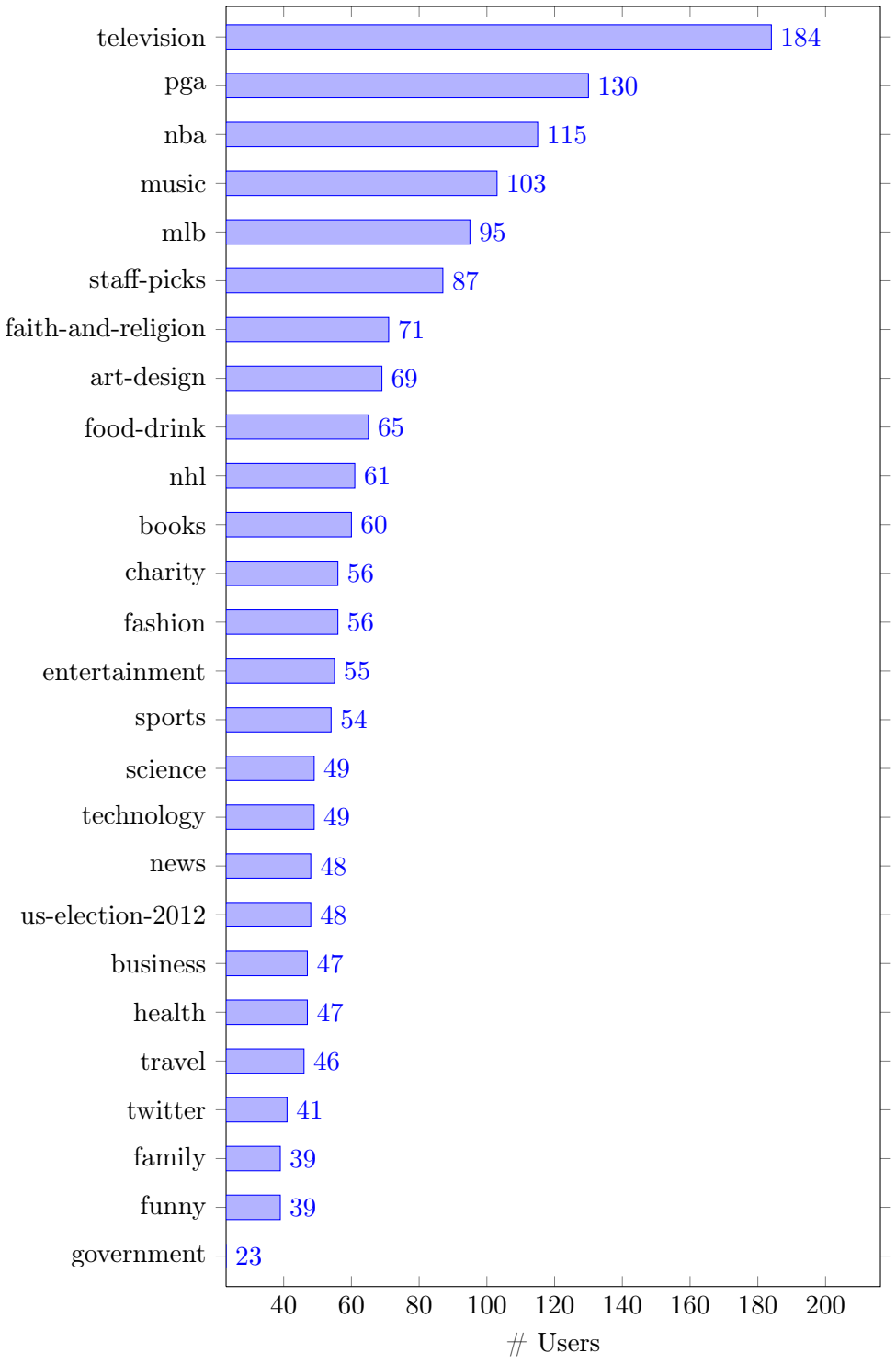


Figure 5.1: A chart of the number of users in each category of the suggested lists data set.

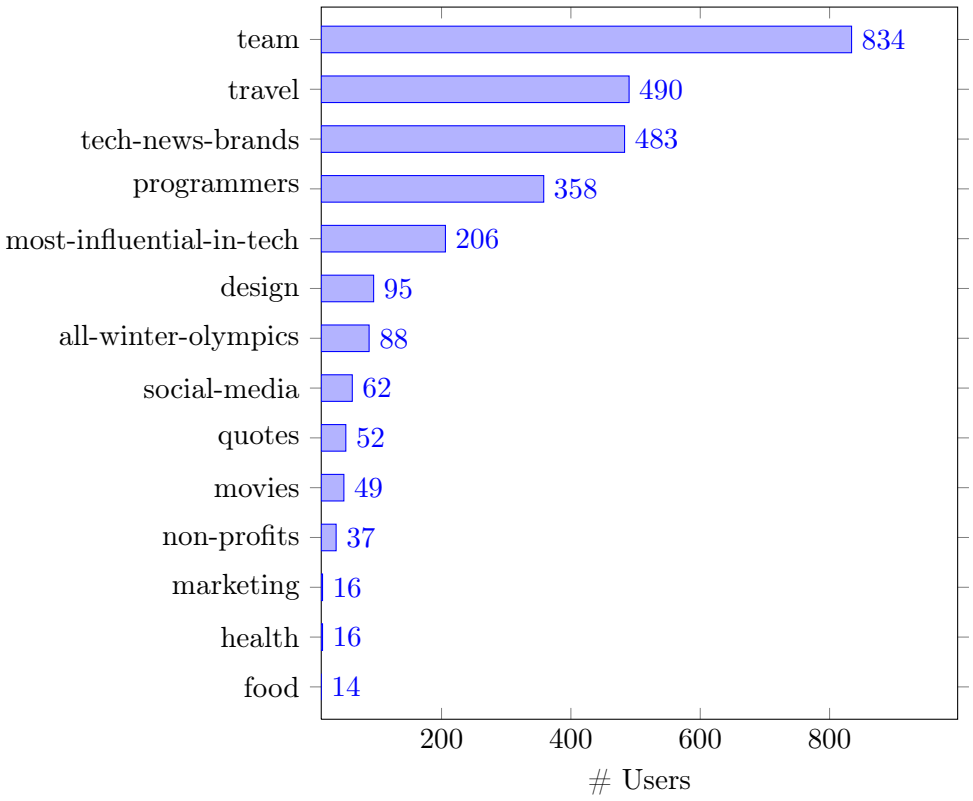


Figure 5.2: A chart of the number of users in each category of the subscribed lists data set.

U/V	V_1	V_2	...	V_C	Sums
U_1	n_{11}	n_{12}	...	n_{1C}	a_1
U_2	n_{21}	n_{22}	...	n_{2C}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_R	n_{R1}	n_{R2}	...	n_{RC}	a_R
Sums	b_1	b_2	...	b_c	$\sum_{ij} n_{ij} = N$

Table 5.2: The structure of the contingency table. Here n_{ij} is the number of objects shared between subsets U_i and V_j.

5.2.1.1 Entropy and purity

Entropy and purity are classification-oriented measures for cluster validity and are commonly used to evaluate the performance of classification models (Tan *et al.*, 2006; Zhao and Karypis, 2004). In terms of cluster evaluation, the degree to which cluster labels correspond to class labels is measured.

The entropy measures the distribution of the various class labels in each cluster (Zhao and Karypis, 2004). We calculate a normalised entropy for each cluster V_j from the contingency table as

$$E(V_j) = -\frac{1}{\log R} \sum_{i=1}^{R} \frac{n_{ij}}{b_j} \log \frac{n_{ij}}{b_j}, \tag{5.2.1}$$

where $\frac{n_{ij}}{b_j}$ is the proportion of documents in cluster j (b_j) belonging to class i. The average entropy of a clustering is then defined as the weighted sum of the entropies for each cluster, with weights based on the cluster sizes, and is calculated as

$$E(V) = \sum_{j=1}^{C} \frac{b_j}{N} E(V_j). \tag{5.2.2}$$

The entropy takes on values in the range of $[0, 1]$. A entropy of 0 indicates that all documents in a cluster belong to the same class. If the document class distribution is uniform over the cluster, the entropy will be 1. Thus a smaller entropy value is better for a clustering result.

The purity measures the extent to which a cluster contains documents of a single class (Tan *et al.*, 2006). We calculate the purity of each cluster V_j as

$$P(V_j) = \frac{1}{b_j} \max_{i=1}^{R} \{n_{ij}\}. \tag{5.2.3}$$

The overall purity of a clustering is

$$P(V) = \sum_{j=1}^{C} \frac{b_j}{N} P(V_j), \tag{5.2.4}$$

a weighted sum of the purity of the clusters in V. A purity of 1 indicates that each cluster only contains documents of a single class.

5.2.1.2 Normalised information distance

The Mutual Information (MI) (Vinh *et al.*, 2010) describes the amount of statistical similarity between two clusterings U and V (Banerjee *et al.*, 2006).

We can calculate the MI from Table 5.2 as

$$I(U,V) = \sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{a_i b_j/N^2}. \tag{5.2.5}$$

We can also define the entropy of clustering U and the conditional entropy of clustering U relative to clustering V as

$$H(U) = -\sum_{i=1}^{R} \frac{a_i}{N} \log \frac{a_i}{N} \tag{5.2.6}$$

and

$$H(U|V) = -\sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{b_j/N}. \tag{5.2.7}$$

The entropy $H(U)$ measures the uncertainty of class allocation in U and the conditional entropy $H(U|V)$ of cluster U given cluster V measures the remaining uncertainty of class allocation in U given that the clustering V is already known.

It is of interest to see how much knowledge of V reduces the uncertainty in U. If V provides information that helps to reduce the uncertainty in U, then $H(U|V)$ will be smaller than $H(U)$. The MI quantifies the reduction in uncertainty: we have $H(U) - H(U|V) = I(U,V)$, which also holds in the reverse direction, $H(V) - H(V|U) = I(U,V)$.

Kraskov *et al.* (2005) noted that one can modify the mutual information to obtain a metric. We calculate the Normalised Information Distance (NID) by normalising the MI and converting it to a distance metric. The NID,

$$\mathrm{NID}(U,V) = 1 - \frac{I(U,V)}{\max\{H(U), H(V)\}}, \tag{5.2.8}$$

is in the range $[0,1]$.

Hubert and Arabie (1985) showed that similarity measures for cluster evaluation should be corrected for chance. Their *corrected-for-chance* property states that an index should have a constant baseline value. This means that the expected value between pairs of independent clusterings, sampled independently at random, should be a constant. Ideally this baseline value should be zero, which indicates no similarity between the clusterings.

To achieve this they propose an index adjustment of the form[3]

$$\text{Adjusted\_Index} = \frac{\text{Index} - \text{Expected\_Index}}{\text{Max\_Index} - \text{Expected\_Index}}. \tag{5.2.9}$$

We thus use the Adjusted Mutual Information (AMI) as shown by Vinh *et al.* (2010) to calculate the adjusted-for-chance NID (ANID). The ANID is

$$\begin{aligned} \text{ANID}(U,V) &= 1 - \text{AMI}(U,V) \\ &= 1 - \frac{I(U,V) - E\{I(U,V)\}}{\max\{H(U),H(V)\} - E\{I(U,V)\}}, \end{aligned} \tag{5.2.10}$$

where $E\{I(U,V)\}$ is the expected MI (Vinh *et al.*, 2010) between two clusterings U and V. This expected MI is defined, from the contingency table (Table 5.2), as

$$\begin{aligned} E\{I(U,V)\} = \sum_{i=1}^{R}\sum_{j=1}^{C} \sum_{n_{ij}=\max(a_i+b_j-N,0)}^{\min(a_i,b_j)} \Bigg(\frac{n_{ij}}{N} \log\left(\frac{N \cdot n_{ij}}{a_i b_j}\right) \times \\ \frac{a_i! b_j! (N-a_i)! (N-b_j)!}{N! n_{ij}! (a_i - n_{ij})! (b_j - n_{ij})! (N - a_i - b_j + n_{ij})!} \Bigg). \end{aligned} \tag{5.2.11}$$

The ANID is zero when clusterings U and V are identical. We use the ANID as our general evaluation metric, further analysing the best results using the purity and entropy.

5.2.2 Classification

In our classification experiments, the task is multi-class classification. The chosen evaluation technique consists of k-fold cross-validation with the *precision*, *recall* and *F1*-measure.

The precision and recall are defined in terms of the *true positive*(TP), *false positive*(FP), *false negative*(FN) and *true negative*(TN) rates. We summarise these four metrics for a class i in Table 5.3.

The precision and recall are defined as

$$\text{P}_\text{i} = \frac{TP_i}{TP_i + FP_i}, \tag{5.2.12}$$

[3]The adjusted measure can be a negative value if the result is worse than the random allocation of data points to clusters.

Class i		Actual class	
		True	False
Predicted class	True	TP_i	FP_i
	False	FN_i	TN_i

Table 5.3: An explanation of the true positive, false positive, false negative and true negative rates.

and

$$\mathrm{R_i} = \frac{TP_i}{TP_i + FN_i} \tag{5.2.13}$$

respectively. The precision score is 1 when all documents labelled as class i are in class i, and the recall score is 1 when all documents in class i are labelled as class i.

The precision and recall are combined to form the F1-measure, which is the harmonic mean of the precision and recall. The F1-measure is

$$F_1 = 2 \times \frac{P_i \cdot R_i}{P_i + R_i}, \tag{5.2.14}$$

and is in the $[0, 1]$ range, where 1 is its best value and 0 its worst.

To calculate the precision, recall and F1-measure for our multi-class classification task these values need to be combined. We achieve this with two techniques, namely using *weighted* scores and *multi-class* scores.

Let $C = \{c_1, c_2, \ldots, c_N\}$ be the set of class sizes for N classes, where each c_i is the number of documents in class i. Let $d = \sum_{j=1}^{N} c_j$ be the sum of the class sizes, in other words the number of documents in the data set. The weighted scores are calculated by weighting each class i proportional to d. The weighted precision and recall are

$$\mathrm{P_w} = \sum_{i=1}^{N} \frac{c_i}{d} \times \frac{TP_i}{TP_i + FP_i}, \tag{5.2.15}$$

and

$$\mathrm{R_w} = \sum_{i=1}^{N} \frac{c_i}{d} \times \frac{TP_i}{TP_i + TN_i} \tag{5.2.16}$$

respectively. The corresponding weighted F1-measure is

$$F_1^w = \sum_{i=1}^{N} \frac{c_i}{d} \times (2 \times \frac{P_i \cdot R_i}{P_i + R_i}). \tag{5.2.17}$$

For the multi-class scores, each class is weighted equally. Therefore, the multi-class precision and recall are

$$\mathrm{P_m} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i}, \tag{5.2.18}$$

and

$$\mathrm{R_m} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + TN_i} \tag{5.2.19}$$

respectively. The corresponding multi-class F1-measure is

$$F_1^m = \frac{1}{N}\sum_{i=1}^{N} 2 \times \frac{P_i \cdot R_i}{P_i + R_i}. \tag{5.2.20}$$

5.3 Conclusion

This chapter introduced the suggested lists and subscribed lists data sets. We discussed the retrieval and construction of these data sets as well as some attributes that may affect our clustering and classification approach. We also discussed the metrics used to evaluate the clustering and classification approaches.

Chapter

6

Experiments

This chapter describes several experiments conducted to evaluate the performance of clustering and classification approaches for constructing topic-based Twitter lists. Overall, each experiment addresses the problem of constructing topic-based Twitter lists, but also evaluates different aspects of the clustering and/or classification approach.

This chapter is structured as follows: In Section 6.1, we evaluate the clustering approach on a Twitter user's tweet content represented as a document, followed by performing classification using libSVM to achieve a baseline for comparison in Section 6.2. In Section 6.3 and Section 6.4, we evaluate the performance of different kLog models on our Twitter data sets. This is followed by a clustering experiment using the feature vectors extracted from the best kLog model in Section 6.5. Finally, the clustering experiment is repeated with a set of graph data for each Twitter user in Section 6.6.

We present preliminary conclusions for each experiment in each results section.

6.1 Clustering short text documents

6.1.1 Purpose

The document clustering approach, introduced in Chapter 3, is governed by four procedures:

- selection of user document content;

- feature vector representation;
- user document similarity calculations; and
- user document clustering.

This experiment evaluates each of the four procedures. In the first case, user tweet content as topic indicator is evaluated. The next three cases are evaluated in terms of best representation, similarity measure and clustering algorithm.

6.1.2 Experimental method

The suggested lists and subscribed lists data sets were detailed in Chapter 5. We calculate the TF-IDF for each user document over the terms in the data set vocabulary. We also train the LDA topic model on the user document set for each data set, using the same vocabulary. The LDA topic model takes as parameter the number of topics to train. We train each data set using 5, 15 and 30 topics.

At this stage, the subscribed lists and suggested lists data sets are each represented in four different ways: the TF-IDF representation, LDA trained on 5 topics, LDA trained on 15 topics, and LDA trained on 30 topics. In our results we refer to these representations as TF-IDF, LDA5, LDA15, and LDA30 respectively. The five similarity measures are then used to calculate the similarities between user documents on the four representations. These similarities are used as input to the k-means and AP clustering algorithms, with varying numbers of clusters, namely 5, 10, 15, 20, 25 and 30. We repeat the experiment for each combination of representation, similarity measure and cluster size for k-means and AP.

To find the specified number of clusters for AP, an extra step is needed. For each combination of representation, similarity measure and number of clusters, the self-preference value p needs to be found that obtains the specified number of clusters. This p is found by searching over a continuous range using a bisection method (Frey and Dueck, 2007).

Section 3.4.2 presented the damping factor as well as termination conditions for the AP algorithm. We set the damping factor to $\lambda = 0.9$ throughout all our experiments. Furthermore, we set the maximum number of iterations to

1000 and set the termination condition for no update to the local result at 30 iterations. For k-means, we set the maximum number of iterations to 300.

6.1.3 Results

We include four tables, Tables 6.1 – 6.4, two of which can be viewed as "expected best result" tables with the other two containing the best results actually obtained. These tables present the ANID (see Section 5.2) for each combination of representation, clustering algorithm and similarity measure. For each row, in other words combination of representation and clustering algorithm, the best performing similarity measure is shown in bold. The table cell containing the table's best result is highlighted in orange and the overall best result for each data set is highlighted in blue.

The "expected best result" table for a data set corresponds to the situation where the number of clusters is set closest to the actual number of lists/categories in the data set.

	Adjusted Normalised Information Distance				
Representation	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means for 25 clusters				
LDA5	0.6873	**0.6808**	0.6879	0.6868	0.7241
LDA15	0.6942	0.6392	0.7647	**0.6146**	0.7136
LDA30	0.7322	0.6295	0.7430	**0.5967**	0.6858
TF-IDF	0.9687	**0.6424**	0.7653	0.6543	0.9131
	Affinity Propagation for 25 clusters				
LDA5	0.6821	**0.6729**	0.8044	0.6738	0.8973
LDA15	0.5776	**0.5652**	0.9500	0.5695	0.9707
LDA30	0.5148	**0.5046**	0.9617	0.5062	0.9923
TF-IDF	0.8858	**0.5168**	0.8455	0.5372	0.9692

Table 6.1: Suggested lists: ANID results for 25 clusters.

The suggested lists data set contains 26 lists and thus we include the results for 25 clusters in "expected best result" Table 6.1. The best clustering result on this data set was obtained for 30 clusters, the results of which are in Table 6.2. The subscribed lists data set contains 14 lists; we include the "expected best result" for 15 clusters in Table 6.3 as well as the best results actually obtained (using 10 clusters) in Table 6.4.

	Adjusted Normalised Information Distance				
Representation	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means for 30 clusters				
LDA5	0.6985	**0.6828**	0.6993	0.6887	0.7199
LDA15	0.6833	0.6420	0.7402	**0.6141**	0.7059
LDA30	0.7556	**0.6453**	0.7516	0.6482	0.6969
TF-IDF	0.9700	**0.6254**	0.7694	0.6393	0.9216
	Affinity Propagation for 30 clusters				
LDA5	0.6914	**0.6788**	0.8238	0.6872	0.8958
LDA15	0.5783	**0.5762**	0.9533	0.5807	0.9706
LDA30	0.5282	**0.5080**	0.9738	0.5203	0.9912
TF-IDF	0.8697	**0.4999**	0.8375	0.5182	0.9701

Table 6.2: Suggested lists: ANID results for 30 clusters.

For the suggested lists data set, the best result obtained was with AP using cosine similarity with the TF-IDF representation for 30 clusters. The second best result was obtained with AP using cosine similarity with LDA30 for 30 clusters.

The results for the subscribed lists data set show that a combination of cosine similarity, LDA15 and 10 clusters with AP achieves the best result. The second best result is a combination of extended Jaccard coefficient, LDA30 and 15 clusters with AP.

	Adjusted Normalised Information Distance				
Representation	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means for 15 clusters				
LDA5	0.6880	0.6776	0.7165	**0.6755**	0.7148
LDA15	0.7151	0.7003	0.7566	**0.6933**	0.7357
LDA30	0.7496	0.7269	0.7547	**0.7151**	0.7530
TF-IDF	0.9097	**0.7149**	0.7900	0.7321	0.9620
	Affinity Propagation for 15 clusters				
LDA5	0.6879	**0.6688**	0.8321	0.6803	0.8840
LDA15	0.6558	**0.6555**	0.9427	0.6588	0.9618
LDA30	0.6543	0.6426	0.9418	**0.6409**	0.9878
TF-IDF	0.8724	0.6872	0.9331	**0.6711**	0.9767

Table 6.3: Subscribed lists: ANID results for 15 clusters.

For both data sets, the cosine similarity tends to perform the best. Figures 6.1 and 6.2 plot the ANID obtained by the cosine similarity against

	Adjusted Normalised Information Distance				
Data set	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means for 10 clusters				
LDA5	0.6667	**0.6338**	0.7189	0.6431	0.7206
LDA15	0.7227	**0.6763**	0.7657	0.7015	0.7468
LDA30	0.7756	0.7130	0.7436	**0.7104**	0.7498
TF-IDF	0.9633	**0.7371**	0.8215	0.7396	0.9710
	Affinity Propagation for 10 clusters				
LDA5	0.6560	0.6420	0.8809	**0.6417**	0.8786
LDA15	0.6988	**0.6117**	0.9420	0.6538	0.9279
LDA30	**0.7028**	0.7217	0.9586	0.7308	0.9269
TF-IDF	0.9941	**0.7918**	0.9492	0.8070	0.9736

Table 6.4: Subscribed lists: ANID results for 10 clusters.

the number of clusters for each combination of representation and clustering algorithm.

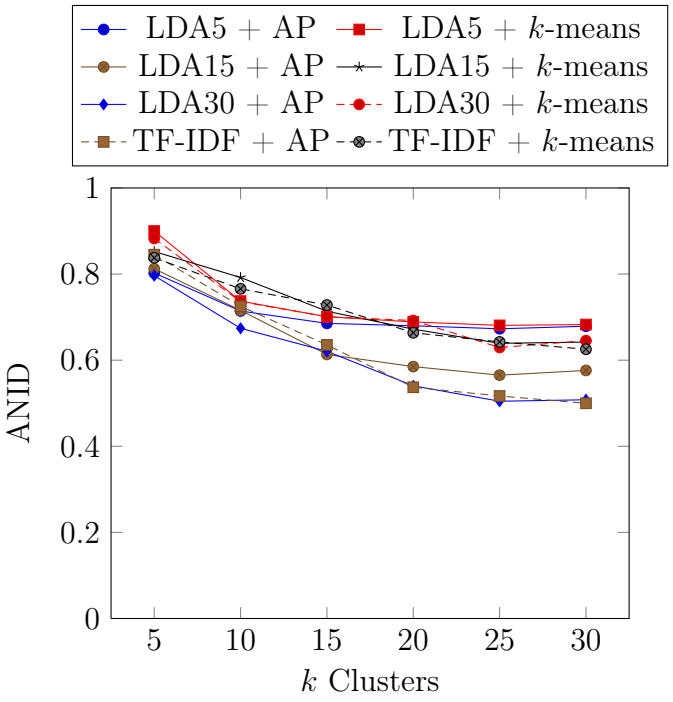


Figure 6.1: Suggested lists data set: ANID obtained using the cosine similarity plotted against the number of clusters for each combination of representation and clustering algorithm.

Figures 6.1 and 6.2 show that the results tend to be similar around the cluster sizes further away from the best result. The difference in performance

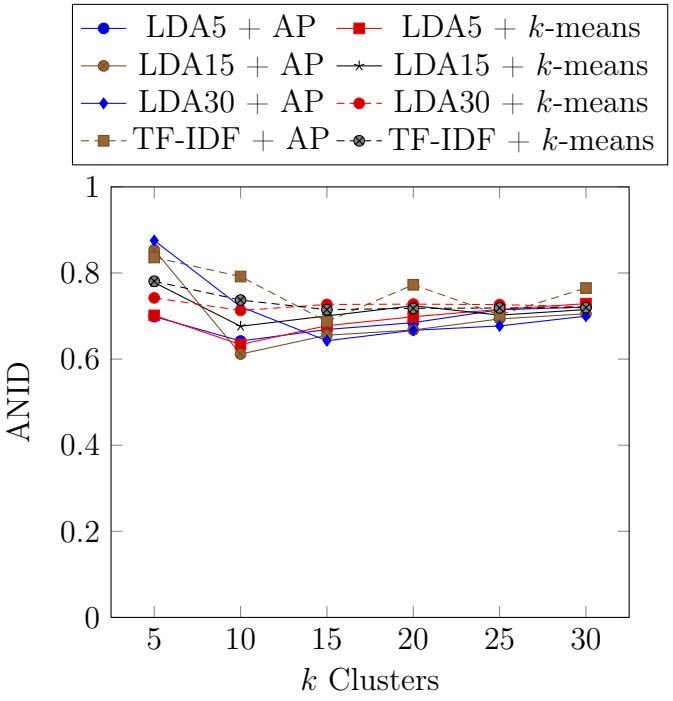


Figure 6.2: Subscribed lists data set: ANID obtained using the cosine similarity plotted against the number of clusters for each combination of representation and clustering algorithm.

closer to the best result for a data set is greater than for the other number of clusters. Figure 6.1 illustrates this tendency with a definite gap visible between the results obtained with k-means and AP closer to the best result. However, in Figure 6.2, this gap between k-means and AP do not exist. The results in Figure 6.2 suggests that in this case the largest contributing factor is the chosen representation technique.

The results indicate that our clustering approach is sensitive to the representation if the number of clusters are close to the best result in each data set. Next, we interpret our results in terms of the similarity measures, representation and clustering algorithms used.

6.1.3.1 Similarity measures

The cosine similarity and extended Jaccard coefficient tend to perform better than the other similarity measures with cosine similarity usually only slightly outperforming the extended Jaccard coefficient. The ANIDs achieved for the cosine similarity and extended Jaccard coefficient generally differ by less than

0.05, with a maximum difference of 0.09.

The results of the Euclidean distance in combination with TF-IDF is poor, while combined with LDA the performance is much improved. The poor performance of the Euclidean distance can be attributed to the relatively small number of overlapping features between two document TF-IDF vectors. The LDA feature vectors have the same topics in common, and do not suffer from the overlapping features problem.

An interesting result is the poor performance of the Pearson correlation coefficient and Kullback-Leibler divergence for the AP clustering algorithm even though their result for k-means is not much worse when compared to the other measures.

If a similarity measure were able to clearly distinguish clusters in a group of points, we would expect a histogram of pairwise similarity values for these points to be multimodal. However, we see that the pairwise similarity values of the Pearson correlation coefficient and Kullback-Leibler divergence on our data sets are concentrated in a small range.

6.1.3.2 Representation

By comparing the document representation techniques, LDA tends to outperform TF-IDF if the number of topics is at least as great as the actual number of lists in the data set. Note that the TF-IDF performs exceptionally well for the suggested list data set, where only the performance of the LDA30 is comparable. This is in contrast with the results for the subscribed lists data set, where the TF-IDF is outperformed by most of the different topic sizes for k-means and AP. The surprisingly good performance of TF-IDF on the suggested list data set supports our hypothesis (Section 5.1.2) that the Twitter categories are created with a representation similar to the TF-IDF.

6.1.3.3 Clustering algorithms

AP consistently achieves better results than k-means for the Euclidean distance, cosine similarity and extended Jaccard coefficient. For the Pearson correlation coefficient and Kullback-Leibler divergence, k-means tend to perform better than AP. This evidence suggest that k-means performs better than AP when the underlying clusters in the data are not well-defined by a similarity measure.

6.1.3.4 Summary

We can conclude that AP in combination with the LDA topic model, using either the cosine similarity or extended Jaccard coefficient achieves the best result. Furthermore, using tweet content as a topic indicator and if care is taken in selecting the number of clusters, one can find sensible clusters that show significant improvement over random assignment.

6.2 Classification of short text documents using libSVM

6.2.1 Purpose

The purpose of this experiment is to set a baseline to be used for comparison with the kLog classification results. We select the well-known SVM library, libSVM (Chang and Lin, 2011), to perform classification on the two Twitter data sets. We apply the classification task to the TF-IDF, LDA5, LDA15 and LDA30 data representations.

6.2.2 Experimental method

This experiment uses the representations of the suggested and subscribed lists data sets to train a classifier. It is necessary to format the representation in such a way that libSVM can be used to train a model. The data is written to a text file where each line is a document and contains the document class and features. We represent a single document for use with libSVM as a single line of the form

```
<label> <index>:<value> <index>:<value> ... <index>:<value>
```

The class of a document is indicated with `<label>`, `<index>` is the feature label and `<value>` is the corresponding value of the feature. These features are the TF-IDF, LDA5, LDA15 and LDA30 representations, which we discussed in the previous section.

Next we apply scaling to our data, which transforms the values to the $[0, 1]$ range. Scaling is performed with the `svm-scale` executable on each file setting the lower bound to 0 and the upper bound to 1. Using the `subset.py` script

provided by libSVM, we partition our data into a training and a holdout set. The holdout set is found by performing stratified sampling (Witten and Frank, 2005) on the whole set.

Using the training set of each data set representation we perform a grid search to find the best parameters for each set. We use the `grid.py` script, which performs this search for C-SVM (Chang and Lin, 2011) using the radial basis function (RBF) kernel (Witten and Frank, 2005). The script performs cross-validation on each parameter combination to evaluate the accuracy achieved with those parameters.

The goal of the grid search for the RBF kernel is to find the SVM hyperparameters C and γ that provide optimal classification performance for C-SVM. The `grid.py` takes the C and γ range values as input on a log scale. We let the C parameter range from $[2, 1024]$ by setting the start to 2, end to 10 and the step size to 1. The γ value ranges from $[0.0625, 1]$ by setting the start to -4, end to 0 and step size to 1.

We repeat the grid search for each of the data set representations and store the resulting parameters. Using these parameters with the `svm-train` executable we train a model for each data set representation. This model is evaluated using the `svm-predict` executable, which reads in the model and predicts the class of each document in the holdout set.

6.2.3 Results

Table 6.5 compares the performance of each data set representation in terms of the weighted precision, recall and F1-measure. We highlight the best F1 score for each data set in bold.

The results show that LDA30 provides the best performance for both data sets. An interesting result is the poor performance of LDA5 on the suggested lists data set. The evidence suggests that when the number of trained topics are substantially less than the number of class labels, performance will be poor. The performance decrease for LDA5 for the subscribed lists data set is smaller than for the suggested lists data set. This indicates that the difference between the number of trained topics and the number of class labels affects the extent of the performance decrease.

Table 6.6 compares our previous clustering result to the libSVM classification result using the entropy and purity measures.

	Precision	Recall	F1
	Suggested lists		
LDA5	0.2723	0.3288	0.2834
LDA15	0.6766	0.6788	0.6603
LDA30	0.6669	0.6981	**0.6766**
TF-IDF	0.7261	0.6481	0.6537
	Subscribed lists		
LDA5	0.6122	0.6558	0.6144
LDA15	0.7043	0.7333	0.7146
LDA30	0.7970	0.7988	**0.7968**
TF-IDF	0.7055	0.7152	0.6900

Table 6.5: The precision, recall and F1 values achieved for the suggested and subscribed lists data sets with libSVM.

	Entropy	Purity
	Suggested lists	
Clustering	0.4406	0.5947
LibSVM	0.2614	0.7288
	Subscribed lists	
Clustering	0.4652	0.6162
LibSVM	0.2684	0.8012

Table 6.6: A comparison of our clustering result to the libSVM classification result using the entropy and purity measures.

For both data sets, the classification result outperforms the clustering result. This result is not unexpected, because it has been shown that SVM classifiers perform well in the document domain (Chang and Lin, 2011). On our data sets, we see that the SVM classifier is better at distinguishing between the categories/lists than the clustering approach.

6.3 kLog: Separate models for content and graph classification

6.3.1 Purpose

In this experiment, we use the kLog system to construct topic-based Twitter lists using a classification approach. In Chapter 4, we introduced the different kLog models that are used to represent Twitter users as documents. These

models are constructed from different Twitter user content, namely

- words,
- topics, and
- social graph information.

In this experiment, variants of these models are each separately evaluated and compared to the libSVM results. A notable difference to the previous representations is the ability to include information from a Twitter user's social graph structure in some of these kLog models.

6.3.2 Experimental method

Table 6.7 summarises the models that we evaluate in this section. We evaluate the performance of each model in the kLog system for the task of constructing topic-based Twitter lists. As with our earlier investigations, we perform these ex-

Words	Topics	Social graph
one-word	lda-topic	simple-friendship
multiple-word	threshold-topic	complex-friendship
tfidf-word		simple-membership
		complex-membership

Table 6.7: A summary of the models, which we use in the kLog system to evaluate the task of constructing topic-based Twitter lists.

periments using the suggested lists and subscribed lists data sets. Furthermore, the two kLog topic models are evaluated with 30 topics.

The `document` entities are the kernel points, and features are generated by the NSPDK. A classifier is trained on these features using the libSVM linear kernel. For those models whose graphicalisation is a star, we set the maximum distance $d = 0$ and the maximum radius $r = 1$. For the complex models these parameters are set to $d = 2$ and $r = 3$.

We found the parameters for the complex models by evaluating the distance and radius over a range of values. These ranges were $\{0, 1, 2, 3\}$ for the distance and $\{1, 2, 3, 4, 5\}$ for the radius. The results showed that for $r > 3$ and $d > 2$ the performance of these models do not improve.

For the libSVM linear kernel, we set the cost parameter $C = 1000$. This C was found by evaluating the classification performance over a range of values.

6.3.3 Results

6.3.3.1 Words

Table 6.8 summarises the result for the word models for the suggested lists data set. The best result was achieved for the multiple-word model, highlighted in blue.

	Precision	Recall	F1
	one-word		
multiclass avg	0.712	0.607	0.631
weighted avg	0.693	0.674	0.659
	multiple-word		
multiclass avg	0.725	0.655	0.673
weighted avg	0.713	0.699	0.693
	tfidf-word		
multiclass avg	0.732	0.635	0.659
weighted avg	0.714	0.691	0.681

Table 6.8: Performance summary of the word models for the suggested lists data set.

Furthermore, for the tfidf-word model, the addition of the TF-IDF values to the multiple-word model do not improve the result. The performance of the TF-IDF may be explained by examining Equation 4.2.14. The equation shows that a product of the TF-IDF values are calculated and added to a discrete match count. The TF-IDF values are generally small, and as such a match has a much smaller contribution to the kernel value than a discrete match. Therefore, the result is still mostly influenced by the discrete match count.

We have previously shown that the TF-IDF performs fairly well in the classification task with libSVM. Therefore, the evidence suggests that kLog struggles to capture the contributions of real-valued components when features are created from a graphicalisation.

Table 6.9 summarises the result for word models for the subscribed lists data set. Again, the multiple-word (highlighted in blue) model performs the best.

	Precision	Recall	F1
	one-word		
multiclass avg	0.593	0.476	0.510
weighted avg	0.742	0.764	0.741
	multiple-word		
multiclass avg	0.727	0.617	0.657
weighted avg	0.790	0.793	0.788
	tfidf-word		
multiclass avg	0.724	0.558	0.595
weighted avg	0.772	0.781	0.768

Table 6.9: Performance summary of the word models for the subscribed lists data set.

An interesting discrepancy between the subscribed lists and suggested lists data set is the difference in the multiclass and weighted average: the suggested lists results for the F1-measure indicate a difference of at most 3% between the multiclass and weighted average, compared to the subscribed lists that has a difference of more than 20%. In the subscribed lists data set, the 4 largest classes achieve high precision and recall scores. However, the precision and recall for a number of the small classes are low. As a result, when weighing the precision, recall and F1 scores equally the average scores are much lower. The difference in class sizes in the suggested lists data set is smaller than that of the subscribed lists data set. Therefore, the impact of the poorly performing smaller classes is not as large as for the subscribed lists data set.

We observe the previous result because libSVM attempts to achieve the best average prediction across all classes. In both data sets, this leads to a bias towards the larger classes (Batuwita and Palade, 2012).

Finally, when comparing these results to those achieved by libSVM in the previous section, the results show similar performance. In particular, the best libSVM result is comparable to the best kLog result. We can thus conclude that the multiple-word model is an adequate representation, since it achieves similar performance to the best libSVM result.

We expand on the best-performing results for each data set by including the precision, recall and F1-measure for each class as shown in Tables 6.10 and 6.11. The worst-performing categories are highlighted in grey. In Table 6.10, these categories are `entertainment`, `staff_picks` and `funny`. In Table 6.11, these categories are `social_media` and `marketing`. In the next iterations of

the kLog models, we will analyse the results to see if there are noticeable improvements for these categories.

	Precision	Recall	F1
art_design	0.830	0.638	0.721
books	0.706	0.800	0.750
business	0.512	0.447	0.477
charity	0.648	0.821	0.724
entertainment	0.111	0.036	0.055
faith_and_religion	0.859	0.775	0.815
family	0.875	0.538	0.667
fashion	0.898	0.786	0.838
food_drink	0.825	0.800	0.812
funny	0.400	0.051	0.091
government	0.818	0.391	0.529
health	0.956	0.915	0.935
music	0.688	0.835	0.754
mlb	0.912	0.768	0.834
news	0.518	0.604	0.558
nba	0.766	0.852	0.807
nhl	0.852	0.754	0.800
pga	0.916	0.923	0.920
science	0.923	0.735	0.818
sports	0.697	0.426	0.529
staff_picks	0.224	0.322	0.264
technology	0.591	0.531	0.559
television	0.526	0.837	0.646
travel	0.955	0.913	0.933
twitter	1.000	0.878	0.935
us_election_2012	0.838	0.646	0.729
multiclass avg	0.725	0.655	0.673
weighted avg	0.713	0.699	0.693

Table 6.10: Per-class results for the suggested lists data set using the multiple-word model.

To summarise the results of our kLog word models, we saw comparable performance to that of our originally proposed user document representation techniques, namely TF-IDF and LDA.

	Precision	Recall	F1
all_winter_olympics	0.948	0.830	0.885
design	0.922	0.747	0.826
food	0.833	0.357	0.500
health	0.700	0.438	0.538
marketing	0.000	0.000	0.000
most_influential_in_tech	0.538	0.476	0.505
movies	0.857	0.735	0.791
non_profits	0.727	0.649	0.686
programmers	0.593	0.679	0.633
quotes	0.957	0.865	0.909
social_media	0.519	0.226	0.315
team	0.823	0.917	0.868
tech_news_brands	0.817	0.851	0.834
travel	0.945	0.876	0.909
multiclass avg	0.727	0.617	0.657
weighted avg	0.790	0.793	0.788

Table 6.11: Per-class results for the subscribed lists data set using the multiple-word model.

	Precision	Recall	F1
	Suggested lists		
	threshold-topic		
multiclass avg	0.312	0.311	0.308
weighted avg	0.345	0.366	0.351
	lda-topic		
multiclass avg	0.004	0.038	0.007
weighted avg	0.011	0.106	0.020
	Subscribed lists		
	threshold-topic		
multiclass avg	0.438	0.389	0.404
weighted avg	0.623	0.647	0.630
	lda-topic		
multiclass avg	0.021	0.071	0.032
weighted avg	0.083	0.289	0.129

Table 6.12: The performance of the threshold-topic model and lda-topic model on both data sets.

6.3.3.2 Topics

Table 6.12 summarises the results achieved for the threshold-topic model and lda-topic model. Overall the results are poor, and in comparison to the multiple-word model there is a large performance decrease. By comparing the two models, we see that the threshold-topic model outperforms the lda-topic model.

In the results of the tfidf-word model, we concluded that the graph structure is important to the performance of a kLog model. In the lda-topic model, each graphicalisation of an interpretation consist of an entity node and 30 nodes that correspond to 30 topics. By disregarding the topic values, these graphicalisations are identical in structure. Therefore, only the topic values are a discriminating factor between two graphicalisations. We have previously observed that kLog struggles to effectively use these real-valued components, which contributes to the poor performance of the lda-topic model.

The results achieved for the threshold-topic model are encouraging, but limited by the number of topics larger than a threshold to provide discriminating information for each user document.

The evidence suggests that what LDA captures cannot be used or that kLog uses word information that is not captured by LDA. As a result the available structure in a graphicalisation is less than that of the word models, which can lead to a performance decrease. Therefore, the extra step of training a topic model on documents is unnecessary because kLog captures adequate information from the words in a document.

6.3.3.3 Social graph

Tables 6.13 and 6.14 show the results for the four social graph models on the suggested lists and subscribed lists data sets respectively.

The results in Table 6.13 indicate that the membership models capture better information to discriminate between user documents than the friendship models. Comparing the F1 scores of the membership models to the friendship models, we observe an increase of more than 10% for both the complex and simple variant. By comparing both membership models to the previous best, multiple-word model, we see an increase of 10% in the F1 score. Furthermore, the complex-membership model and simple-membership model achieve similar results on the suggested lists data set.

	Precision	Recall	F1
	complex-friendship		
multiclass avg	0.728	0.689	0.702
weighted avg	0.741	0.734	0.731
	simple-friendship		
multiclass avg	0.676	0.666	0.666
weighted avg	0.699	0.698	0.694
	complex-membership		
multiclass avg	0.871	0.823	0.837
weighted avg	0.872	0.863	0.862
	simple-membership		
multiclass avg	0.934	0.795	0.844
weighted avg	0.901	0.846	0.854

Table 6.13: The results of the four social graph information model variants on the suggested lists data set.

Table 6.14 shows the results for the subscribed lists data set. Similarly to the suggested lists data set, the two membership models outperform all the previous kLog models in this section. Comparing the F1 scores of the two membership models to the multiple-word model, we see an increase of more than 10%. Furthermore, comparing the F1 score of the complex-membership model to the simple-membership model, we observe an increase of 3% in the multiclass average. Although this increase is small, it may indicate that the complex-membership model is slightly better than the simple-membership model in classifying user documents in the smaller classes.

Tables 6.15 and 6.16 show the results of the complex-membership model for each class in the suggested lists data set and subscribed lists data set. By comparing these tables to those of the multiple-word model (Tables 6.10 and 6.11), we can identify any improvements. Each table includes the word-F1 column, which contains the previous best multiple-word F1 score for the classes. Furthermore, the classes that have previously performed poorly are highlighted in orange and classes that still need improvement are shown in grey.

For the suggested lists data set (Table 6.15) each of the three classes, `entertainment`, `staff_picks` and `funny`, show an improvement in performance. Therefore, we can conclude that the complex-membership model is useful for distinguishing between classes. For the `government` class (highlighted in grey), we observe a small performance decrease. Based on the result, a combination of the multiple-word model and complex-membership model may

	Precision	Recall	F1
	complex-friendship		
multiclass avg	0.785	0.685	0.723
weighted avg	0.846	0.842	0.840
	simple-friendship		
multiclass avg	0.749	0.696	0.717
weighted avg	0.836	0.830	0.832
	complex-membership		
multiclass avg	0.896	0.755	0.792
weighted avg	0.937	0.939	0.934
	simple-membership		
multiclass avg	0.909	0.710	0.762
weighted avg	0.952	0.948	0.943

Table 6.14: The results of the four social graph information model variants on the subscribed lists data set.

improve its result.

Table 6.16 shows the corresponding results for the subscribed lists data set. The results show a large increase in the F1 score of the `social_media` class, and an 11% increase for the `marketing` class. Therefore, the complex-membership model is better than the multiple-word model at distinguishing between user documents for the classification task. The `quotes` class (highlighted in grey), showed a slight performance decrease. Again, a combination of the two models may improve the result.

In conclusion, using the list memberships social graph information to construct a kLog model provides the best classification performance of the models we considered, for both the suggested and subscribed lists data set.

6.4 kLog: A combined model for content and graph classification

6.4.1 Purpose

In the previous section, we created separate kLog models for words, topics and social graph information. The results showed that the membership models performed the best, but the evidence suggests that there may be room for improvement by including the information from the multiple-word model. This

	Precision	Recall	F1	word-F1
art_design	0.968	0.884	0.924	0.721
books	0.944	0.850	0.895	0.750
business	0.889	0.681	0.771	0.477
charity	0.912	0.929	0.920	0.724
entertainment	0.485	0.582	0.529	0.055
faith_and_religion	0.955	0.887	0.920	0.815
family	0.967	0.744	0.841	0.667
fashion	0.926	0.893	0.909	0.838
food_drink	0.965	0.846	0.902	0.812
funny	0.692	0.462	0.554	0.091
government	0.875	0.304	0.452	0.529
health	0.977	0.915	0.945	0.935
music	0.821	0.932	0.873	0.754
mlb	0.939	0.968	0.953	0.834
news	0.537	0.750	0.626	0.558
nba	0.925	0.982	0.953	0.807
nhl	0.984	1.000	0.992	0.800
pga	1.000	0.992	0.996	0.920
science	0.818	0.918	0.865	0.818
sports	0.881	0.685	0.771	0.529
staff_picks	0.774	0.747	0.760	0.264
technology	0.884	0.776	0.826	0.559
television	0.768	0.929	0.841	0.646
travel	0.978	0.978	0.978	0.933
twitter	1.000	0.878	0.935	0.935
us_election_2012	0.792	0.875	0.832	0.729
multiclass avg	0.871	0.823	0.837	0.673
weighted avg	0.872	0.863	0.862	0.693

Table 6.15: The results for the complex-membership model as calculated for each individual class on the suggested lists data set.

	Precision	Recall	F1	word-F1
all_winter_olympics	1.000	0.977	0.989	0.885
design	0.989	0.947	0.968	0.826
food	1.000	0.429	0.600	0.500
health	1.000	0.375	0.545	0.538
marketing	0.500	0.062	0.111	0.000
most_influential_in_tech	0.769	0.825	0.796	0.505
movies	0.911	0.837	0.872	0.791
non_profits	0.941	0.865	0.901	0.686
programmers	0.952	0.994	0.973	0.633
quotes	0.849	0.865	0.857	0.909
social_media	0.784	0.468	0.586	0.315
team	0.987	0.992	0.989	0.868
tech_news_brands	0.895	0.948	0.921	0.834
travel	0.972	0.982	0.977	0.909
multiclass avg	0.896	0.755	0.792	0.657
weighted avg	0.937	0.939	0.934	0.788

Table 6.16: The results for the complex-membership model as calculated for each individual class on the subscribed lists data set.

experiment evaluates such combined models.

6.4.2 Experimental method

Chapter 4 introduced the complex-membership-word model (Figure 4.12) and the simple-membership-word model (Figure 4.13). As with our previous kLog experiments, we use the NSPDK for generating features, and the libSVM linear kernel to train a classifier on these features. For both models, we set the libSVM cost parameter $C = 1000$.

The graphicalisation of the simple-membership-word model consists of only one document entity. As a result, its graphicalisation is a star. Therefore, for this model, the only sensible values for the distance and radius are 0 and 1 respectively.

The graphicalisation of the complex-membership-word model consists of connections between multiple document entities. Therefore, as with our previous complex models, we set the distance and radius equal to 2 and 3 respectively.

6.4.3 Results

The results of the complex-membership-word model and simple-membership-word model are summarised in Table 6.17. This table shows the performance of each model on the suggested lists data set and subscribed lists data set.

Comparing the F1 scores of the complex model to that of the simple model, we observe little difference in performance. Furthermore, the complex-membership-word and simple-membership-word models achieve similar results to those of the complex-membership and simple-membership models respectively.

	Precision	Recall	F1
	Suggested lists		
	complex-membership-word		
multiclass avg	0.871	0.823	0.837
weighted avg	0.872	0.863	0.862
	simple-membership-word		
multiclass avg	0.931	0.790	0.841
weighted avg	0.899	0.842	0.851
	Subscribed lists		
	complex-membership-word		
multiclass avg	0.894	0.752	0.789
weighted avg	0.935	0.938	0.933
	simple-membership-word		
multiclass avg	0.909	0.716	0.769
weighted avg	0.953	0.949	0.944

Table 6.17: A summary of the scores achieved by the complex-membership-word and simple-membership-word models on both data sets.

Overall, the combined models do not show any appreciable improvement over the membership models of Section 6.3. We thus conclude that the words associated with each document do not contribute additional information, which kLog can exploit, to improve the performance of the membership models for the user document classification task.

6.5 Clustering short text documents using kLog features

6.5.1 Purpose

In Section 6.1, we discussed our clustering approach on short text documents to construct topic-based Twitter lists. The results indicated that using tweet content could provide a good initial configuration of Twitter users in topic-based clusters. Next, we performed the same task with a classification approach. kLog was selected as the classification system, due to its ability to incorporate graph structure in the feature construction process. The results showed good performance for the classification task with tweet content, but the list memberships graph information performed the best. This experiment uses the extracted feature vectors from the complex-membership-word model as feature vector when performing the clustering task.

6.5.2 Experimental method

As discussed in Chapter 4, kLog provides us with feature vectors that represent user documents. These feature vectors can be used for our clustering task. kLog is built for supervised classification tasks and as such the clustering task uses data incorporating supervised information. Using supervised information in the feature construction process would effectively invalidate the clustering task. However, after the graphicalisation of an interpretation, the vertices that correspond to the prediction task are removed. The features are then constructed from these so-called *mutilated* graphs. For the complex-membership-model, the vertices in the graphicalisation that corresponds to the `category` relation are removed. The resulting features are thus not dependent on the `category` of each document. We verified this by generating features with the category information and comparing it to the features generated when random categories were assigned to each user document. The results showed no differences between the two sets of features.

For the clustering task, we use the kLog features from the complex-membership-word model for the suggested list data set and the subscribed list data set. We use the NSPDK with $d = 2$ and $r = 3$, to construct the feature vectors from the graphicalisations. Furthermore, we use two clauses from the

kLog Prolog module to extract and save the features vectors to a text file. We use the `generate_all` clause to generate features for each user document, and `save_as_libsvm_file` to save these features.

The saved features are then used as input to our clustering approach. We use the five similarity measures described in Chapter 3, to calculate the similarity between user documents on the feature vectors. The resulting similarity matrices are then used as input for the AP and k-means clustering algorithms. We evaluate the clustering task for 5, 10, 15, 20, 25 and 30 clusters. Similarly to the clustering approach in Section 6.1, we set the damping factor for AP to $\lambda = 0.9$, the maximum number of iterations to 1000 and no update to the local result at 30 iterations. For k-means, we set the maximum number of iterations to 300.

6.5.3 Results

Tables 6.18 and 6.19 presents some results for the suggested lists and subscribed lists data sets respectively. For the suggested lists data set and subscribed lists data sets the results are shown for $\{25, 30\}$ and $\{10, 15\}$ clusters respectively. Furthermore, for those similarity measures that do not convergence for a fixed number of clusters, the table cells are shown in grey. Overall, compared to the results achieved in Section 6.1, the performance is poor.

Given results from Section 6.1 for comparison, an interesting result is that for both data sets the average Kullback-Leibler divergence in combination with k-means achieves the best result. This indicates that the properties of the kLog feature vectors differ from the LDA and TF-IDF feature vectors. The results indicate that the similarity measures do not adequately capture the similarity between documents for our clustering approach. As a result, AP struggles to partition the data points into a small number of clusters.

We conclude that features generated by kLog is not suitable for our clustering approach.

	Adjusted Normalised Information Distance				
Representation + clusters	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means				
kLog + 25 clusters	0.7957	0.7214	0.7801	0.7129	**0.6713**
kLog + 30 clusters	0.7594	0.6974	0.7563	0.7049	**0.6633**
	Affinity Propagation				
kLog + 25 clusters	0.7334	**0.7012**		0.7011	0.8413
kLog + 30 clusters	0.6857	**0.6849**		0.6889	0.8408

Table 6.18: A comparison of the ANID results, for the suggested lists kLog data set, with 25 and 30 clusters.

	Adjusted Normalised Information Distance				
Representation + clusters	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means				
kLog + 10 clusters	0.7958	0.8140	0.7164	0.8141	**0.6783**
kLog + 15 clusters	0.8139	0.7957	0.7419	0.7843	**0.6765**
	Affinity Propagation				
kLog + 10 clusters	0.9536			**0.7024**	
kLog + 15 clusters	0.9490			**0.7211**	0.9232

Table 6.19: A comparison of the ANID results, for the subscribed lists kLog data set, with 10 and 15 clusters.

6.6 Clustering short text documents using social graph information

6.6.1 Purpose

In Section 6.3, the simple-membership model (see Figure 4.11) performed on par with the complex-membership model (see Figure 4.10). A difference between these two models is the way in which the social graph information is included in the graph structure. The complex-membership model effectively incorporates the social graph information by creating links between user documents that are members of the same Twitter list (list memberships). In contrast, the simple-membership model only uses the list membership information to assign a list name to a user document. We can use the simple-membership model as basis for a clustering approach.

6.6.2 Experimental method

We retrieve the first 100 list memberships for each user, in the suggested list data set and subscribed list data set. These 100 list memberships exclude the lists in which we are attempting to cluster users. We use the name of each list to construct a word array of list names that represent each user. The TF-IDF is calculated for each user on the user's word array, followed by LDA for 5, 15 and 30 topics.

The similarities between these user list documents are calculated using the five similarity measures of Chapter 3, yielding five similarity matrices, which are then used as input to the k-means and AP clustering algorithms. This experiment is repeated for 5, 10, 15, 20, 25, and 30 clusters.

Furthermore, for AP we set the damping factor to $\lambda = 0.9$, the maximum number of iterations to 1000 and no update to the local result at 30 iterations. For k-means, we set the maximum number of iterations to 300.

6.6.3 Results

Tables 6.20 and 6.21 show the best-performing results for the suggested lists and subscribed lists data sets respectively. Furthermore, in Figures 6.3 and 6.4, for the cosine similarity (the measure achieving the best result in each case), we show a graph plotting the ANID score for all the cluster sizes.

	Adjusted Normalised Information Distance				
Representation	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means + 25				
LDA5	0.5453	0.5577	0.5685	0.5640	**0.5392**
LDA15	0.3008	0.2942	0.5559	**0.2895**	0.5546
LDA30	0.3498	0.2501	0.5653	**0.2369**	0.5356
TF-IDF	0.7959	**0.2466**	0.5907	0.3386	0.7303
	Affinity Propagation + 25				
LDA5	**0.5361**	0.5601	0.5993	0.5502	0.6332
LDA15	**0.2818**	0.2918	0.4885	0.2875	0.6701
LDA30	0.2321	0.2297	0.6081	**0.2275**	0.7457
TF-IDF	0.4273	**0.2139**	0.9834	0.2482	0.9211

Table 6.20: Suggested lists: ANID results for 25 clusters.

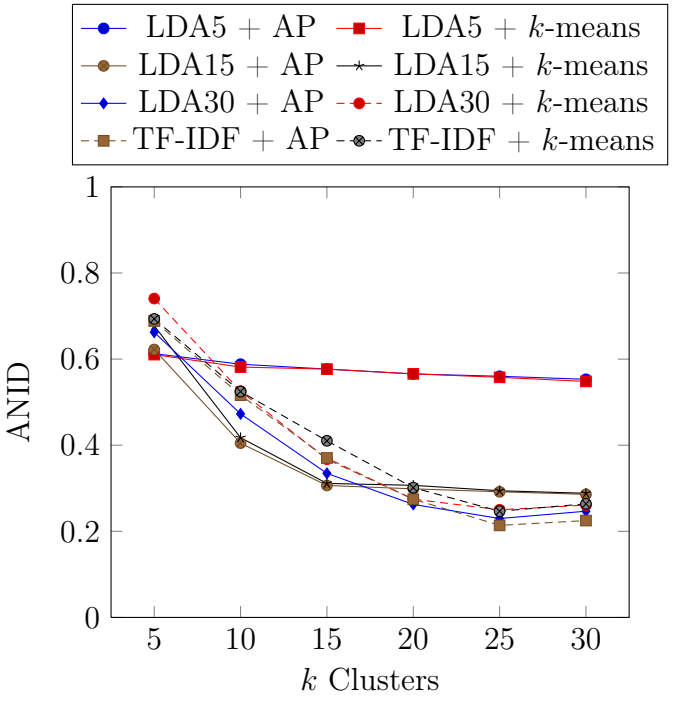


Figure 6.3: Suggested lists data set: ANID obtained using the cosine similarity plotted against the number of clusters for each combination of representation and clustering algorithm.

Table 6.20 shows that the cosine similarity in combination with TF-IDF for 25 clusters, found with AP, performs the best. The plot of the cosine similarity, in Figure 6.3, indicates that LDA5 + AP and LDA5 + k-means perform the worst.

An interesting result is the performance of LDA15: for both clustering algorithms the result improves from 5 to 15 clusters; however, from 20 to 30 clusters the result stays approximately the same. The same phenomenon is observed for LDA30, reaching a minimum at 25 clusters for both k-means and AP. These results support our earlier hypothesis that LDA performs better for a clustering task, if the number of trained topics is equal or larger than the actual number of classes.

For the subscribed lists data set, Table 6.21, the best result is achieved for AP with 10 clusters in combination with cosine similarity and LDA30. The best result in all cases, except TF-IDF + k-means, is achieved at 10 clusters. Beyond 10 clusters, the ANID tends to decrease. Again, the performance of LDA5 does not increase by the same magnitude as LDA30 and LDA15 around 10 clusters, which again suggests that LDA must be trained with a number of

topics larger than or equal to the actual number of classes.

	Adjusted Normalised Information Distance				
Representation	Euclidean	Cosine	Pearson	Jaccard	KL
	k-means + 10				
LDA5	0.4812	**0.4501**	0.5200	0.4507	0.5225
LDA15	0.4748	0.4138	0.6188	**0.4114**	0.5737
LDA30	0.4995	**0.4394**	0.5826	0.4651	0.5383
TF-IDF	0.9492	**0.5631**	0.6545	0.6086	0.5777
	Affinity Propagation + 10				
LDA5	0.4750	**0.4433**	0.6293	0.4450	0.0000
LDA15	0.4208	0.4138	0.7672	**0.3924**	0.8377
LDA30	0.4568	**0.3853**	0.7417	0.4469	0.8684
TF-IDF	0.7047	**0.5037**	0.0000	0.5329	0.9026

Table 6.21: Subscribed lists: ANID results for 10 clusters.

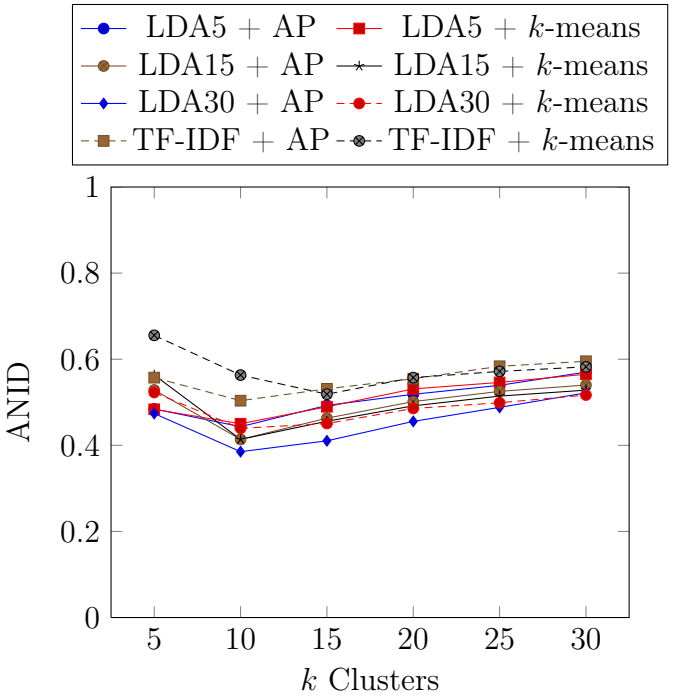


Figure 6.4: Subscribed lists data set: ANID obtained using the cosine similarity plotted against the number of clusters for each combination of representation and clustering algorithm.

On our data sets AP tends to outperform k-means in general, although in some cases the difference in performance is minor. The cosine similarity

achieves the best result, for both data sets, but for each representation the best similarity measures varies. However, this variation is usually small. As a result, this does not impact the clusters found to a large extent.

6.6.3.1 Performance comparison

In Section 6.1, we evaluated the document clustering approach on a Twitter user's tweet content. In this experiment, we performed the same evaluation but replaced the user's tweet content with list membership information. Therefore, each user was represented by the labels of each user's list memberships. Overall, the results indicate a large improvement in performance for both data sets. Table 6.22 compares the results in Section 6.1 to those achieved in this section in terms of the purity, entropy and ANID (see Chapter 3).

	Purity	Entropy	ANID
	Suggested lists		
Tweet content	0.5947	0.4406	0.4999
List memberships	0.8064	0.1941	0.2139
	Subscribed lists		
Tweet content	0.6162	0.4652	0.6117
List memberships	0.7541	0.2654	0.3853

Table 6.22: A comparison of the purity, entropy and ANID for the best results achieved using tweet content and list memberships, for both data sets.

This table shows an improvement of the ANID for the suggested lists data set from 0.4999 to 0.2139, an improvement of 28%. Furthermore, the ANID for the subscribed lists data set shows a 22% improvement from 0.6117 to 0.3853. By examining the purity and entropy obtained for both data sets, we observe a similar improvement to that of the ANID.

As a result, there is strong evidence that list memberships are a better indicator than tweet content to construct topic-based Twitter lists.

Chapter

7

Conclusion

This thesis investigated the problem of constructing topic-based Twitter lists. We introduced a clustering and classification approach to solve this problem, which we evaluated by applying these approaches to data extracted from Twitter. We employed two Twitter data sets consisting of Twitter's suggested categories and popular Twitter lists. We found that representing a Twitter user by his/her list membership information achieved the best results for both the classification and clustering task. This chapter concludes our work by summarising our results, making recommendations on the best approach to use if the data qualities are known, and suggesting future research in this area.

7.1 Investigation and results

We investigated automated procedures to simplify the task of constructing topic-based Twitter lists. To address this task, we proposed clustering and classification approaches.

The clustering approach provides a solution in the case where a user has not created lists of people in his network. In contrast, the classification approach is for the case where a user has created a number of lists with a subset of people in his network.

The proposed clustering approach consists of four steps: using Twitter data to represent a user, representation by a feature vector, document similarity calculations and the application of clustering algorithms. We chose to represent each user as a single document, called the user document. Two

different representations of this user document were used; initially we encoded the user's tweet contents and later on the user's list membership information. The user document conversion to a feature vector was performed using term frequency-inverse document frequency (TF-IDF) and latent Dirichlet allocation (LDA). We evaluated five similarity measures: cosine similarity, Euclidean distance, extended Jaccard coefficient, average Kullback-Leibler divergence and Pearson correlation coefficient. The clustering algorithms, k-means and affinity propagation (AP), were used to find the topic-based clusters.

We found that LDA tends to outperform TF-IDF as long as the number of trained topics were equal to or larger than the actual number of classes in the data set. Furthermore, the cosine similarity usually achieved the best results for our clustering experiments: comparing all the experimental combinations, the results were dominated by the cosine similarity and extended Jaccard coefficient. The performance of the average Kullback-Leibler divergence and Pearson correlation coefficient are particularly poor for this clustering approach. In terms of the clustering algorithms, AP outperformed k-means in most of the clustering experiments. Finally, comparing the results for a user document represented with tweets to the list memberships representation, we observe a large improvement in the clustering result. We conclude that list membership information are a better indicator than tweet content to which topic-based list a user belongs.

The classification approach consisted of experiments using libSVM and kLog. We used LibSVM, a library for support vector classification, on the TF-IDF and LDA representations to find a classification baseline to which kLog could be compared. We selected kLog, a logical and relational language for kernel-based learning, for the classification task due to its ability to model social graph connections between users. Therefore, we would be able to extend the simple content-based approach, as used for clustering, to include links between users from the graph data.

We evaluated different models with the kLog system. In particular, we considered modelling users in terms of words, topics and social graph information. The results showed that the word models performed on par with the libSVM baseline, showing that a simple kLog model achieves similar results to that of TF-IDF and LDA in terms of tweet representation. However, we found that the models of list membership information had the best performance overall.

The results achieved for both the classification and clustering task indicate that representing users by their list membership information performs best when constructing topic-based Twitter lists. In other words it seems that a Twitter user's topic-based expertise is better represented by his/her list membership information than his/her tweet content.

In the course of evaluating methods to construct topic-based Twitter lists, we achieved a number of results related to our main objective. These secondary results influenced the evolution of our clustering and classification approaches, and are therefore important to consider.

We found that with careful consideration of the feature construction process, it is possible to incorporate social graph data in the clustering process. In particular, assigning the list membership information to each user in the representation process effectively provides information about the connections between users.

The clustering approach has also been applied using the feature vectors extracted from the various kLog models. We found that these features yielded poor performance for our clustering approach.

In terms of the kLog classification approach, we showed that the simple multiple-word model performed on par with the TF-IDF and LDA representation in libSVM. We also evaluated kLog on two distinct social graph information sets, the friendships and list memberships data. The list memberships were found to perform the best, but the results achieved for friendship information showed an improvement over the multiple-word model.

Furthermore, in terms of the structure of a graphicalisation, the results showed that for Twitter data there are little to no differences between models generating star graphs and those generating more complex graphs. The complex graph refers to those with explicit links defined between user documents.

Finally, based on our earlier kLog results, we created a model combining the multiple-word model and membership models. The results indicated that the membership information is the major contributing factor to classification performance, with the multiple-word model contributing little to no extra information.

7.2 Recommendations

In this section, we give two recommendations on the implementation of a practical system for the automated construction of topic-based Twitter lists.

For the classification approach, the results showed that membership models performed the best, followed by friendships and finally tweet content models. Similarly, in the clustering approach we showed that using list membership information outperformed tweet content. We did not evaluate the friendships in the clustering approach, but a representation similar to the membership models may provide good results.

Therefore, we recommend that in any system to extract the topic-based expertise of a Twitter user, the user representation should be considered in the following order. Firstly, a user should be represented by his/her list memberships, failing that by his/her friendships and as a last resort the user's tweet content.

As expected, we see that classification performs better than clustering. The necessary data for classification is not always available, and thus a combination of clustering and classification should be used. The clustering approach should be used when there are no list preference data available for a user. In the case where partial list preference data are available, the classification approach should be used.

7.3 Limitations

The results show that list memberships outperform tweets and friendships in both the clustering and classification task. Furthermore, for the classification task we saw friendships outperform tweets.

In the construction of the suggested lists and subscribed lists data sets, we chose distinct lists/categories with minimal overlap between users. This choice led to well-separated communities of Twitter users in each data set. Therefore, using the list memberships and friendships from these well-separated communities may have influenced the quality of the results. If we apply these techniques to a single Twitter user's social graph, we may find that users are not well-separated in distinct communities. As such the effectiveness of list memberships and friendships at finding topic-based Twitter lists may be

diminished.

The choice of data sets was necessary to effectively evaluate the performance of the chosen techniques. However, to comprehensively evaluate the impact of connections between Twitter users, another data set representing the networks of typical Twitter users would be needed.

7.4 Future work

Based on the above recommendations, a number of areas may be considered for future work. Users may have different amounts of data available for list memberships, friendships and tweets. A combination of those three data attributes must therefore be considered to accurately represent these users. To accurately represent a Twitter user with a combination of these attributes is not a trivial task, and could provide significant improvements over the current approaches.

Furthermore, the clustering task is unsupervised and thus the clusters are not labelled. Extending the clustering approach to recommend labels for each cluster can improve the system for an end user.

If we consider future work for the systems and techniques used, we see that the selection of cluster size and the number of topics have a significant impact on the performance of the clustering approach. The selection of these is of paramount importance to the performance of any such system.

In terms of kLog, we discussed the extraction of feature vectors for the clustering task. We gave a brief discussion on the feature vector construction procedure and for future work the suitability of kLog for unsupervised learning can be evaluated. Our clustering approach on kLog feature vectors showed poor results; however kernel-based and other clustering techniques could still be considered.

Furthermore, we found that it is difficult to effectively use real-valued components in our kLog models. For future work, different kernels can be considered that provides more weight to the real-valued components.

In this thesis, we mainly considered kLog for the classification task. However, similar performance to kLog may be achieved with a suitable SVM. We leave the evaluation of SVMs using list memberships and friendships for future work.

Finally, our results showed little to no difference in the performance of a star graph compared to a more complex graph. This result may only be consistent with the characteristics of our data sets and further study is necessary to find the factors affecting the result.

7.5 Summary

We initiated this research to investigate the automated construction of topic-based Twitter lists. We identified two scenarios, which led to a clustering and a classification approach. We evaluated these approaches and found that with careful consideration of features, accurate results can be achieved when automatically constructing topic-based Twitter lists.

List of References

Amine, A., Elberrichi, Z. and Simonet, M. (2010). Evaluation of text clustering methods using wordnet. *International Arab Journal of Information Technology*, vol. 7, no. 4, p. 351.

Arbib, M. (2002). *The Handbook of Brain Theory and Neural Networks: Second Edition*. Bradford Books. MIT Press.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In: *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics.

Banerjee, A., Dhillon, I., Ghosh, J. and Sra, S. (2006). Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, vol. 6, no. 2, pp. 1345–1382.

Batuwita, R. and Palade, V. (2012). *Imbalanced Learning: Foundations, Algorithms, and Applications*, chap. Class Imbalance Learning Methods for Support Vector Machines. Wiley.

Bawden, D. and Robinson, L. (2009). The dark side of information: overload, anxiety and other paradoxes and pathologies. *Journal of Information Science*, vol. 35, no. 2, pp. 180–191.

Bernstein, M., Suh, B., Hong, L., Chen, J., Kairam, S. and Chi, E. (2010). Eddi: interactive topic-based browsing of social status streams. In: *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, pp. 303–312. ACM.

Bigi, B. (2003). Using kullback-leibler distance for text categorization. *Advances in Information Retrieval*, pp. 305–319.

Bird, S., Klein, E. and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Series. O'Reilly.

Blei, D. and Lafferty, J. (2009). Topic models. *Text mining: classification, clustering, and applications*, pp. 71–94.

Blei, D., Ng, A. and Jordan, M. (2003). Latent Dirichlet allocation. *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022.

Borgs, C., Chayes, J., Karrer, B., Meeder, B., Ravi, R., Reagans, R. and Sayedi, A. (2010). Game-theoretic models of information overload in social networks. *Algorithms and Models for the Web-Graph*, vol. 6516, pp. 146–161.

Bousquet, O., Boucheron, S. and Lugosi, G. (2004). Introduction to statistical learning theory. *Advanced Lectures on Machine Learning*, pp. 169–207.

Chang, C. and Lin, C. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, J., Nairn, R., Nelson, L., Bernstein, M. and Chi, E. (2010). Short and tweet: Experiments on recommending content from information streams. In: *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pp. 1185–1194. ACM.

Chen, P. (1976). The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36.

Cherkassky, V. and Mulier, F. (2007). *Learning from Data: Concepts, Theory, and Methods*. John Wiley & Sons.

Costa, F. and De Grave, K. (2010). Fast neighborhood subgraph pairwise distance kernel. In: *Proceedings of the 26th International Conference on Machine Learning*, pp. 255–262.

De Raedt, L. (2008). *Logical and Relational Learning*. Cognitive Technologies. Springer.

De Villiers, F., Hoffmann, M. and Kroon, S. (2012). Unsupervised construction of topic-based twitter lists. In: *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing*, pp. 283–292. IEEE.

Dutton, D.M. and Conroy, G.V. (1997). A review of machine learning. *The Knowledge Engineering Review*, vol. 12, no. 04, pp. 341–367.

Elman, J. (2010). The power of suggestions. `http://blog.twitter.com/2010/01/power-of-suggestions.html`. [Online: Accessed 20-May-2012].

Ertoz, L., Steinbach, M. and Kumar, V. (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: *Proceedings of the Third SIAM International Conference on Data Mining*, vol. 112. SIAM.

Everitt, B., Landau, S., Leese, M. and Stahl, D. (2011). *Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley.

Frasconi, P., Costa, F., De Raedt, L. and De Grave, K. (2012). kLog: A language for logical and relational learning with kernels. *Computer Research Repository*, vol. abs/1205.3981.

Frey, B. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*, vol. 315, no. 5814, pp. 972–976.

Fulekar, H. (2009). *Bioinformatics: Applications in Life and Environmental Sciences*. Springer.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning*. Adaptive Computation and Machine Learning. MIT Press.

Ghosh, S., Sharma, N., Benevenuto, F., Ganguly, N. and Gummadi, K. (2012). Cognos: crowdsourcing search for topic experts in microblogs. In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 575–590. ACM.

Gordon, A. (1999). *Classification, 2nd Edition*. Monographs on Statistics and Applied Probability. Taylor & Francis.

Greene, D., O'Callaghan, D. and Cunningham, P. (2012). Identifying topical twitter communities via user list aggregation. *Computer Research Repository*, vol. abs/1207.0017.

Greene, D., Reid, F., Sheridan, G. and Cunningham, P. (2011). Supporting the curation of twitter user lists. *Computer Research Repository*, vol. abs/1110.1349.

Hodge, V. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126.

Hoffman, M., Blei, D. and Bach, F. (2010). Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, vol. 23, pp. 856–864.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, vol. 42, no. 1, pp. 177–196.

Hong, L. and Davison, B. (2010). Empirical study of topic modeling in twitter. In: *Proceedings of the First Workshop on Social Media Analytics*, pp. 80–88. ACM.

Huang, A. (2008). Similarity measures for text document clustering. In: *Proceedings of the Sixth New Zealand Computer Science Research Student Conference*, pp. 49–56.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, vol. 2, pp. 193–218.

Jain, A. and Dubes, R. (1988). *Algorithms for clustering data*. Prentice Hall advanced reference series. Prentice Hall.

Jain, A., Duin, R. and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37.

Jain, A., Murty, M. and Flynn, P. (1999). Data clustering: a review. *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323.

Jin, O., Liu, N., Zhao, K., Yu, Y. and Yang, Q. (2011). Transferring topical knowledge from auxiliary long texts for short text clustering. In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 775–784. ACM.

Ketkar, N., Holder, L. and Cook, D. (2009). Empirical comparison of graph classification algorithms. In: *Proceedings of IEEE Symposium on Computational Intelligence and Data Mining, 2009. CIDM'09.*, pp. 259–266. IEEE.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 2, pp. 1137–1143. ACM, Morgan Kaufmann Publishers Inc.

Kotsiantis, S., Zaharakis, I. and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Frontiers in Artificial Intelligence and Applications*, vol. 160, p. 3.

Kraskov, A., Stögbauer, H., Andrzejak, R. and Grassberger, P. (2005). Hierarchical clustering using mutual information. *Europhysics Letters*, vol. 70, p. 278.

Kyriakopoulou, A. and Kalamboukis, T. (2006). Text classification using clustering. In: *Proceedings of ECML-PKDD Discovery Challenge Workshop*, pp. 28–38.

Landauer, T. (2007). *Handbook of latent semantic analysis.* University of Colorado Institute of Cognitive Science Series. Lawrence Erlbaum Associates.

Larsen, B. and Aone, C. (1999). Fast and effective text mining using linear-time document clustering. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 16–22. ACM.

Lin, D. (1998). An information-theoretic definition of similarity. In: *Proceedings of the 15th International Conference on Machine Learning*, vol. 1, pp. 296–304. ACM.

Listorious (2012). Listorious. `http://listorious.com`. [Online: Accessed 9-May-2012].

Liu, H. and Motoda, H. (1998). *Feature selection for knowledge discovery and data mining.* Springer.

Markman, V. (2011 August). Unsupervised discovery of fine-grained topic clusters in Twitter posts. In: *Workshops at the 25th AAAI Conference on Artifical Intelligence.* AAAI.

Markovitch, S. and Rosenstein, D. (2002). Feature generation using general constructor functions. *Machine Learning*, vol. 49, no. 1, pp. 59–98.

Michelson, M. and Macskassy, S. (2010). Discovering users' topics of interest on twitter: A first look. In: *Proceedings of the Fourth Workshop on Analytics for Noisy Unstructured Text Data*, pp. 73–80. ACM.

Millar, J., Peterson, G. and Mendenhall, M. (2009). Document clustering and visualization with latent dirichlet allocation and self-organizing maps. In: *Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference*, vol. 21, pp. 69–74.

Nguyen, T. and Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76.

Pennacchiotti, M. and Gurumurthy, S. (2011). Investigating topic models for social media user recommendation. In: *Proceedings of the 20th International Conference Companion on World Wide Web*, pp. 101–102. ACM.

Perez-Tellez, F., Pinto, D., Cardiff, J. and Rosso, P. (2011). On the difficulty of clustering microblog texts for online reputation management. In: *Proceedings of the Second Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, pp. 146–152. Association for Computational Linguistics.

Pinto, D., Rosso, P. and Jiménez-Salazar, H. (2011). A self-enriching methodology for clustering narrow domain short texts. *The Computer Journal*, vol. 54, no. 7, pp. 1148–1165.

Porter, M. (2006). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, vol. 40, no. 3, pp. 211–218.

Ramage, D., Dumais, S. and Liebling, D. (2010). Characterizing microblogs with topic models. In: *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*. The AAAI Press.

Rangrej, A., Kulkarni, S. and Tendulkar, A. (2011). Comparative study of clustering techniques for short text documents. In: *Proceedings of the 20th International Conference Companion on World Wide Web*, pp. 111–112. ACM.

Salton, G. and McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill computer science series. McGraw-Hill.

Sandhya, N., Sri Lalitha, Y., Govardan, A. and Anuradha, K. (2008). Analysis of similarity measures for text clustering. *International Journal of Data Engineering*, vol. 2, no. 4.

Singh, V., Tiwari, N. and Garg, S. (2011 October). Document clustering using k-means, heuristic k-means and fuzzy c-means. In: *Proceedings of the 2011 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 297–301.

Steinbach, M., Karypis, G. and Kumar, V. (2000). A comparison of document clustering techniques. In: *KDD Workshop on Text Mining*, vol. 400, pp. 525–526. Boston.

Strehl, A., Ghosh, J. and Mooney, R. (2000). Impact of similarity measures on web-page clustering. In: *Workshop on Artificial Intelligence for Web Search*, pp. 58–64. AAAI.

Subhashini, R. and Kumar, V. (2010). Evaluating the performance of similarity measures used in document clustering and information retrieval. In: *Proceedings of*

the First International Conference on Integrated Intelligent Computing, pp. 27–31. IEEE.

Tan, P., Steinbach, M. and Kumar, V. (2006). *Introduction to data mining.* Pearson International Edition. Pearson Addision Wesley.

Torres, G., Basnet, R., Sung, A., Mukkamala, S. and Ribeiro, B. (2009). A similarity measure for clustering and its applications. *International Journal of Electrical, Computer and Systems Engineering*, vol. 3, no. 3.

Verbeke, M., Asch, V.V., Morante, R., Frasconi, P., Daelemans, W. and Raedt, L.D. (2012*a*). A statistical relational learning approach to identifying evidence based medicine categories. In: *Proceedings of the 17th International Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 579–589.

Verbeke, M., Frasconi, P., Van Asch, V., Morante, R., Daelemans, W. and De Raedt, L. (2012*b*). Kernel-based logical and relational learning with kLog for hedge cue detection. *Inductive Logic Programming*, pp. 347–357.

Vinh, N.X., Epps, J. and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854.

Wagner, C., Liao, V., Pirolli, P., Nelson, L. and Strohmaier, M. (2012). It's not in their tweets: Modeling topical expertise of twitter users. In: *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing*, vol. 1, pp. 91–100. IEEE.

Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678.

Xu, T. and Oard, D. (2011). Wikipedia-based topic clustering for microblogs. *Proceedings of the American Society for Information Science and Technology*, vol. 48, no. 1, pp. 1–10.

Ye, N. (2004). *The Handbook of Data Mining.* Human Factors and Ergonomics. Taylor & Francis.

Zhang, S., Zhang, C. and Yang, Q. (2003). Data preparation for data mining. *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 375–381.

Zhang, W., Yoshida, T. and Tang, X. (2011). A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert Systems with Applications*, vol. 38, no. 3, pp. 2758–2765.

Zhao, Y. and Karypis, G. (2004). Empirical and theoretical comparisons of selected criterion functions for document clustering. *Journal of Machine Learning*, vol. 55, no. 3, pp. 311–331.

Zhao, Y., Karypis, G. and Fayyad, U. (2005). Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 141–168.

Zheng, Y., Zhang, L., Ma, Z., Xie, X. and Ma, W. (2011). Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, vol. 5, no. 5.