VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

METODY SÉMANTICKÉ PODOBNOSTI VE FOLKSONOMIÍCH

SEMANTIC SIMILARITY METHODS IN FOLKSONOMIES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
JAN KADLEC

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. MAREK SCHMIDT

BRNO 2010

Abstrakt

Folksonomie jsou nový, uživateli řízený přístup ke klasifikaci a důležitá čast Web 2.0. Jsou také jediným přístupem, který je schopen udržet krok s dnešní rychlostí expanze webu, tím že předá uživatelům odpovědnost za klasifikaci. Pokud folksonomie obsahují dostatečné množství dat, dají se k mnohému využít. Tato práce se zaměřuje na metody sémantické podobnosti ve folksonomiích. Cílem této práce bylo odzkoušet mnohé metody na vzorku tří datových sad - delicious.com, Last.fm a medworm.com. Toto bylo vykonáno za pomocí kotvících dat z WordNetu, Open Directory Project a zdravotně orientované ontologie. Výsledky přinesené touto prací indikují, že metody sémantické podobnosti mohou být použity k úspěšnému měření podobností v mnohých doménách.

Abstract

Folksonomies are new, user driven classification structures and an important part of Web 2.0. Folksonomies are the only one approach that can keep up with todays web expansion rate, by utilizing users as classificators of web's content. Folksonomies, when containing sufficient amount of data, can be exploited in several ways. This particular work concentrates on measures of semantic similarity in folksonomies. The aim of this work is to evaluate several semantic similarity measures on a sample of three datasets - delicious.com, Last.fm and medworm.com. Evaluation was done using grounding data from WordNet, Open Directory Project and medical oriented ontology. Results presented by this work indicate, that measures of semantic similarity can be used to successfully measure the similarities in folksonomies in several domains.

Klíčová slova

folksonomie, tagování, web 2.0, podobnost

Keywords

folksomy, tagging, web 2.0, similarity

Citace

Jan Kadlec: Semantic Similarity Methods in Folksonomies, bakalářská práce, Brno, FIT VUT v Brně, 2010

Semantic Similarity Methods in Folksonomies

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Marka Schmidta

. . . .
Jan Kadlec
July 23, 2010

Poděkování

I would like to sincerely thank all the helpful people from Department of Computer Science, Aalborg University, specifically to Peter Dolog and Fred Durao. And many thanks go to my supervisor Marek Schmidt as well.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Folksonomies, sometimes also referred to as *social tagging*, are relatively new, but extremely fast evolving approach to data classification. Folksonomies are nowadays at the very centre of many Web 2.0 projects. Web 2.0 is a term first used by Tim O'Reilly. He describes the term more closely in [29], and there is also description of how various techniques should evolve from classical web to Web 2.0. Among others, [29] describes evolution from directories(taxonomies) to folksonomies as a part of Web 2.0 creation process.

1.1 What folksonomies are

The exact expression *folksonomy* was first used around 2004 to identify two already existing sites, delicious and Flickr. It is a composition of words *folk* and *taxonomy.* A definition of taxonomy according to [4] is "a study of the general principles of scientific classification" or, which makes more sense in our case, "orderly classification of plants and animals according to their presumed natural relationships". Of course, plants and animals have to be substituted with something that can be organized using "presumed natural relationships". It is, however, not very exact to use the word taxonomy in process of building the word *folksonomy*, because folksonomies tend to overcome limitations of taxonomies and other classification approaches. What are those limitations and how are folksonomies dealing with them will be described in the following articles.

One might try to describe a folksonomy informally as a collection of labels, called tags, assigned to resources by users, with no restrictions applied to what the label represents, except for it being a string. This freedom is both major advantage and disadvantage of folksonomies, when compared to other data classification approaches, like taxonomies. When classifying a resource, one is no longer limited to a certain set of categories and can therefore expand the label set with new label easily. On the other hand, due to no limits to labels, they might not be related to the subject or could just be a noise, or worse,a spam. Despite their downsides, folksonomies are the only approach that can cope with web's expansion rate. By transferring the classifying role from experts to users we lower the degree of precision, but gain incomparably bigger reach of the classifications.

Formal definition of a folksonomy, as described in [24], is as follows: Folksonomy F is a 4-tuple, containing a set of users U, a set of resources R, a set of tags and a set of relations between those - Y. Y is a subset of cartesian product of U, T and R. What exactly a *resource* represents depends on a domain. It might for example be a web page, a music artist, a movie, a picture or a video.

$$F = (U, T, R, Y), Y \subseteq U \times T \times R \tag{1.1}$$

Nowadays, folksonomies are at the center of numerous web pages. Some are just implementations of folksonomies in different domains, some pages use them to improve their functionality, for personalization, keyword predictions and many other applications. Probably the most known folksonomy based services are delicious [6], Last.fm [26], digg [5], Youtube [1] and last but not least, Flickr [14]. Annotated resources range from web pages to pictures. Some web pages do not provide tagging, but users seem to find a way how to tag anyway. This happened in Twitter([34]), where people started tagging their tweets with so called *hash-tags*, by putting a hash character in front of the word they want to tag with(i.e. #politics).

For our experiment we chose delicous, Last.fm a and Medworm [17]. The reason for this choice were mainly grounding data available for the domains covered by these three pages. Of course, there were many other sites to choose from, including Bibsonomy [21] and MovieLens [28] which already had their content available to download. Bibsonomy's resources are partly similar to the ones in delicious, and since experiment with Bibsonomy's dataset has already been done in [24] we decided to try delicious instead. Movielens, which gathers information about movies, would be more than suitable for use with similarity methods but would be very hard to ground. More on a subject of grounding in the chapter4.1.

Folksonomies, when containing sufficient amount of data, can be exploited in several ways. Computing similarity between components of folksomy's sets, clusterizing and inferring taxonomies are some of the practical applications. Some of these approaches will be described in the chapter 2.

1.2 Semantic similarity and similarity measures

In this part we will elaborate on the terms semantic similarity and semantic relatedness. [4] defines "similar" as "having characteristics in common : strictly comparable" and also "alike in substance or essentials", whereas "related" is described as "connected by reason of an established or discoverable relation". This definition also comes from [4]. Additionally, we need to describe the term "semantic" as well. Webster's definition is as follows: "semantic: of or relating to meaning in language". We should state now, that the measures of semantic similarity described in this text do not only capture the language meaning of connections between entities, but other connections as well. Some sources use semantic similarity interchangeably with the term semantic relatedness even when it comes to the measures. There are cases when these two are technically the same thing, however a clear distinction has to be made. In [31] author states, that when two entities are similar, they are also related. From computers' point of view, terms "doctor" and "syringe" are not similar at all, however they are strongly related. On the other hand "doctor" and "nurse" are similar (they share several similar attributes, for example they both work in a hospital, wear similar uniforms, and last but not least, they both have some kind of medical based education) and very related as well.

It is hard to answer the question, whether measures of semantic similarity capture relatedness as well, but since the principle of most of the measures is based on comparing shared attributes of entities, whose similarity we want to measure, we can probably state right now that relatedness, at least in most cases, is not going to be captured.

This work concentrates on semantic similarity measures in folksonomies. These measures often have their origin in classical measures of semantic similarity, which are commonly used to measure similarity of documents and have been developed for this task. Some of the measures can be even used in non changed form.

The aim of this project is to implement various semantic similarity measures and aggregation methods, mostly as described in [24], [15] and [25] and to collect datasets to apply these measures to. Once the outcome has been computed, correlation with measures of similarity in human created classifications or *grounding* for short will be done and we will try to give interpretation of the results, and suggest convenient measures for particular domains.

1.3 How aggregation methods affect similarity measures

Various methods of aggregation have been introduced to better adapt the measures for use with folksonomies. In process of aggregation, information from one of folksonomy's sets is inevitably lost. In our case it is information about users, because we want to concentrate only on similarities between resources and tags. Result of an aggregation is an aggregation matrix. This is also an input of a similarity measure.

Each measure has its basic formula, usually identical to one that would be used in document similarity. This type of formulas is compatible with a so called *projection* aggregation (more on that in the section 5.2). Outcome of this method is a matrix containing only ones and zeros. This aggregation method makes no difference between tagging a resource one or several times with one tag, therefore it does not matter, if only one user annotates the resource with a particular tag or thousand users do. Exactly this kind of behavior is captured by a *distributional* or a *fuzzy* aggregation method(see section 5.3). This method counts the number of occurrences of each tag/resource.

Other approach uses a projection aggregation, but works with probabilities rather than with simple sets. Formulas stay almost the same, but are expanded by log likelihood parts. What would be an intersection in a basic projection case becomes a summation of log likelihoods of members of the intersection. This will be closely explained in the section 5.2. This is also called distributional aggregation. It will be referred to as distributional aggregation using probabilities in the following text.

This of course calls for some changes in the formula. As far as we know, there is no rule to convert measures to work with other aggregations, however some patterns can be seen. We applied those patterns in cases, where no suitable measure already existed or could not be found elsewhere, but results given by these measures might not be correct. These measures, except for the cosine similarity, which is defined on vector space, were not mentioned in [24] either. Since projected measures work with a set representation and distributional with a vector representation, typically an intersection of two sets is "converted" to the dot product of two vectors, and a set length becomes a vector norm.

Whereas with previous aggregation methods there is no distinction as to what user made certain annotations, because every annotation is written to the same matrix, in so called *macro* and *collaborative* aggregations we first compute a similarity for each user and the concluding similarity is a result of a summation. Of course, this is not an aggregation in the same sense as in a projection and distributional case. It is not important, which aggregation we choose for users, it only needs to be identical for each user. Informally, collaborative and macro aggregations do not affect formulas of similarity measures, although some formal adjustments have to be made, especially in case of the collaborative aggregation, which adds "implicit" annotations (more on that in the chapter 5.1).

1.4 How to measure performance of used methods

We can use several approaches in order to capture, how well the similarity methods perform. The best way would surely be to create a set of few very well known tags or resources and let several people assign similarities to all the pairs these sets allows. This would, however, require a lot of human resource, which we were not in possession of. We then had to reside to use such measures that have already been verified by user studies. Since no such study has been done in the domain of folksonomies, other semantic similarity measures in different approaches than folksonomies had to be used; typically, these were taxonomies or ontologies. In the following text, this will be called *grounding*.

Basically, grounding is a process, in which we try to find an intersection as big as possible with folksonomy's dataset and the taxonomy which relates the best to the given folksonomy. Once we have obtained this intersection, we can compute similarities using both semantic similarity measures in the folksonomy and in the taxonomy, resulting in two similarity matrices which we can later correlate. Intersections between the folksonomies' datasets and grounding data are typically only a fraction of an actual size of the original folksonomies, however still big enough to perform the correlations. Correlation techniques are described in the chapter 6.

We have evaluated several measures, using 5 aggregations. For the results, please see the chapter 6.

1.5 Structure of this document

This document is divided into 7 chapters. You are now reading the first introductory chapter, which is followed by the chapter 2, where reader can find a description of work related to this project. The most vital part of the project, the datasets, are described in the chapter 3. Detailed information on grounding techniques and grounding data is in the chapter 4. Chapter 5 is the biggest chapter, explaining aggregation methods, actual similarity measures and information about their implementations. Results of correlations between the measures of semantic similarity and grounding approaches can be found in chapter 6. Finally, project evaluation and suggestions for future work are at the end of this document, in the chapter 7 followed by the appendix, where can reader find a list of used literature along with some figures characterising the datasets along with example similarity vectors.

Chapter 2

Related work

From our point of view, the most important work is [24], which is, as far as we know, one of the only works that have introduced some changes to the usual similarity metrics to better fit them into the domain of folksonomies. We have adapted most of the similarity measures and aggregation methods from this work, however most of these measures existed before, but were not adjusted to fit into the domain of folksonomies. Yet changes done to these measures are not very wide.

Research activity in this domain is indeed vast. One can say that whenever the social tagging is mentioned then the work is somehow related to our subject. Nevertheless we will only describe a small portion of this domain, we will for example describe measures of semantic similarity in taxonomies. Although this might not seem related to our subject, semantic similarity methods in taxonomies are vital for assessing methods in folksonomies, mainly because numerous were validated by user studies.

2.1 Semantic similarity in other structures than folksonomy

First measures of semantic similarity in taxonomies were based on edge counting approach. If we think about *is-a* taxonomy as a graph, such measures were functions of the distance between two nodes. Philip Resnik in [31] proposed measure, that not only exploits structure of taxonomy but utilizes information contained within each concept. Full definition of this can be found in [31], but we can informally say that the lower in taxonomies' structure one gets, the more information concepts in that dept contain. For example, in Wordnet, the uppermost concept is usually called an "entity" and since everything in Wordnet is an "entity", that fact caries no information. Oppositely, the bottommost concepts in a taxonomy are the most special ones(they have no subordinates) and therefore give the highest information. All the information theoretic measures use log probabilities, saying that information carried by term is equal to the negative logarithm of the probability of the term's occurrence. Also, Resnik's measure operates with the function $common(a, b)$ which returns the common concept in taxonomy for terms a, b whose position in the taxonomy is the lowest. As Lin in [22] explains by Wordnet's example: $common(a, b)$ of terms "hill" and "coast" is the term "geological formation". Resnik's measure is defined as:

$$sim_{resnik}(a, b) = \log P(common(a, b)) \tag{2.1}$$

where $I(x)$ is the function returning information held by concept x.

In this work we will use another, similar measure, for grounding purposes. It was introduced by Wu and Palmer in [36]. With respecting all the definitions given in the previous paragraph, its formula is as follows:

$$sim_{WuPalmer}(a,b) = \frac{2 \cdot N_3}{N_1 + N_2 + 2 \cdot N_3} \tag{2.2}$$

where N_1 and N_2 are the numbers of edges to $common(a,b)$ and N_3 is the distance from term $common(a,b)$ to the highest term(i.e. root) of the taxonomy.

Chapter 3

Datasets

In order to be successful with the experiment, we needed quite large amount of data. As described earlier, we chose delicous, Last.fm and Medworm. Since delicous and Last.fm are one of the oldest tagging services, and because several researchers needed their datasets as well, we were able to find already existing datasets, which were free to download, adapt and use. In fact, there are several datasets obtainable for both of these services, but some were more suitable than others, some contained too much data, while some did not contain enough data for our experiment. As for Medworm, since the service itself is quite new and research in medical based social tagging systems is rather sparse, we had to create a new dataset by crawling the page.

3.1 Last.fm dataset

Last.fm is a social music site that describes itself as a "social music revolution". It developed from two separate services, one of which concentrated on creating a personalized music library while the other was a service used for storing information about what music its users listen to on their PCs.

Users tag artists, songs, music events such as concerts or festivals as well as music albums. On top of that, they can join various groups, make friends and so on. The taggable content of the site is either manually generated or created by special process of *scrobbling*. Scrobbling basically means uploading information about music users are listening to on their computers or portable music players. This music will appear on user's profile and can later be tagged.

Although Last.fm offers quite sophisticated API, which can be used to collect tags about resources, the maximum number of tags returned is only 50. This number of tags is not adequate for the task at hand. Moreover information about how each user tagged particular resources would be too hard, if not impossible, to get. We will, however, find a way to use this API to extract artists out of the dataset. Dataset satisfying our needs is an outcome of 6 month-long crawl of Last.fm service and therefore contains vast amount of data. This dataset was first used in [32]. We decided to use only the portions of the dataset concerning annotations about artists, mainly for computational complexity reasons and also because of the lack of appropriate grounding data .

Originally, the amount of data was too big to be handled in a reasonable time, so we decided to limit it quite radically, while still leaving more than enough information to make meaningful computations. That left us with more than 280000 tags, used by almost 21000 users and applied to approximately 350000 artists. All this sums up to more than two and a half million triples. Such large number of triples does not have to have great impact on outcome, when compared to substantially lower one. Although dataset contained users' friends and groups we did not use this content.

Typically, tags represent music styles, but sometimes express sentiment (tags like: *awesome*, *best song*), feelings or activities connected to song/artist for example tags "breakfast music" or "relaxing music", or indicate that user has seen some artist performing live. The reason for introducing the tagging in Last.fm service was probably the lack of staff to classify scrobbled music, but as we've explained before, it has evolved into more than just plain classification.

To better describe the dataset, we refer to histograms A.7, A.8 and A.9. Those depict tag, resource and user occurrences respectively. On the x-axes there are numbers of annotations when concerning tags, resources or users. In each case we set boundaries for x-axis according to what we want to capture from the dataset so that it can be discussed here. Let us make this clear by an example. In A.8, the majority of resources (in this case more than 1000) were only annotated 20 times. In this case, we only show resources annotated more than 20 times, although most of the resources were tagged less times. However, these ones are not very important for us, because by having less than 20 annotations (tags) assigned to them they lack sufficient data to actually compute substantive results. We can call these resources "unique" resources and they are a majority in every dataset we had.

Evidently, a trend can be seen in every figure that shows decreasing amount of occurrences as the x-axis progresses. Surprisingly, this applies to Last.fm's tags as well. Since Last.fm's annotations are primarily meant to describe what style of music artists perform, it would be quite logical not to have so many "unique" tags, because an amount of music styles, however always growing, is still somewhat limited value. Nevertheless, these tags are present as much as in other datasets. This might be so because people tend to tag artists with their names, for example people tag the artist "John Lennon" by the tag "John Lennon", creating these "unique" tags in the process. In the figure A.9 we can see that users of Last.fm are much more active than users of other datasets.

3.2 Delicious dataset

In delicious, users tag exclusively web pages and its main aim is to replace or improve bookmarking provided in web browsers. Surprisingly, as far as we know, there are not many snapshots of delicious freely available on the internet at this time. The one used in this work can be found on the page [27]. It contains 214 000 bookmarks, which is more than enough for the kind of experiment described here.

Dataset is in the JSON format, which can be directly imported into Python data structures, so there was no need to parse. A simple extractor was then written to get the data into format usable with our code. This format will be described in section 5.8.

Roughly 135 000 resources are present, along with unique 70 000 tags, applied by approximately 80 000 users. The number of triples (meaning relations between user, resource and tag) is about 650 000. Since the dataset was created by reading a new bookmarks feed, tag frequencies are sometimes rather sparse, but are sufficient in case of really common resources(pages) as showed in the figure A.2. Example of one bookmark in this dataset can be found in the code sample 3.1. Quotes were deleted and some addresses were shortened to make the example clearer. It is obvious that we did not need most of the information stored, however, some of the fields could be used in other research. We only parsed web adresses, authors and tags.

```
{
updated: Sun, 06 Sep 2009 19:48:28 +0000,
links: [{href: http://www.cnn.com/, type: text/html, rel: alternate}],
title: CNN.com,
author: Andree,
comments: http://delicious.com/url/006d5652f4c43ab9e69328ab5e74f7e4,
guidislink: false,
title_detail: {base: http://feeds.d...s.com/v2/rss/re...=100,
type: text/plain,
language: null, value: CNN.com},
link: http://www.cnn.com/, source: {},
wfw_commentrss: http://feeds.delicious.com/v2/rss/url/006...,
id: http://delicious.com/url/0...4#Andree,
tags:[
{term: news, scheme: http://delicious.com/Andree/, label: null},
{term: politics, scheme: http://delicious.com/Andree/, label: null}
]
}
```

Code sample 3.1: Example of delicous dataset

We can see more about the datasets' content in figures A.1, A.2 and A.3. Phenomenon of an "unique" tag, a unique resource and even unique users can be seen in all of the figures and this is common for all the datasets. The reason for that in case of resources is probably that users not bookmark pages as in bookmarking the main page(e.g. "www.bbc.co.uk"), but rather part of the web page they find interesting enough to bookmark, for example "http://news.bbc.co.uk/2/hi/science/nature/default.stm".

A question comes to mind whether to treat bookmarks sharing the same main page as if they were about the same topic and therefore we could join tags given to various parts of page and assign them only to the home page. While there are surely some examples where this would be valid, we cannot generalize this, therefore it could lead to unwanted results.

From the figure A.1 is clear that many tags have been used only from one to five times in this dataset, which is probably due to the creation of the dataset by reading the RSS feed. It is possible that many tags might actually be quite common in delicious, although they are not in our dataset. If there were a way how to get data directly from delicious, we would probably not have this problem.

Same thing can be said about users' activity in delicious, although a peak can be seen around 15 annotations given by users with almost 500 users falling into that category, suggesting that this might be the threshold after which users loose their interest in using delicious.

3.3 Medworm dataset

Medworm collects data from more than 6 000 medical oriented blogs, journals and news services. However, tagging data is mainly present in a blog section. Since we had to parse Medworm to create the dataset, we used a list of almost 14 000 tags present on one of

Medworm's pages and then performed a search for each tag, collecting data about articles, blogs articles came from and categories they belong to in scope of Medworm.

A simple parser in Python was used and data initially stored in JSON format. We collected almost 200000 articles from about 1000 blogs adding up to more than one million triples. We should now note, that the term "user" in scope of this dataset actually means blog. There is not always a way to get info about the actual user by whom the article was written or, more importantly, tagged. But for the sake of compatible terminology we will continue on using the "word" user in connection with Medworm dataset as well. It also makes sense when we look at a blog as a set of users creating its content. One of these users then writes a blog article, tags it and the article, along with tags, gets imported into Medworm. So we can say that blog is a collection of users and these users are both authors of articles(which is not really relevant for this project) and authors of tags as well. This will only affect macro and collaborative measures in a way that each users' annotations will not typically be as sparse as they would be in normal case.

In the figure 3.3 you can see a part of one element parsed from Medworm's website. Again, web addresses were shortened. Initially all elements had been tag oriented and were later converted into our triple model. As reader can see, dataset contains articles' url, its name, name of the blog it came from and a category it belongs to in Medworm. Some of the articles also have information about author within the blog, by whom the article was written, however this is not present every time.

```
{''cancer'': [
[''http://www.medworm.com/index.php?...gerd-not-about-acid.html'',
''A new approach for treating reflux?'',
''The ND Blog: Notes from the Nutritionista by Monica Reinagel, L.D.N., C.N.S.'',
''Nutritionists and Food Scientists''],
[''http://www.medworm.com/index.php?...AHeartyLife%2F%7E3%2FI5SRXWimUSw%2F'',
''Insomnia Common with Chemotherapy'',
''A Hearty Life'',
''Nurses''] ... ]}
```

Figure 3.1: Example of Medworm dataset

From the figures A.4, A.5 and A.6 it is clear that patterns visible in the other datasets apply here as well, with exceptions of blog activity. Blog activity has its peak with one article only, which is quite surprising, because of the fact that blog that only has one article does not seem useful. This might have been caused by Medworm's import mechanisms, which, to our knowledge, are not publicly available, so we can not be entirely sure. However, there is a significant portion of blogs with 100 to 150 articles, as it can be seen in the figure A.6.

3.4 Comparison of the datasets

We have already given some information about the datasets, summarized here by the table 3.1. This table also shows information about usefulness of tags contained in the datasets. We made a short list of words that we thought were not important for describing resources. This included mainly sentimentally oriented tags as "like", "love", "great" and so on. Of course, we could not filter all the non desired tags and therefore percentages in 3.1 are probably lower in reality. Full scale analysis of this phenomenon is however beyond the scope of this project. It is logical that Last.fm includes highest portion of such tags, since its resources are pieces of music, it comes as no surprise that people associate it with feelings more often than in the other domains.

From the pictures shown before we can tell right now that the dataset that was most suitable for our needs is Last.fm's dataset, above all because of its long creation time and also method of parsing, which made it the most complete sample we had. Medworm looked promising as well, but did not provide sufficient data for our grounding case. More about that subject in the section 4.1. Similarly, delicious proved not to be as good as Last.fm. This is probably due to the concept of its creation, as described before.

	users	resources	tags	annotations	% of useful tags
delicious	80000	135000	70000	650000	97%
Medworm	1000(blogs)	200000	14000	1000000	99%
Last.fm	21000	350000	280000	2500000	95%

Table 3.1: Table summarizing the datasets

Chapter 4

Resource grounding techniques, data and implementation

4.1 Grounding data and techniques

Although the goal of this project is to test similarity measures in folksonomies, the part of task which concerns grounding of obtained results is almost as big as the part which has already been described. In all the cases we needed to obtain human created dataset, which contained intersection with given folksonomy dataset that was big enough to make a valid comparison. Grounding dataset must have some kind of hierarchic structure. Such structures, apart from taxonomies, are sometimes part of ontologies. Such ontology, apart from other information, contains a set of entities and their "is-a" relationships with other entities. Having this data, we can infer hierarchies, store them as a graph and then use graph based similarity measures to compare with folksonomy based similarity measures.

We wanted to have a special grounding case for each dataset. If we had used one grounding mechanism for all the datasets, there would not have been so much diversity between the results and all the results would possibly have been very similar to each other.

We chose Lin's information theoretic measure [22] in all the cases except Wordnet. There are several implementations of Wordnet's similarity measures freely available, for example [30] and [13]. We decided to use Lin's measure because of its simplicity and relatively good performance, which was proved by several user studies, as described in [22] and [35].

For reasons justified before, we used Lin's measure. However, the main challenge here was not to implement the measure, which is quite trivial, but to extract hierarchies from available data. This will be described in the chapter 4.1. For the graph representation, when implementing Lin's measure, we chose Python graph library called iGraph [3].

Once we have build the tree from the hierarchy we computed all the paths from tree's root, the root being usually the "Top" category of given hierarchy. In some cases we added the root node ourself to make the connections between various branches of the graph possible. Since Lin's measure uses information theoretic definition of probability, sometimes called *log probability*, as mentioned for example in [10], we have to compute these probabilities somehow. In [36] the probability of one entity is simply defined as the number of occurrences of given entity divided by the number of all entities. In some cases we had sufficient data to compute the probability, in some cases we had not, so we resided to computing the probabilities only from hierarchical structure. We did that by cycling through all paths from tree's root and counting occurrences of each visited node. This yielded frequencies and we obtained the actaul probabilities by dividing the frequencies by the number of nodes.

Lin's measure is defined as follows (described(among others) in [19]):

$$Lin_{(a,b)} = \frac{\log p(common(a,b))}{log(p(a) + \log p(b)} \tag{4.1}$$

where function $common(a, b)$ returns the *least common subsumer* of a and b in a graph, p is the function returning probability of the given node in the graph. This definition can be applied to all following cases. The figure 4.1 shows a small part of music genre hierarchy. In the figure 4.1 the least common subsumer of nodes "Indie" and "Alternative" is "Rock" and similarly least common subsumer of "Rock" and "Pop" is "Top".

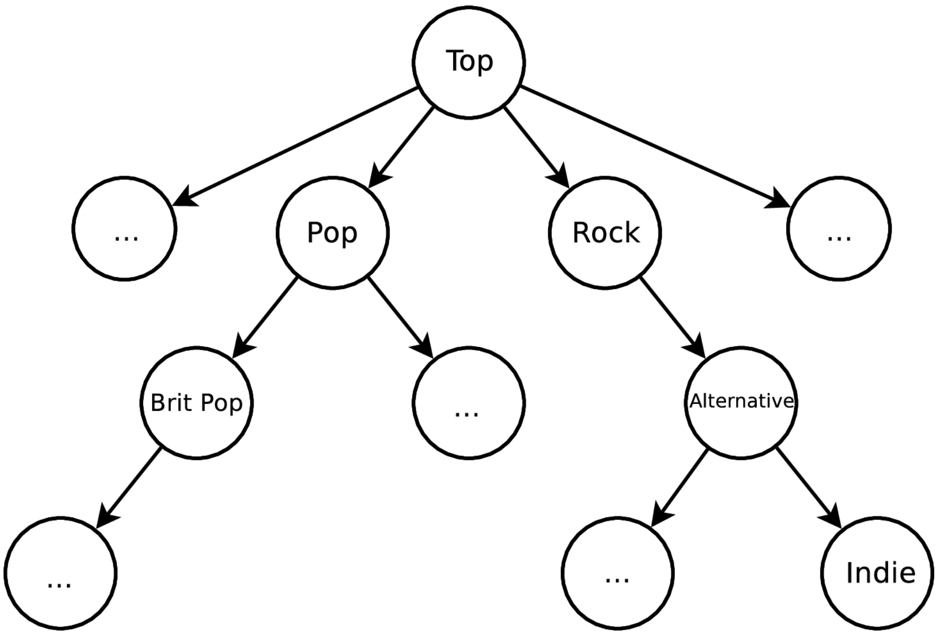

Figure 4.1: Example of music oriented taxonomy

Essentially, in all the following sections of this chapter, the result was a similarity matrix that was later used for grounding.

4.2 Delicious grounding

4.2.1 Tag grounding

Delicious' bookmarks have no limits, so if we want to compare significant amount of the data included in the dataset, we have to use very common database such as WordNet. We used implementations of similarity in Wordnet included in NLTK - Natural Language ToolKit [12], set of natural language processing oriented modules for Python. NLTK implements 6 similarity measures in Wordnet.

Another option would be to use Wordnet::Similarity module([30]), written in Perl. This module contains 10 similarity measures and is accessible online, which makes it convenient tool for quick random calculations. Nevertheless, we chose NLTK over Wordnet::Similarity mainly because the rest of our code was already in Python, so there was no need to convert anything.

Similarity measures in Wordnet work with so called synsets - sets of synonyms. Querying Wordnet returns typically a set of synsets, as showed in the code sample 4.1. Main problem when computing similarities in Wordnet is therefore choosing the right synsets, which are hard to determine if we do not know actual meaning of the tag. There is no way to obtain this knowledge without going back to the resource the tag is assigned to and trying to guess, what the page is about and then according to that choosing appropriate synset. This kind of analysis is beyond aim of this project. Moreover similarity methods in Wordnet work with only one type of speech, meaning it is impossible to measure similarity between noun and verb, for instance.

For reasons justified above, we did not perform any attempts to guess the right synsets. We simply choose the first pair of synsets with matching word classes. There was also a possibility to compute similarities between all the matching synsets, but that would consume much more time and benefits of this approach were not provable.

Intersection of Wordnet and delicious dataset includes more than 14000 tags, we only considered those occurring 5 and more times when computing similarities. This left us with 5000 tags. We used Wu-Palmer similarity measure, whose formula we already described in the section 2.1. We decided to use this measure because it performed very good in [22] and also to easily compare our results with the ones given presented [24].

4.2.2 Resource grounding

For the same reason as in the case of tags we needed to have a hierarchy as broad as possible. ODP - Open Directory Project([2]) is indeed big enough to meet our expectations. It is open, growing, multilingual and is being used by Google and other big companies.

ODP's structure is more complicated than a classic is-a taxonomy. Apart from hierarchy links, it also contains symbolic links and related links. Symbolic link expresses that two categories connected with such link are basically the same. Symbolic links connect for example categories with same content, only in different language. Related links join the categories that are related and are usually used for "see also" type of recommendation. First we wanted to implement measure described in [23] because it has been developed specially for ODP and therefore fully exploits its architecture. However, this measure is both implementation-wise and computation-wise too expensive (mainly because of its utilization of symbolic and relative links and inferring transitive relations in ODP's graph using those links),therefore we used Lin's measure, which uses just hierarchical information contained in ODP.

4.3 Last.fm grounding

We considered only Last.fm's artists' annotations, although similarities between songs and events could be calculated too, however there are little to none grounding data for latter two cases. We used the hierarchy inferred from Musicmoz, project similar to ODP([8]), as grounding basis for both tags and artists.

Another option would be to use Wordnet in tags case. While many of the styles/tags are indeed present in Wordnet it is better to use more specific hierarchy. Since semantic similarity measures in Wordnet use synsets as input, it would be hard to judge which one to pick. This may become clearer by example in the figure 4.1, which contains output of

Wordnet queried for a term “rock”. As the reader can see, there are many synsets returned, but only one (number 6 in the noun section) that captures desired meaning of the word “rock” in this case, therefore it is better to use hierarchy from Musicmoz - it is certain that the term “rock” in Musicmoz’s hierarchy refers to music genre.

```
$ wn rock -over

Overview of noun rock

The noun rock has 7 senses (first 2 from tagged texts)

1. (14) rock, stone -- (a lump or mass of hard ...)
2. (6) rock, stone -- (material consisting of the aggregate of minerals ...)
3. Rock, John Rock -- (United States gynecologist ...)
4. rock -- ((figurative) someone who is strong and stable and dependable ...)
5. rock candy, rock -- (hard bright-colored stick candy ...)
6. rock 'n' roll, rock'n'roll, rock-and-roll, rock and roll, rock, rock music --
(a genre of popular music originating in the 1950s; a blend of black rhythm-and-blues
with white country-and-western;
''rock is a generic term for the range of styles that evolved out of rock'n'roll.'')
7. rock, careen, sway, tilt -- (pitching dangerously to one side)

Overview of verb rock

The verb rock has 2 senses (first 1 from tagged texts)

1. (6) rock, sway, shake -- (move back and forth or sideways ...)
2. rock, sway -- (cause to move back and forth ...)
```

Code sample 4.1: Example of Wordnet output

4.3.1 Tag grounding

Information extracted from XMLs available on MusicMoz’s web page included music styles hierarchy with 550 styles. Aside from that, we needed a frequency for each style, so that we could compute their probability. This was done together with counting frequencies as described in the subsection 4.3.2. We mapped style names directly to Last.fm tags acquiring intersection containing 480 tags/styles. It is interesting that 70 styles from MusicMoz hierarchy were not used as tags in our dataset.

4.3.2 Artists grounding

Every entry about music artist in MusicMoz includes one to three music genres assigned to artist. Genres are not the same as categories in the hierarchy. Luckily, Musicmoz provides mapping to the majority of hierarchy categories, we had to map the rest ourselves. When computing similarity of the artist with more than one genre assigned we simply use mean of similarities computed for each category. The intersection between the dataset and artists stored in MusicMoz includes about 36000 interpreters.

For example, the music group “The Beatles” has following styles assigned : [“British Invasion”, “Rock”, “Skiffle”] and “Lenny Kravitz” has only [“Rock”]. Overall similarity is therefore as follows (TB stands for The Beatles, LK for Lenny Kravitz and so on):

$$sim(TB, LK) = \frac{sim(BI, R) + sim(R, R) + sim(S, R)}{3} \tag{4.2}$$

where $sim()$ is any similarity measure giving result in interval $< 0, 1 >$. Resulting Lin's measure similarity matrix is more than 100 MB large. We should probably note here, that although previous definitions of artists similarity make sense, it will not perform as good as it should, mainly because just 3 styles are not enough to throughly describe an artist. We can therefore expect that this grounding technique will not produce the best of results and results given in the chapter 6 should be viewed with this knowledge.

4.4 Medworm grounding

Since Medworm is a medical oriented service, we needed to find some appropriate grounding data. We could have used Wordnet, but for reasons same as in Last.fm's case we decided to find some specialized ontology. Only medical ontology with sufficient hierarchy structure was Human disease ontology from [18]. A small part of this ontology as displayed by Protégé([33]) can be seen in the figure 4.2. As reader can see, its structure consists only of is-a relationships and therefore Lin's measure can be applied. The term with DOID_4 is the term labeled "disease" and has only one superordinate. We considered this term as a root of taxonomy's tree in our implementation. Not the whole structure is visible in the picture. Maximum depth of the hierarchy in this particular ontology is 12.

This ontology, although only with 50 intersecting tags was still the best we could find. Ontology was stored in OBO format, whose description can be found in [11]. Yet for our purpose we did not need to know much about the format, because we only wanted to extract hierarchical information from ontology. This is stored in form of is-a relationships as showed in the code sample 4.2, where we can see that "Chapare hemorrhagic fever" is both a "viral hemorrhagic fever" and a "arenaviridae infectious disease".

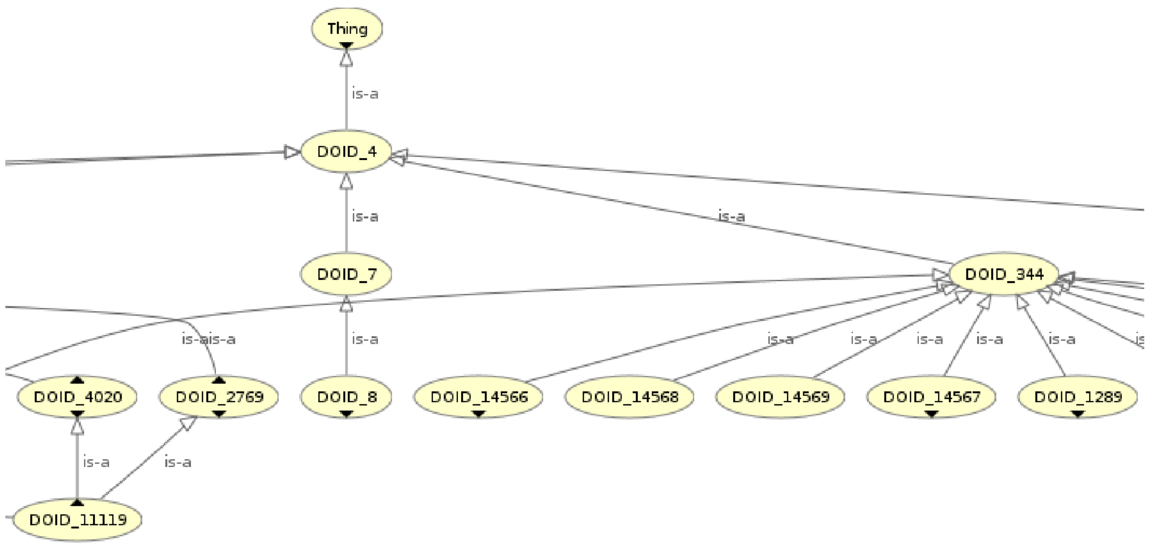

Figure 4.2: Portion of human disease ontology

```
[Term]
id: DOID:0050198
name: Chapare hemorrhagic fever
def: ''A viral hemorrhagic fever that involves ...''
is_a: DOID:1330 ! viral hemorrhagic fever
is_a: DOID:3944 ! arenaviridae infectious disease
```

Code sample 4.2: Example of a term in Human disease ontology

Because resources in the Medworm dataset are blog articles, it is impossible to find grounding data for those. Hence we only did tag grounding in case of Medworm.

4.5 Comparison of used grounding approaches and grounding data

We used similar grounding approaches, however there was a big difference between sizes and structures of our grounding data. This is summarized in the table 4.1. If we consider the structure of grounding data, the "richest" hierarchy is no doubt the one of ODP, followed by Wordnet, Human disease ontology and finally by Musicmoz. The richer structure does not automatically mean more precise similarities. ODP is also the only grounding dataset, that contains more complex structure than just a hierarchy.

	number of terms	intersecting terms
Wordnet	150000	14000
Human Dis. Ont.	2500	40
MusicMoz	550	500
ODP	more than 5 million	7000

Table 4.1: Table summarizing the intersections with the datasets

Chapter 5

Semantic similarity measures, aggregation methods and implementations

In this chapter we will use example folksonomy created by two users, Bob and Alice. It contains three resources: “seznam.cz”, “spectrum.ieee.org” and “guardian.co.uk”. Users used following tags to annotate resources: “czech”, “politics”, “news”, “science”, “magazine” and “british”. Its graphical representation can be viewed in the figure 5.1.

Formally, this folksonomy contains following sets (see definition in the chapter 1): U={Bob, Alice}, R={seznam.cz, spectrum.ieee.org, guardian.co.uk}, T={czech, politics, news, science, magazine, british}, Y={{Bob, guardian.co.uk, politics}, {Bob, guardian.co.uk, british},{Bob, guardian.co.uk, news}, {Bob, spectrum.ieee.org, magazine}, {Bob, spectrum.iee.org, science} {Alice, seznam.cz, news}, {Alice, seznam.cz, czech},{Alice, guardian.co.uk, british},{Alice, guardian.co.uk, news}}

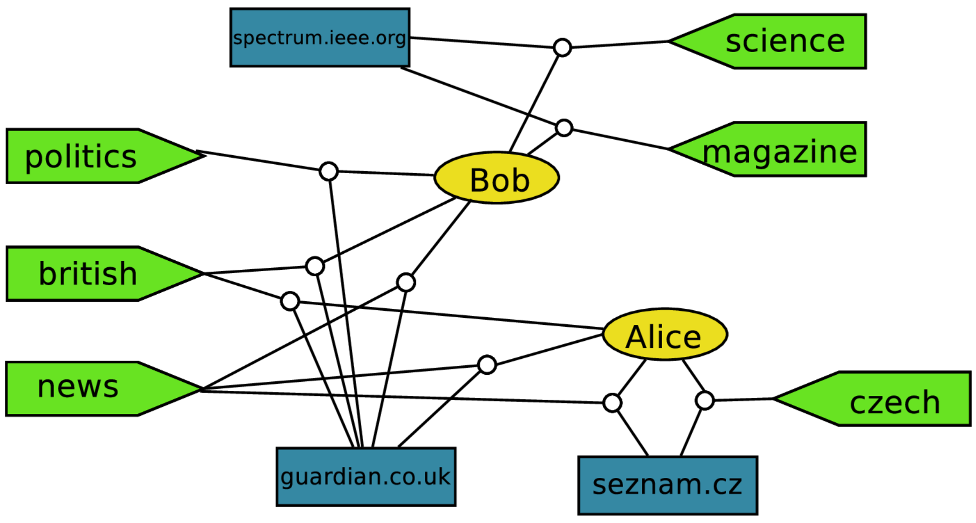

Figure 5.1: Graphical representation of a small folksonomy

5.1 Aggregations and probabilities

Since semantic similarity measures commonly consider only two terms as an input, and we have a triple representation, we need to narrow triples by one. We performed resource-resource and tag-tag similarity, therefore we aggregated over users. Although user-user similarities could have been computed, we did not preform such task. Application of all aggregation methods results in a two dimensional matrix. Following section use definitions given in [24].

5.2 Projection aggregation

This aggregation method is rather straightforward. As said before, this method does not capture how many annotations were given to certain object creating binary matrix. Example of projection aggregation matrix of the folksonomy 5.1 is in the table 5.1. To show how transposed matrix looks like, we refer to the table 5.2. First case can be used to compute resource-resource similarity while the second case is used for computing tag-tag similarities. Notice in case of guardian.co.uk that in spite of being tagged twice by “news” and “british” it only has one annotation marked in the matrix.

	british	news	czech	science	magazine	politics
guardian.co.uk	1	1	0	0	0	1
seznam.cz	0	1	1	0	0	0
spectrum.ieee.org	0	0	0	1	1	0

Table 5.1: Projection aggregation, case 1

	guardian.co.uk	seznam.cz	spectrum.ieee.org
british	1	0	0
news	1	1	0
czech	0	1	0
science	0	0	1
magazine	0	0	1
politics	1	0	0

Table 5.2: Projection aggregation, case 2

5.3 Distributional aggregation

Distributional, or sometimes referred to as *fuzzy*, aggregation is similar to the projection case with the distinction that it counts occurrences of annotations. Example is in the table 5.3.

Although it might seem that this method will capture state of a folksonomy better, it has some downsides in comparison to the projection. Most importantly, computing similarity measures with distributional aggregation requires more computational time because main operations involved in such measures are more complex than in projection’s case.

Implementations have showed that computing similarity with distributional measures can be up to 4 times slower than doing the same task using projection aggregation. This will become clearer after reading the section 5.6. Both methods share one major disadvantage: as folksonomy gets bigger, each change to the aggregation matrix has an impact on resulting similarities, eventually making use of these aggregations in "alive" folksonomies problematic.

	british	news	czech	science	magazine	politics
guardian.co.uk	2	2	0	0	0	1
seznam.cz	0	1	1	0	0	0
spectrum.ieee.org	0	0	0	1	1	0

Table 5.3: Distributional aggregation applied to the folksonomy 5.1

Other case of aggregation is also called distributional, but uses a projected matrix with probabilities of each element in the matrix. For example, probability of a tag in a folksonomy is the number of resources annotated with the tag divided by the number of all resources in the folksonomy. This definition is identical for resources. So, the probability p of the resource *seznam.cz* is: $p(\text{seznam.cz}) = 2/6$, because it has been annotated with 2 out of six tags present in the folksonomy.

5.4 Macro aggregation

Macro aggregation is more user oriented than preceding methods. Macro aggregation is not technically an aggregation method in the same sense as projection and distributional methods are. Basically, in the case of macro aggregation, we first aggregate every user's annotations separately and compute similarities using every matrix. Resulting similarity matrix is obtained by summing all the similarity matrices. It is unimportant, if we use projection or distributional method to obtain users' matrices. In [24] authors only use distributional measure with probabilities and we decided to do the same.

Macro aggregation shares some issues with the collaborative aggregation method. This will be described in the section 5.5.1.

Alice	british	news	czech	science	magazine	politics
guardian.co.uk	1	1	0	0	0	0
seznam.cz	0	1	1	0	0	0
spectrum.ieee.org	0	0	0	0	0	0

Table 5.4: Macro aggregation applied to the folksonomy 5.1, case 1

Bob	british	news	czech	science	magazine	politics
guardian.co.uk	1	1	0	0	0	1
seznam.cz	0	0	0	0	0	0
spectrum.ieee.org	0	0	0	1	1	0

Table 5.5: Macro aggregation applied to the folksonomy 5.1, case 2

If we want to combine macro aggregation and distributional aggregation we need to redefine probabilities. It comes as no surprise that each user will have his own set of probabilities. The formula stays the same, we just apply it to each user's respective folksonomy. Therefore probability of tag *british* in case of user *Alice* is : $p(\text{british}|\text{Alice}) = 1/3$.

5.5 Collaborative aggregation

Collaborative aggregation works on a principle not too different from one used in the macro aggregation. There is one exception, which was introduced to strengthen a bond between a user and all the resources/tags he/she annotates. If we add implicit resource/tag as seen in tables 5.6, 5.7 and 5.8, similarity measures will then yield non zero similarity for all the resources/tags user has used, even if they did not originally share any common properties (tags, resources) and would have not been similar at all, had we not used implicit annotation.

Alice	british	news	czech	science	magazine	politics	Alice
guardian.co.uk	1	1	0	0	0	0	1
seznam.cz	0	1	1	0	0	0	1
spectrum.ieee.org	0	0	0	0	0	0	0

Table 5.6: Collaborative aggregation, case 1

Bob	british	news	czech	science	magazine	politics	Bob
guardian.co.uk	1	1	0	0	0	1	1
seznam.cz	0	0	0	0	0	0	0
spectrum.ieee.org	0	0	0	1	1	0	1

Table 5.7: Collaborative aggregation, case 2

Alice	guardian.co.uk	seznam.cz	spectrum.ieee.org	Alice
british	1	0	0	1
news	1	1	0	1
czech	0	1	0	1
science	0	0	0	0
magazine	0	0	0	0
politics	0	0	0	0

Table 5.8: Collaborative aggregation, case 3

Again, we have to redefine the probabilities like we did in macro aggregations' case. We must not forget that we added implicit annotation to all the resources/tags therefore the fact that resource/tag is annotated with this implicit tag/resource carries no information. We can eliminate impact implicit annotation has on probabilities by slightly modifying the probability formula. This can be done by incrementing the divisor in the formula by one. Using this modification, probability of the tag *news* in Alice's folksonomy is: $p(\text{news}|\text{alice}) = 3/(4+1)$

5.5.1 Issues of collaborative and macro aggregations

Both collaborative and macro aggregations have a minor glitch. If we keep passive users' annotations in the folksonomy, collaborative and macro aggregations will yield very high similarities (often even 1.0) for those tags/resources annotated only by these passive users and no one else. Let us say, that user only tags one resource with 5 tags and these tags are unique in folksonomy. This will result to each of these tags being completely similar to each other even if this would hardly be truth in a reality. For example, normed matching similarity coefficient for any pair from these 5 tags would yield 1.0, because they all share one resource and this resource is the only one in small users' folksonomy. Whereas with the projection aggregation, it would be 1/[the number of all resources in folksonomy], and this would of course output a very small similarity. In our experiments, we generally only considered those users with more than 20 annotations and therefore limiting unique tags/resources to an acceptable level.

We should however state, that this problem is not so strong in the collaborative aggregation. All the similarities are lowered a little by the implicit annotations. Reader will see in the chapter 6, that this has a vast impact on the performance of the measures.

Another issue is the fact, that results of macro or collaborative aggregated similarity methods do not give results in range 0 to 1. Results of these measures have to be normalized. We can either divide each value by number of users in the whole folksonomy, or only consider those users, who added some similarity to the resulting matrix. We decided to use the latter case, because it better reflects the state a folksonomy is in, especially in those folksonomies, where users annotations are sparse and/or unique.

5.6 Similarity measures

Let us describe some notations that will be used in following sections. Respecting notations given in [24], measures are symmetric when used for computing resource similarities or tag similarities. We will use following annotations: x_1 and x_2 are objects whose similarity we want to compute. X_1 and X_2 are sets(in case of projection aggregation) or vectors(in other cases) of attributes(or features) of these objects. If we use the example given in 5.1, a set of attributes of a resource "seznam.cz" is the set containing tags "czech" and "news". Similarly the set of attributes of the tag "news" contains resources "seznam.cz" and "guardian.co.uk". Whenever a function $p(y)$ is mentioned, it is the function returning probability of certain attribute y. If $y|u$ is present it means the probability of attribute occurring in case of the user u.

Formulas ending with **S** are most basic projection cases, those ending with **P** use the projection aggregation with probabilities while formulas with **D** at the end work with weighted representations. Finally **U** means that formula describes macro or collaborative measure for one user.

5.6.1 Matching coefficient

Matching coefficient is the simplest thinkable measure; it is only based on counting how many common features objects share. For basic projection case its formula is as follows:

$$\text{matchingS}(x_1, x_2) = |X_1 \cap X_2| \tag{5.1}$$

when using probabilities:

$$\text{matchingP}(x_1, x_2) = -\sum_{y \in X_1 \cap X_2} \log p(y) \tag{5.2}$$

for distributional case:

$$\text{matchingD}(x_1, x_2) = X_1 \cdot X_2 \tag{5.3}$$

and finally for macro and collaborative instances:

$$\text{matchingU}(x_1, x_2) = -\sum_{y \in X_1^u \cap X_2^u} \log p(y|u) \tag{5.4}$$

This measure will not produce similarities between 0 and 1. If we want to obtain such number, output needs to be normalized by simply dividing the result of measure by number of tags in resource similarity case and vice versa.This will, typically, procude very low similarities in general.

5.6.2 Overlap coefficient

Overlap coefficient, as its name suggests, is based on measuring the overlap between the intersection of representation sets and one of the cardinalities of those sets. Notice how in the projection case it is the lower one, while in the distributional case with probabilities it is the higher one, because in latter case we use log probabilities and clearly the higher number in this case is the one representing lower probability.

Overlap coefficient, projection aggregation:

$$\text{overlapS}(x_1, x_2) = \frac{|X_1 \cap X_2|}{\min(|X_1|, |X_2|)} \tag{5.5}$$

Distributed overlap coefficient using probabilities:

$$\text{overlapP}(x_1, x_2) = \frac{\sum_{y \in X_1 \cap X_2} \log p(y)}{\max(\sum_{y \in X_1} \log p(y), \sum_{y \in X_2} \log p(y))} \tag{5.6}$$

For use with macro and collaborative aggregations we define following formula:

$$\text{overlapU}(x_1, x_2) = \frac{\sum_{y \in X_1^u \cap X_2^u} \log p(y)}{\max(\sum_{y \in X_1^u} \log p(y|u), \sum_{y \in X_2^u} \log p(y|u))} \tag{5.7}$$

5.6.3 Jaccard coefficient

Jaccard coefficient, named after its developer, botanist Paul Jaccard, who defined it in [20], is similar to Dice similarity, but usually gives slightly higher similarities. Jaccard coefficient for projection case is as follows:

$$\text{jaccardS}(x_1, x_2) = \frac{|X_1 \cap X_2|}{|X_1 \cup X_2|} \tag{5.8}$$

followed by the distributional case with probabilities:

$$\text{jaccardP}(x_1, x_2) = \frac{\sum_{y \in X_1 \cap X_2} \log p(y)}{\sum_{y \in X_1 \cup X_2} \log p(y)} \tag{5.9}$$

and finally macro and collaborative aggregated formula:

$$\text{jaccardU}(x_1, x_2) = \frac{\sum_{y \in X_1^u \cap X_2^u} \log p(y)}{\sum_{y \in X_1^u \cup X_2^u} \log p(y)} \tag{5.10}$$

5.6.4 Dice coefficient

Dice coefficient is similar to Jaccard coefficient and was described in [16]. It is defined as follows for projection aggregation:

$$\text{diceS}(x_1, x_2) = \frac{2 \cdot |X_1 \cap X_2|}{|X_1| + |X_2|} \tag{5.11}$$

if we want to use Dice coefficient with probabilities, we use following formula:

$$\text{diceP}(x_1, x_2) = \frac{2 \cdot \sum_{y \in X_1 \cap X_2} \log p(y)}{\sum_{y \in X_1} \log p(y) + \sum_{y \in X_2} \log p(y)} \tag{5.12}$$

and finally for user-concious aggregations:

$$\text{diceU}(x_1, x_2) = \frac{2 \cdot \sum_{y \in X_1^u \cap X_2^u} \log p(y|u)}{\sum_{y \in X_1^u} \log p(y|u) + \sum_{y \in X_2^u} \log p(y|u)} \tag{5.13}$$

5.6.5 Cosine similarity measure

Cosine similarity is normally defined on vector space. However definitions can be found for special cases. Cosine similarity is widely used in many domains.

Projected cosine similarity:

$$\text{cosineS} = \frac{|X_1 \cap X_2|}{|X_1| + |X_2|} \tag{5.14}$$

distributional frequency weighted representation:

$$\text{cosineD} = \frac{X_1}{||X_1||} \cdot \frac{X_2}{||X_2||} = \frac{\sum_y w_{x_1 y} w_{x_2 y}}{\sqrt{\sum_y w_{x_1 y}^2} \sqrt{\sum_y w_{x_2 y}^2}} \tag{5.15}$$

user-oriented aggregations:

$$\text{cosineU} = \frac{X_1^u}{\sqrt{|X_1^u|}} \cdot \frac{X_2^u}{\sqrt{|X_2^u|}} = \frac{|X_1^u \cap X_2^u|}{\sqrt{|X_1^u| \cdot |X_2^u|}} \tag{5.16}$$

5.6.6 Mutual Information

Mutual information measure is one of the more complex similarity measures, mainly because it introduces the term of joint probability. Joint probability function (py_1, y_2) returns probability of attributes y_1 and y_2 coinciding and it has a different formula for the projection case and for the distributional case.

Mutual Information with projection or distributional aggregation:

$$\text{miSD} = \sum_{y_1 \in X_1} \sum_{y_2 \in X_2} p(y_1, y_2) \log \frac{p(y_1, y_2)}{p(y_1) p(y_2)} \tag{5.17}$$

where joint probabilities are yielded by the following function:

$$p(y_1, y_2) = \frac{\sum_x w_{xy_1} w_{xy_2}}{\sum_x 1} \tag{5.18}$$

For macro and collaborative case:

$$\text{miU} = \sum_{y_1 \in X_1^u} \sum_{y_2 \in X_2^u} p(y_1, y_2|u) \log \frac{p(y_1, y_2|u)}{p(y_1|u)p(y_2|u)} \tag{5.19}$$

where joint probabilities are defined as:

$$p(y_1, y_2) = \frac{\sum_x \min(w_{xy_1}, w_{xy_2})}{\sum_{r,t} w_{rt}} \tag{5.20}$$

and classical probability having the following formula:

$$p(y) = \frac{\sum_x w_{xy}}{\sum_{x,t} w_{rt}} \tag{5.21}$$

Although we have implemented this measure, computation of the joint probabilities was so computationally expensive that this measure could not be used with datasets as big as we used.

5.6.7 Maximum Information Path

In [25] Maximum Information Path measure is only described for distributional case with probabilities. This is the first and only measure described here which was developed specifically for use with folksonomies, namely with collaborative aggregation.

Maximum Information Path for distributional case:

$$\text{mipP} = \frac{2 \log(\min_{y \in X_1 \cap X_2}[p(y)])}{\log(\min_{y \in X_1}[p(y)]) + \log(\min_{y \in X_2}[p(y)])} \tag{5.22}$$

and the same measure using user oriented aggregations:

$$\text{mipU} = \frac{2 \log(\min_{y \in X_1^u \cap X_2^u}[p(y|u)])}{\log(\min_{y \in X_1^u}[p(y|u)]) + \log(\min_{y \in X_2^u}[p(y|u)])} \tag{5.23}$$

Although this measure performs exceptionally well according to [25], it did not outperform other measures in our experiment.

5.7 Programming languages and tools used

We decided to implement measures and aggregation methods in Python mainly because of SciPy, Python module for scientific computations, which is written partially in C and Fortran and is therefore rather fast. Specifically we applied SciPy's module for computation with sparse matrices. There is another sparse matrix module for Python called Pysparse ([9]). It specializes in sparse matrix data types and efficient sparse matrix operations. However, most of its functions are undocumented and its memory efficiency compared to SciPy is extremely bad. Because most of the vital function from SciPy we used were available in Pysparse as well, we tried to switch SciPy for Pysparse, but that implementation was both slow and memory inefficient (we only could calculate similarities between few hundred elements). There are some sparse matrix modules for Java as well, but none of them was in the usable state when we were implementing. Still, Python was the preferred language and SciPy the best choice available.

Sparse matrix is convenient data structure for storing both aggregated matrices and similarity matrices. We will try to describe types of matrices we used in our project. Usage of sparse matrices was not necessary, but it made manipulation with datasets easier. Other option would be to use memory mapped file for matrix storage, although this is typically slower than sparse matrices, because whole sparse matrix can be stored in memory (in our case). If we were to apply similarity measures on whole datasets, there would be no difference between using sparse matrix or normal matrix. Moreover, these matrices would probably exceed memory size. But since we used only portions of datasets, we could use sparse matrices with no problems, saving memory, space and time.

5.7.1 SciPy's sparse matrix structures

SciPy implements several sparse matrix types, each of which fits differently to different matrix operations. Each sparse matrix type has different use. Some take long time to build, but once created operations done with such matrix are quick, while some are built fast, but then often have limited set of operations available and are not so fast. In our implementation we used three types of matrices: list of lists format, dictionary of keys format and compressed sparse row format.

5.7.2 Uses, advantages and disadvantages of used matrix data structures

From now on, we will refer to List Of Lists matrix as LIL matrix, Dictionary Of Keys as DOK matrix and Compressed Sparse Row format as CSR matrix. LIL matrices take relatively long time to build but support efficient changes to sparsity structure(meaning adding values into matrix) as well as effective row slicing(we didn't use column slicing in our project). These two operations(adding values and row slicing) are the only two we need and LIL matrix supports both and is quickest in them too compared to other types. However, its long creation time prohibits its use in macro and collaborative aggregations, because we need to create matrix for each user and that takes too much time with LIL matrix. Our solution was to use DOK matrix when creating user's matrices and then convert them to CSR matrices. DOK matrices are created quickly and support adding values while CSR matrices implement fast row slicing.

5.8 Data model

We wanted input format to be as minimalistic as possible, because input data are typically very large. In the most basic case, the input into our application consist of four files. Three files represent tags, users and resources with IDs and fourth file contains triples connecting the IDs from other three files. Tags, users and resources files are stored in following format: [user/resource/tag id][tabulator delimiter][user/resource/tag id label][newline] and triples are stored similarly, but use only IDs: [user id][delimiter][resource id][delimiter][tag id][newline]. In the case of macro and collaborative aggregations, one more input file with user ids has to be provided.

5.9 Implementation of aggregation methods

Before we start computing actual similarities, aggregations need to be applied to inputs. In our implementation, we built all aggregations during one passage through input file for time conserving purposes. Implementation of projection, distributional and macro aggregations was quite straightforward, strictly following definitions given in 5.1. We created matrices ready to use for computing resource-resource similarities. Prior to computing tag-tag similarities, matrices had to be transposed. This was conveniently done with SciPy's method. In macro aggregation's case we iterated over all users' matrices and replaced them with theirs respective transposals.

This approach does not apply to collaborative aggregation. When implementing collaborative aggregation we needed to resolve how to store implicit tag/resource. We decided to enlarge resulting aggregation matrix by one column and use that as a column for implicit annotation. Because of this we could not use simple transposal. We wrote custom transpose function which omitted implicit annotations when creating transposed matrix and added them where they should be instead. We refer to the section 5.2 to show that transposing matrices with implicit transpose method would lead to wrong results.

5.10 Calculation of probabilities

Once we possess aggregated matrices, we need to calculate appropriate probabilities for those methods that use them. It is more convenient for the implementation to calculate these probabilities prior to computing similarity, because it saves processor time later and makes implementation of actual methods clearer.

The calculation itself is fairly straightforward, following definitions given in the section 5.1 and it is being done right after the computation of aggregation matrices. We used only projection aggregation matrix to compute probabilities - tag probabilities were obtainable instantly, for resource probabilities we had to transpose the matrices.

The situation was similar in macro and collaborative aggregations, we just had to compute the probabilities for each user separately.

The main challenge here was to implement joint probabilities used in the mutual information measure. There are two cases of joint probabilities, each one for different aggregation case. We followed the formulas given in [24] to be sure that our implementation will produce correct results, but in doing so, we had to iterate over all the elements in the projection matrix several times, resulting in very slow solution. It has been stated before in this text, that slow calculation of the joint probabilities was the reason we did not include mutual information measure in our experiment. This is surely a topic for future work.

5.11 Implementation of similarity measures

During the implementation of the measures we followed definitions given in the section 5.6. Although the formulas themselves were mostly quite simple, we had to implement some of the mathematical operations present in those formulas. These operations included set union, set intersection, vector norm and vector dot product. Luckily, these set operations were already implemented in SciPy so we had to implement only the vector operations.

Since the main goal in our implementation was to produce a similarity matrix and this task is extremely computationally demanding, because we have to compute similarities between all the elements in the aggregation matrix, we spent quite a lot of time trying to minimize the amount of operations needed. One fact we can utilize is that similarity of two terms in matrix, $sim(a, b)$, is the same for $sim(b, a)$. This means that we do not have to cycle through all the elements in the matrix, but only through half. This would be easy to do if we iterated over all the elements in the matrix, but since we only have to go through the non-zero elements in the matrix we had to have a mapping between actual elements in the sparse matrix, so that we could implement effective passage through matrix easily.

Each time we computed similarity between two elements, we needed to obtain two rows from the aggregation matrix. These two rows along with some other information are inputs of the similarity measure function. Getting a row from a sparse matrix is typically a very expensive operation and needs to be minimized. Since we essentially use two nested for-cycles to iterate over the elements of the matrix, we can minimize access to the matrix by getting the rows from it only once in the first for-cycle.

Another option would improve performance as well, but would require changes in the functions computing the similarities. As said before, the input of similarity measures are two rows from the aggregation matrix. We compute similarities all the combinations in the matrix, therefore the first row changes with much lower frequency than the second row. Function implementing the measures typically uses the elements in the rows to compute some kind of a constant. The part of the constant influenced by the first row does not change as long as the row does not change. If we precalculate that constant and passed it to the modified function, we can reduce the time needed to calculate similarity of one pair to the half of the original time in some cases.

5.12 Performance issues

The worst performance issue was not caused by the implementation, but rather by limitations of SciPy's data structures. This concerned computing measures in macro and collaborative aggregations. As described in the section 5.7.2 we had to use DOK matrices for storage of user-oriented aggregations. This was the only solution available, if we wanted to keep on using SciPy. However, the performance of DOK matrix is much worse than the one of LIL matrix. This resulted in very long computation times. While in some cases it took projected and distributed measures few minutes to calculate the whole similarity matrix, macro and collaborative aggregated measures ran for more than 24 hours.

This could be partly solved by reducing the number of users in a folksonomy. As discussed in the subsection 5.5.1, this might even improve the performance, but it depends strongly on the particular domain of a folksonomy. As showed in figures in the chapter 3 majority of users is not very active and if we do not consider these users in the computations, it will not have a great impact on resulting similarities.

Apart from these problems, naturally, the time needed to calculate a similarity matrix rises with the number of objects in the folksonomy, but is also influenced by the number of properties each objects has assigned to it (i.e. tags in resource's case and vice versa).

Normal computer was enough to fulfill the needs of our experiment, but as the number of objects in folksonomy grows, the complexity of computation rises quadratically. Therefore for use in the real world we would have to have much more powerful computation force than just a normal computer which is not only limited by frequency of its CPU but has insufficient memory for really big folksonomies.

Chapter 6

Results of the experiment

6.1 Correlation techniques

The best method to evaluate the performance of semantic similarity methods we implemented would be by means of a user study. Naturally, we did not have resources to perform such a study, so we had to settle for algorithmic evaluation.

Originally, we wanted to use direct correlations, measuring the differences between the pairs directly. However, this did not work as expected, mainly because distributions of similarities given by measures using folksonomies differed from those given by measures using grounding data. For instance, similarities computed by the Jaccard coefficient have their mean around 0.1, whereas Lin's similarity measure used in most of the grounding cases has mean of its similarities near 0.4. Although it would be possible in some cases to calculate the difference between the means and shift the values to make direct correlations possible, it is better to use a correlation methods that does not rely on direct comparisons between pairs.

Such approach is used in Kendall rank correlation coefficient, also referred to as Kendall's τ coefficient. It is a statistic used to measure the association between two measured datasets and it does not correlate the two datasets directly, but rather by counting numbers of concordant and discordant pairs. We used Python implementation from [7]. Its formula is as follows (according to [7]):

$$\tau = \frac{\text{number of concordant pairs} - \text{number of discordant pairs}}{\frac{1}{2}n(n-1)} \tag{6.1}$$

Given two pairs (x_i, y_i) and (x_j, y_j), the *concordant* pair is such pair that the following can be applied: if $x_i > x_j$ and $y_i > y_j$ or if $x_i < x_j$ and $y_i < y_j$. Every other pair is *discordant* and n is the total number of pairs. Results of Kendall rank correlation coefficient are between -1 and 1 where -1 means that all the pairs are discordant a 1 means that all are concordant.

This correlation approach also enables us to use non-normalized results given by matching coefficient and all the measures in macro and collaborative aggregations directly. Although we used normalized results in the actual correlation, but there would be no difference between using these results and the non-normalized results. We also chose this approach so that we can easily compare our results to the one in [24], since authors used the same correlation coefficient.

6.2 Last.fm dataset results

Results given by the measures applied to Last.fm dataset were correlated against Lin's measure, which was calculated from Musimoz directory (desribed in the subsection 4.3.1).

6.2.1 Correlations of tag-tag similarities

Results of the Kendall rank correlation coefficient between tag similarities from Last.fm and those in Musicmoz are in the figure 6.1. The figure shows how various measures perform when using certain aggregations. Note that the cosine in one of the distributional cases as well as the Maximum Information Path in projection case are not present, because these measures are not defined in those cases. We used projected cosine in macro and collaborative aggregations.

From correlations in the figure 6.1 we can see that Maximum Information Path measure performs the best along with the overlap measure and, surprisingly, the simplest, matching coefficient. The Jaccard coefficient, the Dice coefficient and the cosine similarity measure did not perform as well, but still managed to capture substantial amount of pairs. Notice how the Dice coefficient and the Jaccard coefficient yielded the same results. Although their formulas are quite similar, this is still surprising fact and it can be seen in all the following results.

As anticipated, macro aggregation drastically decreases the ability of measures to obtain high correlation scores. Still, these measures manage to yield positive aggregation scores, which cannot be said about the same measures in the other datasets.

Maximum Information Path measure performs the best in the collaborative case, probably because it is its natural aggregation (it was defined on it). All the other measures are now closer with their correlation scores to the best aggregations, sometimes even performing better.

It is interesting to confront projection and distributional(the one using probabilities) aggregations. While implementation of the distributed measures is no doubt more difficult and their performance in sense of computation time worse than in case of projected measures, their scores are in favor of projection aggregation.

Since calculating distributionally aggregated measures with frequency weighted representations requires more resources than in case of the other distributional aggregation, we decided not to calculate results of these measures any longer in the most of the other cases.

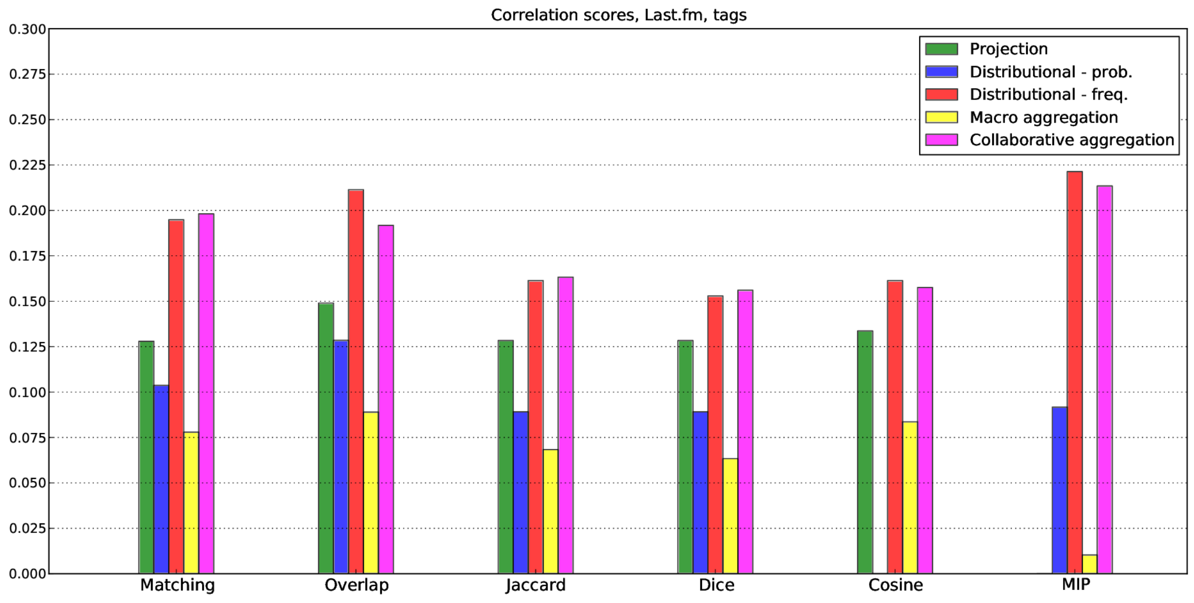

Figure 6.1: Kendall's τ correlation scores in Last.fm dataset, tag similarities

Before going any further, we should state that correlations we have just described were by far the best in the whole experiment. There are several reasons why this is so. The following has already been discussed elsewhere in this text, but let us repeat it. Intersection between Last.fm's tags and Musicmoz directory was quite small, including roughly around 500 terms. Out of these 500 terms the majority was composed by those tags that are very frequent in Last.fm - music styles. There are many other purposes of tagging in the Last.fm' dataset than just to assign music styles to artists, but this particular purpose is with no doubt one of the most important ones.

Basically, we only had 500 objects all of which were extremely rich in annotations. If we take the tag 'rock' for instance, it had a lot of artists attached to it and this was the case for virtually any tag. This is not so in the whole Last.fm dataset, as reader could see in the chapter 3. There are many unique tags, like in all the datasets. To sum up the results of correlations in case of Last.fm's tags, the good results obtained are probably caused by the nature of the folksonomy, which is almost ideal for use with semantic similarity measures. The results are probably partly caused by appropriate grounding data, too.

6.2.2 Correlations of artist-artist similarities

Results of correlating artist similarities in the Last.fm dataset can be seen in the figure 6.2. Obtained correlations are slightly worse than in the preceding case, but still rather high considered the grounding approach we used, which, as far as we can say, is far from ideal and it is surprising that only three music styles(sometimes even two or one style) are enough to distinguish between the artists(again, see the section 4.3.2 for details).

Notice that we did not compute the similarities using distributional aggregation with frequency weighted representation. This was justified in the last subsection. Also note that Maximum Information Path measure is not present in the projection case and the cosine measure is not among the results of distributional aggregation. This, too, was explained in the preceding subsection.

It is interesting that all the measures performed similarly in this case with the overlap measure having slightly higher correlation scores than the other measures. Again, notice how the Jaccard coefficient and the Dice coefficient share almost the same results.

Since the grounding technique and data used in this case were technically the same in the last case, all of the comments given in the last subsection apply here as well.

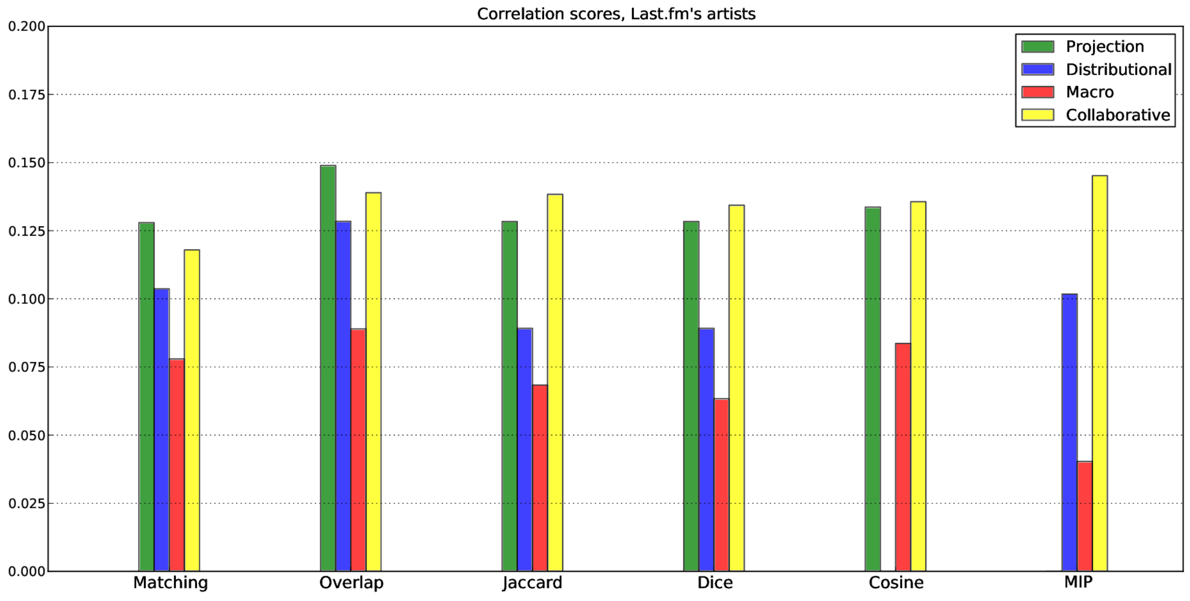

Figure 6.2: Kendall's τ correlation scores in Last.fm dataset, artist similarities

6.3 Delicious dataset results

6.3.1 Correlations of tag-tag similarities

These results are the only ones that share a grounding domain with [24]. Although the actual dataset is different(authors used Bibsonomy datasets in [24]), the domain is the same. Results are similar, but not quite the same. Namely, they differ in macro aggregation, which performed worse in our case, even yielding negative correlations for some measures. Also collaborative aggregation does not perform as good as in [24]. We will try to discuss why this is so later.

In the figure 6.3 reader can see that the scores obtained here are the worst so far. There are several possible reason why. For one, the distribution of tag frequencies is far from ideal, with the majority of tags only used a few times.

Macro aggregated measures perform exceptionally bad in this case - they produce even negative correlations. The reason for this has already been partially discussed in the section 5.5.1. The fact that the dataset contained so many unique resources, resulted in big portion of tag pairs having similarity equal to 1. This, of course, is not good for correlations. Notice that in the last section, Last.fm dataset did not have this problem (although the macro aggregation was the worse aggregation method in that case as well). This is probably due to convenient distribution of Last.fm's resources.

In [24], collaborative aggregated measures usually outperform the other aggregation methods, but as reader can see, this did not happen in our experiment. This, again, is likely caused by the structure of the dataset. By experimenting with results we have discovered that collaborative aggregated measures need sufficient information about each object, or at least the majority of objects, otherwise their performance will be very poor. Since there are many unique, or not very frequent resources in our delicious dataset, this is probably the reason for correlation scores so low.

When a folksonomy has a similar structure like the one just described, it is better to compute the similarities using projected methods, which does not seem to be affected with this.

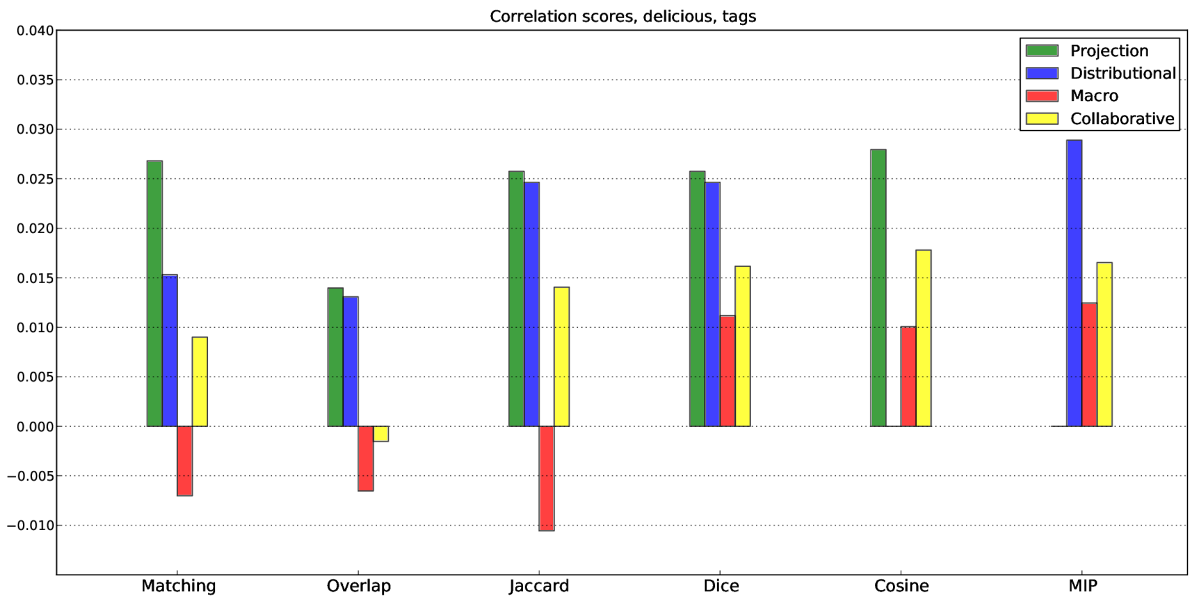

Figure 6.3: Kendall's τ correlation scores in delicious dataset, tag similarities

6.3.2 Correlations of resource-resource similarities

Following results are also comparable to the ones from [24]. Authors of the article used the same grounding data, but the grounding measure is different. It is more simple in our case, to be exact. But surprisingly, we have obtained higher correlation scores.

In order to minimize the impact of unique and not very frequent tags, we considered only ones assigned to more than 35 resources. This drastically improved the performances of user-oriented aggregations when compared to the subsection 6.3.1.

In the figure 6.4 can reader see Maximum Information Path measure performing the best. This is the only case, when this happened and it would suggest, that this measure will perform better as the number of objects in a folksonomy grows, because this number was the highest of all the cases we have calculated.

All the other rules already described in the other cases apply here as well.

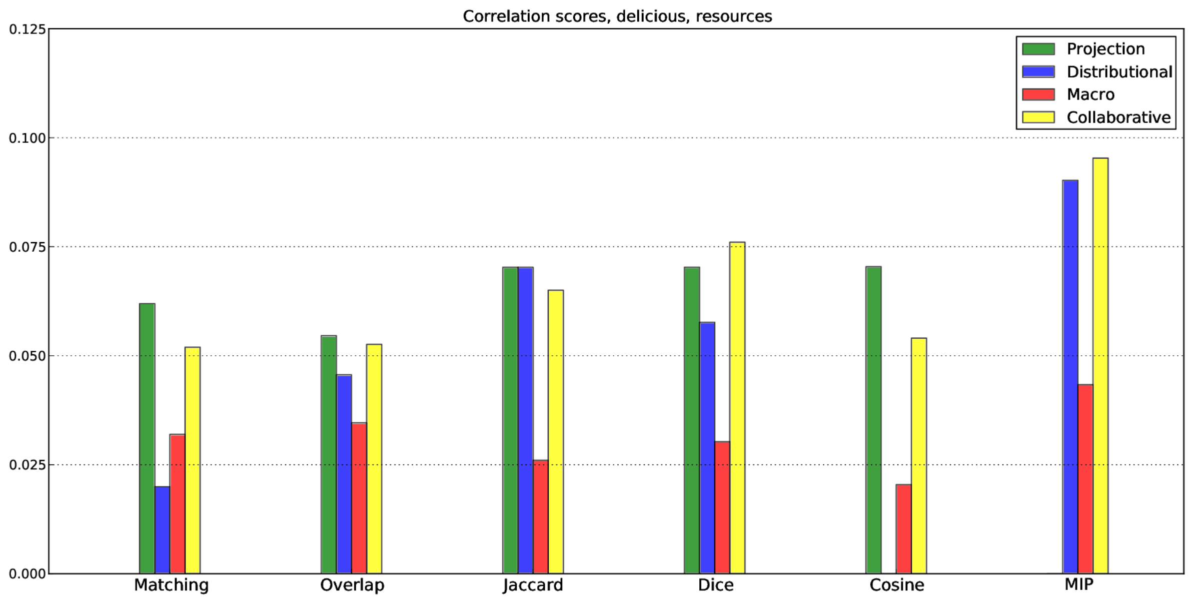

Figure 6.4: Kendall's τ correlation scores in delicious dataset, resource similarities

6.4 Medworm dataset results

As said before, using our grounding case, we did not obtain enough data to calculate the similarities properly, therefore it made no sense to perform the Kendall's τ. Suggestions how to utilize the Medworm dataset will be presented in the following chapter.

Chapter 7

Project evaluation and future work

We can say that the experiment was successful. We saw that measures of semantic similarity in folksonomies are applicable in several domains. From results in the preceding chapter, we can say that aggregations have very high impact on performances of semantic similarity measures. We have discovered that the simplest projection aggregation performs the best in most of the cases. However, it is problematic to maintain representations of big folksonomies using this aggregation method. When folksonomy is too big for projection aggregation, we can recommend using the collaborative aggregation method, as it was usually second in performance.

Future work contains bigger experiments with other datasets and more measures as well as experimenting with all the possible combinations of aggregations. We would also like to facilitate the Medworm dataset. One of the possible ways, how to do this, would be to search Medworm specifically for the diseases included in the Human Disease Ontology, collect tags assigned to articles about the diseases and then redo the experiment in the same fashion described in this text.

Bibliography

[1] Youtube: Broadcast Yourself. `http://www.youtube.com`, accesed on 2010-03-04.

[2] ODP - Open Directory Project. `http://www.dmoz.org/`, accesed on 2010-03-07.

[3] The iGraph library. `http://igraph.sourceforge.net/`, accesed on 2010-03-17.

[4] Dictionary and Thesaurus - Merriam-Webster online. `http://www.merriam-webster.com`, accesed on 2010-04-07.

[5] Digg - The Latest News Headlines, Videos and Images. `http://www.digg.com`, accesed on 2010-04-07.

[6] Delicous: Social Bookmarking. `http://www.delicious.com`, accesed on 2010-05-01.

[7] Fast (O(n log(n))) implementation of Kendall's Tau. `http://projects.scipy.org/scipy/ticket/999`, accesed on 2010-05-17.

[8] Musicmoz. `http://musicmoz.org/`, accesed on 2010-05-17.

[9] Pysparse. `http://pysparse.sourceforge.net/`, accesed on 2010-05-17.

[10] The free dictionaty: Log probabilty. `http://encyclopedia2.thefreedictionary.com/Log+probability` , accesed on 2010-05-17.

[11] The OBO Flat File Format Specification, version 1.2. `http://www.geneontology.org/GO.format.obo-1_2.shtml`, accesed on 2010-05-21.

[12] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly, Beijing, 2009.

[13] Steven Bird, Ewan Klein, and Edward Loper. Natural Language Toolkit. `http://www.nltk.org/`, accesed on 2010-03-04.

[14] Stewart Butterfield and Caterina Fake. Flickr. `http://www.flickr.com`, accesed on 2010-05-01.

[15] Sam Chapman. SimMetrics. `http://www.dcs.shef.ac.uk/ sam/simmetrics.html`, accesed on 2010-05-17.

[16] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945.

[17] Frank Dolan. Medworm: Medicine RSS. `http://www.medworm.com`, accesed on 2010-04-01.

[18] OBO Foundry. The Open Biological and Biomedical Ontologies: Human disease ontology. `http://www.obofoundry.org/cgi-bin/detail.cgi?id=disease_ontology`, accesed on 2010-05-01.

[19] Prasanna Ganesan, Hector Garcia-Molina, and Jennifer Widom. Exploiting hierarchical domain structure to compute similarity. Technical Report 2001-27, Stanford InfoLab, 2001.

[20] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[21] Knowledge and Data Engineering Group of the University of Kassel. Bibsonomy. `http://www.bibsonomy.org`, accesed on 2010-05-20.

[22] Dekang Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[23] Ana Maguitman, Filippo Menczer, Fulya Erdinc, Heather Roinestad, and Alessandro Vespignani. Algorithmic computation and approximation of semantic similarity. *World Wide Web*, 9(4):431–456, December 2006.

[24] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Gerd Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *18th International World Wide Web Conference*, pages 641–641, April 2009.

[25] Benjamin Markines and Filippo Menczer. A scalable, collaborative similarity measure for social annotation systems. In *HT '09: Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 347–348, New York, NY, USA, 2009. ACM.

[26] Felix Miller, Martin Stiksel, and Richard Jones. Last.fm. `http://www.last.fm`, accesed on 2010-05-01.

[27] Arvind Narayanan. Folksonomy dataset for NLP (delicious.com bookmarks). `http://arvindn.livejournal.com/115182.html`, 2009-09-07 [cit. 2009-09-07].

[28] University of Minnesota. Movielens. `http://movielens.umn.edu/`, accesed on 2010-05-20.

[29] Tim O'Reilly. What is Web 2.0. `http://oreilly.com/web2/archive/what-is-web-20.html`, 2005-09-30 [cit. 2009-09-30].

[30] Ted Pedersen. WordNet::Similarity. `http://wn-similarity.sourceforge.net/`, accesed on 2010-05-17.

[31] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

[32] Rossano Schifanella, Alain Barrat, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. Folks in folksonomies: social link prediction from shared metadata. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 271–280, New York, NY, USA, 2010. ACM.

[33] Stanford Center for Biomedical Informatics Research. The Protégé Ontology Editor and Knowledge Acqusition System. `http://protege.stanford.edu/`, accesed on 2010-05-21.

[34] Twitter, Inc. Twitter. `http://www.twitter.com/`, accesed on 2010-05-21.

[35] Christian Wartena and Rogier Brussee. Instanced-based mapping between thesauri and folksonomies. pages 356–370. 2008.

[36] Zhibiao Wu and Martha Stone Palmer. Verb semantics and lexical selection. In James Pustejovsky, editor, *Proceedings of the 32th Annual Meeting on Association for Computational Linguistics (ACL '94), June 27-30, 1994, New Mexico State University, Las Cruces, New Mexico, USA*, pages 133–138. Morgan-Kaufman Publishers, San Francisco, CA, USA, 1994.

Appendix A

Figures portraying contents of datasets

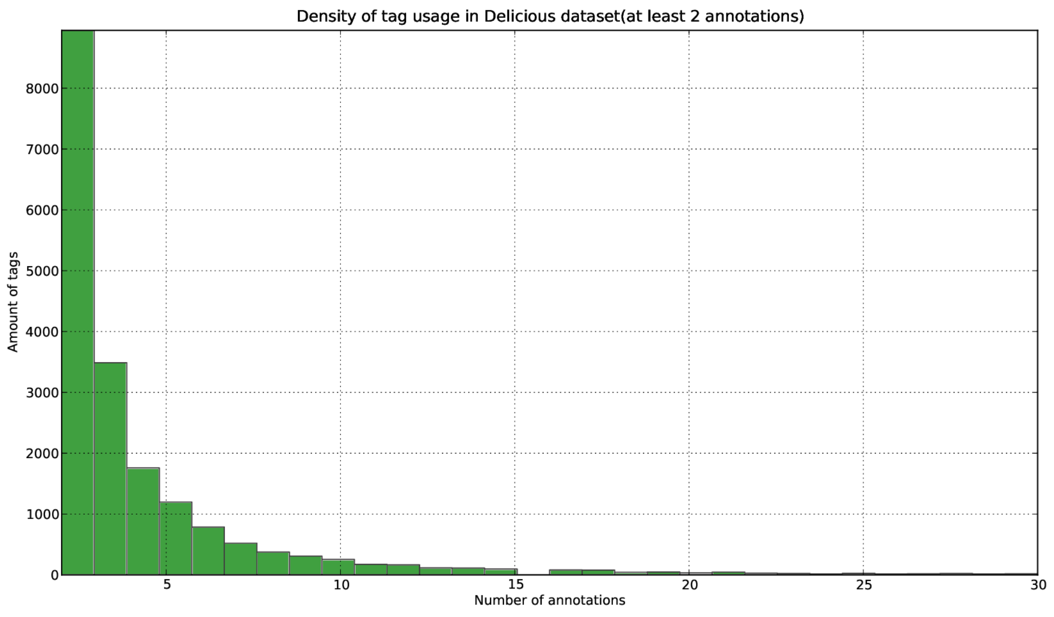

Figure A.1: Tag occurrences in delicious dataset

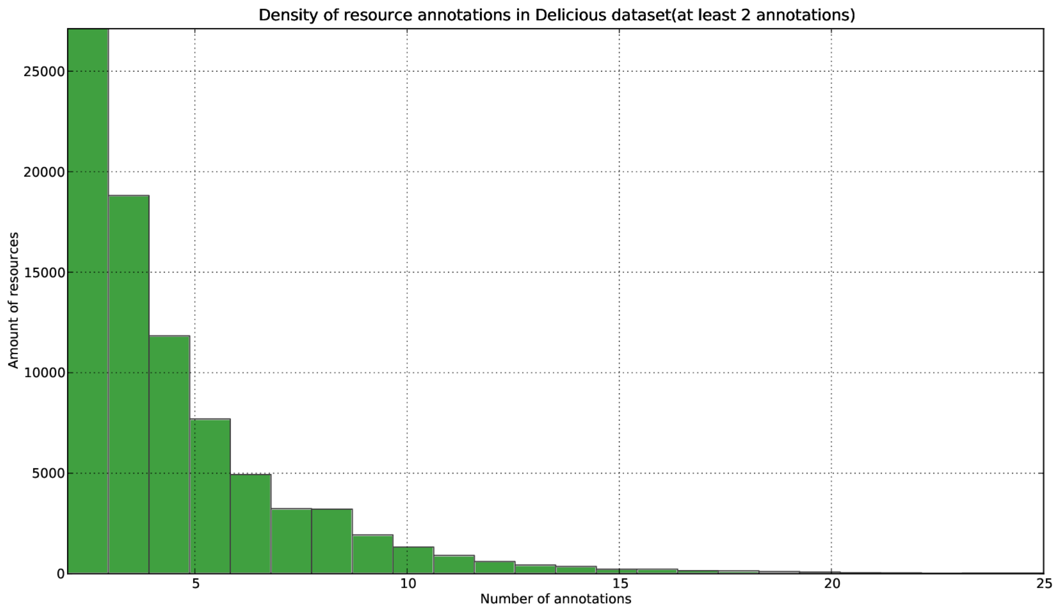

Figure A.2: Resource occurrences in delicious dataset

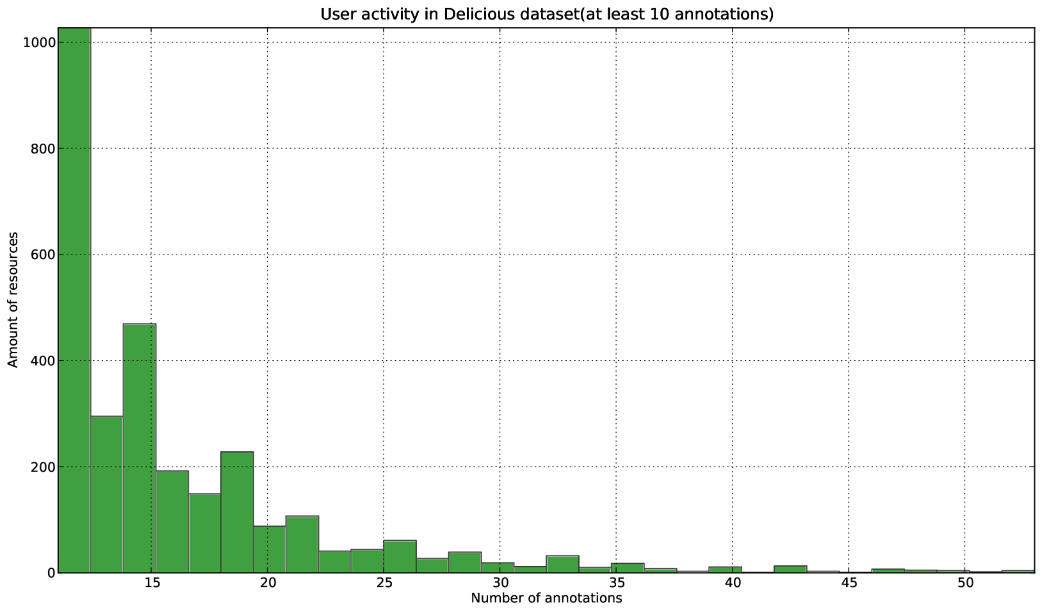

Figure A.3: User activity in delicious dataset

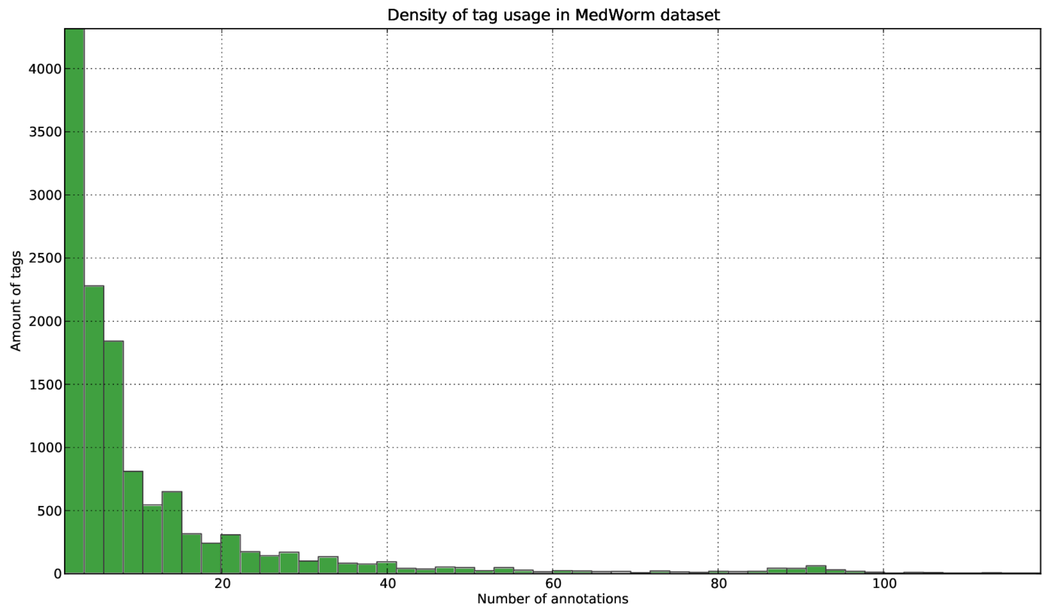

Figure A.4: Tag occurrences in Medworm dataset

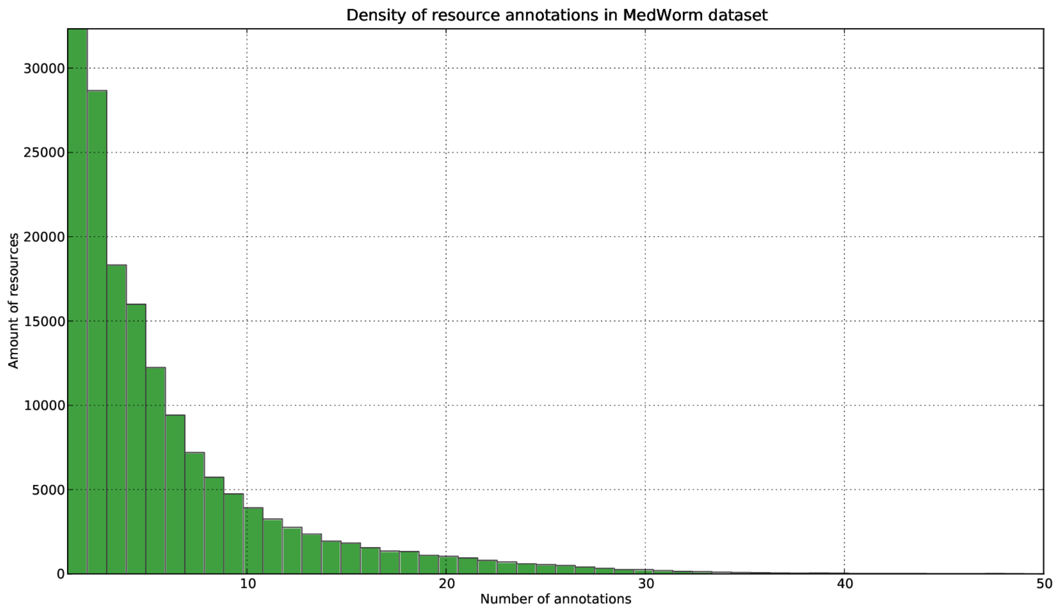

Figure A.5: Resource occurrences in Medworm dataset

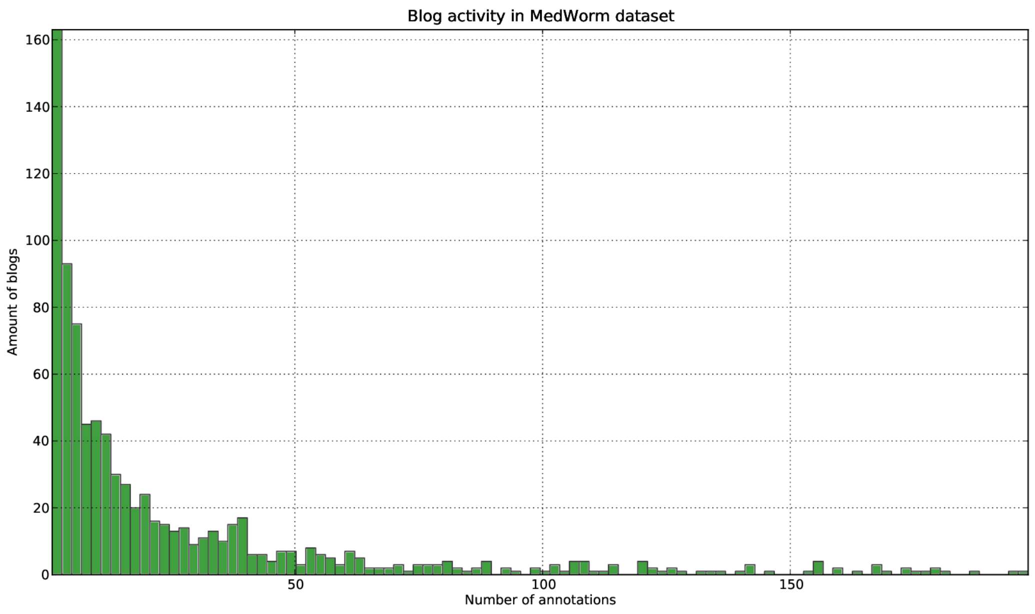

Figure A.6: Blog activity in Medworm dataset

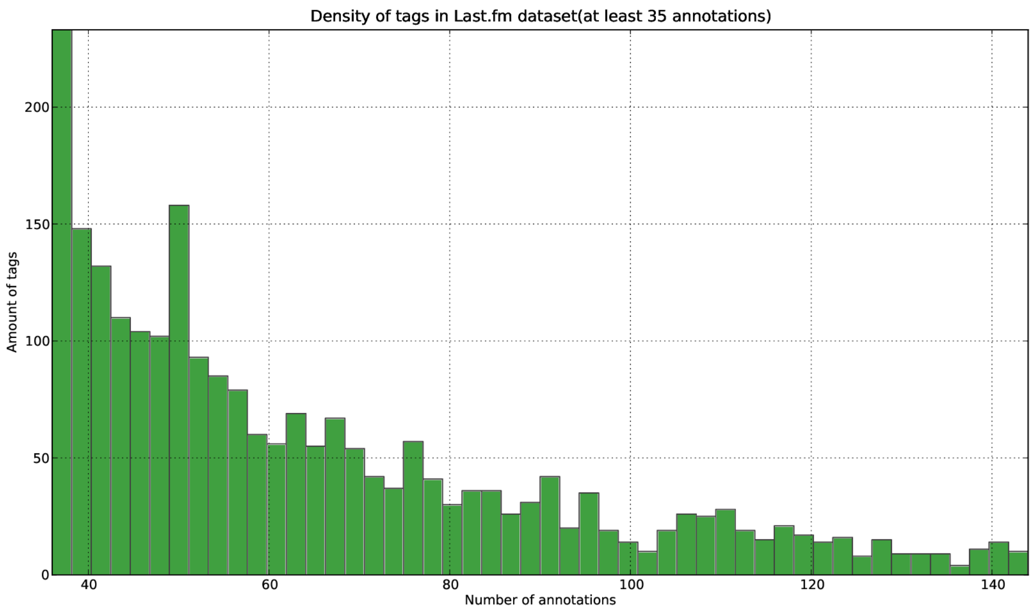

Figure A.7: Tag occurrences in Last.fm dataset

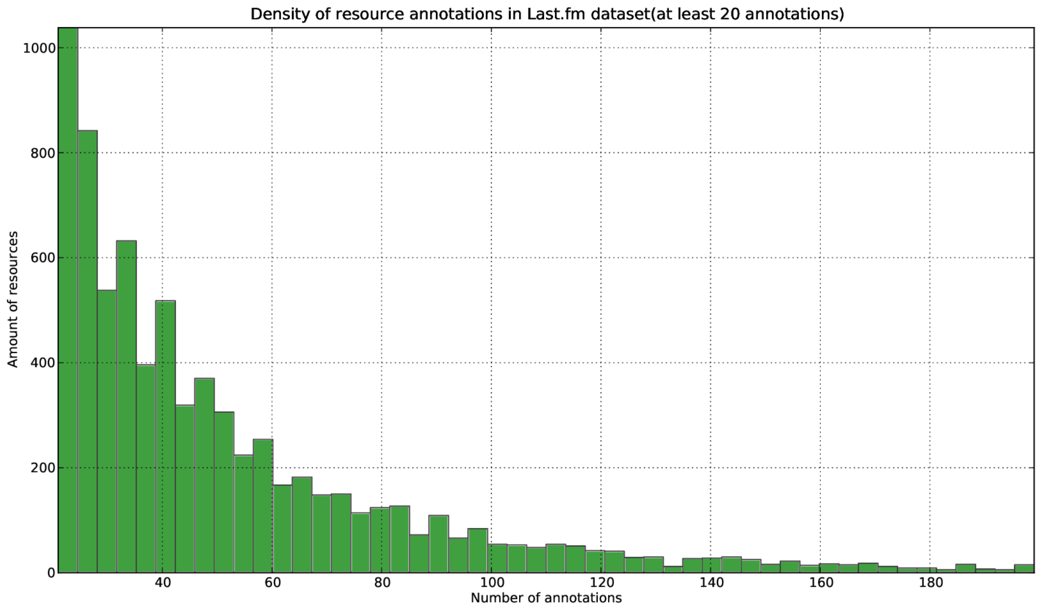

Figure A.8: Resource occurrences in Last.fm dataset

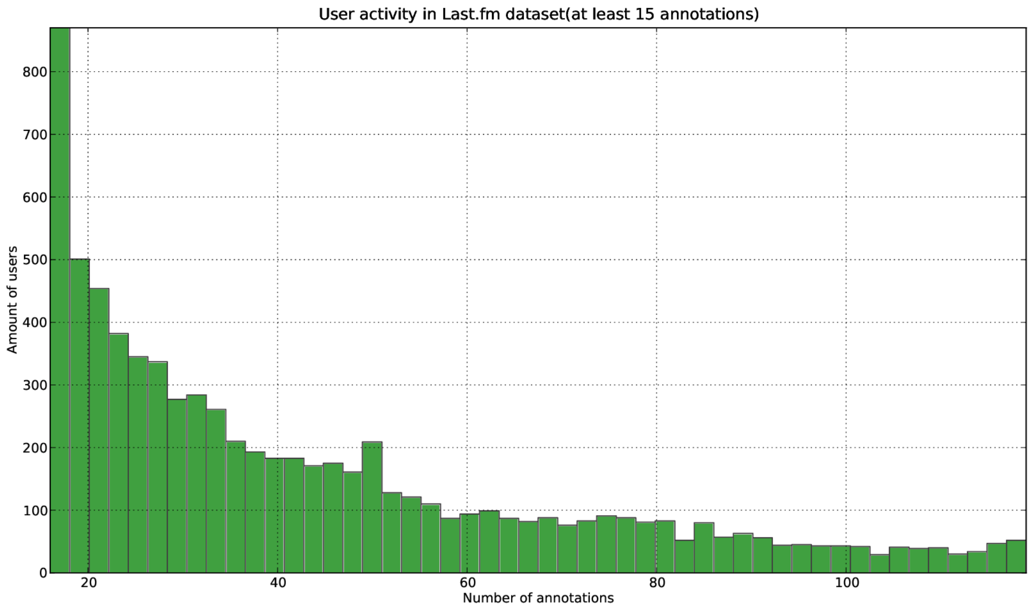

Figure A.9: User activity in Last.fm dataset

Appendix B

Example similarity vectors

artist	similarity
duke ellington	0.334905660377
ella fitzgerald	0.317647058824
billie holiday	0.307952622673
frank sinatra	0.299424184261
count basie	0.294117647059
chet baker	0.283687943262
ray charles	0.270916334661
charlie parker	0.263736263736
miles davis	0.257225433526
john coltrane	0.253164556962
nat king cole	0.248677248677
nina simone	0.243827160494
otis redding	0.241860465116
thelonious monk	0.238938053097
charles mingus	0.237006237006
benny goodman	0.233236151603
bill evans	0.232779097387
eva cassidy	0.227848101266
sarah vaughan	0.218527315914
chuck berry	0.214797136038
glenn miller	0.21472392638
marvin gaye	0.214022140221
django reinhardt	0.213457076566
james brown	0.209708737864
eric clapton	0.208897485493
john lee hooker	0.205655526992
sonny rollins	0.205263157895
norah jones	0.204081632653

(a) Dice coefficient, query: “Louis Armstrong“

artist	similarity
paul mccartney	0.141723263392
george harrison	0.140424769503
keith moon	0.0823378591302
suzi quatro	0.081771933173
john lennon	0.0648848485208
tom jones	0.0634460973821
the rutles	0.0626795594308
bill haley and the comets	0.0584866590598
three dog night	0.056067959477
bob seger	0.0559885224111
the doobie brothers	0.0558891931504
the guess who	0.0545497406456
electric light orchestra	0.052629781474
the searchers	0.0520066556907
robert palmer	0.0512775482645
the hollies	0.0494015924841
styx	0.0489090959196
smokie	0.0488329239163
barclay james harvest	0.0486986637921
the monkees	0.0477223149402
boston	0.0476449716736
neil diamond	0.0471838050516
jackson browne	0.0468865297302
slade	0.046660297013
the band	0.0464008713414
pat boone	0.0462982208769
pete townshend	0.0459930074101
phil collins	0.0459736040497

(b) Jaccard coefficient, query: “Ringo Starr“

Table B.1: Vectors obtained from Last.fm dataset, artist similarities

music style	similarity
alternative	0.739907997794
alternative rock	0.735932686233
classic rock	0.599588463921
hard rock	0.449813476425
indie rock	0.420326345427
britpop	0.346254783078
pop	0.326336072435
arena rock	0.310755315731
blues-rock	0.307499731673
progressive rock	0.295584742729
experimental rock	0.288555961069
psychedelic rock	0.284092756558
grunge	0.268979454273
metal	0.234887821593
blues	0.234132656063

(a) Distributed Jaccard coefficient

music style	similarity
alternative	0.42721499982
alternative rock	0.324558747617
pop	0.296433311584
indie rock	0.259361208986
hard rock	0.218894302679
punk	0.2071467544
classic rock	0.200696616448
metal	0.175829638554
progressive rock	0.171459796394
experimental	0.167755156663
folk	0.158586653758
electronic	0.150977298082
post-punk	0.121583555471
heavy metal	0.116267538366
electronica	0.116158004504

(b) Distributed Dice coefficient

Table B.2: Vectors obtained from Last.fm dataset, style similarities, query : “rock”

music style	similarity
rap	0.498255711889
underground hip-hop	0.19591270919
soul	0.151825413846
trip-hop	0.145643693357
funk	0.12120628023
underground rap	0.107136363462
urban	0.10474938279
electronica	0.104217012263
dance	0.101311116644
downtempo	0.0939935709022
reggae	0.0884420597297
electronic	0.0866522211842
jazz	0.0862012474577
pop	0.080758841329
alternative	0.0787230777526

(a) Distributed Jaccard

	similarity
rap	0.840465027813
underground hip-hop	0.476924329006
underground rap	0.434825584777
east coast rap	0.320865500009
old school rap	0.31907420988
hardcore rap	0.291135693536
jazz-rap	0.241679343475
west coast rap	0.240826421442
urban	0.218592107905
ambient breakbeat	0.125543658614
electronic	0.0988594383794
trip-hop	0.0972454558386
work songs	0.0932610035419
electronica	0.0929776264392
cumbia	0.0902934978776

(b) Distributed cosine

Table B.3: Vectors obtained from Last.fm dataset, style similarities, query: “hip-hop“

tag	similarity
development	4997.88919176
tutorial	2791.21791003
reference	2372.45252098
software	2237.36390851
tools	1972.29346237
web	1924.68338259
java	1696.38993443
python	1625.10609129
code	1286.86567368
design	1194.51303287
library	1063.03987527
tutorials	1006.25327268
linux	879.33149539
.net	870.541554346
blog	855.547284837
framework	819.151380948
tips	781.190626738
ruby	774.887035362
ajax	675.020956
graphics	596.27177037
language	578.704483429
free	495.691745629
documentation	477.916823233
database	470.973947074
rails	449.994275804
mac	432.200487887
flash	425.089962945
technology	416.937937249
book	413.777961978
tool	400.890328223

(a) Distributed matching, query: “programming“

tag	similarity
game	0.215004574565
gaming	0.13940415964
interactive	0.139130434783
fun	0.104793756968
math	0.101855848079
flash	0.0898954703833
kids	0.0891530460624
education	0.0867984548966
online	0.0630876554443
maths	0.05039193729
free	0.048632218845
literacy	0.0484210526316
resources	0.0458281444583
learning	0.0423060970552
science	0.0419490158116
reading	0.0402999062793
3d	0.0356489945155
teaching	0.0319240724763
entertainment	0.031185031185
software	0.0310842932894
freeware	0.0310746655158
activities	0.0306965761511
puzzles	0.029702970297
programming	0.0295819935691
retro	0.0285551113649
download	0.02795145274
technology	0.0270508673919
cool	0.0268514508445
indie	0.0266990291262
simulation	0.0257510729614

(b) Projected Dice - query: “games“

Table B.4: Vectors obtained from delicious.com, resource similarities

resource	similarity
www.diynetwork.com/	0.452217967874
channel.nationalgeographic.com/	0.452217967874
www.weather.com/	0.452217967874
www.channelone.com/	0.444527957042
www.abc.net.au/	0.444527957042
www.pbs.org/wgbh/pages/frontline/	0.436448552308
www.charlierose.com/	0.436448552308
www.televisionwithoutpity.com/	0.399690285519
www.foodnetwork.com/	0.395751480366
www.bravotv.com/	0.395751480366
science.discovery.com/	0.395751480366
www.nickjr.com/	0.395751480366
www.fox.com/	0.395352203142
www.cbs.com/	0.37378309765
abc.go.com/	0.37378309765
www.cbc.ca/	0.37378309765
www.nbc.com/	0.37378309765
pbskids.org/	0.335985458244
www.cnbc.com/	0.312290982818
www.fox.com/24/	0.301276151264

(a) Lin's measure, ODP dataset

resource	similarity
www.comedycentral.com/	0.203637883222
www.un.org/news/	0.200890220788
news.bbc.co.uk/	0.189982317246
abc.go.com/	0.18305398268
edition.cnn.com/	0.164095319198
www.abc.net.au/	0.163287049647
www.bbc.co.uk/worldservice/	0.151909334488
www.greenleft.org.au/	0.151425309199
www.einnews.com/	0.142042308082
www.economist.com/	0.141871044513
www.newser.com/	0.135498482325
www.casttv.com/	0.134395440973
www.theworld.org/	0.127018561364
www.eurozine.com/	0.124704341468
home.disney.go.com/tv/	0.122592394726
www.biography.com/	0.119333858889
www.pbs.org/wgbh/	0.11724505478
www.allmovie.com/	0.115584411981
www.alternet.org/	0.113120261559
www.dominionpaper.ca/	0.113086159974

(b) Distributed Jaccard, delicious

Table B.5: Comparison of different similarity measure approaches, query: "cnn.com"