WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Master Thesis

Categorization and comparison of datasets across Open Data portals

Georg Prohaska

Date of Birth: 14.11.1983
Student ID: 0325904

Subject Area: Information Business

Studienkennzahl: J 066 925

Supervisors: Univ.Prof. Dr. Axel Polleres, Dr. Jürgen Umbrich

Date of Submission: 16. October 2017

Department of Information Systems and Operations, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR INFORMATIONS-
VERARBEITUNG UND PROZESS-
MANAGEMENT DEPARTMENT
OF INFORMATION SYSTEMS AND
OPERATIONS

Contents

List of Figures

List of Tables

Listings

Abstract

The success of the Open Data movement which strives to provide free access to all kinds of data on the web has led to an increased amount of available Open Data. The OpenData@WU project is harvesting metadata of 260 portals that provide access to such data. Among other aspects, this metadata contains the title, description and keywords of each dataset. In our work we apply Babelfy, a state of the art natural language processing technique, to these natural language fields in order to gain insights into the current state of the content and structure of Open Data. We present an approach to automatically categorize datasets into one of 34 categories based on their textual descriptions. Furthermore, we employ clustering techniques on the results to classify Open Data portals and discover similarities among them.

Zusammenfassung

Die Open Data Bewegung, welche den offenen Austausch frei nutzbarer Daten anstrebt, hat zu einer großen Menge an verfügbaren Open Data geführt. Das OpenData@WU Projekt sammelt Metadaten von 260 Open Data Portalen. Unter anderem enhalten diese Metadaten Titel, Beschreibung und Schlagworte zu den jeweiligen Datensätzen. In der vorliegenden Arbeit setzen wir Babelfy, eine moderne Natural Language Processing Methode, ein um diese Texte maschinell zu verarbeiten und dadurch Aufschluß über die Inhalte und die Struktur von Open Data zu erlangen. Weiters stellen wir einen Ansatz zur automatischen Kategorisierung von Datensätzen in eine von 34 Kategorien basierend auf deren textuellen Beschreibungen vor. Die Ergebnisse dieses Kategorisierungsprozesses werden mit Hilfe von Clusteringverfahren untersucht um Ähnlichkeiten zwischen Open Data Portalen zu finden.

Acknowledgements

I want to use this space to thank my supervisors Dr. Jürgen Umbrich and Dr. Javier David Fernandez Garcia. They always had time for me, answered all my questions and gave me very helpful feedback and suggestions.

I also would like to thank my family for their moral support, especially my wife Gabriele who is always there for me and helped me keep my motivation up throughout this study.

1 Introduction

In the years to come, digitalization will impact our society as well as economy, science and culture even further than it is today. Digital data has become a modern commodity and technologies like Social Media, Big Data, Internet of Things, Internet of Services and Open Data present completely new opportunities for designing services, developing business ideas and increasing productivity.
In this thesis, we will focus on Open Data, which is a movement to make more data publicly available. Open Data facilitates transparency as well as improvements in effectiveness and efficiency of public administration and the private sector [37]. It is a major source for innovation and can, therefore, contribute to designing new and creative products and services. In recent years, this concept of Open Data has become more and more relevant. Every day an increasing amount of data is made available for the public to use. There are various providers of Open Data: A lot of different governments and international institutions, such as the EU, have been publishing their data about numerous subjects including geography, transport, environment, economics and many more. There are also private organizations and other institutions, such as universities, that contribute further to the amount of Open Data that is available for everyone to use. Usually, each of these parties manages its own portal to provide access to their individual data sets. Often there are even multiple portals within a country managing different regions or themes of Open Government Data.
OpenData@WU[1] is a project of the Institute for Information Business at the Vienna University of Economics and Business that is dedicated to researching the various aspects of Open Data. In the course of this project the Open Data Portal Watch framework is being developed, which provides access to the metadata of over 260 Open Data portals [40]. This metadata includes, among other attributes, the title, description and keywords of each dataset of all covered portals. These fields contain mostly plain text written by the publisher of the respective dataset. Since this kind of information is intended to be read by a human user no machine-readable data can directly be dervied from them. Hence, knowledge about the context and contents of a dataset are not directly attainable for a computer, which increases the difficulty of analysing and searching the available data. Full-text search is the only method that could be applied directly to this metadata, however, research shows that this is not a very effective way of searching because of contextual ambiguity of simple search terms leading to low precision and re-

[1] `http://data.wu.ac.at/opendata/`

call for such methods [11]. This is especially true for Open Data, since end users often take a browsing or exploratory approach rather than looking up specific information [67].
Translating the textual descriptions of datasets into information that can be processed and analysed by a machine would open up various possibilities ranging from advanced searching methods like faceted-search or concept-based search to automatic classification of datasets and content-based comparison of Open Data portals.

1.1 Research Question and Approach

Natural Language Processing (NLP) is a field of research that aims to create machines that are able to understand natural human language. This is a complex task for which findings of linguistic science need to be combined with the newest methods of computer science and artificial intelligence. For this thesis, we will research methods of Natural Language Understanding (NLU) and, consequently, apply them in order to extract processable semantic information from the plain text descriptions of datasets. NLU involves multiple challenges, the two main ones being *polysemy* – the same word can have multiple meanings depending on the context - and *synonymy* – two or more distinct words or phrases can have the same meaning [24].
Tackling the problem of polysemy involves identifying the specific meaning of a word in context. In literature, this is known as word-sense disambiguation (WSD). There are various approaches to solve this problem, including dictionary-based, statistical and machine learning approaches, all of which are going to be discussed thoroughly in this thesis [64]. The result of a NLU algorithm, including proper WSD, is a semantic representation of a text that can be interpreted by a computer. Such a meaningful representation must be grounded on a structured lexical knowledge base. The creation of such a resource is another very challenging aspect of NLP. While multiple sources for lexical knowledge and ontologies such as Cyc [42] or WordNet [20] exist our special focus will lie on BabelNet [54], since that is the resource that will be used as a base for processing the descriptions of the datasets.
In order to achieve automatic understanding of natural-language text another task needs to be performed and that is Entity Linking (EL)[19]. EL tries to recognise entities within a text and link them to a reference knowledge base such as, for instance, BabelNet. There, all possible entities are catalogued and recorded together with their meaning. Additionally, BabelNet provides semantic relations between the entities, such as is-a, instance-of or part-of relations. We will discuss different approaches for WSD as well as EL in this thesis. One particularly interesting approach that addresses both of these

issues is Babelfy [50]. It is a graph-based algorithm that provides disambiguated entity recognition of natural-language text. Babelfy utilizes the BabelNet semantic network as a knowledge base. The state-of-the-art performance of this algorithm together with the fact that BabelNet is currently the largest multilingual knowledgebase lead us to the decision to directly apply Babelfy to the textual description of datasets and analyse the results.
In this thesis, we are first going to give an introduction to some NLP tasks, namely WSD, EL and text categorization. Furthermore, we are going to discuss the clustering techniques which come back into use in the analysis part of our work (section 7). Of course, related work of fellow researchers is going to be outlined. Then, we are going to present a solution that makes use of the Babelfy algorithm to achieve entity recognition in Open Datasets. Moreover, a thorough analysis of the resulting data is going to be given. Here the focus lies on comparing the outcome for different Open Data portals. These goals can be summarised into a research question for this thesis as follows:

Can automatic entity recognition be used to discover the subject area of Open Datasets?

Another aspect that makes NLP difficult is language diversity. Since the Open Data movement has spread all over the globe there are an increasing number of datasets available in various languages. The Open Data Portal Watch project, which provides the required meta data for our research, covers datasets published in over 20 distinct languages. This is another advantage of BabelNet as well as Babelfy: they both cover all the required languages, and many more.
There are multiple potential use cases for discovering entities (or concepts as we will call them from here on out) in Open Datasets: The subject area of datasets could be discovered and, by aggregating this information, the key topics and themes of the portals themselves could be derived. These insights could be used by portal administrators to better understand and improve the structure of their content. Utilizing the concepts for improved search is another interesting aspect. Since BabelNet provides multiple relations between the concepts, an idea would be to leverage them to create a faceted search hierarchy. A use case that we will investigate even further is automated categorization of datasets. Many concepts in BabelNet belong to one of 34 distinct categories. We developed an algorithm that utilizes this information to derive the category of a dataset. Results and comparisons for 150 portals and over 170.000 datasets will be presented.

1.2 Goals

The overall goal of this study is to discover the potential of the semantic information "hidden" in the natural language descriptions of open datasets. After deciding for Babelfy as a NLU technique to achieve entity recognition within the datasets we tried to explore the various possibilities these discovered entities presented.
Initially, we investigated the use case of search, in particular facetted search, since simply using the concepts as facets seemed reasonable at first glance. However, the substantial number of distinct concepts found by Babelfy made it clear that some sort of hierarchy or mapping between the concepts would be necessary to achieve a useful solution. We decided that such a task is out of scope for a single master thesis, especially because the semantic network of BabelNet is very large and complex with over 30 different relations between the concepts and no clear or easily derivable hierarchy. That said, this is definitely an aspect that should be investigated further in future work, since the potential for improving search over Open Data is unquestionable.
When reviewing the API of BabelNet we discovered that many of the concepts are labelled with one of 34 categories. These categories range from general ones like Business Economics and Finance to more specific ones line Numismatics and Currencies. There is no hierarchy and the set of categories is fixed, however, they provide a good basis for a high-level classification of any content. This feature of BabelNet lead us to the idea to use the discovered concepts indirectly to classify datasets into these 34 categories, which hopefully would give a good overview of the content of the datasets. Since the set of categories is fixed a comparison between portals was not far to seek.
In summary, the goal was to (i) use Babelfy to discover the concepts of a dataset, (ii) exploit the predefined categories of BabelNet concepts to classify a dataset and (iii) to employ clustering methods to the results to find similarities between Open Data portals. During the study, it became clear that manual tweaking of parameters for such a categorization algorithm is somewhat necessary in order to achieve reasonable results. Therefore, we decided to build a prototype called *Categorizer* that provides a graphical user interface for parameter setting and exploration of the results of categorization. Finally, of course, a thorough analysis of the outcome had to be conducted.

2 Theoretical foundations

In this section, we are going to discuss the theoretical foundations that are necessary to tackle our problem. We are going to explain what Open Data is exactly and what kinds of Open Data are distinguished. Furthermore, we are going to take a look at the RDF data model, which is closely related to Open Data. Then, we are going to explore natural language processing and discuss some tasks specific to that area, namely word-sense disambiguation, entity linking and text categorization. Finally, some clustering techniques which were used for this study are going to be explained.

2.1 Open Data

Open Data refers to *"A piece of content or data is open if anyone is free to use, reuse, and redistribute it – subject only, at most, to the requirement to attribute and share-alike."* [3] The motivation behind this idea is that Open Data will facilitate innovation, creativity and new businesses, which will consequently lead to economic growth. The amount of data that is available is growing at an ever increasing rate. The Open Data Barometer of 2015 [23] in Figure 1 shows that especially countries of the western world have high capacity for Open Data, which means that they all have established policies for this subject.

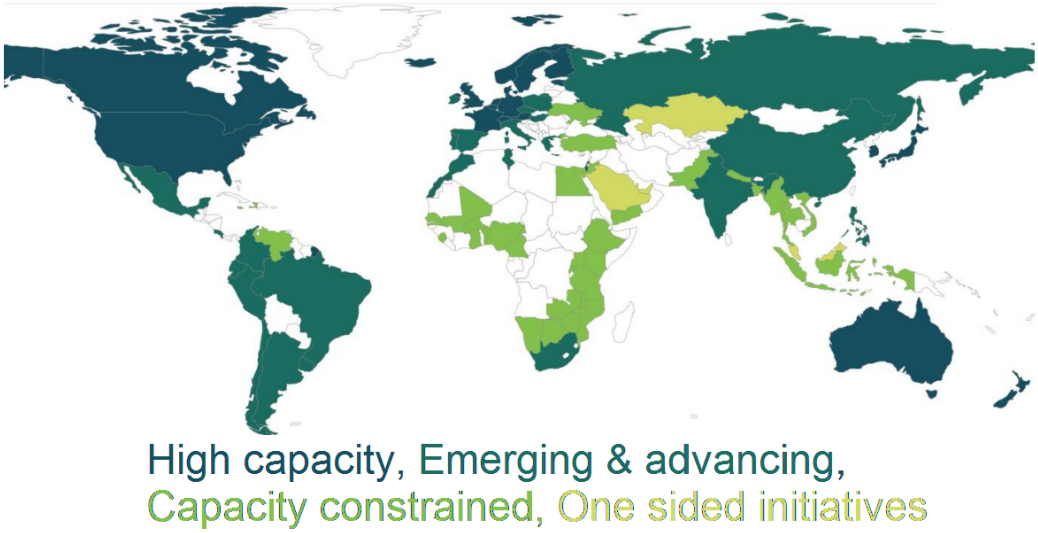


Figure 1: Country clusters based on Open Data Barometer Readiness and Impact questions [23].

However, the availability of data alone does not contribute much to eco-

nomic growth. Creative individuals or businesses that take advantage of the possibilities of Open Data are needed to achieve that goal. This is a process that takes some time. Nevertheless, some believe that the number of individuals and businesses that are already engaged in Open Data has reached the critical mass necessary to trigger a set-change in business attitudes towards this matter [29]. Goods and services are being enhanced by the use of Open Data. Furthermore, more and more new businesses, like Duedil, i3 Education Services, and more emerge that base their entire business model around finding new ways to make use of the available data. Big players like Google are also starting to open up their own data in order to increase customer satisfaction and to facilitate easier supplier management. In the future Open Data might have a significant impact on our society. Yet, this process is still in its infancy and a lot of work remains to be done in order to reach the full potential of the available possibilities.

2.1.1 Open Data Portals

Open Data is usually distributed over public websites which we are calling Open Data portals. Thousands of distinct portals are currently available from all over the world. There are many different approaches on how to present the data to the user. A lot of portals provide interfaces that only allow searching over the metadata of the datasets, while the actual data is stored in downloadable files. There are numerous examples for this approach, one would be the Open Data Portal of the Vienna University of Economics and Business[2]. Other websites, such as Worldatlas[3], provide a more integrated type of interface that also utilizes some visualization techniques.
At this time, according to the Portal Watch project 260 Open Data portals are active around the world. A large portion of these portals make use either CKAN[4] or SOCRATA[5] two software frameworks that provide means to publish and search Open Data. Apart from the evident title, description and keywords most datasets include further metadata such as date of issue, date of last update, type of license the dataset is published under, format of downloadable files, name of the publishing organization, et cetera. It should be clear, that, at this time, Open Data is rather scattered across numerous access points, which is probably why little research has been conducted on the content and structure of its entirety. With this thesis we try to make one step into that direction.

[2] http://data.wu.ac.at/
[3] http://www.worldatlas.com/
[4] http://ckan.org/
[5] http://socrata.com/

2.1.2 Open Government Data

Open Govenment Data (OGD) is that part of the worldwide Open Data that is provided by different governments. The goal is to increase transparency of government and administrative structures and to encourage citizens to take part in the democratic process. This notion became increasingly popular around the world since the G8 leaders signed an Open Data Charter in 2013 where they agreed to *"Establish an expectation that all government data be published openly by default, ..."* [2]. This lead many countries to launch Open Data intitiatives publishing many different kinds of data. However, some critizise that many countries have yet to open up significant core data about government spending, company registers or public sector contracts, which would contribute alot to transparency and consequently help reduce corruption [23]. While, in many regions around the world, there is still a lot of improvement possible in this matter, the trend seems to go into the right direction. OGD is published mostly in formatted files. TSV, XML, JSON, CSV are just some of the many different open formats that are currently used to store the data. Evidently, this represents a considerable barrier for the end user. A standard format has not yet been established. Furthermore, inconsistencies or poor description of the datasets is very common in OGD[40], which also renders searching the available data hard for the end user. These issues are most likely not going to be solved by the providers of the data in the near future, since, from a technical standpoint, government institutions tend to improve rather slow. However, since OGD is in fact open, third parties have the opportunity to reach improvements or rather to create completely new solutions using OGD.

2.2 Linked Open Data

Linked Open Data (LOD) is another form of Open Data. It refers to data that is organized in such a way that its meaning can be interpreted by a computer and that it can be linked to other external data sets [71]. This concept was introduced by Tim Berners Lee in 2006 in a Web architecture note [10]. The Resource Description Framework (RDF) [5], which is a data model for describing relationships between resources, has become the de facto standard for creating LOD. In recent years the lines between the Semantic Web, which is a term that has been utilized frequently in the past, and LOD have become blurred [71]. The ultimate goal is to create the so called Web of Data that can be read and interpreted by a machine. This is also commonly called "The Linked Open Data Cloud"[6] which has been growing rapidly in

[6] `http://linkeddata.org/`

recent years. Figure 2 shows a graph of the current state of the LOD cloud. Each circle represents a dataset and the radius depends on the number of triples that source provides. DBPedia for example has about 3,000,000,000 triples. The arrows indicate the existence of at least 50 links between two datasets [4]. It should be clear now that there is already a massive amount of information out there and that the potential for innovative applications is particularly high. However, because of the complexity and in-homogeneity of the LOD cloud even a basic task such as searching poses major difficulties for software developers. Hence, different research topics have emerged aside, such as semantic search on LOD, integrating large number of linked data sources, mining the web of linked data, quality evaluation of linked data, et cetera. As a result, we are definitely going to see more and more diverse and interesting applications being developed in the near future.

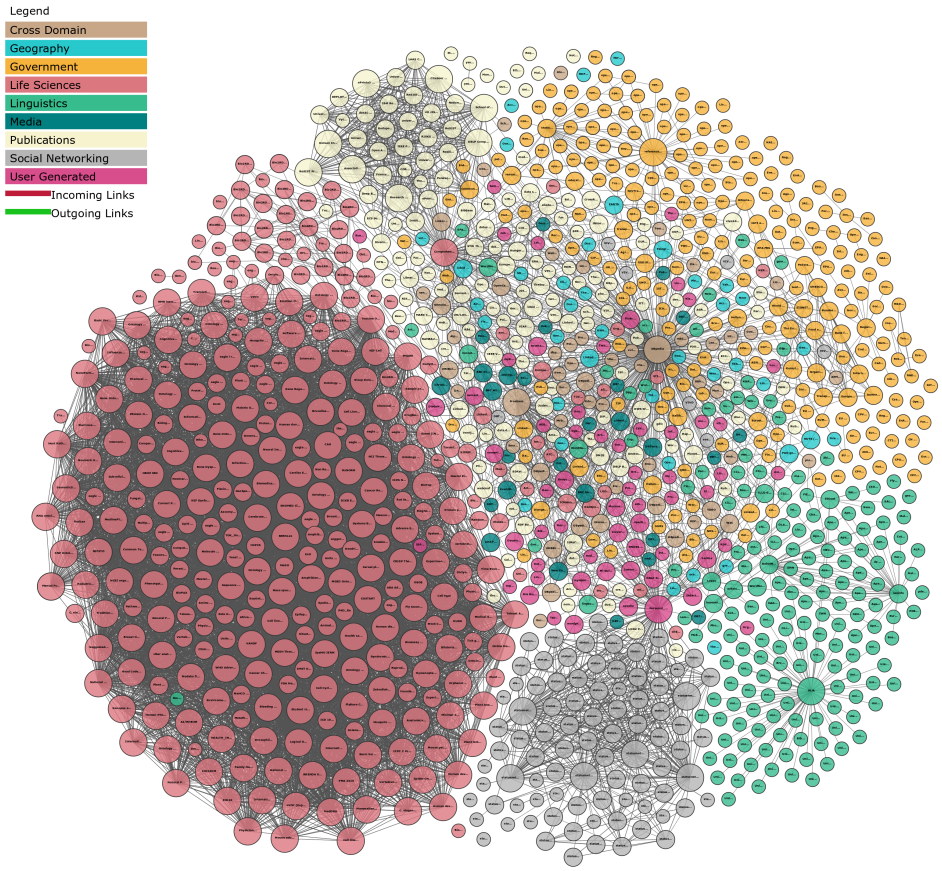


Figure 2: The Linked Open Data cloud [4].

2.2.1 RDF

The Resource Description Framework (RDF) [5] is a model to express logical statements about resources. A resource is something that is unique and that one wants to make a statement about. It can be anything from a webpage to a physical or abstract entity. Resources are identified by so called Uniform Resource Identifiers (URI) which are unique strings of characters. Since a lot of resources are in fact websites URIs are often denoted in the form of Uniform Resource Locators (URL). However, URIs do not have to necessarily be reachable on the web. In the latest recommendation of the RDF-model the International Resource Identifier (IRI) has been introduced, which is a generalization of the URI. It allows non-ASCII characters to be used in the character string.
In the RDF-model each statement consists of three parts: subject, predicate and object. The subject – a resource – is described using the predicate and the object, which can be resources, just values expressed by literals or even blank nodes. Blank nodes represent something that is not specified concretely. They are used like simple variables in algebra [5]. The predicate characterizes the relationship between the subject and the object. These three units are called a RDF-triple. The RDF-model can be mathematically understood as a labeled direct graph, where the directed arcs point from subjects to objects via predicate labels with meaning. The following code shows a simple example for three RDF triples:

Listing 1: An RDF example in RDF/Turtle notation

```
:dataset1
        a dcat:Dataset;
        dct:title "Imaginary dataset";
        dcat:keyword "water" .
```

This defines "dataset1" as a dataset under the DCAT Vocabulary [17]. The second triple states that the title of this specific dataset is "Imaginary dataset". The quotations imply that the object is a literal. The last triple indicates that the keyword for this dataset is "water", again a literal.
RDF was originally developed by the World Wide Web Consortium as a standard to describe metadata. However, it has been widely employed for building the Semantic Web i.e. the LOD. There are various common serialization formats for RDF, namely N-Triples, N-Quads, JSON-LD, N3, RDF/XML and Turtle (see main RDF formats in [5]). They each have different advantages, for example Turtle is often used by developers of the Semantic Web since its syntax is very easily readable for a human.
By now, a lot of different RDF-vocabularies have been developed that define

various predicates and objects in order to provide a base line for formulating statements. For instance, the very commonly used FOAF vocabulary[7] specifies characteristics of people and social groups, such as name, age or title.

2.2.2 SPARQL

To query RDF-data the SPARQL language [63] has become the most commonly used standard. Its syntax and structure resembles SQL. SPARQL has been officially recommended by the World Wide Web Consortium in 2008. It is based on graph pattern matching. A SPARQL query normally consists of one or more sets of triple patterns. These patterns stand for the RDF subject, predicate and object. The components of each triple may be replaced with a variable instead of an IRI or literal. These patterns may also be joined in order to create more complex queries. Consequently, these sets are matched against the RDF data. The results of SPARQL queries can be result sets or RDF graphs. SPARQL also supports aggregation, subqueries, negation, creating values by expressions, extensible value testing, and constraining queries by source RDF graph [63].

2.2.3 DCAT

For this project we needed a way to describe the metadata of all the datasets coming from different Open Data portals. For this purpose the Data Catalog Vocabulary (DCAT) [17] was a good choice, since it covers everything from title over keywords to a description of the catalog i.e. the portal. Figure 3 shows the structure of the DCAT vocabulary. It essentially consists of three main classes:

1. dcat:Catalog
2. dcat:Dataset
3. dcat:Distribution

A Catalog contains one or many datasets, in our case it translates to one of the Open Data portals. In DCAT a Dataset is defined as a *"collection of data, published or curated by a single agent, and available for access or download in one or more formats"*[17], so it does not necessarily have to be a downloadable file. However, in our project all datasets have that property.

[7] `http://www.foaf-project.org/`

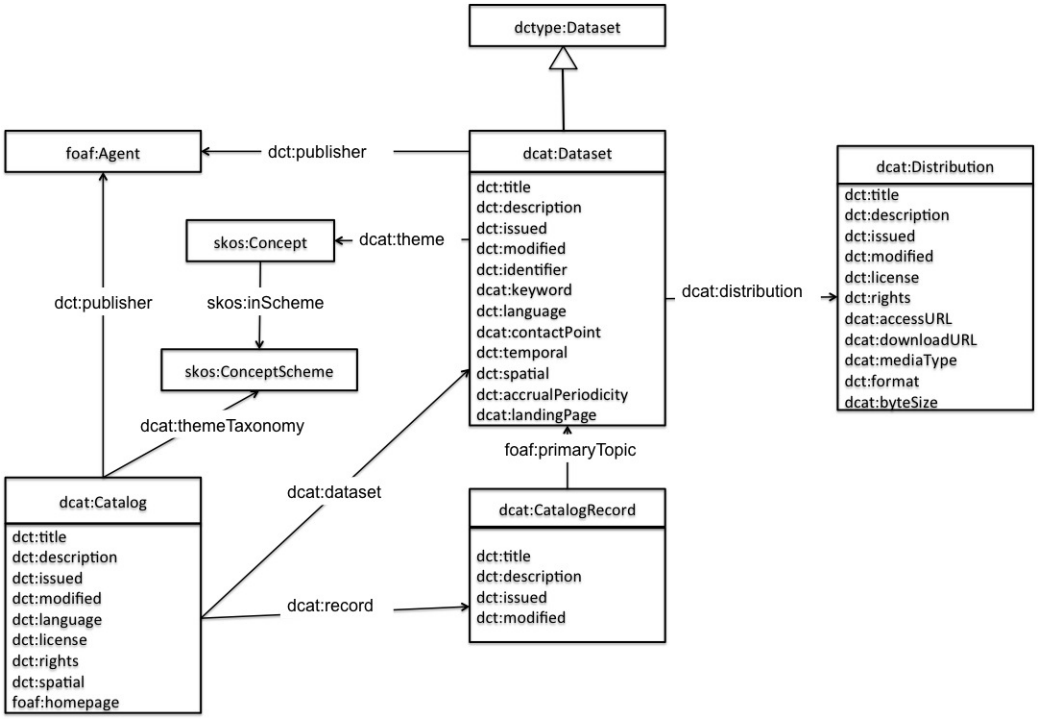


Figure 3: Structure of the DCAT vocabulary [4]

Hence, each has a related distribution which represents the file itself. Another RDF-vocabulary that is employed for describing datasets is the VoID vocabulary[8]. It was also created for expressing metadata about datasets, however, it focuses mainly on RDF-datasets and most of the files managed in this project are not actual RDF-files.

2.3 Natural language processing

Natural language processing (NLP) is a field in computer science that focuses on digitally representing and processing natural languages such as English. The enormous and ever-growing amount of human-written text that is available on the World Wide Web makes new methods for search and automatic categorization necessary. NLP uses artificial intelligence methods to address this problem. The goal is to enable a computer to communicate with a human via natural language, be it speech or written text. NLP may be divided into two parts: Natural language generation (NLG) and natural language understanding (NLU). In the following, we are going to discuss each aspect in more detail.

[8] `https://www.w3.org/TR/void/`

2.3.1 Natural Language Generation

The goal of NLG is to automatically generate consistent and meaningful sentences from some digital representation. The best possible outcome being sentences that cannot be distinguished from ones that were produced by a human. One of the main challenges of NLG systems is making choices [57]. Such a system has not only to decide *what* it puts out but also *how*. This concerns for example the correct use of pronouns as the following example shows:

(1) (a) Niklas talks about Niklas.
(b) Niklas talks about himself.

Binding theory [14] is a method that could be used to come to the right choice here. Another example shows that in other cases choices must be made between two linguistically correct alternatives:

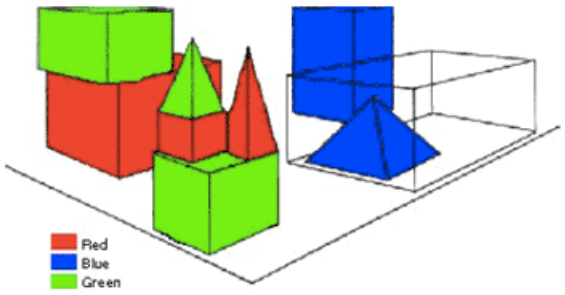
(2) (a) I bought a shampoo. I used it.
(b) I bought a shampoo. I used the shampoo.

Both possibilities are valid, however, a NLG system still has to decide which one to take. This choice can be made for example based on readability factors, which would suggest the use of (2a). If the NLG system is used in a more critical context such as medical manuals more precise language might be required which would make (2b) the more appropriate choice.
Thus, a big part of NLG is concerned with making correct choices. Analyzing specific decisions, aggregating them in a meaningful way and providing methodologies for deducing rules and constructing choice-making systems are some of the challenges of NLG. It is closely connected to linguistic research and often makes use of techniques from the field of artificial intelligence.
An example for an application of NLG is generating textual weather forecasts from digital data. Different meteorological statistics can be used to predict temperature, wind speed/direction, rain probability, et cetera. SumTime [62] is a NLG system that takes this data as input and generates textual weather forecasts. Since the amount of available data is very large, the system first must decide which information will be included in the text and how the document will be structured. This step is called *document planning*. Then, the appropriate words, syntax and sentences must be chosen. This is referred to as *microplanning*. Finally, in the *realization* step, the order an applicable form of the words must be selected.
NLG is applied in many other contexts as well. For instance, using numeric data and event records to create summaries of medical, financial or sports data. Generating document templates for legal, clinical or business docu-

ments based on a knowledge base is another application. NLG can even help improving the lives of disabled people, for example, by aiding blind users in exploring graphs [22].
In our work, we are interested in understanding the already existing text descriptions of open datasets, which is why we focus on NLU, although, NLG could be an interesting approach to explain the data content to users.

2.3.2 Natural Language Understanding

The ultimate goal NLU research is to create a machine that is able to understand fluent text written in a natural language. This is an incredibly difficult task due to the complexity and ambiguity of human language. In 1971, Terry Winograd, a PhD student at MIT, made a first attempt at implementing a NLU engine. The program called SHRDLU[9] could communicate with its user in plain English. The knowledge of the system was limited to a small world with different colored blocks that could be moved by giving commands. Figure 4 shows an example of a conversation between SHRDLU and its user.



Person: Pick up a big red block.
Computer: OK.
Person: Grasp the pyramid.
Computer: I don't understand which pyramid you mean.

Figure 4: A conversation example with SHRDLU

Between 1979 and 1982, Fernando Pereira and David Warren developed a system called CHAT-80 that could answer questions posed in English about a geographical dataset. The grammar of this system was based on logical formulas that translated questions of the user to queries for the database. Listing 2 shows some examples.

[9] `http://hci.stanford.edu/winograd/shrdlu/`

Listing 2: A conversation example with CHAT-80

```
Q: What is the capital of Upper Volta?
A: ouagadougou
Q: Which country's capital is London?
A: united kingdom
Q: What are the capitals of the countries bordering the
    Baltic?
A: denmark:copenhagen; east germany:east berlin;
   finland:helsink; poland:warsaw; soviet union:moscow;
    sweden:stockholm; west germany:bonn
Q: What coutries border Denkmark?
A: I don't understand!
```

Of course, these were still very limited approaches. In recent years, the interest in NLU has increased significantly in both academia and industry. Availability of computing power and advances in artificial intelligence have made many new applications possible: From voice-driven assistants such as Apple's Siri or Google Assistant, over natural-language search and question answering to many others. A remarkable achievement in this field is for example the win of the AI system "Watson" by IBM in the popular gameshow Jeopardy in 2011.

NLU is currently very present in the realms of financial trading. Unstructured data such as news, tweets or analyst reports are processed by computers which consequently conduct the appropriate trades. These transactions are often executed within milliseconds. Leveraging these automated systems has in some cases lead to very abrupt and unpredictable market movements. For example, in 2013 the S&P 500 temporarily lost $136B in market capitalization due to a hacked twitter feed of the Associated Press in the USA.

One issue that must be resolved by any NLU system is how to represent natural language in a way that can be interpreted and manipulated by a computer. A simple solution is the so-called *bag-of-words* approach, where a text is represented as an unordered list of words. Statistical methods can be used on these data structures to perform tasks such as search [9] or text categorization [60]. The drawback of this approach is that it does not capture any semantics which makes it unusable for many applications. Furthermore, even for the task of text categorization it performs poorly as soon as the input texts become short, which is why we refrained from using such techniques in our approach.

Two fundamental challenges of NLU are *synonymy*, two different words that have the same meaning, and *polysemy*, words that can have multiple meanings in different contexts, e.g. the word "church" can refer to a building or

an organization. *Latent Semantic Analysis* (LSA) [18] is a mathematical approach that tries to find the main concepts of a document by manipulating the term-document matrix. This matrix contains all documents of the base corpus of text together with an entry for the frequency of each word in the corresponding document. By applying singular value decomposition to this matrix LSA is able to identify the most important concepts of the data. This elegantly solves the problem of synonymy; however, polysemy remains an issue.

Efforts have been made to create databases that store words or concepts and their relations such as synonymy, hypernymy (type-of relation, e.g. "pigeon", "crow", "eagle" are hyponyms of "bird") or meronymy (part-of, e.g. "finger" is a meronym of "hand"). BabelNet [54] is one of the biggest so-called *knowledge bases* including over 14 million different concepts. Such knowledge bases can be used for different NLU tasks such as word-sense disambiguation or entity-linking. Since this is the approach we used we will discuss it in more detail in a later section of this thesis.

Many NLP challenges can be reduced to a classification task. For instance, in part-of-speech tagging each word must be assigned to a part of speech. Word-sense disambiguation has to identify the correct meaning of a word from a set of meanings. One method to solve such tasks are *decision trees*. Figure 5 shows an example for disambiguation of periods. A period can

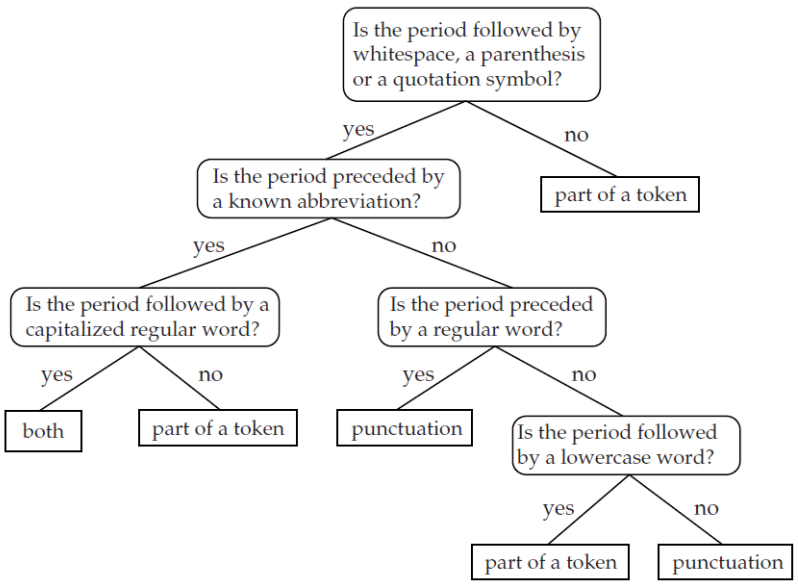
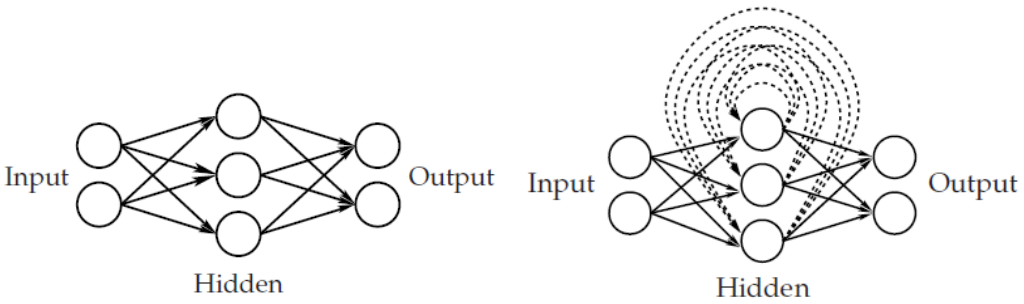


Figure 5: A decision tree for period disambiguation [57]

either be the mark of the end of a sentence ("I drove home."), part of an

abbreviation ("e.g.") , or both ("It was proven by Brecht et al."). The decision tree classifies periods into these three categories. A new classification always starts at the root node. Each node represents a rule that leads to a decision. The different classes lie at the leaves of the tree. Decision trees can be automatically generated from already classified training data. This data includes different features for each object and its corresponding class. Different algorithms to recursively deduce decision trees from such training data exist. The main drawback of this approach is that it needs a rather large set of preclassified data, which is often hard to obtain just as in the case of Open datasets. Furthermore, they often show weaker performance for real data than other approaches like neural networks.

Artificial neural networks (ANNs) are networks that consist of multiple so-called *perceptrons* [57]. A perceptron is an artificial neuron that maps some input to a binary output based on a weight and a threshold. Multiple perceptrons are connected to create an ANN that can learn from training data. Such networks often have multiple layers of perceptrons between the input and the output which allows more complex types of classification. Figure 6a shows an example of such a multi-layered perceptron. The hidden



(a) A multi-layered ANN (b) A recursive multi-layered ANN

Figure 6: The structure of an artificial neural network (ANN) [57]

layers allow a mapping of the input space to a new space of features which consequently can be used by the output layer. ANNs can even have multiple interconnected hidden layers. Training such a network is done by adjusting the weights of the perceptrons until the input leads to the desired output. In *feed-forward* networks the directed graph of the perceptrons does not contain loops, thus, the computation of the output can be achieved in a single pass. *Recurrent* ANNs, on the other hand, can contain such loops. Figure 6b shows an example. These are used when inputs and outputs are sequences with an arbitrary length, e.g. the words in a sentence. Such feedback-loops provide a "memory" to the network by connecting one position in the time sequence

with the subsequent position. ANNs show high performance for many classification tasks in NLU such as word-sense disambiguation or part-of-speech tagging and they are currently in the focus of a lot of research.

2.4 Word-sense Disambiguation

Word-sense disambiguation (WSD) has always been a topic of interest in the research area of NLP. It is concerned with selecting the correct meaning of a word in context. To clarify this, we look at the following two sentences:

(a) The suitcase is light.
(b) Please turn off the light.

The word "light" has two completely different meanings: In (a) it is an adjective meaning "easy to lift" and in (b) it refers to "illumination". The meaning is given by the context. It even has a different part of speech, although the spelling is identical in both cases. This is a difficult problem for a NLP system. It needs to process the unstructured text, convert it to some internal representation and, subsequently, analyze it and thereby find a way to deduce the correct meaning from context. John Mallery identified WSD to be an AI-complete problem [45], that means that it is as least as hard to solve as other fundamental problems of artificial intelligence such as, for instance, the Turing Test [68]. This is due to various factors.
First, it involves fundamental questions like how to represent word senses. Example approaches for such representations are enumeration of finite set of senses or automatic generation of senses based on certain rules. Furthermore, it must be determined how fine grained the distinctions between senses are, i.e. differences can be very subtle. WSD might be simplified if the domain context of the input texts is known, of course, that is not the case for many applications. In addition, selecting the set of words that need disambiguation is a non-trivial task.
Another major aspect that makes WSD challenging is that it requires knowledge. In fact, every WSD system needs some sort of knowledge in order to select fitting senses for a word. In other words, the labels that are assigned to words must be defined somewhere. Such knowledge can have various forms: Text corpora with preliminarily annotated word senses are one possibility. Over the last decades, some more structured machine-readable dictionaries respectively semantic networks have been build. However, creating such resources is a very difficult task in itself and it can never be fully completed due to the ever-evolving nature of the world and human language. This fundamental problem is called *knowledge acquisition bottleneck* [25].

As mentioned before, the rapid growth of the World Wide Web has led to an enormous amount of unstructured texts and data. Thus, means to process this data automatically become increasingly necessary. Traditional techniques, which are often based on lexicosyntactic analysis of text and do not consider WSD at all, often perform poorly when applied to such vast amounts of data [52]. WSD is a crucial step in solving multiple current NLP problems and could contribute to realizing the Semantic Web.
Machine translation is an area where WSD plays an integral role. In fact, it contributes significantly to the quality of results. For example, "penna" in Italian can be translated as "pen", "feather" or "author" depending on the context. Clearly, choosing the correct sense makes a big difference for the outcome of automated translation.

2.4.1 Supervised WSD

Supervised WSD makes use of a preliminarily classified training set, i.e. a list words together with some defining features and the correctly assigned word sense. This is called the *ground truth* from which supervised WSD algorithms try to learn a classifier. Usually, these algorithms classify a single word at a time. Various machine-learning techniques are used to achieve that goal. Decision trees or artificial neural networks are two examples that were already discussed in the previous section. Other methods include Naive Bayes classifiers, Exemplar-Based Learning or Support Vector Machines. Again, the common drawback of these approaches is the need for usually manually created sets of training data.

2.4.2 Unsupervised WSD

Unsupervised WSD methods could be the answer to the knowledge acquisition bottleneck. The main idea of such approaches is that words with similar senses will occur close to one another in a text. They use clustering to group similar words, i.e. senses, in the input text and, subsequently, try to classify new occurrences into these induced clusters. Thus, no machine-readable knowledge base is required for such methods. To be exact, an unsupervised WSD system cannot perform the common WSD task of *sense labeling*, since it has no knowledge about any specific word senses or labels. It accomplishes *word sense discrimination*, i.e. it divides "the occurrences of a word into a number of classes by determining for any two occurrences whether they belong to the same sense or not" [59]. Thus, such approaches usually do not result in clusters that would be found in a dictionary, which makes them hard to compare and evaluate. However, once found, such clusters can still

be reused at a later point of time. The currently most popular methods for unsupervised WSD are:

- *Context Clustering* represents each word occurrence as a *context vector*. This vector contains all senses the corresponding word can have. Clustering is used on such vectors to find groups of similar sense of the target word.
- *Word Clustering* tries find to clusters based on words which have similar meanings. Similarity between words is defined by occurrences of syntactic dependencies (such as, subject-verb, verb-object, et cetera) [52].
- *Cooccurrence Graphs* are graphs that represent words as nodes and syntactic relations as edges. To disambiguate a target word, a local graph is built around it. Normalizing this graphs adjacency matrix leads to a Markov chain which can, subsequently, be clustered using the Markov clustering algorithm [69] to discover the word senses.

2.5 Entity Linking

In our work, we are interested in Entity Linking (EL). Recently, EL has gained increased attention in the research area of NLP. It refers to the task of identifying entities in a text and connecting them to an entry in a knowledge base. Sometimes, it is also called *record linkage, entity resolution* or *entity recognition*. Some of the numerous applications of EL include information retrieval, information extraction or knowledge base population.
The available data on the World Wide Web is growing exponentially and a big part of it is formulated in natural language. This includes many occurrences of named entities in various forms. Furthermore, a significant effort has been made to create big knowledge bases such as Wikipedia as well as machine-readable resources like DBpedia [8], YAGO [65] or BabelNet [54]. The goal of EL is to automatically establish correct connections between this data and such knowledgebases.
Figure 7 shows an example of the task an EL system must solve. The preliminarily identified named entity “Michael Jordan” needs to be linked to the correct entry in the knowledge base. Very often multiple candidate entries exist. In this instance, the Michael Jorden referred to in the text is an American scientist, whereas, in other texts the same name might be a mention of a football player or a mycologist. Of course, the context in which an entity is mentioned must be considered for making such a decision. This disambiguation task is one of the core challenges of EL and is very similar to WSD

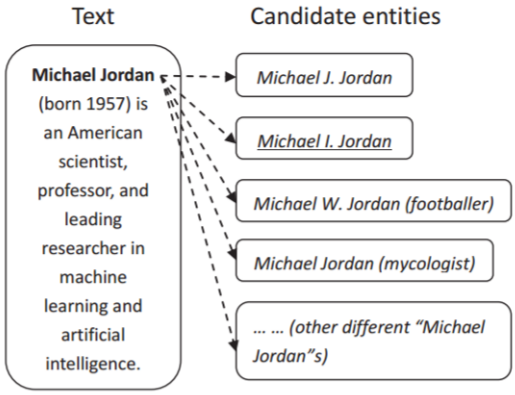


Figure 7: An example for the task of Entity Linking [61]

where the possible senses of a word must be disambiguated. Furthermore, an entity can have multiple name variations, for example abbreviations (e.g. "USA" for United States of America) or nicknames (e.g. "Big Apple" for New York City). Thus, simply matching the complete string will lead to many missed potential links.
Deciding whether a mention of an entity actually has a corresponding entry in the knowledge base or is unlikable, because no such entry exists yet, is another challenging aspect. Coming back to the example in Figure 7; Suppose the input text is about a Michael Jorden who is some politician for whom no entry in the knowledge base has been created yet. An EL must determine that none of the available entries are a match and, therefore, return a null value. This becomes especially difficult when using large knowledge bases with many potential matches.
Most EL systems consist of three major components: *Candidate Entity Generation, Candidate Entity Ranking* and *Unlinkable Mention Prediction*. The following paragraphs will discuss each of them briefly.

2.5.1 Candidate Entity Generation

Candidate Entity Generation is the initial step for any EL approach. For each mention m of an entity a set of entries in the knowledge base must be generated that consists of all possible matches for m. Most approaches are based on some sort of string comparison between the mention and names of entities in the knowledge base. We are now going to discuss three of the most used techniques:

- *Name Dictionary Based Techniques* make use of online resources (in most cases Wikipedia) to create a name dictionary. Specifically, such a

dictionary is a key/value map where the keys are names and the values are sets of possible entities. For example, the key "Micheal Jordan" would map to the value {*Michael Jordan (footballer), Michael Jorden (mycologist)*, ...}. Furthermore, different keys can map to the same value, thus, allowing entities to have multiple names, e.g. the key "USA" as well as the key "America" have the value *United States of America*. Such a dictionary can be constructed automatically from Wikipedia by leveraging its entity pages and inter-article hyperlinks [27].

- *Surface Form Expansion from the Local Document* is a technique that tries to address the problem of acronyms and abbreviations. The goal is to identify other mentions of the same entity within the input document in expanded form (e.g. the full name of a person who was previously mentioned using just initials). This information can then be leveraged to generate more accurate candidate entity sets. Heuristic based methods use heuristic pattern matching to discover acronyms. For example, a popular pattern is an entity followed by its acronym in parenthesis (e.g. "New York City (NYC)"). Supervised learning methods can be used to tackle more complex acronyms where some letters are changed or missing (e.g."Communist Party of China (CCP)") [72].

- *Methods Based on Search Engines* try to take advantage of the capabilities of modern search engines such as Google. The mentioned entity is forwarded to the search engine, sometimes even together with its context [30], and, subsequently, the results are searched for Wikipedia references, which are then added to the candidate entity sets.

2.5.2 Candidate Entity Ranking

After the candidate entity set has been found for a particular mention the candidates must be compared and ranked in order to be able to determine the most appropriate link. Usually, the size of the candidate entity set is significantly larger than 1, which would of course make this step trivial. For instance, it has been shown that the average size of such sets for the TAC-KBP2010 dataset (a benchmark dataset manually created by researchers) is 12.9 [38]. Therefore, this is a crucial step in EL. Methods to achieve such a ranking can be divided into three categories [61]:

- *Independent Ranking Methods* refrain from the idea that different mentions of entities within a text are somehow related. They treat each

mention independently and mainly focus on similarities between the entities and their surrounding text, i.e. the local context.

- *Collective Ranking Methods*, in contrast, assume that entities within a document mostly refer to a set of related topics. These approaches link all entities of an input text in one pass, trying to leverage relations between them.

- *Collaborative Ranking Methods* try to discover relationships between entity mentions and their surrounding context across documents. Candidate entities are ranked based on the similarity of the mention and its context with other mention-entity links that have been found previously in other documents.

2.5.3 Unlinkable Mention Prediction

When candidate entity ranking is complete, the top ranked entity may be chosen to establish a link with the respecting mention. However, in real world applications some entities will most likely not have corresponding entries in the knowledge base. Especially when working with large amounts of data such cases are inevitable. Therefore, such unlinkable mentions need to be predicted. Many studies simply ignore this fact and work with the assumption that the knowledge base contains all possible entities [61]. Others only assign a null value whenever the candidate entity set is empty. Some EL systems include a score in their method for candidate entity ranking which can be used to establish a threshold for a link to be established. If the score of the top ranked entity is below that threshold the null value is returned instead. Other approaches use supervised machine learning to filter out unlikable mentions [73].

2.6 Text categorization

Since we try to categorize Open datasets based on their textual descriptions *text categorization* (TC) is an especially interesting research area for us. The goal of TC is to automatically assign natural language texts to a set of predefined categories. Sometimes the term TC is also used for automatic discovery of such a set of categories based on a given text, however, here we will focus on the former aspect. TC is essential for several higher-level tasks ([41], [33], [70], [44]). Especially in the area of machine learning TC has become a growing topic of interest. Much of the progress in this field can be attributed to the availability of large test collections. These consist of a multitude of documents which have been manually assigned to distinct

categories by human indexers. Such collections allow researchers to quickly investigate novel approaches without the costly task of acquiring new training data. Furthermore, they allow for quality assessment and meaningful comparison of different methods. Examples for such collections are OHSUMED [32], a medical information database, TIPSTER [31], a collection of news stories, and the Reuters Corpus Volume I [58], another large archive of over 800,000 categorized news stories.
The task of TC can have multiple properties based on the context of the its application [60]:

- *Single-label* vs *multi-label*: Single-label categorization assigns exactly one category to each text. *Binary* TC is a special case of single-label TC where a text either belongs to a category or to its complement. In multi-label TC, a text may belong to multiple categories. This means that some categories overlap. Binary TC is the most general form; a binary classifier may be used for multi-label classification by applying it to every category and determining whether an input text belongs to it or not. In contrast, multi-label algorithms cannot always be applied to single-label problems, since the resulting categories do not necessarily have to be ordered. Therefore, it cannot be directly determined which of the categories is "most fitting".

- *Category-pivoted* vs *document-pivoted*: Category-pivoted TC (CPV) iterates over all available categories and tries to find all matches in a given set of texts. Document-pivoted TC (DPC) on the other hand, tries to find the appropriate category for every text given as an input, thus, iterating over the texts. This is a rather subtle distinction; however, it can be important when for example the set of categories or the set of input texts are not accessible in their entirety. DPC is usually employed when the input texts become available at distinct points in time, for example when filtering for spam e-mails. CPC is the appropriate choice when the set of categories is subject to change over time, for example when new categories are added after some texts have already been categorized.

- *Hard* vs *ranking*: Depending on the application a TC algorithm must produce a definitive true or false answer to the question whether a text belongs to a category or not. This is called *hard categorization*. Alternatively, an algorithm could just give a *ranking* of categories for each input text. This approach is popular for semi-automated TC where the final decision is taken by a domain expert. Such approaches

are useful when the expected quality of a fully automated system is low due to limited training data.

In the 1980s, TC was addressed mostly using so called *knowledge engineering* (KE) techniques. KE systems consist of a set of logical rules that were manually created by domain experts. One rule of the following form is created for each category:

if ⟨*DNF formula*⟩ **then** ⟨*category*⟩

In such a system, a text belongs to a category if and only if the respective disjunctive normal form (DNF) formula evaluates to true. Figure 8 shows an example for such a rule. Such systems can be very effective for

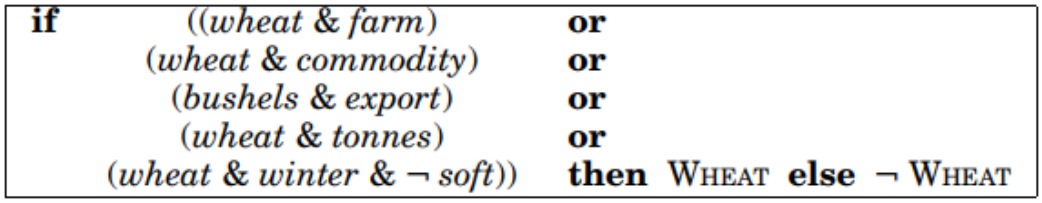


Figure 8: Rule-based classifier for the WHEAT category; key words are indicated in *italic* [6].

specific domains; however, it should be clear that constructing them is very labor-intensive and therefore costly. Moreover, this approach suffers from the *knowledge acquisition bottleneck*, i.e. the whole systems needs to be adapted manually whenever a category is added or for usage in a different domain.
Nowadays, almost all approaches to TC are based on *machine learning* (ML). In general, these techniques include a so-called *learner* that automatically constructs a classifier based on a set of preliminarily categorized training data. Since the training data is usually manually created this is called *supervised* learning. The big advantage of this approach is that such a learner can be applied to any domain if sufficient training data is available. Furthermore, the categories themselves are not hard coded into the learner; therefore, it is not necessary to update it whenever they change; just automatic retraining with an updated set of training data is necessary. In the following sections, we are going to review some of the most common ML approaches to TC.

2.6.1 Naïve Bayes Classifier

Naïve Bayes classifiers belong to the probabilistic classifiers and are based on Bayes' theorem from probability theory. Objects are assigned to categories

based on a probability that is computed from their feature vectors. The naïve assumption is that all features are independent from one another, which is a necessity for the use of the Bayes' theorem. This premise is mostly false in practice, nonetheless, such classifiers still achieve reliable results as long as the attributes are not strongly correlated.
Since filtering spam e-mails is a popular application of naïve Bayes classifiers we will use this context as an illustrative example [1]: An e-mail can either be spam or not, so, our set of possible categories consists of two entries $C = \{spam, \overline{spam}\}$. The words $\{w_1, \ldots, w_n\}$ of an e-mail can be seen as its feature vector. From the preliminarily classified training data we can estimate the probability that a word w_i appears in a spam e-mail or a non-spam e-mail:

$$P(w_i|spam) = \frac{\text{number of spam e-mails including } w_i}{\text{total number of spam e-mails}}$$
$$P(w_i|\overline{spam}) = 1 - P(w_i|spam)$$

To classify a new e-mail E we must determine whether $P(spam|E) < P(\overline{spam}|E)$. If this evaluates to true then E is not a spam e-mail. These probabilities can be computed based on Bayes' theorem:

$$P(spam|E) = \frac{P(spam \cap E)}{P(E)} = \frac{P(E|spam) \cdot P(spam)}{P(E)}$$

$P(E)$ is the probability that E occurs which is independent of $P(spam|E)$ and $P(\overline{spam}|E)$ and is therefore negligible. Consequently, we just have to evaluate the following expression:

$$Q = \frac{P(spam|E)}{P(\overline{spam}|E)} = \frac{P(E|spam) \cdot P(spam)}{P(E|\overline{spam}) \cdot P(\overline{spam})}$$

Is $Q > 1$ then E is classified as spam, otherwise as non-spam. The probability $P(spam)$ can be again estimated from the training data:

$$P(spam) = \frac{\text{number of spam e-mails}}{\text{total number of e-mails}}$$
$$P(\overline{spam}) = 1 - P(spam)$$

In practice, it is more common to use a higher threshold like $Q > 10$ to make sure that only e-mails that are very likely to be spam are filtered out and put into the spam folder. These classifiers learn by adjusting the probabilities $P(w_i|spam), P(spam)$ whenever new training data is available. Naïve Bayes classifiers are popular because their evaluation only requires linear time and they perform reasonably well even for small sets of training data. A drawback is that they cannot be interpreted very easily by a human. Due to the necessity of preclassified training data this approach is not feasible for our particular situation and goals.

2.6.2 Decision Tree Classifiers

An approach that is more easily interpretable by a human are *decision tree* (DT) classifiers. In this context, a DT is a tree where the edges are labeled with words that may or may not appear in a text. The leaves of the tree represent the categories a text can belong to. A DT classifier starts at the root and tests which words appear in the input text. As soon as a leaf is reached the corresponding category is assigned to the text. Figure 9 shows an example for such a tree.

Such a DT for a category c can be learned from training data T in the

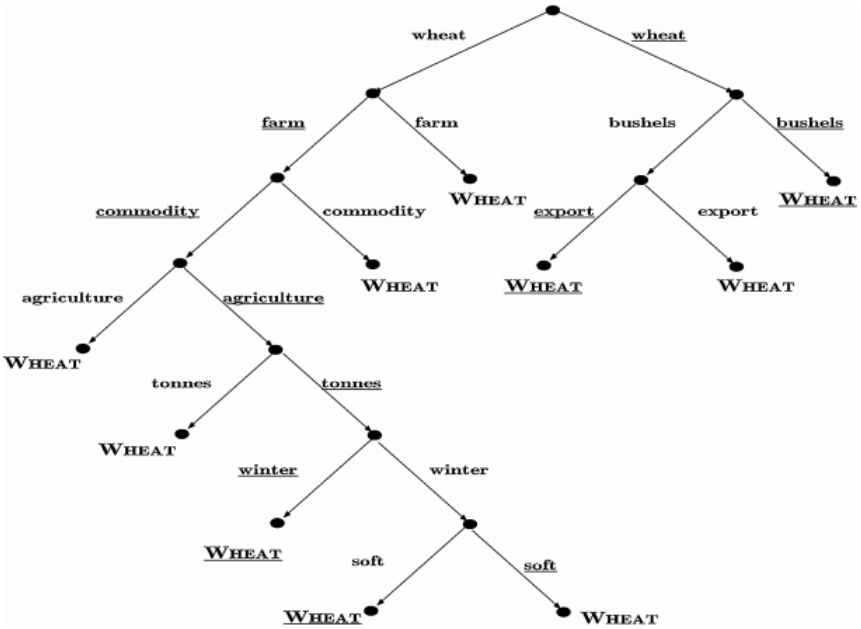


Figure 9: A decision tree determining membership to the WHEAT category (underline represents negation) [6].

following way:

1. If all elements in T belong to the same category (c or $\overline{c}$) assign this category to the leaf.
2. Otherwise, select a term t and partition T into sets that have the same value for t. Each of those sets is put into its own subtree.

This procedure is repeated recursively until each leave of the tree contains a category. Here, the crucial part is the selection of the term t, since it

determines the partitioning. This is often based on a measure for information gain or misclassification, e.g. Gini impurity. DT generated in such a way are relatively prone to overfitting to the training data, which is why many DT learning approaches include methods for trimming the tree, i.e. removing some branches that are too restrictive.
A big advantage of decision trees is that it is easy for humans to work with them. Very little explanation is necessary to interpret a visualization of a DT. A drawback is that they are rather sensitive to changes in the training data. A small adjustment in the training data may lead to a substantial change in the tree [13].

2.6.3 Support Vector Machines

Support Vector Machines (SVM) were introduced to TC by Thorsten Joachims in 1998 [39]. A SVM classifies a set of objects in such a way that the empty margins between the classes are as wide as possible. This idea can be illustrated well for linearly separable training data. Figure 10 shows such an example. Each training example is represented as a vector in the feature

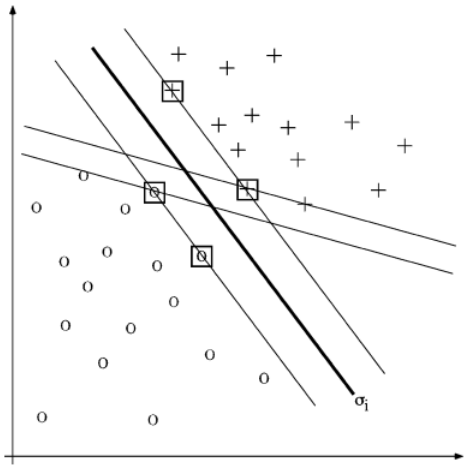


Figure 10: Classification with a Support Vector Machine [60].

space (crosses and circles represent the two classes). Various lines (*decision surfaces*) could be chosen to separate the two classes. In a multidimensional feature space the decision surfaces are hyperplanes. The SVM selects the hyperplane (here the line σ_i) with the largest distance to the nearest training examples. The small subset of training examples that determine the decision surface are called *support vectors* (indicated in the figure by little boxes).
This method is also applicable to datasets that are not linearly separable by

the use of the so-called *kernel trick*. This method entails transforming the data to a higher dimensional space where it becomes linearly separable.
SVM are usually not prone to overfitting and are able to scale to a large number of dimensions. Furthermore, no parameter tuning is necessary to reach optimal results [39]. A drawback of SVMs is their high resource consumption in large-scale applications [66]

2.7 Clustering

One of the goals of our work is to find similarities between Open Data portals and discover a possible contextual grouping of them. To that end we employ clustering techniques on the results of the dataset categorization process. The task of clustering is to assign objects to groups based on their features. These groups or *clusters* should contain elements that are similar, so, members of the same cluster should be more similar than members of distinct clusters. Different clustering algorithms exist that find such clusters automatically. Figure 11 shows an example; The input dataset (Figure 11a)

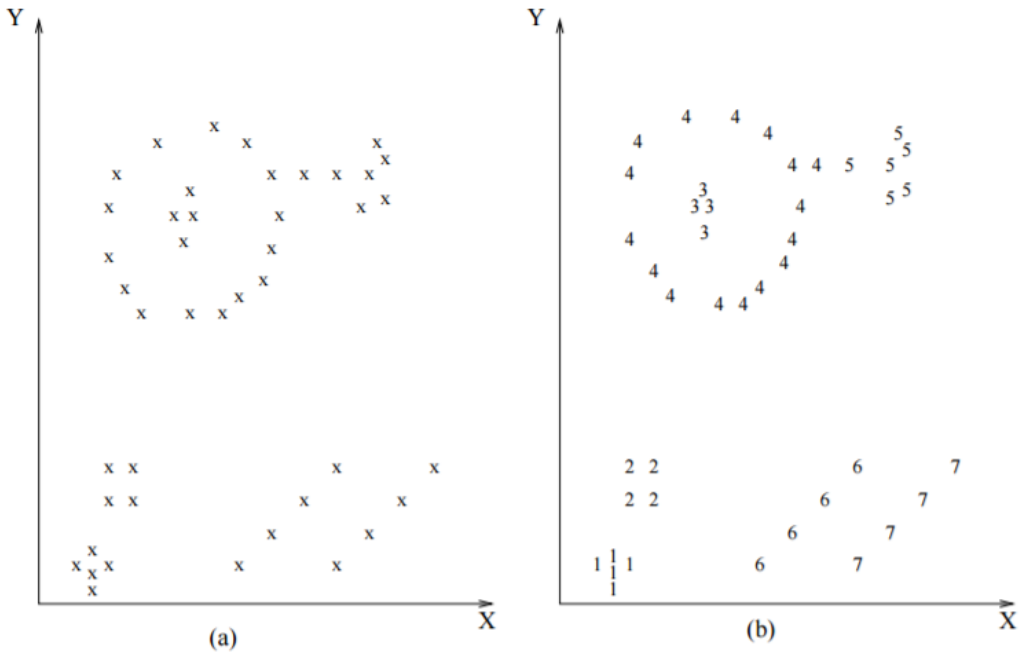


Figure 11: An example for clustering [36]

consists of multiple objects (points) which are defined by two features X and Y. In this example (Figure 11b) clustering resulted in seven clusters denoted by numbers from one to seven. Clustering algorithms are *unsupervised* which

means that they try to identify groups in unlabeled datasets, so, they do not need any training data. On the other hand, in supervised classification, also called *discriminant analysis*, clusters are given as an input and the task is to assign a new object to one of them.
Clustering is an important tool for data analysis and data mining, especially when working with large datasets. In fact, it is used in a wide variety of areas, ranging from pattern-recognition in image processing over market segmentation and analysis to machine-learning applications like document retrieval or decision-making. Clustering is especially useful when not much information about the data under study is available. It is a great tool for exploring a dataset and revealing its structure respectively the relations of its objects.
Since clusters should contain similar objects, it is essential to specify a calculatable measure for similarity. Usually, this is done by defining the *distance* between two objects in the feature space. The most common method for computing the distance in a numeric feature space is the *Euclidean distance*. For two objects $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ it is defined as follows:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

If the distance between two objects is small they are considered similar. The Euclidean distance is best suited for datasets with "compact" or "isolated" clusters [46]. Examples for other distance measures are the *Mahalanobis distance* or the *Hausdorff distance*.
A wide variety of clustering techniques exist and they differ in various aspects [36]:

- *Agglomerative* vs *divisive*: In an agglomerative approach, each object is initially assigned to its own cluster. Subsequently, the clusters with the smallest distance between them are joined together. This process is repeated until it arrives at some stopping criterion. In contrast, a divisive algorithm starts with all objects in a single cluster which is then divided until reaching a stopping criterion.

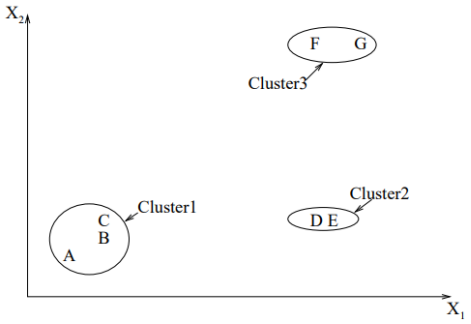
- *Polythetic* vs *monothetic*: Polythetic algorithms consider all features of an object to compute the distance. Thereafter, classification is conducted based on this distance. A monothetic approach first divides objects into clusters based on a single feature. The so-found clusters are then further refined based on the second feature. This process is repeated until all features are incorporated into the solution. Most algorithms are polythetic, since monothetic approaches lead to clusters

that are often too small to be meaningful respectively interpretable when applied to feature-rich datasets.

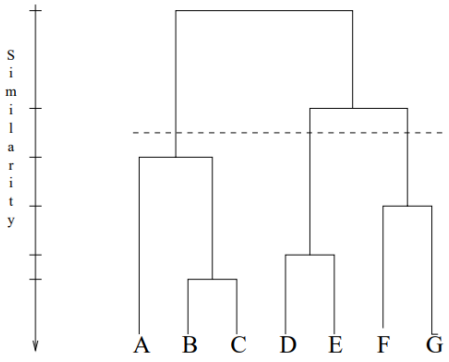
- *Hard* vs *fuzzy*: In hard clustering, each object is assigned to exactly one cluster. Fuzzy techniques assign a score for every cluster to each object. This score indicates the degree with which the object belongs to the respective cluster. It can also be interpreted as a probability of belonging to a cluster.

2.7.1 Hierarchical Clustering

Hierarchical Clustering can be agglomerative or divisive. In both cases the result is a cluster hierarchy. Figure 12a shows an example of seven data objects in three clusters. Results of hierarchical clustering can be visualized



(a) An example dataset with three clusters.



(b) Hierarchical clustering of the example dataset. The dotted line represents the cut that results in the three clusters.

Figure 12: An example for hierarchical clustering [36].

as a *dendrogram*. It shows the partitioning process for the dataset into smaller and smaller subsets. The leaves of a dendrogram represent the data objects. At each node, the two child subsets are combined to a new and more general cluster. The height of the node is determined by the distance between the combined clusters. "Cutting" the dendrogram at distinct levels of similarity leads to a different number of clusters. Figure 12b shows a dendrogram for the example dataset and a cut resulting in three clusters.
Hierarchical clustering can be used to find the appropriate number of clusters for a dataset which is the method we use in our approach. The size of the steps from top to bottom in the dendrogram can be seen as the percentage of

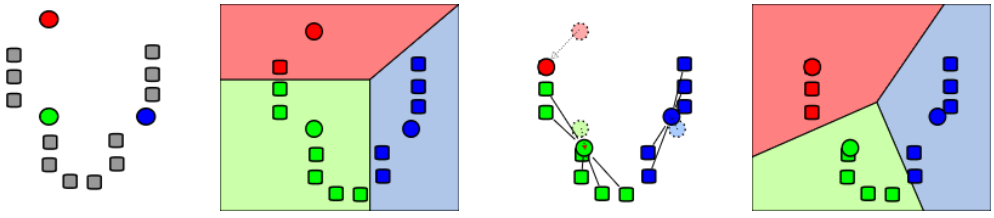
variance explained by the respective number of clusters. Therefore, as soon as adding another cluster does not give a significantly better model for the data the appropriate number of clusters is reached. This is known as the *Elbow method*, since we look for an "elbow" in the plot of variance explained vs number of clusters.

2.7.2 K-means Clustering

K-means Clustering is one of the most commonly used clustering algorithms. It partitions a dataset into k clusters based on the distance between objects and the mean of each cluster. The parameter k is an input parameter that needs to be set manually. The algorithm tries to minimize the sum of squares (variance) within the clusters. Formally, this means to find the minimum of the following function:

$$Y = \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - c_i||^2$$

with the data objects x_j and the cluster centers c_i. Since $||x_j - c_i||^2$ is the squared Euclidean distance the k-means algorithm effectively assigns each object to its nearest cluster center. Finding the optimal solution for this problem is NP-hard [26], therefore, heuristic algorithms such as Lloyd's algorithm [43] are most commonly used ("Lloyd's algorithm" and "k-means algorithm" is mostly used synonymously). Figure 13 shows an example of Lloyd's algorithm.



(a) (b) (c) (d)

Figure 13: Lloyd's algorithm [56].

Initially (13a), the cluster centers are positioned randomly (colored dots). Here, we have three centers ($k = 3$). Then, clusters are created by assigning each data object to its nearest center (13b). In the next step, the centers are moved to the mean of the clusters (13c). Steps (13b) and (13c) are repeated until the position of the centers remains stable.

Since the k-means algorithm is sensitive to the initial position of the centers, different methods are used for initialization. *Forgy* and *Random Partition* are most common [28]. The Forgy method randomly selects k objects of the dataset and uses them as initial centers. In Random Partitioning, all objects are assigned to a random cluster and the means of each of these clusters is used as initial centers. The Forgy method usually distributes the centers more evenly, whereas Random Partitioning favors the center of the dataset. K-means is a heuristic algorithm, which is why, it can happen that a local minimum instead of a global minimum is reached, depending on the position of the initial centers. Figure 14 shows an example where the algorithm converges to a local minimum, i.e. a "wrong" solution. A common solution

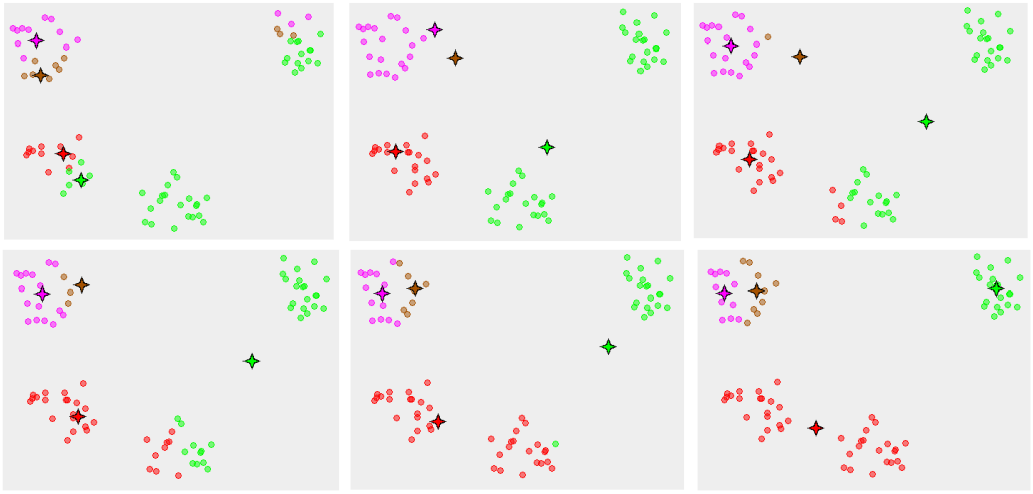


Figure 14: Example of the k-means algorithm converging to a local minimum. Figure created with [49]

for this problem is to repeat the whole algorithm multiple times with different initializations and compare the achieved sum of squares. This is in many cases possible, since the algorithm is usually very fast. Another disadvantage of the algorithm is that the number of clusters must be set preliminarily. Of course, this is problematic for an unknown dataset, making other techniques for determining the appropriate number of clusters necessary. As for all other clustering techniques, the quality of the results of the k-means algorithm depends on the dataset. It performs well on datasets with spherical clusters of the same size. It produces worse results when the density within the dataset varies or the shapes of the real clusters are irregular.

3 Related Work

In this section, we are going to give an overview of the related work of our concrete goal of categorizing short natural language texts, namely descriptions of open datasets. Since they are the basis of this study we are first going to give an introduction to the inner workings of BabelNet and Babelfy. Then, we are going to present another very interesting approach to WSD and topic labelling using centrality measures of sense graphs extracted from DBpedia. At this time, there is no work that we are aware of that addresses topic detection or other forms of NLU specifically on Open Data. Therefore, in the last part, we are going to briefly discuss other approaches to NLU in different contexts.

3.1 BabelNet

BabelNet was created in 2011 by Roberto Navigli and Simone Ponzetto at the Sapienza University of Rome [54]. It is a very large multilingual lexicalized semantic network and ontology that was automatically generated initially by linking the web encyclopaedia Wikipedia with the electronic lexical database WordNet. As of today, multiple other sources of knowledge, such as Open Multilingual WordNet[10] or GeoNames[11] have been integrated into the project[12]. This is done automatically by estimating mapping probabilities between the encyclopaediae and entries in the machine-readable lexica. Multiple methods including bag-of-words approaches and graph-based techniques are employed for this estimation. Machine translation is used when encyclopaedia entries are not available in a specific language. The accuracy of the WordNet-Wikipedia mapping in the current version (BabelNet 3.6) has been evaluated to over 90% on open-text words [53]. Since their methodology is efficient and fully automated the researchers were able to build a huge network with currently over 14 million entries, connected by multiple semantic relations and covering 271 languages.
The semantic network of BabelNet is represented as a labelled directed graph. The vertices of this graph represent concepts such as *city* and named entities such as *Vienna*. They are called *synsets* in BabelNet. Translations for multiple languages are stored for each synset. The edges of the graph are labelled with one semantic relation that was found between the corresponding vertices (synsets). WordNets lexical and semantic relations are directly assumed (hypernymy, hyponymy, meronymy, ...), whereas for Wikipedia hyperlinks

[10] `http://compling.hss.ntu.edu.sg/omw/`
[11] `http://www.geonames.org/`
[12] For a complete list see `http://babelnet.org/about`

are the basis for relations. Multiple types of relations have been derived, including is-a, part-of, similar-to, et cetera. Unspecified semantic relations occur as well.

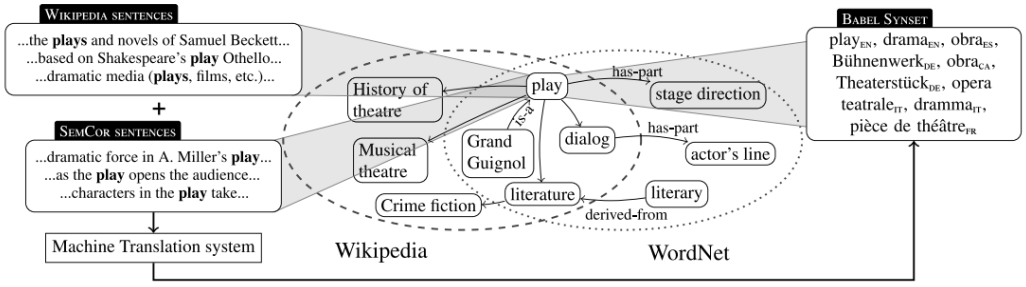


Figure 15: An overview of the construction of BabelNet [54]

Figure 15 gives an overview of how BabelNet is constructed. First, concepts are extracted from WordNet senses and Wikipedia pages. Since these resources overlap and duplicate synsets must be avoided merging becomes necessary. This is done automatically via a mapping algorithm. Translation of the synsets is achieved by, firstly, using the inter-language links of Wikipedia. Secondly, machine translation is employed to fill the remaining gaps. SemCor [48], a sense-tagged annotated corpus for WordNet that was manually created, and the Google Translation API[13] served as resources for this purpose.
The enormous size as well as the support for multiple languages makes BabelNet the ideal knowledge base for applying NLP to Open Datasets. Furthermore, the fact that it includes over 7.7 million named entities contributes to this choice, since Open Datasets are often related to specific regions, cities, institutions, landmarks et cetera. This is an assumption that will be investigated further in this study. That said, BabelNet is still an automatically generated network, which means the contained relations and mappings are not entirely accurate. This entails a natural boundary of correctness for the approach that will be presented in this thesis.

3.2 Babelfy

In 2014 Roberto Navigli et al. presented Babelfy, a system that addresses both Entity Linking (EL) and Word Sense Disambiguation (WSD) [50]. The goal of their work is to automatically understand the meaning of text. BabelNet serves as a knowledgebase. Figure 16 shows an example result of

[13] `https://cloud.google.com/translate/`

the Babelfy process. It annotates words or phrases in the input text with disambiguated BabelNet concepts. It can detect concepts as well as named entities such as ⟨*Nintendo*⟩. Sometimes, both a named entity and a standard concept are assigned to one phrase, e.g. ⟨*Mario Kart 8*⟩ and ⟨*Kart*⟩ in this example.

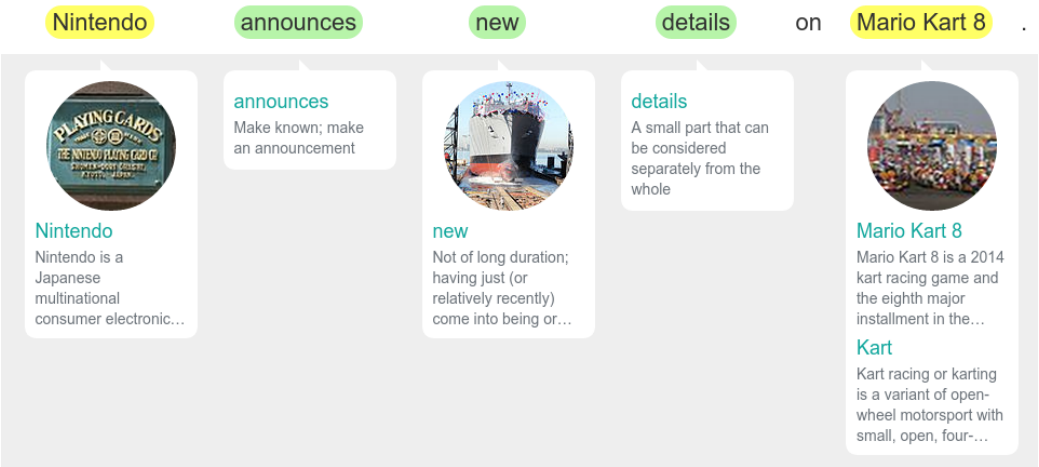


Figure 16: An example result of the Babelfy process.

Babelfy is a graph-based approach that uses random walks and a densest subgraph heuristic to address WSD and EL simultaneously. In a first step a semantic signature, that is, a set of related synsets, is created for every synset in BabelNet. This is necessary because of data quality. BabelNet synsets have an average of 50 incident edges, therefore, often edges between semantically unrelated synsets exist. This is addressed by weighting the different relations and, consequently, using random walks with restart to establish a measure of relatedness between synsets. This procedure is just a preliminary step and, therefore, not executed for every input text.
When an input text is provided, Babelfy extracts all text-fragments and seeks out their possible senses in BabelNet. Consequently, a graph is constructed by relating these candidate meanings using the semantic signatures mentioned above. This graph represents all possible meanings of the input text. In a final step, a dense subgraph heuristic is used to derive the best candidate meanings for each fragment.

3.3 Topic labelling using DBpedia

In 2013 Ioana Hulpus et al. presented a novel approach for automatically finding appropriate topic labels for natural language input documents using

graph-based techniques and DBpedia as a knowledge base [35]. Their work is part of a larger framework for automated topic analysis called *Canopy*. Figure 17 shows an overview.

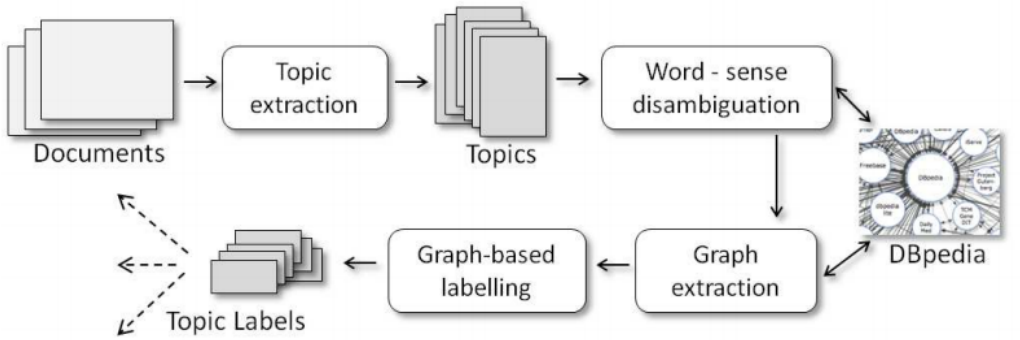


Figure 17: The Canopy framework [35].

Topic extraction is achieved by applying Latent Dirichlet Allocation (LDA) to the input documents [12]. LDA is a probabilistic model for text or image corpora. In this model each document of the corpus is seen as a combination of a small number of different *latent topics*. Furthermore, each word in the document is probabilistically associated with one or more of these topics. The number of topics per document is preliminarily defined and they explain similarities between documents.
For WSD Canopy uses an eigenvalue-based approach that disambiguates all words of a topic at the same time [34]. In that way relations between the senses are leveraged to improve results. DBpedia is used as a knowledge base to find candidate meanings and related words. Table 1 shows an example to explain the process. In a first step, a bipartite graph is derived

Table 1: Example WSD task for three target words [34].

Target	**Candidate**	**Related words**
web	*web#1*	computer, network, application
	web#2	feather, net, flat, part
	web#3	world, English, bible, work, public
internet	*internet#1*	computer, connected, http, net
page	*page#1*	computer, media, public
	page#2	paper, flat, side
	page#3	boy, work, knight, part

from the relations between the related words and the possible senses. One

graph represents one combination of meanings. Figure 18 shows all graphs for our example. The weights of the edges are determined by the number of other senses a sense shares the related word with. For instance, in the top left graph *web#1* shares the word "computer" with both *internet#1* and *page#1*, whereas in the top right *web#1* shares it only with *internet#1*. WSD is achieved by calculating the dominant eigenvalues for each adjacency matrix of these graphs. The eigenvalue is used as a score for the sense combinations of all words. The graph with the highest eigenvalue contains the senses with optimized relations between them. The apparent advantage of this approach is that, in contrast to many other algorithms which score each sense separately, it takes relations between senses into account. This is especially useful for disambiguating small set of words, such as the topics found by LDA.

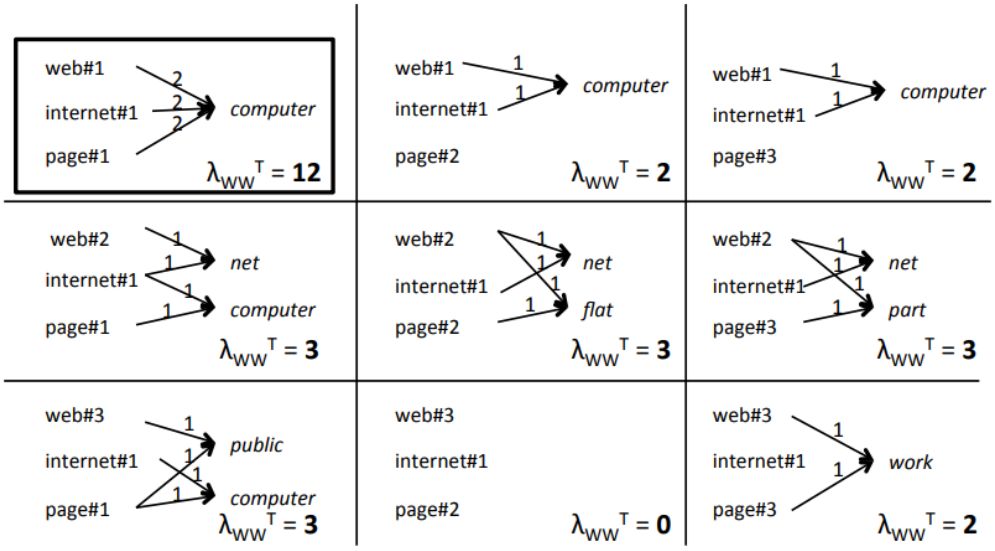


Figure 18: Bipartite graphs for every combination of senses [34].

The result of WSD is a set of concepts C for each topic. In the next step, a *sense graph* G_i for each $C_i \in C$ is extracted from DBpedia. This graph is created by following certain edges, e.g. `skos:broader`, `rdfs:subClassOf`, etc, from the seed concept C_i and adding all nodes to G_i which are at most two hops away. Subsequently, all sense graphs are merged into one *topic graph* G. Figure 19 shows an example for four concepts.

Centrality measures are applied to G to find an appropriate topic label. The researchers experimented with several types of centrality measures. The best results were achieved using the *focused betweenness centrality*, a slight

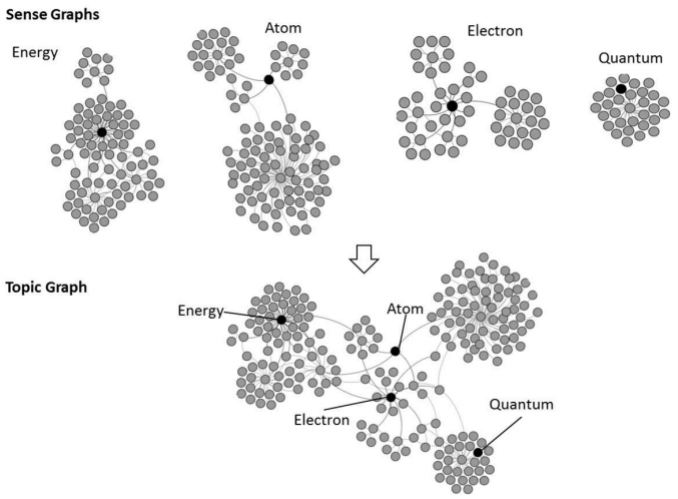


Figure 19: The merge of four sense graphs into a topic graph [35].

variation of the betweenness centrality which assigns high scores to nodes that are part of many distinct shortest paths.
An Evaluation based on human judgements of this method showed better results than multiple standard text-based approaches. The advantage is a very good exploitation of the knowledge base which leads to the ability to identify broader labels. This is a very interesting approach that could potentially be translated to the concepts of BabelNet in order to find suitable labels for open datasets which is a promising possibiliy for future work.

3.4 Other approaches to NLU

Of course, there are various other approaches to NLU. For example, TagMe [21] also tackles WSD and EL at the same time. It is able to annotate short texts with corresponding Wikipedia pages. However, TagMe is limited to English, German and Italian and is linked solely to Wikipedia as a knowledge base. MetaMap [7] is another example, it was specifically developed to identify concepts in a biomedical text and map them to the UMLS Metathesaurus[14]. Of course, this is a very specific use of NLP techniques that would not work in the context of this study.
R. Collobert et al. [16] take a completely different approach: The researchers propose the use of a neural network to address the various NLP challenges, including WSD, EL or part-of-speech tagging. All of these can be seen as the task of assigning labels to words. The developed system is able to learn on

[14] `http://umls.nlm.nih.gov/`

the basis of largely unlabelled test data, and, therefore, does not need a big man-made knowledge base. This a very interesting and cunning approach that has achieved remarkable results. It should be investigated further in future work.

4 Categorizer: Overview and Methods

In this section, we are going to give an overview of our solution and the methods that were used to develop it. First, we will give a brief introduction to the initial situation and a rough overview of the problems that needed to be solved. Second, an overview of the accomplished solution, including a formal description, is going to be discussed. Finally, we are going to present the methods and tools that have been employed to attain the solution.

4.1 Onset

The starting point for this thesis was the Open Data Portal Watch project of the Vienna University of Economics and Business. The question was how the harvested metadata could be meaningfully exploited in order to gain insights into the structure and content of currently available Open Data. Since the most interesting information about each dataset, i.e. its context and content, is only available in the form of natural language text it was clear that some sort of natural language understanding (NLU) technique was going to be necessary to automatically access and process it. The limited time frame of a master thesis plus the fact that multiple very effective and publicly available methods for NLU already existed lead us to the decision to simply apply one of them instead of developing something from scratch. Our requirements for such a method were that (i) it performs well on short texts, (ii) it supports multiple languages, (iii) it provides references to many named entities, (iv) it is publicly available and free. Since a lot of open datasets have rather short descriptions (median of 144 characters for the 150 portals in this study) (i) was necessary to achieve meaningful results. Portal Watch covers Open Data portals from all over the world, which is why the textual data comes in a multitude of languages. We could have restricted our analysis to just a few portals using a single language, however, when working with Open Data the multilingual aspect is immanent. Therefore, we looked for an approach that supports (ii). Another aspect of Open Data is that it is often related to named entities such as regions, cities, institutions or organizations, which is why we included (iii). Of course, the financial means of a student are limited which made (iv) a practical necessity. The Babelfy framework paired

with the BabelNet knowledgebase fulfil all the requirements above with some slight limitations regarding accessibility. We are going to discuss how we applied these methods later in section 4.3.
One can think of various possible use cases for machine readable open datasets. Apart from the possibilities for the Portal Watch project to dig deeper into their metadata collection it could be used to develop a concept based search engine, compare portals based on content and discover categories for datasets and even portals. In the course of getting familiar with the chosen frameworks we decided to focus on the last two points. They seemed feasible within the given time frame and available resources.

4.2 Solution overview

We are now going to present a general overview of our approach given the input data described above.

Management of input text There are three fields of the metadata of a dataset that contain natural language text:

1. *Title*: The title usually contains a brief description of the dataset (with a median of 40 characters for the datasets in this study). It usually that text that is displayed in the list of search results for most Open Data portals.

2. *Description*: The description is usually more detailed and often, in contrast to the title, contains full natural language sentences. For the datasets in our study the median of the number of characters in the description is 144. Often, descriptions are only displayed in full when navigating to the webpage specific to the dataset.

3. *Keywords*: Keywords are mostly single words or tags that should describe the content of a dataset. Most Open Data portals use keywords directly as facets to limit search results. Datasets in this study contain on average 4.5 keywords.

All three fields contain information about the same dataset, hence, in a first step, the three fields are concatenated into a single string. That way, a word sense disambiguation algorithm applied to this string can leverage the context of the description to disambiguate the keywords as well. Keywords are simply concatenated and separated by whitespaces. Since keywords are in a way more important than other words in the description and title – they are supposed to concisely describe a dataset – we want to distinguish

between concepts that were discovered in the keywords and concepts that belong to the rest of the string. Therefore, we insert a delimiter, in the form of a special character, between the keywords and the rest. Thus, the final string s_d that is sent to Babelfy for each dataset d looks as follows: "*keywords*~~*title description*".

Language detection In order to be able to employ the disambiguation service of Babelfy we first have to discover the language of each dataset. This is necessary since Babelfy requires the language of the input text as a parameter. Many Open Data portals provide all datasets in a single language, however, there are some portals that offer datasets in different languages. The portal of the Netherlandish government for example[15] includes datasets in Dutch as well as English. Therefore, the second step is to automatically discover the language of each dataset. We achieve this by forwarding the string s_d to an external open source library that provides language detection based on naïve Bayesian filters [51].

Multi-concept detection Formally, Babelfy provides us with a mapping between d and a set of concepts $C = \{c_1, c_2, \ldots, c_n\}$ with all concepts $c_i \in B$ the semantic network of BabelNet. However, strictly speaking C is not a regular subset of B. While Babelnet is a set of distinct concepts (synsets) a synset can occur multiple times within the natural language text of a dataset. For example, if the state of New York is mentioned in the title as well as the description the concept $\langle New\ York \rangle$ will appear twice in C. Using the delimiter mentioned above we can divide C into two sets C_k for concepts discovered in the keywords and C_t for concepts in the description and title. We will use this distinction later for categorization.

Relevance score Babelfy provides a set of scores for each discovered concept in C that reflect how important it is in the given input text. These scores are delivered together with each discovered concept. After receiving them from Babelfy results are saved to a database. The essential information that is stored about each dataset includes the title, description and keywords as well as the two sets C_k and C_t of discovered concepts together with their respective scores. This data represents the basis for our categorization algorithm which will be described in detail in section 5.

Category assignation To give a rough overview of our approach: BabelNet provides a mapping from synset to one of 34 fixed categories. This

[15] `https://data.overheid.nl/`

mapping exists for about 2.7 mio of the 14 mio synsets in the network, which is still enough to arrive at leastwise a few concepts with associated categories for most datasets. The idea is to leverage the relevance scores of the concepts, the frequency of occurrence of a category as well as the distinction between concepts in the keywords and concepts in the rest of the input string in order to compute a confidence with which a dataset belongs to each of the 34 categories. Based on this confidence the categories can be ranked and the one with the highest scores is selected as the categories of the dataset. Some categories overlap semantically, e.g. ANIMALS and BIOLOGY. Furthermore, it can be argued that a dataset belongs to multiple categories, e.g. a dataset about state production of crops could be assigned to FARMING as well as POLITICS AND GOVERNMENT. Therefore, it makes sense to establish a threshold and assign all categories that receive a score higher than the threshold to each dataset. This threshold determines how selective the various categories are in respect to the complete set of datasets.
Due to the limitations of Babelfy, i.e. the limit of 15.000 API calls per day, we were not able to run all datasets provided by the Portalwatch project through the entity recognition process. However, we reached a high enough number of dataset and portals to conduct a meaningful analysis. In particular, the data for this study contains over 174.000 datasets in 24 different languages belonging to 150 Open Data portals.

4.3 Methods and Tools

We will now have a look at the methods and tools that were used to create our solution called *Categorizer*. BabelNet and Babelfy are the two main tools our solution relies on for any NLP task. All input data is stems from the Portalwatch project.

4.3.1 Portalwatch

The Open Data Portal Watch project provides quality assessment and monitoring of 260 Open Data Portals. Snapshots of the metadata of all portals are taken regularly. This information is made publicly available through a REST API[16]. It provides operations to retrieve information about specific datasets in different formats, e.g. DCAT or Data Quality Vocabulary (DQV)[17] which is a vocabulary to express data quality. Furthermore, the API provides access to information specific to a portal respectively all the snapshots of it that have been made. This includes a list of datasets, some portal specific

[16] `http://data.wu.ac.at/portalwatch/api`
[17] `https://www.w3.org/TR/2015/WD-vocab-dqv-20150625/`

information as well as aggregated dataset quality measures for the portal. Of course, operations to access the set of all portals in the system are available as well. Finally, some functionality to search over resources and metadata is provided, also via a REST call. Results for any API call are packaged into the common data exchange format JSON.
We wrote a short Python script to access the metadata per portal. We wanted to cover as many portals as possible hoping to gain interesting insights during analysis, e.g. finding common themes of Open Data portals or clusters in category distributions at the portal level. For this reason, together with the fact that Babelfy API calls were limited, we decided to first sort the list of portals ascendingly with respect to their number of datasets. Subsequently, the datasets were downloaded per portal. This process takes up quite a bit of time since a separate API call is necessary for every single dataset, however, since access is not limited it was still possible to download enough data in a couple of days.

4.3.2 BabelNet API

BabelNet provides access to its services via a HTTP/REST API, a Java API and a SPARQL endpoint. Since we worked solely with the Java API we are not going to discuss the other options in detail. Unsurprisingly, parameters and capabilities of all three options are very similar. In order to use any of the APIs one must first register at the BabelNet homepage[18]. Subsequently, a key which identifies the account is sent to the registered e-mail address. This key must be inserted into a specific configuration file located in the working directory of the project where BabelNet is to be used.
The number of queries to BabelNet is limited to 1.000 for an account registered this way. However, for research purposes, i.e. when affiliation with a research institution can be demonstrated, this limit is increased by the developers. Alternatively, under the same precondition, the complete BabelNet indices (with a size of 16 Gigabytes) are available for download. We elected this option because once the indices are downloaded there are no more restrictions regarding API calls.
One class that is implemented following the singleton pattern, i.e. only one instance of it is allowed at the time, serves as entry point to all functionality of BabelNet. *Synsets* are represented by a class which encapsulates a lot of information including an alphanumeric ID that identifies it, its part-of-speech, its relations to other synsets, whether it is a named entity, et cetera. A synset contains one or more *senses*, i.e. terms that can express the respec-

[18] `http://babelnet.org/`

tive synset in a given language. For example, the synset $\langle Car \rangle$ contains the English senses "automobile", "car", "auto", "machine" and "motorcar". Senses are language specific, so $\langle Car \rangle$ contains many more senses in different languages (e.g. "PKW", "Wagen", et cetera for German). This is how synonymy is modelled in BabelNet.
The framework also covers polysemy. In fact, the same term can appear in multiple senses belonging to different synsets. For instance, the term "church" can be a sense of the synset for the building as well as the synset for the Christian church or the church service. Functionality is provided to retrieve all possible synsets a given term belongs to. This is the point where word-sense disambiguation becomes necessary. BabelNet does not provide any means to rank or score the results of a query for a word.
Another aspect of BabelNet that is important for this study is the mapping between synsets and categories. The version of BabelNet (3.7) which was used in this study did contain this mapping already. However, the researchers who developed BabelNet put out a newer, more precise mapping in the form of a downloadable list. This list contained over 2,7 mio synset IDs together with the associated category and a confidence score of the category annotation. A small number of synsets had more than one category label which was the case when the confidence for these secondary categories was close to the main category. Since this represented the most accurate mapping we decided to use it as a basis for our categorization approach.
Table 2 shows a complete list of the 34 available categories. The origin of

Table 2: The set of 34 categories

Animals	Art, architecture, and archaeology	Biology
Business, economics, and finance	Chemistry and mineralogy	Computing
Culture and society	Education	Engineering and technology
Farming	Food and drink	Games and video games
Geography and places	Geology and geophysics	Health and medicine
Heraldry, honors, and vexillology	History	Language and linguistics
Law and crime	Literature and theatre	Mathematics
Media	Meteorology	Music
Numismatics and currencies	Philosophy and psychology	Physics and astronomy
Politics and government	Religion, mysticism and mythology	Royalty and nobility
Sport and recreation	Textile and clothing	Transport and travel
Warfare and defense		

these is the Wikipedia featured articles page[19] plus FARMING and TEXTILE AND CLOTHING which were added by the authors. This page provides a set of links to associated Wikipedia articles for every category (127 on average).

[19] `https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`

This resource together with some graph based heuristics was the basis for the automatic mapping of synsets to categories [15].

4.3.3 Babelfy API

Accessing the services of Babelfy works the same as for BabelNet: First, one must register at the webpage[20] and then, upon arrival of the key, the API is open to use. As a standard, the number of calls to the API is again limited to 1.000. There is no downloadable version for Babelfy. Hence, we requested an increase of the daily limit for research purposes. Luckily, the developers were kind enough to grant this request and increase the limit to 15.000.
The entry point for Babelfy is – the same as Babelnet – a single class following the singleton pattern. Passing it a string and a parameter that specifies the language results in a list of *semantic annotations*. Each annotation contains an ID that identifies the respective sysnet, an attribute that marks the position of the annotated text fragment in the input string. Furthermore, Babelfy provides three separate scores for all discovered synsets:

1. *Disambiguation score*: Since Babelfy does not do just Entity Linking but Word Sense Disambiguation as well a score is given for every mapping between a word or phrase and the chosen concept (synset) in BabelNet. This score reflects the confidence for the selected sense of the word.

2. *Coherence score*: The coherence score measures the level of connectedness of the disambiguated synset in context. It is computed based on the number of connections that the synset has with other synsets in the same text.

3. *Relevance score*: The relevance score measures the relevance of the concept within the input document. This is solely dependent on the position of the node in the graph representation of the text, it has nothing to do with respective synset itself.

Babelfy has quite a few parameters that can be changed. For this study, we used mostly the standard settings. We selected the option that makes Babelfy interpret all adjectives as nouns to reduce the number of distinct synsets in the results. An important parameter is whether Babelfy considers only exact matches or both exact and partial (e.g. "Donald" leading to ⟨*Donald Trump*⟩ as a potential match) matches for disambiguation. After some debate, we opted for the stricter option (only exact matches) since it reduces noise in the results.

[20] `http://babelfy.org/`

5 Categorizer: Architecture

In this section, the architecture of our approach is going to be presented. First, we are going to give an overview of the framework and discuss its internal and external components. The second part will give a detailed description of the categorization algorithm we developed, including a discussion about all its parameters.

5.1 The Framework

The framework we developed consist of two main components: One that imports the data from Portalwatch and performs language as well as concept detection to the datasets. This component is called *Concept Finder*. The second component is called *Categorizer*. It uses the pre-processed data to perform categorization. It also provides a graphical user interface for parameter setting and result visualization.

Figure 20 shows an overview of the architecture of our framework. The starting point are the Open Data portals from where Portalwatch harvests all available metadata of open datasets. After their metadata is semantified into the DCAT format the datasets serve as an input for the Concept Finder component. Here, the string that serves as input for the next steps is assembled. Furthermore, this component is responsible for parameter setting of the external libraries for language detection and concept detection. Moreover, it can detect the language and discover the concepts of a dataset via these libraries. Also, it creates the necessary data structures that represent the annotated datasets.
The second component, named Categorizer, receives the annotated datasets by portal. This process is initiated by the user who selects the desired portals. Categorizer allows the user to navigate through the datasets and displays the relevant fields as well as all concepts, scores and respective categories belonging to each dataset. At this point, the knowledgebase is used to establish the mapping between concepts and categories. Furthermore, this component manages setting parameters and executing the categorization process. Moreover, it enables the user to manually define weights for specific concepts as well as edit the confidence scores and category for them. All settings of the applications can be saved to and loaded from a file by the user. Finally, it is possible to apply categorization using the current parameters to any set of portals and export the resulting category distribution, i.e. the frequency of each category for every portal in the set.

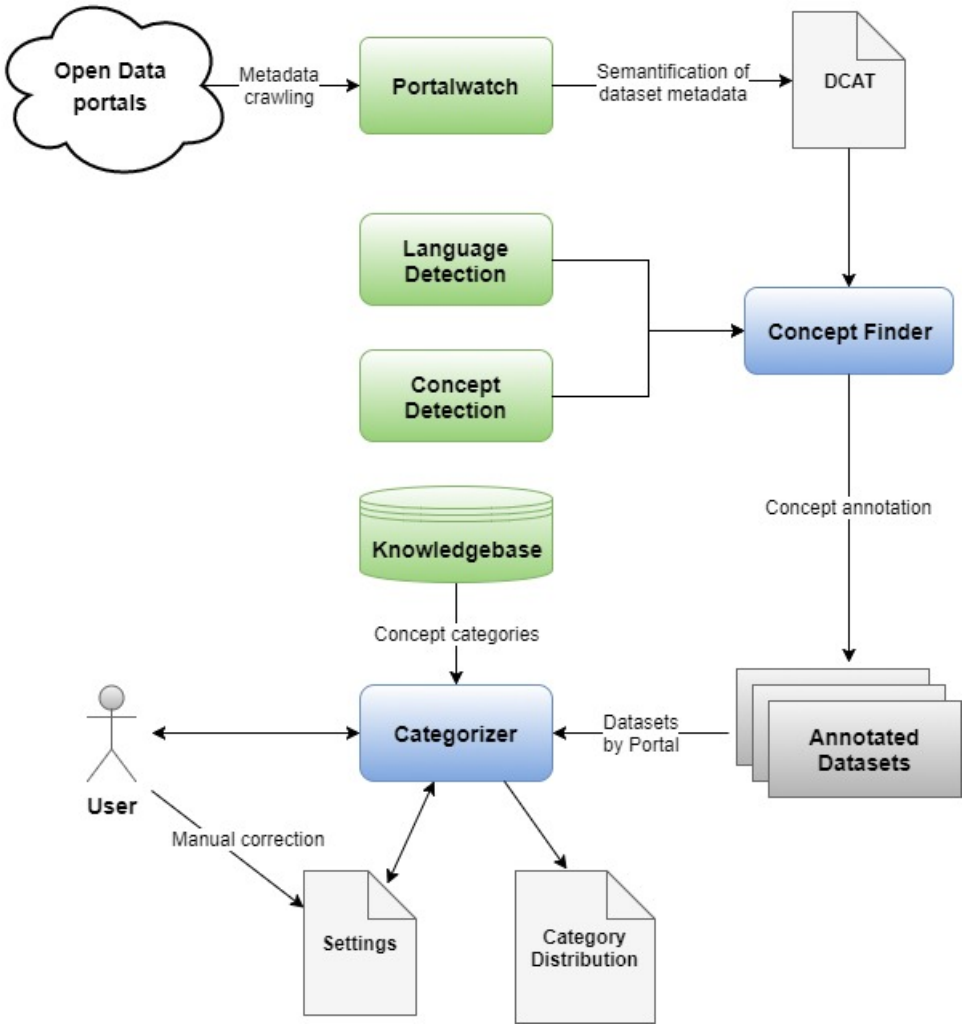


Figure 20: The architecture of the prototype.

5.2 The Categorization Algorithm

The prototype we build during this study is based on the algorithm that is going to be discussed in the following. Its purpose is to derive a set (map) of relevant categories M for a dataset d with $M \subseteq K$ where K is the set of the possible categories. Furthermore, every entry in M has a corresponding score indicating the relevance of the respective category for the dataset.

5.2.1 Requirements, Input, Output

A prerequisite for application of this algorithm is that concepts have been discovered and scored for the input dataset and that a mapping between concept types, i.e. synsets, and categoires is available. In principle, this approach does not require Babelfy and BabelNet, which we used in our implementation, as resources for concept detection. If a different entity recognition service would provide a similar structure, i.e. concepts, scores and category mapping, then it could be used just as well.
Algorithm 1 shows a pseudocode representation of our approach. The input is a single dataset d and a set of parameters that affect the outcome of the categorization process. The output is a map S that contains all categories relevant for d together with scores indicating their relevance. The first step is to initialise M and gather all concepts discovered for the dataset (Line 3). Next, we iterate over all concepts of d. Here, we check whether the current concept belongs to a category. If it does not we move to the next concept (Line 6) since no categorical information can be gained. Table 3 shows an example dataset providing some employment statistics about the Australian Capital Territory Public Service (ACTPS). We can see that for this dataset seven concepts were discovered in total, with six having an associated category.

5.2.2 Scores

For concepts where relevance- and coherence-scores are available our confidence score (*score*) is initialised by a weighted average between them (Line 8). The parameters w_r and w_h determine these weights. Their sum is required to be 1 to ensure that *score* is in $[0, 1]$ (relevance- as well as coherence-scores are also in $[0, 1]$). The values of w_r and w_h are not strongly influential on the outcome of the algorithm since relevance- and coherence-scores are somewhat strongly correlated (correlation coefficient = 0.65 in the data of this study). However, we still kept them as parameters because we deem both scores relevant and found identifying an optimal set of weights out of scope for this thesis.

Algorithm 1 Categorization of a dataset

Input: dataset d;
weights for relevance score w_r and coherence score w_h;
weight w_k for concepts found in keywords;
weight w_c for category confidence;
map S containing weights for concept types (synsets)

Output: map M containing a set of categories with scores

Require: $w_r + w_h = 1; \forall x\, 0 \leq S[x] \leq 1$

function CATEGORIZE(d, w_r, w_h, w_k, w_c, S)
  $M := \emptyset; N := \emptyset$
  $C := concepts(d)$ ▷ all concepts discovered for d
  for each $c \in C$ **do**
    if $category(c) = \emptyset$ **then**
      continue
    end if
    $score := relevance\_score(c) \cdot w_r + coherence\_score(c) \cdot w_h$
    if $keyword\_concept(c)$ **then**
      $score = \frac{score \cdot w_k \cdot (1 + category\_confidence(c) \cdot w_c)}{w_k \cdot (1 + w_c)} \cdot S[c]$
    else
      $score = \frac{score \cdot (1 + category\_confidence(c) \cdot w_c)}{w_k \cdot (1 + w_c)} \cdot S[c]$
    end if
    if $category(c) \in M$ **then**
      $M[category(c)] = M[category(c)] + score$
      $N[category(c)] = N[category(c)] + 1$
    else
      $M = M \cup (category(c), score)$
      $N = N \cup (category(c), 1)$
    end if
  end for
  for each $k \in M$ **do**
    $M[k] = \frac{M[k]}{N[k]}$ ▷ Normalize scores
  end for
  return M
end function

Table 3: Example dataset with concepts, scores and categories. **Bold** indicates keyword concepts.

Title	*ACTPS Workforce Indicators*
Description	*Statistics on employment in the ACT Government Public Service from 2007-present*
Keywords	*ACTPS, employment*

Concept	Rel.	Coh.	Category	Conf.	Weight
⟨***Employment***⟩	0.01	0.2	Business, Economics and Finance	0.283	1.0
⟨*Index*⟩	0.02	0.4	Business, Economics and Finance	0.448	1.0
⟨*Statistics*⟩	0.08	0.7	Mathematics	0.389	0.0
⟨*Act*⟩	0.02	0.4	Philosophy and Psychology	0.496	1.0
⟨*Governance*⟩	0.01	0.8	Politics and Government	0.931	1.0
⟨*Service*⟩	0.2	0.8	Business, Economics and Finance	0.871	0.3
⟨*Public*⟩	0.03	0.3			

The next step is to check whether the current concept c was found in the keywords of the dataset (Line 9). In the example dataset only the concept ⟨*Employment*⟩ was found in the keywords. When that is the case *score* is multiplied by the parameter w_k which indicates the influence of concepts found in the keywords compared to concepts found in description and title. Usually, w_k should be larger than 1 since the nature of a keyword is that it is supposed to concisely describe the content of a dataset. A value for w_k between 1.5 and 3 is recommended in order to not discard the information of the other fields completely.

5.2.3 Annotation Confidence

The $category\_confidence(c)$ of a concept c is the confidence score for the mapping from concept type to concept and is provided by the knowledgebase. It is in $[0, 1]$ as well and reflects the precision of the category annotation (**Conf.** column in Table 3 for the example dataset). The influence of this confidence value on *score* is determined by yet another parameter, namely w_c. For BabelNet the method to acquire the categories and confidence scores

performs well [15], however it is still automatic and far from perfect. The parameter w_c is intended as another means to reduce variance in the output. However, it is still recommended to keep $category\_confidence(c)$ in the calculation since these scores vary quite a bit (standard deviation 0.28 in our data) and it is reasonable to include a measure of "correctness" of the category annotation into the equation.

5.2.4 Semi-automatic scoring

During experimentation, we found that some concept types are very frequent and at the same time general. For instance, the concept type $\langle Data \rangle$ appears in over 24% of all datasets which is not surprising considering the Open Data context. At the same time, this is a very general concept, i.e. it does not reveal much information about the content of an open dataset. Therefore, its annotated category COMPUTING should perhaps not be considered when deciding on the category label for a dataset. There are numerous other examples of inherently general concepts such as $\langle Year \rangle$, $\langle Type \rangle$ or $\langle Information \rangle$. When we first tested our approach the frequency of category labels like COMPUTING and PHYSICS AND ASTRONOMY was rather high and often inaccurate. On closer inspection, we discovered the reason to be precisely these general concepts that occurred frequently and brought categories into the equation that had little to do with the context of the dataset. Furthermore, we noticed some relevant and recurrent concepts with annotated categories which were clearly suboptimal. For example, the concept $\langle Politics \rangle$ was labelled with the category PHILOSOPHY AND PSYCHOLOGY although POLITICS AND GOVERNMENT is certainly the better option. It was clear that some manual correction would be necessary to achieve reasonable results.
For the reasons just discussed we introduced another parameter, namely S, that allows to adjust the influence of specific concepts on the confidence score ($score$) for category annotation of datasets. S is a map that contains an individual weight for every concept type, i.e. synset. This weight is applied after normalization to every $score$ of the respective concept type. For example, if we set $S[\langle Year \rangle] = 0$ then any concept for year will always receive a score of 0 no matter what values any of the remaining parameters have. For the example dataset this is indicated in the **Weight** column. We see that $\langle Statistics \rangle$ is disabled completely and the influence of $\langle Service \rangle$ is reduced by 70%.

5.2.5 Conclusion

Once the *score* of a discovered concept is calculated it is added to the result map M. If the category of the current concept has not occurred before it is simply added to the map together with the score. Otherwise, the current entry of that category is increased by the value of the score. Thus, categories for a dataset earn higher scores if many concepts associated with that category are found or if the *score* of a particular concept is high due to high relevance, being a keyword concept, et cetera. In this way, a ranking of categories is created. In a last step, to put out comparable result maps M is normalized by the number of contributing concepts (Algorithm 1 line 23).
Table 4 shows the results of the categorization process when applying the parameters we used in the analysis part of our work (i.e. $w_r = 0.8$, $w_h = 0.2$, $w_k = 2.5$, $w_c = 0.5$; see section 7 for more details) on our example dataset from Table 3. As we can see the manual correction of the concept type $\langle Service \rangle$ has quite an influence on the result; This concept detected in the description has rather high scores in every aspect, however, we felt that its generality warrants an influence reduction. Our approach results in a reasonable category selection for this example dataset.

Table 4: Categorization results for example dataset.

Concept	**Category**	*score*
$\langle Governance \rangle$	Politics and Government	0.0937
$\langle \boldsymbol{Employment} \rangle$	Business, Economics and Finance	0.0365
$\langle Service \rangle$	Business, Economics and Finance	0.0345
$\langle Act \rangle$	Philosophy and Psychology	0.0319
$\langle Index \rangle$	Business, Economics and Finance	0.0313
$\langle Statistics \rangle$	Mathematics	0.0

Resultmap

Category	**Final score**
Politics and Government	0.0937
Business, Economics and Finance	0.0341
Philosophy and Psychology	0.0319

We want to note that it is intentional that the resulting map M does not contain an entry for every possible category. Only categories that are associated to at least one concept in the dataset appear in M. Thus, a distinction can be made between categories with a score of 0 and categories that do not occur in M at all. This distinction can be useful for cases where the input

text, i.e. description, title and keywords, is very short. In such cases where only a handful of concepts are detected it can happen that just one category with a score of 0 is found in total. Then it is advantageous to have at least some hint where the dataset might belong to, although the precision of the result is of course expected to be low.
This algorithm is tailored specifically to categorize open datasets, however, the general idea of using entity recognition and leveraging them for categorization could be easily adapted to other input documents. Of course, categorization only makes sense when applied to more than a single dataset. It should be clear that the same values must be used for all parameters when using this approach on multiple datasets. Since we have not found optimal values for any of the current parameters yet manual adjustment is going to be necessary depending on the use case. This was the main reason for building the Categorizer prototype, a graphical interface for fine-tuning parameters and visualizing results. It is going to be discussed in detail in the next section.

6 Categorizer: Practical Implementation

In this section, we are going to present the Categorizer component of our prototype. First, we are going to give a brief overview of the means we used for implementation. Second, we are going to present the adaptions to the categorization algorithm discussed in the previous section that were necessary for practical implementation. Then, the graphical user interface is going to be shown and its functionality is going to be discussed. Finally, we are going to explain the current limitations of the solution.

6.1 Overview

The Java programming language was used to create our prototype due to personal preference and the availability of Java APIs for Babelfy and BabelNet. The most important the design principles that we tried to follow include minimal complexity, ease of maintenance, loose coupling, extensibility, reusability and information hiding [47]. Proper modelling of the system using the Unified Modelling Language aided in reaching these goals. These class models can be reviewed in the appendix of this thesis. One general goal was to design the classes and their interdependencies in such a way that replacing the services of Babelfy and BabelNet with some other form of entity recognition framework would require as little change to the code as possible. For that reason, popular design patterns such as the factory pattern and

other information hiding techniques were used to encapsulate all code specific to those APIs. Of course, there are quite a few aspects that are rather specific to Babelfy, e.g. the most-common sense heuristic, and still had to be integrated deeper into the system, so, we do not claim to have reached complete lose coupling of the different components. Hopefully, this will still enable relatively quick testing of other resources and knowledge bases.
We used a relational database management system (PostgeSQL 8.4) to store all relevant data. Connection from the java program to the database was established via the current version (42.0.0) of the PostgreSQL JDBC Driver which allows for rather uncomplicated submitting of queries and retrieving of results. The database schema consists of two main and three auxiliary tables. The first main table stores all information about a dataset including its ID, title, description, keywords, language, portal and a column for its categories. In general, the ID of a dataset is also the URL where it can be found on the web. The second main table stores all concept related data. Here, an entry is made for every concept that was detected. Of course, a reference to the dataset the concept was found in is stored, as well as the synset-ID and the three scores provided by Babelfy (disambiguation-, relevance- and coherence-score). Furthermore, a tag is kept that encodes whether the concept was detected in the keywords and whether it is a named entity. The auxiliary tables store the mapping between synsets and categories as well as a short name for each synset. These were created to limit the calls to BabelNet in order to increase performance of the application.
The JavaFX framework was used to create the graphical user interface of Categorizer. A big advantage of JavaFX is that it is portable to a wide variety of operating systems such as Windows, Linux and macOS without the need of any adjustment. It is designed to completely replace Swing as the standard GUI library for Java SE. The framework provides easy to use functionality for visualizing data in the form of a table, which we employed for our prototype. Performance stays at a very high level even when the underlying data structures grow large. Furthermore, JavaFX includes FXML, a declarative XML-based markup language that offers an alternative way of defining the scene graph of a user interface.

6.1.1 Language Detection

We use an open-source Java library written by Shuyo Nakatani for language detection [51]. It is based on naïve Bayes filters and comes with profiles, i.e. training data, for 53 languages, covering all languages we needed for this study. It is very easy to use: Simply passing a string that contains the text to a language-detection object results in the language code of the most

likely match. It is possible that no matching language is found, however, this happened rather rarely for our data. Nonetheless, false positives were quite frequent, especially when the input string was short and contained many uncommon words. There seems to be a little bias towards Afrikaans and Romanian, which were the most frequent wrong mappings. Overall this method worked reasonably well, although some manual corrections were necessary. Still, for future work and larger scale application of our approach we would recommend a commercial service, e.g. Google's Translation API[21]. The simple reason being that all wrong classifications at this level lead to poorer results at the next stages and precision of commercial products is expected to be much higher. Furthermore, manual checking of results uses up quite a bit of time that could be better spend elsewhere.

6.2 Adapted Categorization Algorithm

The categorization algorithm described in the previous section is, in principle, independent of the method for concept detection and the knowledgebase. Employing Babelfy for concept detection makes a small adaption to the algorithm necessary due to the so-called most common-sense heuristic it uses.

6.2.1 Most common-sense heuristic

Babelfy offers an option to enable the so-called "most common-sense heuristic" (MCS) which finds additional concepts in the input string (it is enabled as a standard in the Babelfy API). Unfortunately, there is no clear description in the documentation of Babelfy what this heuristic actually does. However, when experimenting with this setting we found out that this heuristic discovers a lot of appropriate and therefore valuable concepts, which is why we decided to utilize it.
The problem with the MCS heuristic is that it does not provide any scores for the concepts it discovers, i.e. disambiguation-, coherence- and relevance-score are all 0 for every concept it finds. Since these scores are the basis for the calculation of our category confidence scores some sort of replacement for concepts found by the MCS heuristic is necessary. Unfortunately, we could not find a way to correctly determine these missing scores, which is why we decided to shift this decision to the user of our algorithm via a parameter, namely s_m. Algorithm 2 shows the adapted categorization algorithm we used in our implementation. This parameter sets the score that is used for every concept found by the MCS heuristic (Line 9). It is required to be in the

[21] `https://cloud.google.com/translate/docs/detecting-language`

interval $[0, 1]$ since that is the range of relenvance- and coherence-scores.
s_m determines how strong the influence of concepts found by the MCS heuristic is when calculating the confidence scores for categories of a dataset. This parameter has a substantial impact on the outcome of the algorithm since almost half of all discovered concepts are found by this heuristic. This was the case for our data, however, there is no reason to believe this would change for other input texts. When using our approach for categorization one should take particular care when choosing s_m. Experimenting with different values and checking samples of datasets is in order to get a sense of the quality of concepts the MCS heuristic finds. Simply turning it off by setting $s_m = 0$ is also a viable option when less ambiguous results are sought-after.

6.3 User-Interface and Features

Categorizer is a desktop Java application that can load open datasets from a local PostgreSQL database. The code is open source and available online[22]. It has no dependencies other than the Java Runtime Environment 8. This application is intended to provide a user interface for the categorization algorithm described in the previous section. It allows to fine-tune any parameter and to explore an arbitrary set of datasets and the concepts that have been discovered for them by Babelfy. Furthermore, some statistics about aggregated datasets and categories can be reviewed as well as individual settings saved to and loaded from a file.

6.3.1 Graphical User-Interface

Figure 21 shows a screenshot of the main window of the application. In the top part (A) we see a couple of textfields where all the parameters can be adjusted. Most of them have been discussed thoroughly in the previous section, so, here only the four fields in the middle are going to be explained.
In the first version of our categorization algorithm an additional parameter was implemented that decreased the influence of concepts that recurred in a dataset. However, during testing we discovered that such a weighting does not improve the results and, therefore, unnecessarily increases complexity. Although this is the case the current version of Categorizer still offers this parameter for testing purposes.

The categorization algorithm as described in section 5.2 results in a map of categories and associated scores. This application is intended as a tool to apply this algorithm to a set of datasets and study the results. In many cases

[22] `https://github.com/Gepro83/MasterThesis`

Algorithm 2 Categorization of a dataset

Input: dataset d;
default score s_m for the most common-sense heuristic (MCS);
weights for relevance score w_r and coherence score w_h;
weight w_k for concepts found in keywords;
weight w_c for category confidence;
map S containing weights for concept types (synsets)

Output: map M containing a set of categories with scores

Require: $w_r + w_h = 1; 0 \leq s_m \leq 1; \forall x\, 0 \leq S[x] \leq 1$

function CATEGORIZE($d, s_m, w_r, w_h, w_k, w_c, S$)
  $M := \emptyset; N := \emptyset$
  $C := concepts(d)$ ▷ all concepts discovered for d
  for each $c \in C$ **do**
    if $category(c) = \emptyset$ **then**
      continue
    end if
    if $MCS(c)$ **then** ▷ concept was found by MCS heuristic
      $score := s_m$
    else
      $score := relevance\_score(c) \cdot w_r + coherence\_score(c) \cdot w_h$
    end if
    if $keyword\_concept(c)$ **then**
      $score = \dfrac{score \cdot w_k \cdot (1 + category\_confidence(c) \cdot w_c)}{w_k \cdot (1 + w_c)} \cdot S[c]$
    else
      $score = \dfrac{score \cdot (1 + category\_confidence(c) \cdot w_c)}{w_k \cdot (1 + w_c)} \cdot S[c]$
    end if
    if $category(c) \in M$ **then**
      $M[category(c)] = M[category(c)] + score$
      $N[category(c)] = N[category(c)] + 1$
    else
      $M = M \cup (category(c), score)$
      $N = N \cup (category(c), 1)$
    end if
  end for
  for each $k \in M$ **do**
    $M[k] = \dfrac{M[k]}{N[k]}$ ▷ Normalize scores
  end for
  return M
end function

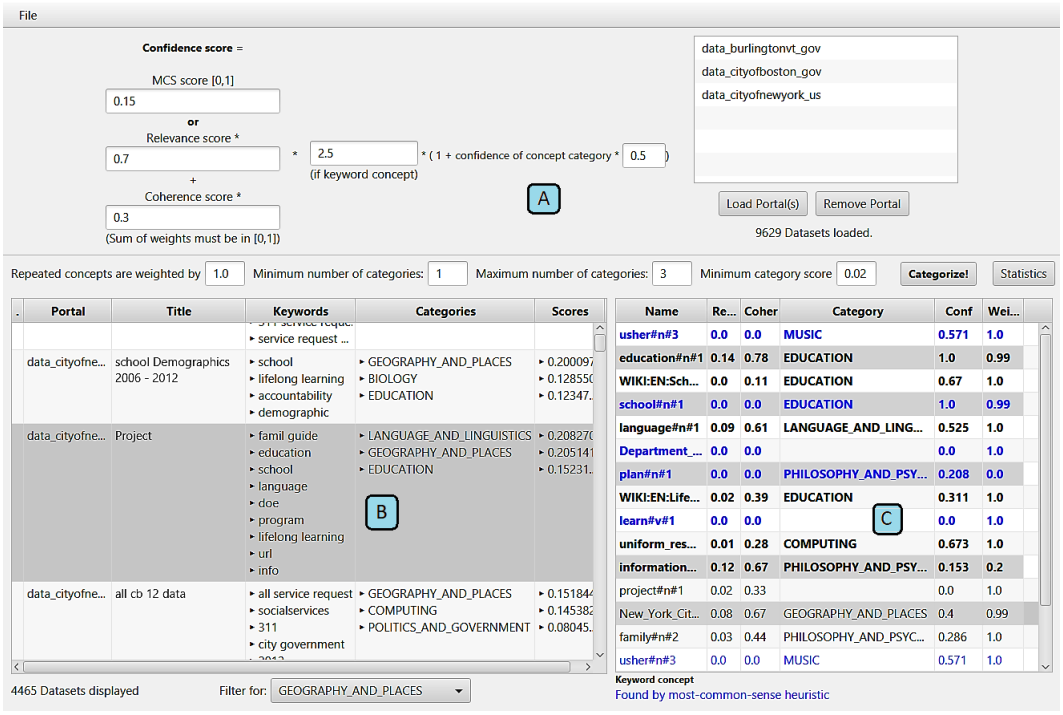


Figure 21: The main window of Categorizer

just a subset of all categories that have been discovered for a dataset will be of interest to a user. Usually, the top ranked categories will be selected or some threshold for the confidence score will be defined. Categorizer allows to specify these settings by adjusting the corresponding parameters in the middle of the window. The first one determines the minimum number of categories a dataset will be annotated with. In fact, the only way that a dataset receives less distinct category labels than that threshold is that not enough categories were found for the respective dataset, which is rarely the case for a minimum of 1 or 2. The next parameter specifies how many different categories a dataset can be maximally assigned to. The last field allows the user to set a score threshold. Only categories that earn a score above this threshold are assigned to a dataset with the exception that the minimum number of categories cannot be reached. In such a case category labels are assigned until the minimum number of categories is reached regardless of the achieved score.

The application allows to load datasets based on a portal. When a selection is made from the list of available portals the corresponding datasets are loaded into the application. The set of portals the user can chose from depends on the data that has been stored preliminarily in the database, since datasets,

concepts as well as the list of portals are obtained directly from it. Portals, i.e. their datasets, can also be removed by clicking the respective button.
Once a portal has been loaded into Categorizer the table in the bottom left (B) shows a list of all corresponding datasets. Each row in the table represents a single dataset displaying the name of the portal it belongs to, its title and keywords, its categories and corresponding scores. The columns for the categories and scores are initially empty. They are filled once a categorization process has been executed by clicking on the "Categorize!" button. The displayed datasets can be filtered by selecting a category from the combobox at the bottom of the window. As soon as a selection is made only datasets that belong to that category are displayed. By pressing *ctrl+c* on the keyboard the ID of the currently selected dataset is copied to the system clipboard. This presents an easy way to navigate to the dataset on the web, since, in almost all cases, this ID is a valid URL to the respective dataset.
When a dataset is selected the table on the bottom right (C) displays all concepts that were discovered for it. The names of the concepts stem from BabelNet, including the code for the part-of-speech ("n" for noun, "v" for verb). Relevance- and coherence-scores as well as categories and confidences scores are displayed. The "Weight" column contains the user defined weight for the corresponding concept type (the parameter S described in the previous section). Concepts that were detected in the keywords are marked with a bold font. Furthermore, concepts that were found by the most common-sense heuristic are indicated by the colour blue. Concepts whose values have been manually changed by the user are marked with a grey background.
A click on the "Categorize!!" button first triggers a parameter validity-check and then initiates the categorization process, which is usually completed within a few seconds. The "Statistics" button opens another window of the application (Figure 22). Here, some statistics about the loaded datasets are given (A). The left table (B) shows as list of all categories that were assigned during categorization. The second column contains the frequency of the category, i.e. the fraction of datasets that belong to it. In this instance 47.9% of the loaded datasets are labelled with POLITICS AND GOVERNMENT. The table on the right (C) shows a list of all concept types that occur in the loaded datasets. Again, the frequency column indicates what fraction of datasets contain a corresponding concept. The columns "AvgRel" and "AvgCoh" contain the average relevance- and coherence score of each concept (concepts found by the MCS heuristic are taken out of this calculation since they always have scores of 0). These can be interpreted as an indicator of the average importance of a concept type.
The columns for the category, the annotation confidence and the weight are editable by the user. This represents our realization of the semi-automatic

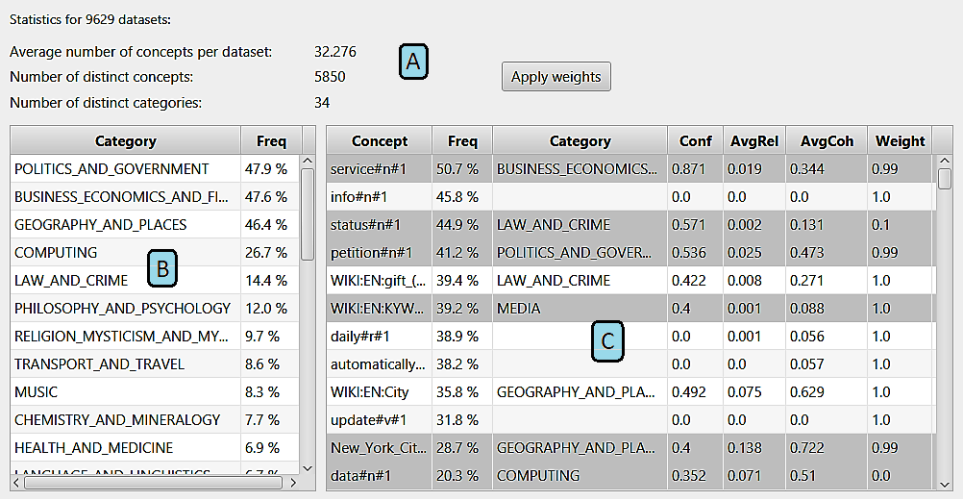


Figure 22: The statistics window of Categorizer.

scoring described in the previous section. Again, concepts that have been altered manually are marked with a grey background. Furthermore, it is possible to reset a concept to its standard values via the context menu. The "Apply weights" button is just a shortcut; it has the exact same functionality as the "Categorize!" button. Selecting a row in the concept table and pressing *ctrl+c* copies an URL to the system clipboard that leads to the corresponding BabelNet page which includes all available information about the concept. This also works in the concept table of the main window.
All settings, including the adjustments of parameters of specific concept types, can be saved to and loaded from an external file. Furthermore, a possibility is provided to generate a CSV file containing the category frequency distributions of a set of portals that is selected by the user. To create this file the categorization algorithm is applied to each portal using the current settings. The CSV file contains one column for the name of the portal plus a column for every category indicating its frequency in the datasets of the portal (The same values that are displayed in the left table of the statistics window). This CSV export feature as well as saving and loading settings can be accessed via the menu bar at the top of the main window.

6.3.2 Use Cases

Initially, Categorizer was intended to facilitate easy testing of different parameter values for our categorization algorithm. When it became clear that editing the categories and weights of individual concepts would be necessary we decided to invest a little more effort into building a proper application.

Furthermore, just looking at aggregated statistics does not suffice for determining the impact any of the parameters has on the outcome of the categorization process. The application provides a means for inspecting specific datasets and quickly analysing why certain category labels were assigned to it. The statistics window provides an easy way to determine which concepts are most influential and, if necessary, adjust some of their parameter values. Apart from the obvious task of automatically categorizing large numbers of datasets we think this application provides a good opportunity for Open Data portal administrators to analyse the quality of their metadata. If the outcome of the categorization process does not align with the expectations of someone who knows the content of the datasets some investigation could help to uncover datasets with insufficient descriptions or ambiguous keywords. Using the filter option datasets belonging to a specific category can be located quite quickly.
Another use case for Categorizer is the comparison of Open Data portals. Exporting category distributions provides a decent possibility to compare the content of multiple portals, which is exactly what we did in the analysis phase of this study (for a through discussion refer to section *Analysis*). Furthermore, the possibility of loading multiple portals at once allows to compare whole sets of portals. Moreover, the statistics window gives a good overview of the most influential concepts, thus, giving some insight into the general content of the datasets.

6.4 Limitations

Categorizer has been developed in the course of this master thesis with the specific goal to test and apply our approach of using the services of Babelfy and BabelNet to categorize open datasets. Therefore, some aspects of the application are rather limited and many potential improvements remain future work. For example, the process of loading datasets from the database into the application is relatively slow. Depending on the hardware it can take several minutes to load all datasets if their number increases over 20.000. This is due to the large number of concepts that must be loaded from the database (32 concepts per dataset on average). Certainly, some restructuring of the database or optimization of the code could improve performance, however, since the focus of this study lies on entity recognition and categorization only moderate efforts have been made to address this issue.
A missing aspect that becomes apparent quite fast when using the application is the lacking navigational link between the concept types in the statistics window and the loaded datasets. Locating datasets that contain a certain concept type would be a useful feature. Utilizing the concept types as facets

to filter the displayed datasets would be an efficient way to improve usability of the application.
Of course, the need of having the data that is used pre-processed in a very specific way that is not publicly available and stored in a custom format in a database can be viewed as a limitation as well. Moreover, one can easily think of additional useful features and improvements. However, Categorizer was developed in the course of this master thesis, thus, the scope of possibilities is inherently limited.

7 Analysis

In this section, we are going to present the findings of this study. First, the data corpus we used is going to be described. Then, we are going to look into some statistics and present some visualizations of different frequency distributions for concepts, scores and categories. Of course, we are going to explain which parameter values for the categorization algorithm we decided upon and why those choices were made. Furthermore, we are going to apply k-means clustering to the category frequency distributions of all Open Data portals in our data and try to find contextual similarities at the portal level. Finally, an evaluation of the precision of the algorithm is going to be presented. To that end, we conducted a survey were participants were asked to manually categorize datasets. The results are going to be discussed in the last part of this section.

7.1 Statistics

Table 5: Key figures of the data corpus.

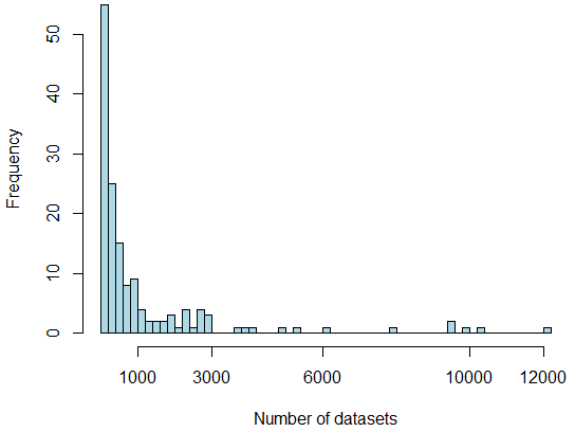
			Occurrences	**Types**
Portals	150	**Concepts**	5,091,871	52,476
Datasets	174,308	**Named entities**	467,763	27,088
Languages	24	**Total**	5,559,634	79,564

Table 5 shows the key figures of our data corpus. All data originates from the Portalwatch project[23]. We managed to appropriately pre-process, i.e. run through Babelfy, a total of 174,308 dataset. These belong to 150 different portals (a full list is provided in the appendix). Figure 23a shows the distribution of the count of the datasets per portal. We can see that

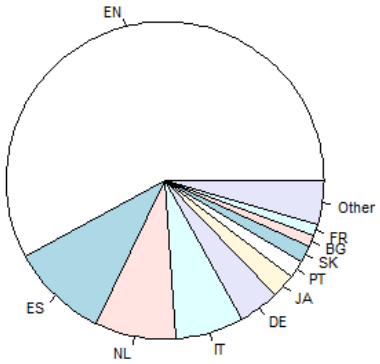
[23] `http://data.wu.ac.at/portalwatch/`

most portals contain less than 1,000 datasets and only a few with a significantly higher amount are part of this study. We selected the small portals intentionally because of the limitations of Babelfy, i.e. only allowing to process a fixed number of datasets per day. Since one of our goals was to find contextual similarities among portals we decided to process all small portals first to cover as many as possible.

Figure 23: Distribution of datasets and languages.



(a) Distribution of the number of datasets per portal.



(b) Distribution of languages of datasets.

A total of 24 distinct languages were detected in the datasets. Figure 23b shows their distribution. We can see that over half of the datasets have English descriptions. With English being the most popular language in the Web this result is not surprising, however, its dominance is still remarkable. One of the goals of Open Data is to make it easily accessible for a big audience, which is probably another reason why the English language is often chosen as a means of communication.

The amount of textual information that is available for each dataset is, of course, crucial for our study. Figure 24 shows the distributions of the number of characters for titles and descriptions of the datasets. Titles are in general much shorter - most of them contain less than 100 characters with the median at 40. Descriptions contain more characters, 63.6% of datasets have descriptions between 50 and 500 characters long, with a median at 144. The tail of the distribution for descriptions is noticeably longer. There are quite a few datasets that have lengthy descriptions, 6.6% of them include over 1.000

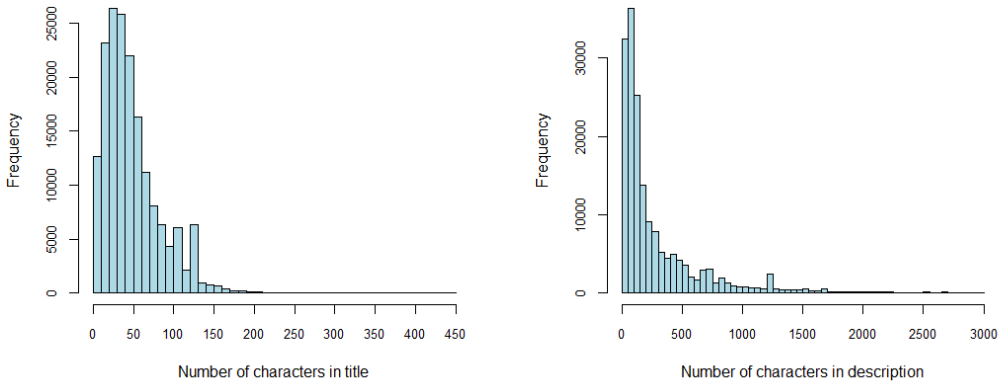
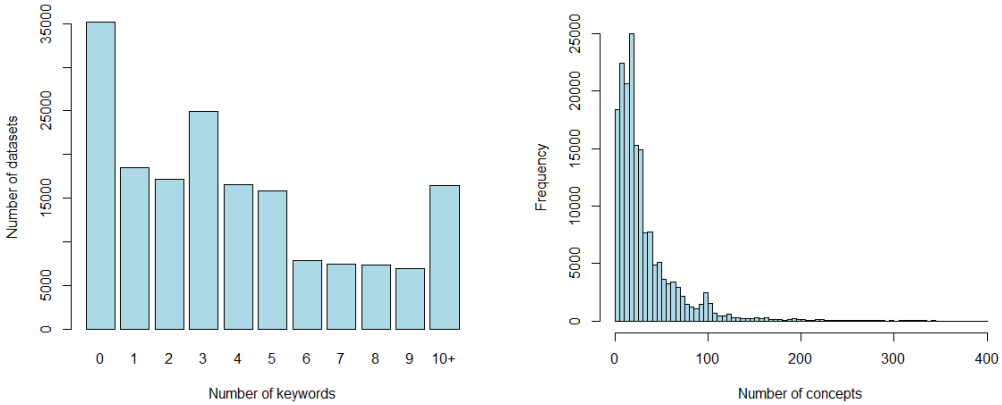


Figure 24: Distributions of the number of charaters.

characters. Furthermore, it is noteworthy that the fraction of datasets that contain less than 10 characters for the title is 6.4% and for the description 8.2%. Some of the datasets (2.4%) even contain less than 20 characters for title and description combined. Clearly, such datasets pose a great challenge for any categorization algorithm due to the limited amount of information to work with.

Figure 25: Distribution of keywords and concepts.



(a) Distribution of the number of keywords.

(b) Distribution of the number of detected concepts.

Another interesting aspect is the number of keywords since concepts de-

tected in this field are valued higher by our categorization approach. Figure 25a shows their distribution. The median number of keywords is 3. Interestingly, over 20% of the datasets do not contain any keywords. This is not due to some portal leaving out this field; 138 out of the 150 portals contain at least some datasets without any keywords, thus, this seems to be a common phenomenon.
Now, we are going to analyse the concepts discovered by Babelfy. A total of 5,559,634 concepts were detected in our data corpus. Figure 25b shows the distribution of concepts per dataset. Of course, it looks similar to the distributions of title and descriptions since the length of the input string and the number of detected concepts is highly correlated. The median for the number of concepts is 21. For almost all datasets (99.2%) at least one concept could be detected. The distribution of distinct concepts, i.e. excluding duplicate types, looks almost identical, it has its median at 15.

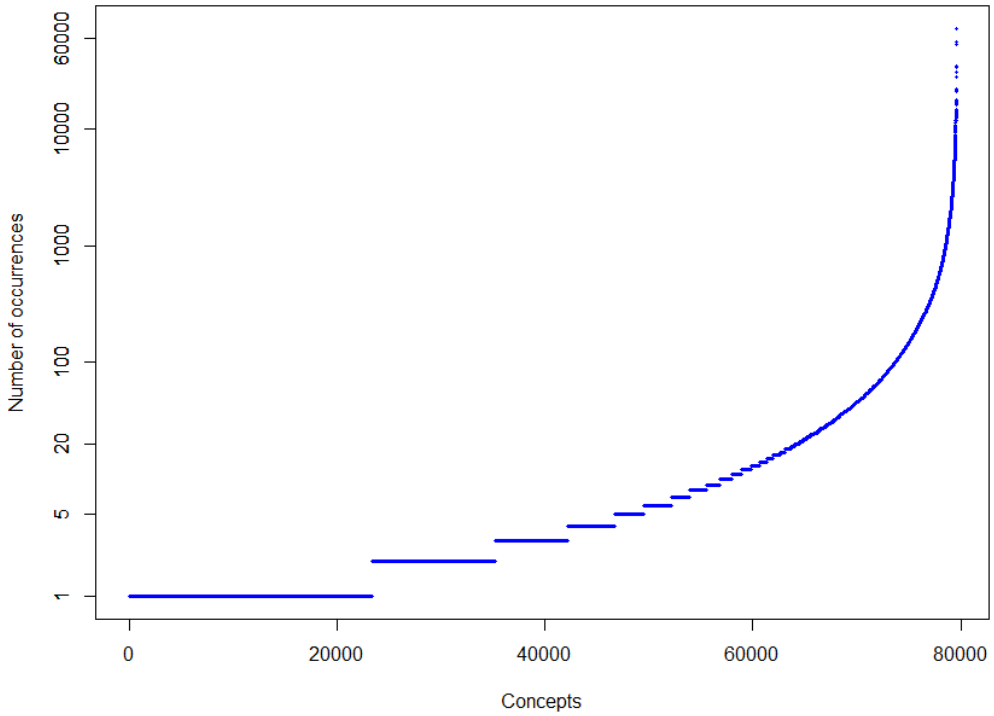


Figure 26: Occurrence of concept types.

Figure 26 shows a plot of the rate of occurrence of each distinct concept, i.e. concept type (multiple occurrences within a single dataset are included

here). A total of 79,564 distinct concepts have been detected. It is remarkable that the vast majority of concept types occur only a few times; 88.6% of concept types only appear 50 times or less, whereas a few types are extremely frequent. Research shows that word frequencies in natural language follow a similar distribution known as *Zipf's law* [55]. As the evidence suggests, this translates to the frequency of concept types. The most frequent types are very generic, which is why we disabled their influence on our categorization process. Table 6 lists the ten most frequent concept types with their total occurrence-count.

Table 6: The most frequent concepts with a count of occurrence.

Concept Type	**Count**
data	72,222
service	55,995
data point	53,412
year	34,809
petition	34,050
data set	30,406
information	28,019
department	27,615
municipality	21,997
map	21,423

Table 7: The most frequent named entities with a count of occurrence.

Named Entity	**Count**
KYW News Radio	20,965
New York City	10,190
Engineering and Physical Sciences Research Council	5,093
Los Angeles	4,768
Aragon	4,173
Journal of Chemical Physics	3,926
City Of (Tv show)	3,690
Spain	3,631
Dallas Police Department	3,319
StatBank	3,088

In BabelNet synsets that represent named entities, e.g. $\langle Vienna \rangle$ or $\langle Bob\ Dylan \rangle$, are marked as such. Around 8.4% of the discovered concepts in

our data are named entities. Interestingly, these are comprised of 27,088 distinct entities, thus, 34% of the total concept types are named entities. This indicates that a lot of variation in the content of Open Data is due to the focus on a multitude of different named entities. The distribution occurrences of named entities follows Zipf's law as well. Table 6 shows the ten most frequent named entities. Some of them seem peculiar, e.g. the most frequent named entity being a news radio station. Investigation revealed that these are indeed almost entirely false positives, the same is true for $\langle City\ Of \rangle$ which refers to the name of a TV show. Babelfy seems to have a bias for some synsets. This goes to show that the Babelfy algorithm is far from perfect and that at least some human inspection of the results is appropriate.
Out of the 79,564 concept types (synsets) 42,168 (53%) possess an annotated category label. Thus, a dataset contains approximately 10.5 concepts on average that can contribute to the categorization process. The percentage of labelled synsets is significantly higher than the percentage for BabelNet as a whole; About 2.7 mio out of the 14 mio total, i.e. only around 19%, synsets possess category labels. Presumably, this discrepancy stems from the fact that Open Data is generally described using rather common language, whereas BabelNet includes a large amount of very specific synsets which are often skipped by the automated label annotation process [15].

7.2 Categorization of Datasets

Now, we are going to discuss the results of applying our categorization algorithm to the data corpus. Our approach is tailored specifically for Open Datasets and, unfortunately, there are no publicly available annotated datasets which could be used as a reference set. Therefore, we had to manually choose all the parameters. Due to the scope of a master thesis we did not have enough time to invest the amount of effort that would be necessary to find the optimal values. This will remain future work for the time being. Nevertheless, we tried to select reasonable values which we are going to argue for in the following.

7.2.1 Parameters

Our selection of parameter values is based on experimentation and manual inspection of categorization results. Table 8 shows the settings we selected to conduct the analysis of this section. The first one is the default score for the MCS heuristic. This parameter is at the same time very impactful and hard to select. Almost half (49%) of all detected concepts were found by the MCS heuristic, which is why, this parameter influences the categorization of almost

Table 8: Categorization parameters used for analysis.

Parameter	Value
default score for MCS (s_m)	0.07
weight for relevance score (w_r)	0.8
weight for coherence score (w_h)	0.2
weight for keyword concepts (w_k)	2.5
weight for category confidence (w_c)	0.5
minimum number of categories	1
maximum number of categories	3
minimum category score	0.1
number of edited concept types	232

every dataset. On analysing various samples, we discovered that relevance of the concepts detected by the MCS heuristic to the context of a dataset varies just as much as for the other concepts. Unfortunately, there is no way to determine the relevance of such a concept from the information made available by Babelfy. The only solution other than disabling the heuristic, which would entail a substantial information loss about the datasets, was to choose some value that aligns with the relevance and coherence scores of the other concepts. Therefore, we chose the value for s_m close to the weighted average relevance (0.025) and coherence scores (0.279). This reflects our assumption that on average concepts detected by the MCS heuristic are as relevant as concepts detected the standard way.
Although the possible range of the relevance score for a concept is $[0, 1]$, in general, this number is very low for concepts detected by Babelfy in our data. In fact, the average relevance score is only 0.025 and only 1.7% of datasets contain a concept with a score above 0.5. We were not able to determine the reasons due to limited access to the inner workings of Babelfy. Still, the score reflects the relevance of concepts in the input text which is why we wanted to keep it in the calculation. The coherence score is distributed more normally in its range. Since this score is based on the connectedness of the synset it favours concepts like $\langle Nation \rangle$ or $\langle Water \rangle$ which have many relations to other concepts. Although we consider both scores evenly important we opted for a weighting that favours the relevance score in order to achieve a somewhat even influence of both of them on the categorization process.
Our selection of weights for keyword concepts (w_k) and the category confidence of concept types (w_c) is based on experimentation with different samples of datasets. Of course, the information quality of keywords is quite variable; However, in general, most publishers of Open Data try to use them

correctly as concise descriptors of the context. Thus, we selected a value for w_k that reflects the importance of keywords while not rendering the concepts detected in the description insignificant. Since we found that it contributes quite a lot to variance in the results we reduced the influence of the category confidence by 50% to maintain a balance between variance and precision.
We opted for a range of the possible number of category labels that annotates datasets with one to three labels. We wanted to maximise the number of datasets that receive at least one label while keeping the categories well defined. Thus, we selected a relatively strict threshold for the category score but still allowing up to three labels per dataset. The fact that 0.1 is a high value is due to the general low values for relevance and coherence scores as well as the low value of s_m which lead to small numbers for the category confidence score of our algorithm.
In total, we edited 232 concept types using our Categorizer prototype. This was done by going through the most frequent types and making adjustments where they felt appropriate. In most cases, we simply disabled very general concepts such as $\langle Number \rangle$ or $\langle Year \rangle$. In addition, some adjustments of category labels were made such as changing the category of $\langle Politics \rangle$ to Politics and Government from Philosophy and Psychology. We tried conduct such overrides only for very clear cases. Lastly, the influence of some very frequent concept types such as $\langle Data \rangle$ was decreased in order to achieve higher selectivity of the different categories.

7.2.2 Categorization Results

Applying our categorization algorithm using the parameters discussed above to our data corpus results in the distribution of categories depicted in Figure 27. At least one category was found for 98.3% of all datasets. We can see, that the most frequent category label is Geography and Places. This result is plausible considering the fact that a lot of Open Data, especially Open Government Data, is published by authorities responsible for certain regions, e.g. cities, resulting in datasets that contain information related to these regions. Computing being a frequent category stems from frequent mentions of data related terms in descriptions as well as many datasets containing statistical information which often entails concepts with this category label. In addition, the portal of the University of Bristol[24] is included in our data and it provides over 12.000 datasets of research data, 5224 of which belong to this category. Politics and Government as well as Business Economics and Finance being part of the four most frequent categories

[24] `https://data.bris.ac.uk/data/`

is also very probable since most of the datasets are Open Government Data.

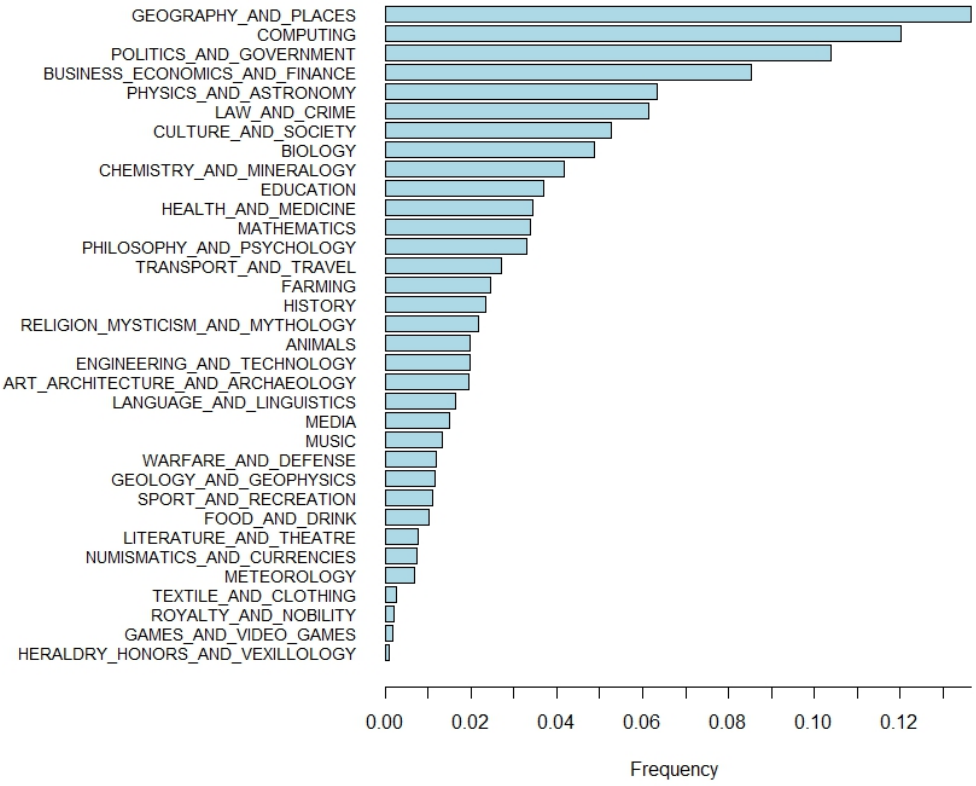


Figure 27: Frequencies of categories among datasets.

Figure 28 shows the distribution of the category confidence score per label assigned. Most of the datasets (87.2%) receive just a single label due to the threshold at 0.1. Still, close to half (46%) of the labels only achieve a confidence score between 0.05 and 0.1. These are only assigned because of the condition for the minimum number of categories. The mean lies at 0.123. As mentioned above, this is due to the generally very low values of relevance, coherence and MCS scores which influence the final score the most.

7.2.3 Clustering of Open Data Portals

We employed k-means clustering in order to find contextual commonalities between the 150 Open Data Portals in our data corpus. As a basis, we utilized the category distributions our prototype is able to produce: A portal is represented as a 34-dimensional vector that contains the frequencies of each of the 34 possible categories. Thus, any two of these vectors can be

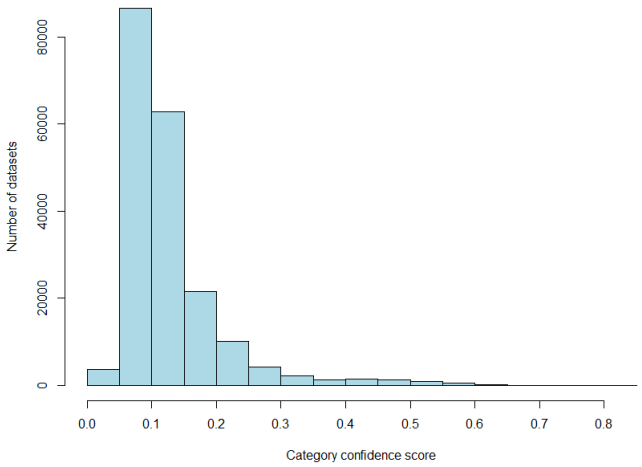


Figure 28: Distribution of category confidence scores.

compared to each other. Since this is purely numerical data and the ranges of all dimensions are equal, i.e. the Euclidean distance is a sensible distance measure in this vector space, k-means is a suitable clustering method.

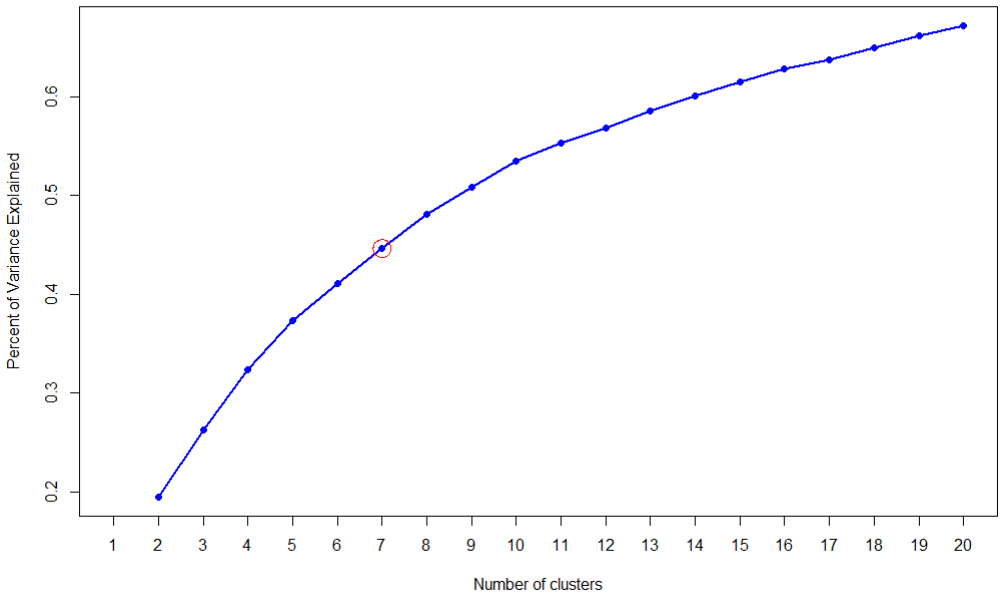


Figure 29: Variance explained by number of clusters.

The elbow criterion discussed in section 2.9.1. was used to find an appropriate number of clusters. Figure 29 shows a plot of the percentage of variance explained by the number of clusters. Unsurprisingly, there is no clear elbow in this plot which is often the case for real-world data. We chose a number of clusters that explains close to 50% of the variance while resulting in relatively distinct clusters with clear contextual focuses.

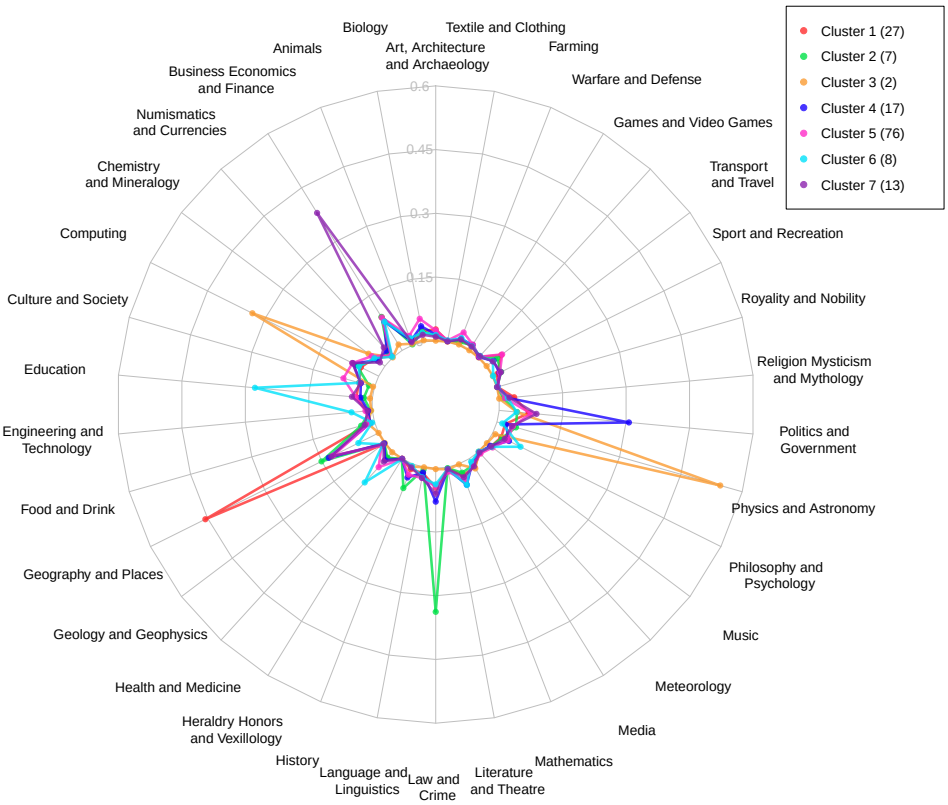


Figure 30: Cluster centers of k-means.

The centers of the seven clusters found by k-means is are depicted in a radar chart in Figure 30. One can immediately see that most clusters have tendencies towards certain categories while one (cluster 5) has relatively even distribution of categories. The plot of all 150 portals in Figure 31 reveals a moderate amount of variance remaining within the clusters, yet, the cluster centres seem to represent the focus of the portals quite well.
The one cluster that stands out is cluster 5 having no clear categorical tendency. In fact, looking at Figure 32 which depicts all portals belonging to cluster 5, we can see that there are only a few portals in this cluster that exceed the 30% mark for any single category. Moreover, this cluster is the

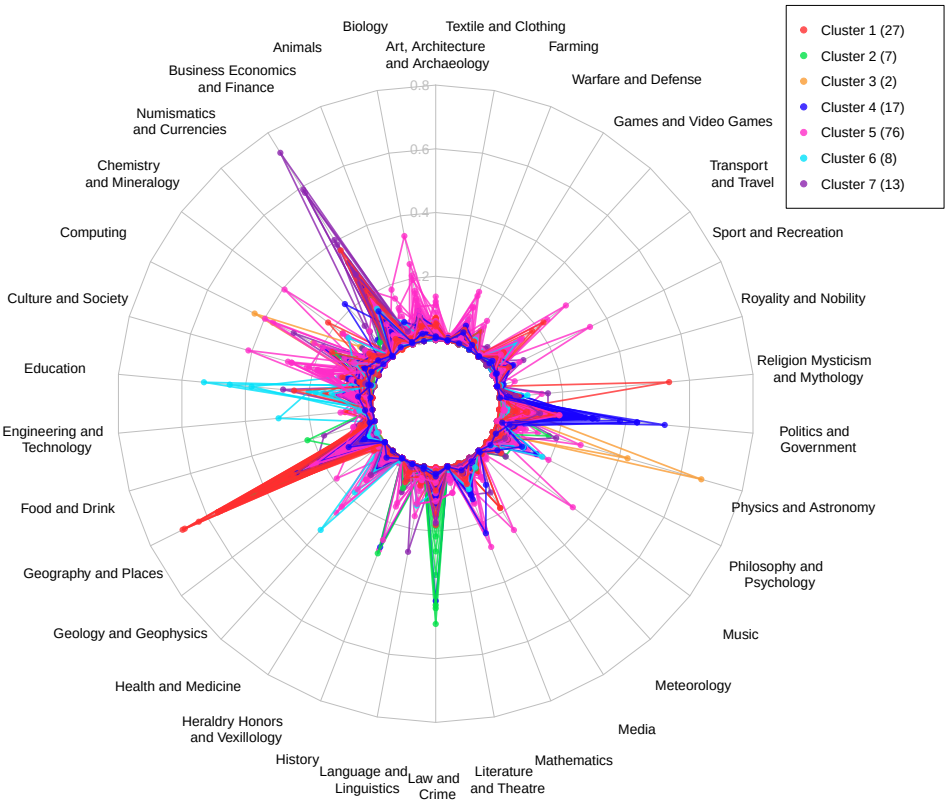


Figure 31: Category distributions of all portals.

largest one; It contains over half of all portals of our data corpus. This data indicates that our algorithm produced categories with a high selectivity for the portals in this cluster which is in general a desired outcome for any automatic categorization algorithm.

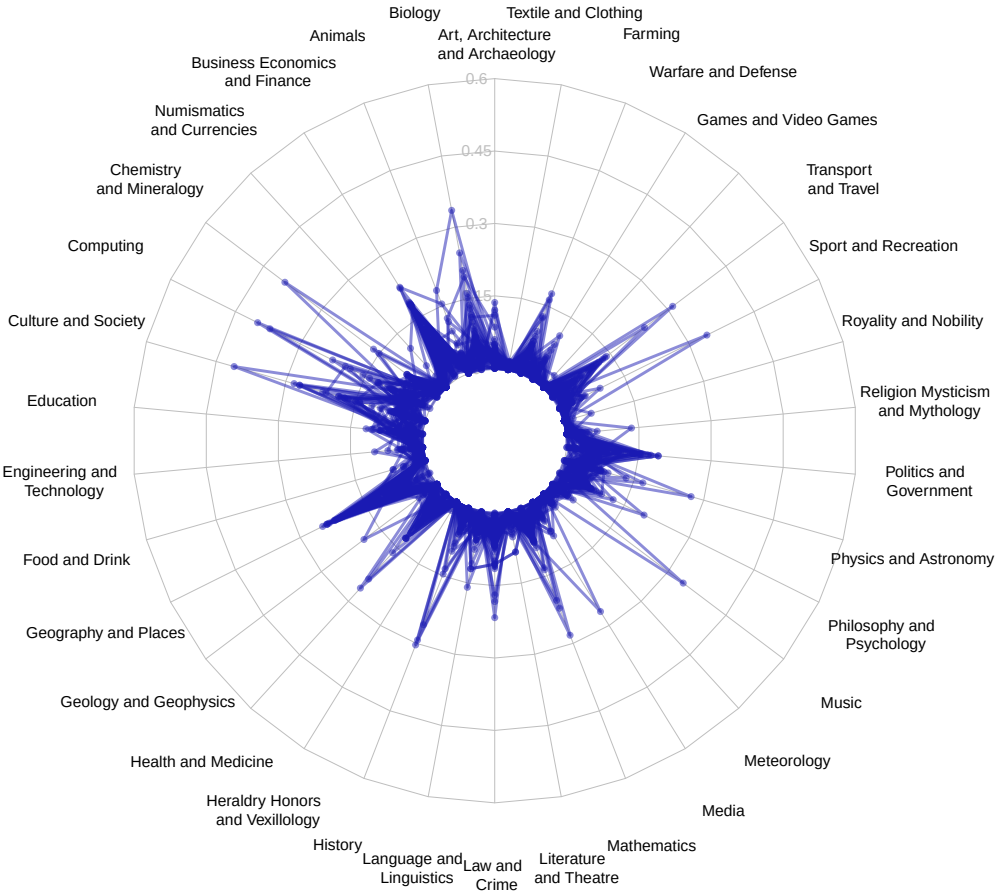


Figure 32: Category distribution of cluster 5.

Let us inspect two of the smaller clusters, namely cluster 3 and cluster 7. Figure 33 shows a plot containing all portals of these two clusters. Cluster 3 contains two portals that each contain datasets belonging mainly to two categories: Computing and Physics and Astronomy. The two portals are (i) an American government portal providing data about the energy sector[25] and (ii) the portal of the University of Bristol[26] which provides mostly scientific research data. While the categorization results for these specific portals

[25] https://data.energystar.gov/
[26] https://data.bris.ac.uk/

are not very useful due to the low selectivity of the categories, the clustering is certainly sensible. Both portals offer similar content, i.e. analytical data based on physical science.

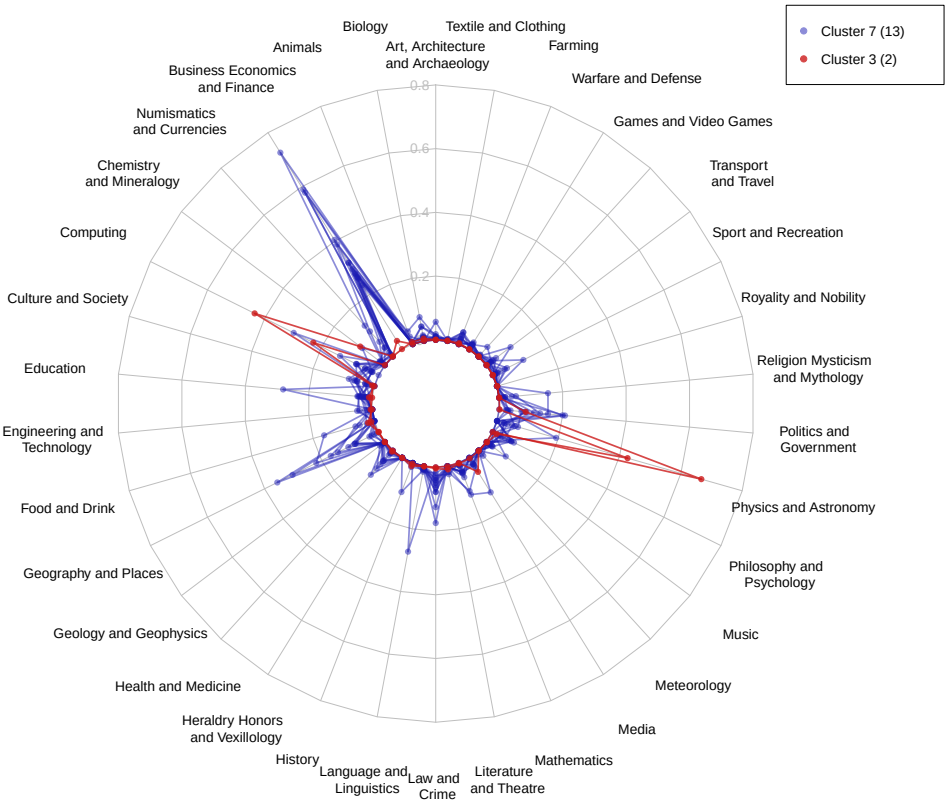
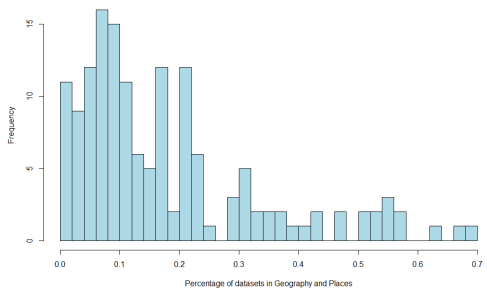


Figure 33: Category distributions of cluster 3 and 7.

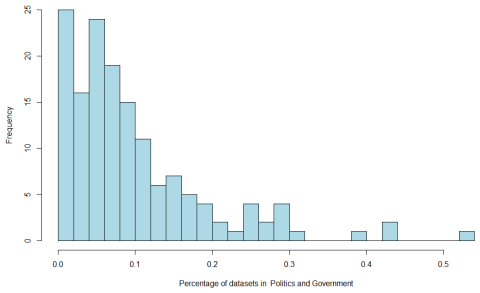
The emphasis of cluster 7 lies on Business Economics and Finance. Although there are a few portals with over 50% of datasets in that category most of the portals have somewhat wider category distributions. The focus on the main category is still noticeable for all of them, i.e. at least 20% of the datasets of all portals were assigned to it. Examples of this cluster are (i) the Cook County government portal[27], (ii) the portal of Culver City in USA California[28] and (iii) the Open Data portal of the Vienna University of Economics and Business[29]. At this time, (i) provides 528 datasets out of which 172 are classified under "Finance & Administration" by the portal

[27] https://datacatalog.cookcountyil.gov/
[28] https://data.culvercity.org/
[29] http://data.wu.ac.at/portal

administrators which indicates a correct cluster allocation. All of the categories ("Payroll", "Expenditures", "Revenue", "City Businesses") advertised on the website of (ii) are business or finance related, thus, it represents another correct allocation.
Since we are familiar with the content of the portal we can state that this category label is a false positive for all datasets of (iii). The majority of datasets are about lectures and events and have little to do with economics or finance. The reason for this categorization result is the mention of the full name of our university in the rather brief descriptions of almost all datasets. The name contains the terms "economics" and "business" which are discovered as concepts by Babelfy and tagged with the BUSINESS ECONOMICS AND FINANCE label. A much more appropriate category for these datasets would be EDUCATION which is at least the second most frequent (28%) category assigned by our algorithm; However, the frequency was not high enough to allocate the portal to its "correct" cluster, namely cluster 6 with a focus on education. This goes to show much influence single concepts can have on the categorization process.



(a) Distribution of the frequency of GEOGRAPHY AND PLACES per portal.



(b) Distribution of the frequency of POLITICS AND GOVERNMENT per portal.

The second biggest cluster is cluster 1 with a focus on GEOGRAPHY AND PLACES. As mentioned before this is also the biggest category at the dataset level. Reviewing the distribution of the frequency of this category per portal in Figure 34a we see that most portals have at least a few percent of datasets labelled with that category. The mean is 17%, i.e. on average each portal has 17% of datasets in that category. Most concept types that represent some geographical region, e.g. cities, are labelled with GEOGRAPHY AND PLACES. In addition, concepts like ⟨*Municipality*⟩ and ⟨*Area*⟩ are very frequent and belong to this category as well. That said, this label is somewhat general, i.e. it does not convey a lot of information about the content of a dataset other than that the data is specific to a region. However, this confirms our

assumption that a lot of Open Data is associated to a corresponding geographical area.
Although not as much as GEOGRAPHY AND PLACES another frequent category is POLITICS AND GOVERNMENT. Figure 34b shows its distribution. The mean here is at 9.7%. The majority of portals have at least 5% of datasets in that category which is expected since Open Government Data is a significant part of Open Data. The most frequent concept types of this category are ⟨*Petition*⟩, ⟨*Governance*⟩ and ⟨*Election*⟩.
It is noticeable that some of the categories such as MUSIC, TEXTILE AND CLOTHING or FOOD AND DRINK are very infrequent. This is an indication that such contexts are currently not covered by Open Data, at least for the portals in this study. Confirming this assumption for all existing portals will remain future work.

7.2.4 Evaluation of Precision

Since no reference dataset for Open Data exists it is very hard to properly verify these results. Manual investigation indicates a reasonable level of precision. Moreover, the distribution of categories seems very plausible. Still, at this point, we are merely able to formulate educated guesses about the content and structure of the analysed open datasets. Nevertheless, we tried to evaluate our approach with means that are in the scope of a master thesis: We drew up a questionnaire asking participants to assign up to three categories to datasets based on the title, description and keywords. Naturally, the set of possible categories was the same as the one used throughout this study. No information about the categories other than the labels was provided. The 100 datasets in the questionnaire were selected randomly from 5 different portals. In total 8 participants with basic knowledge in Open Data took part in this evaluation. Each dataset was categorized by two participants.
To calculate the precision, we took the union of the categories for each dataset and compared this set to the results of our approach. Using the parameters discussed above we reached a precision of 59.6% and a recall of 27.8%. The low recall stems from the rather strict minimum category score which leads to a single category label for most datasets. In contrast, participants mostly chose different sets of categories for any dataset. In fact, only for 17 out of the 100 datasets two participants chose the exact same set of categories. This goes to show how subjective such a categorization can be, thus, rendering evaluation of an automatic approach difficult.
Decreasing the minimum category score to 0.06 lead to a recall increase to 41.4% while decreasing precision to 50%. Of course, the size and type of this evaluation does not permit any definite conclusions about our approach,

however, it indicates that it is at least reasonable and valid. The evaluation set and results can be reviewed online[30].

8 Conclusion and Future Work

Open Data is a growing movement dedicated to making various kinds of data available to the public. Multiple countries have launched Open Data initiatives leading to serval institutions providing access to their data via Open Data portals. Since all of these portals have their own web pages, i.e. points of access the entirety of currently available Open Data is rather scattered. The OpenData@WU project has harvested the metadata of over 260 portals making the study of Open Data as a whole possible.
In this study, we applied a state-of-the-art entity recognition technique to the natural language descriptions of open datasets. We utilized the concept-category mapping of BabelNet to develop a heavily parameterized categorization algorithm. To be able to test various parameters values we developed a prototype that provides a graphical user interface for the algorithm and means to explore detected concepts and categorization results. Yet, finding the optimal set of parameters turned out to be a very difficult task and remains future work. Nevertheless, we conducted an analysis of over 170.000 datasets from 150 Open Data portals using a set of parameters that yielded satisfactory results.
We found the most frequent categories to be Geography and Places, Computing, Politics and Government and Business Economics and Finance. While the high amount of datasets in Computing is attributable to two larger portals providing primarily statistical research data the other three categories occur frequently in most Open Data portals. Since most concepts representing cities and regions are labelled with Geography and Places our findings confirm the assumption that Open Data is often specific to a geographical region. BabelNet offers further data such as the geolocation for many of these concepts. An interesting aspect for future work would be utilizing this information to analyse the geographical distribution of Open Data.
Employing clustering techniques on the categorization results at the portal level we were able to find groups of portals with different contextual focuses. The largest cluster containing over half of the portals showed a rather balanced category distribution which indicates high selectivity of the category labels derived by our approach. The other clusters show clear emphasis on a single category. Here, the most frequent categories were Geography and

[30] `https://goo.gl/xJJ1UK`

Places, Politics and Government and Business Economics and Finance. While these results may not be surprising the complete lack of presence of other categories is noteworthy. Our findings indicate that, at this time, Open Data is contextually limited to a few key topics. Data associated to themes like music, food or sports seem to not have found their way into Open Data yet.
Other than the categorization approach presented in this thesis the concepts detected by Babelfy open up a variety of possibilities to analyse Open Data. Ioana Hulpus et al presented an interesting graph-based technique to achieve topic labelling. Applying their approach to the concepts we discovered in open datasets could provide more nuanced insights into the content structure of Open Data. Moreover, leveraging the concepts for search, e.g. building a facet hierarchy, is an interesting aspect for further research.
Overall, this study, and the presented categorization of open datasets, constitutes a first step in the analysis of the content and structure of Open Data, an open, interesting and timely challenge which deserves further research and efforts.

References

[1] Spam oder nicht spam? *c't*, 17:150–153, 2003.

[2] G8 open data charter and technical annex. URL: `https://www.gov.uk/government/publications/open-data-charter/g8-open-data-charter-and-technical-annex`, 2013. [Online; accessed 06-August-2017].

[3] The open definition. URL: `http://opendefinition.org/`, 2016. [Online; accessed 25-July-2017].

[4] Paul Buitelaar Anja Jentzsch Andrejs Abele, John P. McCrae and Richard Cyganiak. The linking open data cloud diagram. URL: `http://lod-cloud.net/`, 2017. [Online; accessed 25-July-2017].

[5] Andrew Wood Antoine Isaac, Pierre-Antoine Champin and Sandro Hawke. Rdf 1.1 primer. URL: `https://www.w3.org/tr/2014/note-rdf11-primer-20140624/`, 2014. [Online; accessed 30-July-2017].

[6] Chidanand Apté, Fred J. Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Trans. Inf. Syst.*, 12:233–251, 1994.

[7] Alan R. Aronson and François-Michel Lang. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association : JAMIA*, 17 3:229–36, 2010.

[8] Soeren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.

[9] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. Modern information retrieval. 1999.

[10] Tim Berners-Lee. Linked data. URL: `https://www.w3.org/designissues/linkeddata.html`, 2006. [Online; accessed 08-August-2016].

[11] David C. Blair and M. E. Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the ACM*, (3), March 1985.

[12] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[13] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Advanced Lectures on Machine Learning*, 2003.

[14] Daniel Büring. *Binding Theory*. Cambridge University Press, 2005.

[15] Jose Camacho-Collados and Roberto Navigli. Babeldomains: Large-scale domain labeling of lexical resources. 2017.

[16] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[17] Richard Cyganiak. Data catalog vocabulary. URL: `https://www.w3.org/TR/vocab-dcat/`, 2014. [Online; accessed 25-July-2017].

[18] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41:391–407, 1990.

[19] Paul McNamee Delip Rao and Mark Dredze. *Entity Linking: Finding Extracted Entities in a Knowledge Base*, pages 93–115. Springer Berlin Heidelberg, 2013.

[20] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[21] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, 2010.

[22] Leo Ferres, Gitte Lindgaard, Livia Sumegi, and Bruce Tsuji. Evaluating a tool for improving accessibility to charts and graphs. *ACM Trans. Comput.-Hum. Interact.*, 20:28:1–28:32, 2010.

[23] World Wide Web Foundation. Open data barometer global report. URL: `http://opendatabarometer.org/`, 2015. [Online; accessed 25-July-2017].

[24] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, March 2009.

[25] William A. Gale, Kenneth Ward Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439, 1992.

[26] M. R. Garey, David S. Johnson, and Hans S. Witsenhausen. The complexity of the generalized lloyd - max problem. *IEEE Trans. Information Theory*, 28:255–256, 1982.

[27] Stephen Guo, Ming-Wei Chang, and Emre Kiciman. To link or not to link? a study on end-to-end tweet entity linking. In *HLT-NAACL*, 2013.

[28] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In *CIKM*, 2002.

[29] Richard Hammell. Driving growth, ingenuity and innovation. Technical report, Deloitte LLP, 2012.

[30] Xianpei Han and Jun Zhao. Nlpr_kbp in tac 2009 kbp track: A two-stage method to entity linking. In *In Proceedings of Test Analysis Conference 2009 (TAC 09)*. MIT Press, 1999.

[31] Donna Harman and Mark Liberman. Tipster complete ldc93t3a. URL: `https://catalog.ldc.upenn.edu/LDC93T3A`, 1993. [Online; accessed 25-July-2017].

[32] William R. Hersh, Chris Buckley, T. J. Leone, and David H. Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR*, 1994.

[33] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sanz, and Francisco Carrero García. Content based sms spam filtering. In *ACM Symposium on Document Engineering*, 2006.

[34] Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. An eigenvalue-based measure for word-sense disambiguation. In *FLAIRS Conference*, 2012.

[35] Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. Unsupervised graph-based topic labelling using dbpedia. In *WSDM*, 2013.

[36] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31:264–323, 1999.

[37] Marijn Janssen, Yannis Charalabidis, and Anneke Zuiderwijk. Benefits, adoption barriers and myths of open data and open government. *IS Management*, 29:258–268, 2012.

[38] Heng Ji, Ralph Grishman, and Trang Dang. Overview of the tac2011 knowledge base population track. 2012.

[39] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML*, 1998.

[40] Sebastian Neumaier Jürgen Umbrich and Axel Polleres. *Quality assessment & evolution of Open Data portals*. In IEEE International Conference on Open and Big Data, Rome, Italy, August 2015, Rome, Italy, 2015.

[41] Wai Lam, Miguel E. Ruiz, and Padmini Srinivasan. Automatic text categorization and its application to text retrieval. *IEEE Trans. Knowl. Data Eng.*, 11:865–879, 1999.

[42] Douglas Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38:33–38, 1995.

[43] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Information Theory*, 28:129–136, 1982.

[44] H J Lowe, I Antipov, W Hersh, C A Smith, and M Mailhot. Automated semantic indexing of imaging reports to support retrieval of medical images in the multimedia electronic medical record. *Methods of information in medicine*, 38 4-5:303–7, 1999.

[45] John C. Mallery. Thinking about foreign policy: Finding an appropriate role for artificially intelligent computers. In *Master's thesis, M.I.T. Political Science Department*, 1988.

[46] Jianchang Mao and Anil K. Jain. A self-organizing network for hyperellipsoidal clustering (hec). *IEEE transactions on neural networks*, 7 1:16–29, 1996.

[47] Steve McConnell. Code complete - a practical handbook of software construction, 2nd edition. 2004.

[48] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. 1993.

[49] E.M. Mirkes. K-means and k-menoids applet. [Online; accessed 20-June-2017].

[50] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2:231–244, 2014.

[51] Shuyo Nakatani. Language detection library for java, 2010.

[52] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41:10:1–10:69, 2009.

[53] Roberto Navigli, David Jurgens, and Daniele Vannella. Semeval-2013 task 12: Multilingual word sense disambiguation. In *SemEval@NAACL-HLT*, 2013.

[54] Roberto Navigli and Simone P. Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, December 2012.

[55] Steven T Piantadosi. Zipf's word frequency law in natural language: a critical review and future directions. *Psychonomic bulletin & review*, 21 5:1112–30, 2014.

[56] K. Reghunath. Real-time intrusion detection system for big data, 2017.

[57] Ehud Reiter. *The Handbook of Computational Linguistics and Natural Language Processing*, chapter Natural Language Generation. Wiley-Blackwell, 2010.

[58] Tony Rose, Mark Stevenson, and Miles Whitehead. The reuters corpus volume 1 -from yesterday's news to tomorrow's language resources. In *LREC*, 2002.

[59] Hinrich Schuetze. Automatic word sense discrimination. *Computational Linguistics*, 1998.

[60] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, 2002.

[61] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27:443–460, 2015.

[62] Somayajulu G. Sripada and Ian Davy. Sumtime-mousam: Configurable marine weather forecast generator. 2003.

[63] Andy Seaborne Steve Harris and Eric Prud'hommeaux. Sparql 1.1 query language. URL: `https://www.w3.org/TR/sparql11-query/`, 2013. [Online; accessed 02-August-2017].

[64] Mark Stevenson and Yorick Woliks. *Word-sense Disambiguation*, pages 249–262. Oxford Handbooks, 2005.

[65] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge unifying wordnet and wikipedia. 2007.

[66] Basu S. Micchelli C. Vandewalle J. Suykens J.A.K., Horvath G. *Advances in Learning Theory: Methods, Models and Applications*. IOS Press, 2003.

[67] M. Swamiraj and L. Freund. *Facilitating the discovery of open government datasets through an exploratory data search interface*. 2015 Open Data Research Symposium, Ottawa, Canada, 2015.

[68] A. M. Turing. Computing machinery and intelligence. 1950.

[69] S. van Dongen and Stijn van Dongen. A cluster algorithm for graphs. 2001.

[70] Mark Wasson. Large-scale controlled vocabulary indexing for named entities. In *ANLP*, 2000.

[71] Liyang Yu. *A Developer's Guide to the Semantic Web*. Springer, 2010.

[72] Wei Zhang, Yan Chuan Sim, Jian Su, and Chew Lim Tan. Entity linking with effective acronym expansion, instance selection, and topic modeling. In *IJCAI*, 2011.

[73] Wei Zhang, Chew Lim Tan, Yan Chuan Sim, and Jian Su. Nus-i2r: Learning a combined system for entity linking. In *TAC*, 2010.

A List of Portals

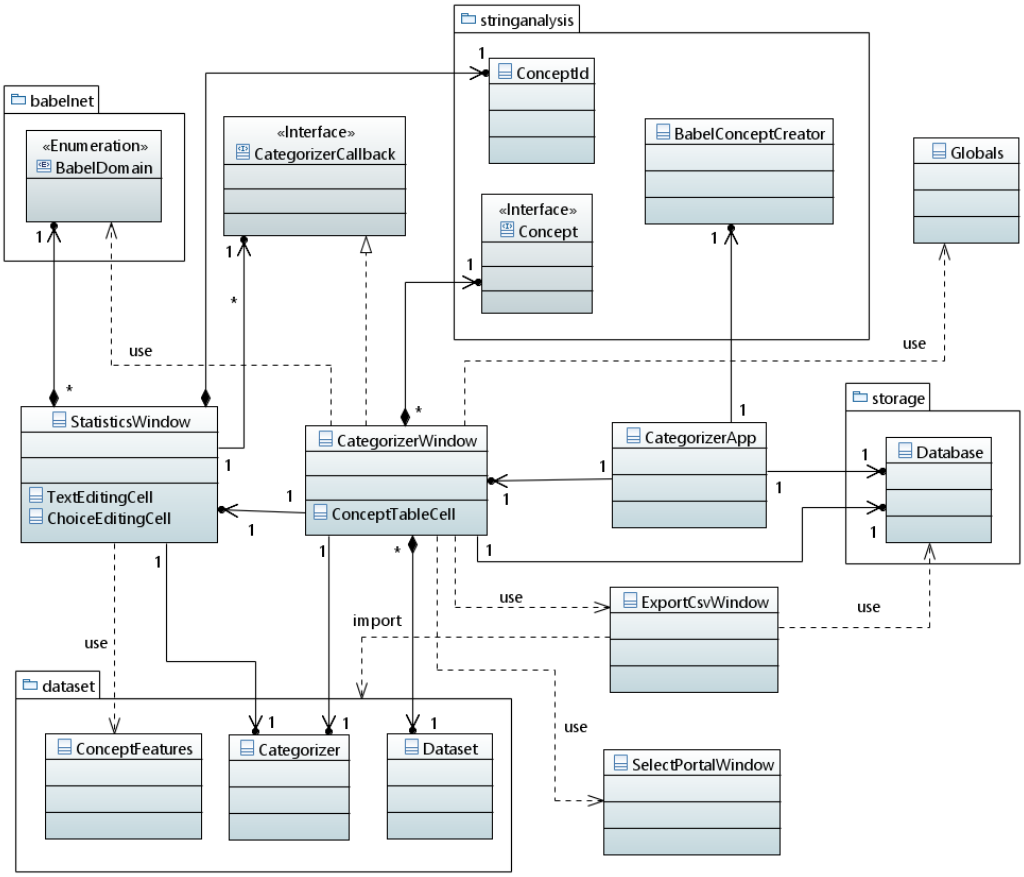
bristol.azure-westeurope-prod.socrata.com
gisdata.mn.gov
data.nj.gov
data.providenceri.gov
www.yorkopendata.org
ckan.odp.jig.jp
opendata.socrata.com
data.oaklandnet.com
data.vermont.gov
data.kingcounty.gov
data.openva.com
www.civicdata.io
dados.recife.pe.gov.br
opendata.rubi.cat
data.somervillema.gov
data.gov.sk
performance.smcgov.org
data.mo.gov
opendata.hu
datos.gob.es
data.nsw.gov.au
dati.lazio.it
gobiernoabierto.valencia.es
data.openpolice.ru
datamx.io
opendata.opennorth.se
data.wu.ac.at
data.gv.at
data.nola.gov
data.redmond.gov
data.illinois.gov.champaign
rdw.azure-westeurope-prod.socrata.com
hampton.demo.socrata.com
opendata.go.tz
www.opengov-muenchen.de
data.cityofnewyork.us
data.gov.hk.en

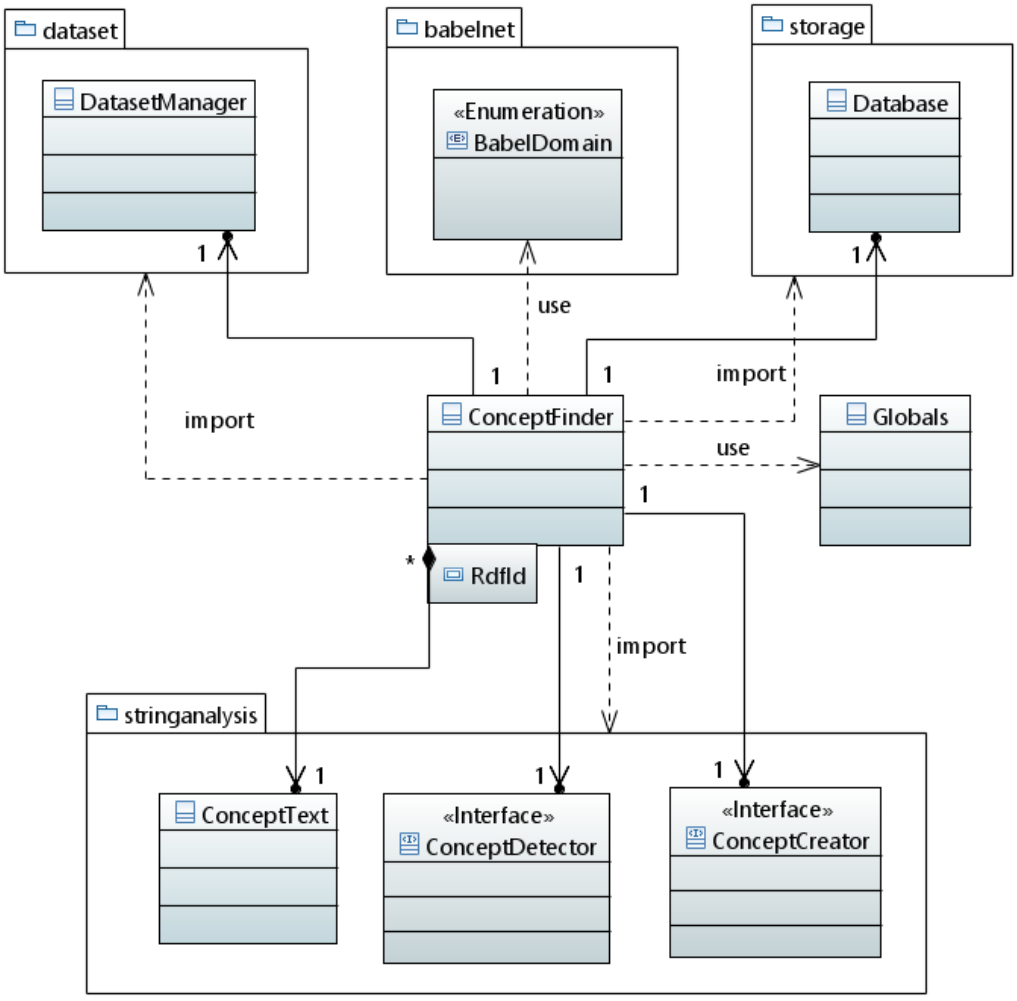
data.kcmo.org
data.zagreb.hr
datacatalog.cookcountyil.gov
donnees.ville.montreal.qc.ca
controllerdata.lacity.org
data.illinois.gov.belleville
www.opendataforum.info
www.odaa.dk
www.dallasopendata.com
www.data.vic.gov.au
data.ohouston.org
dati.trentino.it
opendata.ayto-caceres.es
data.kk.dk
healthdata.nj.gov
data.edmonton.ca
data.stadt-zuerich.ch
stat.cityofgainesville.org
data.hawaii.gov
data.dcpcsb.org
opendata.awt.be
data.hartford.gov
data.baltimorecity.gov
data.gov.bf
www.opendataportal.at
opendata.aragon.es
data.cityofboston.gov
data.surrey.ca
opendata.government.bg
bythenumbers.sco.ca.gov
data.medicare.gov
finances.worldbank.org
data.act.gov.au
130.179.67.140
www.rotterdamopendata.nl
data.qld.gov.au
data.graz.gv.at
data.gov.hr

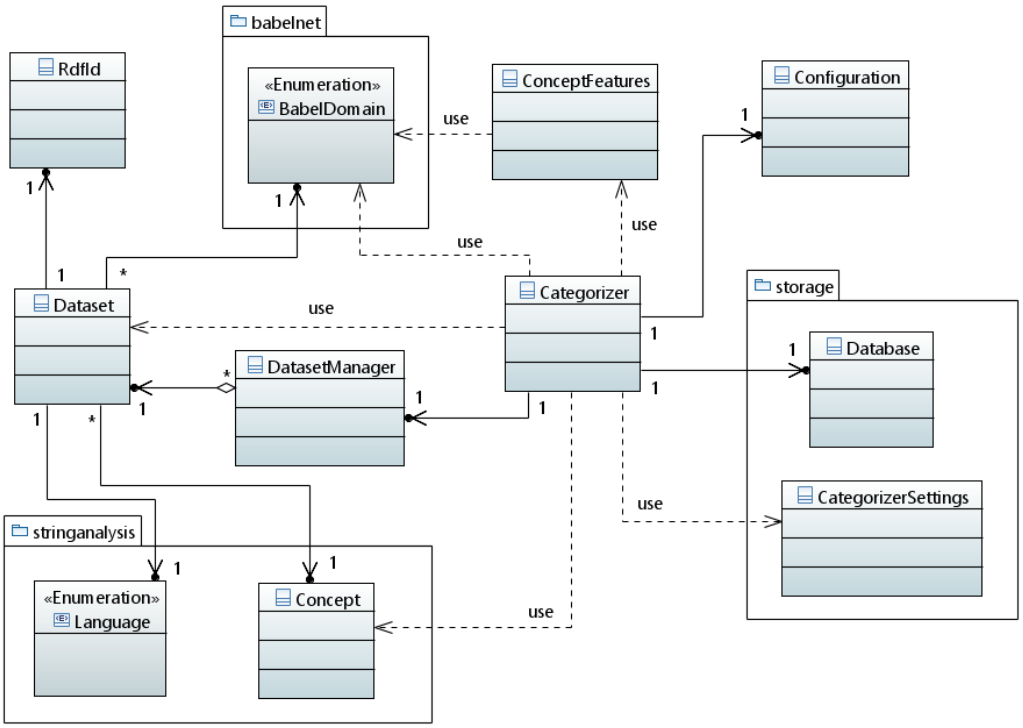
data.sa.gov.au
nycopendata.socrata.com
data.oregon.gov
performance.westsussex.gov.uk
performance.chattanooga.gov
opendata.bayern.de
data.overheid.nl
www.opendataphilly.org
datos.codeandomexico.org
drdsi.jrc.ec.europa.eu
dati.toscana.it
annuario.comune.fi.it
data.sfgov.org
data.ct.gov
data.grcity.us
catalogodatos.gub.uy
datosabiertos.malaga.eu
datameti.go.jp.data
dados.rs.gov.br
data.ug
datos.alcobendas.org
data.acgov.org
nats.demo.socrata.com.login
opendata.comune.bari.it
www.edinburghopendata.info
data.austintexas.gov
data.wa.gov
data.gov.ie
dati.lombardia.it
data.illinois.gov
beta.avoindata.fi
data.honolulu.gov
portal.openbelgium.be
www.dati.friuliveneziagiulia.it
linkeddatacatalog.dws.informatik.uni-mannheim.de
opendata.caceres.es
data.maryland.gov
data.upf.edu.en.main

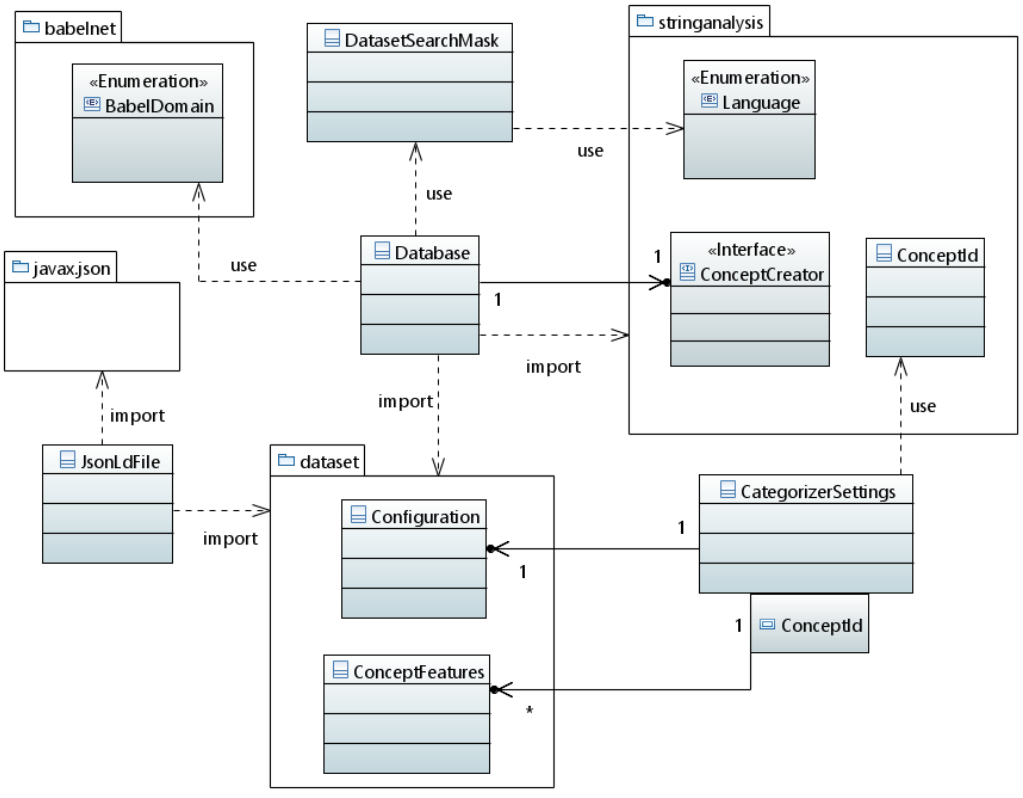
ckan.okfn.gr
dados.al.gov.br
data.bris.ac.uk.data
data.seattle.gov
africaopendata.org
www.opendata-hro.de
westsacramento.demo.socrata.com
data.winnipeg.ca
opendata.lasvegasnevada.gov
data.energystar.gov
data.datamontana.us
opendatahub.gr
www.nosdonnees.fr
data.montgomerycountymd.gov
dados.gov.br
www.hri.fi
opencolorado.org
datosabiertos.ec
rs.ckan.net
data.nhm.ac.uk
health.data.ny.gov
data.gov.ro
dati.veneto.it
daten.buergernetz.bz.it.de
danepubliczne.gov.pl
data.cityoftacoma.org
oppnadata.se
gavaobert.gavaciutat.cat
data.illinois.gov.rockford
data.culvercity.org
data.raleighnc.gov
tourisme62.opendatasoft.com
opendata.terrassa.cat
bronx.lehman.cuny.edu
data.burlingtonvt.gov
datos.gob.mx
data.michigan.gov

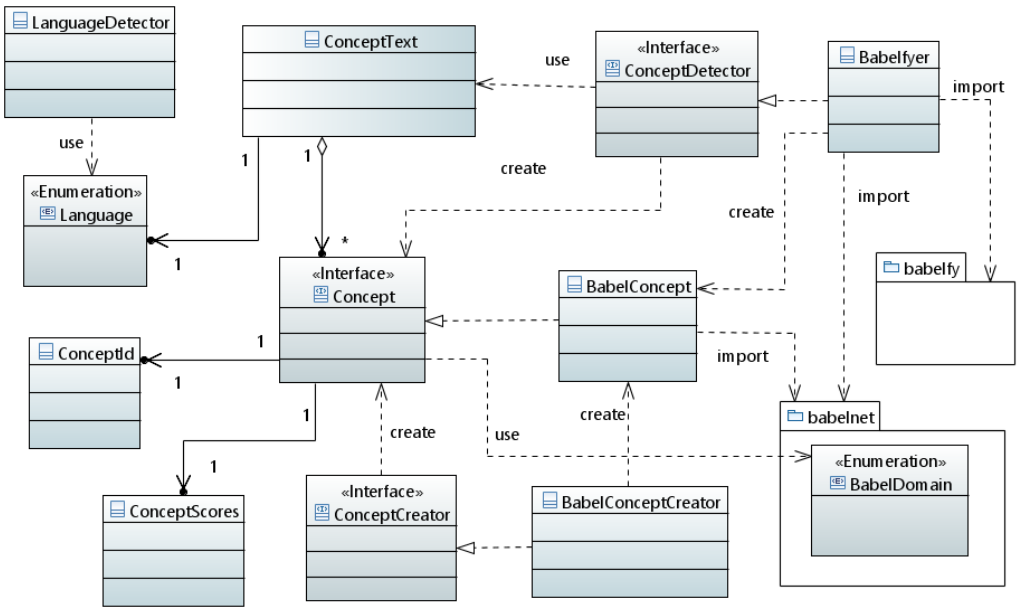
B Class Diagrams




dataset
DatasetManager
babelnet
«Enumeration»
BabelDomain
storage
Database
1
use
1
1
import
import
ConceptFinder
Globals
use
1
*
RdfId
1
import
stringanalysis
1
1
1
ConceptText
«Interface»
ConceptDetector
«Interface»
ConceptCreator


babelnet
RdfId
«Enumeration»
BabelDomain
ConceptFeatures
Configuration
use
1
*
Dataset
Categorizer
storage
DatasetManager
Database
CategorizerSettings
stringanalysis
«Enumeration»
Language
Concept


babelnet
«Enumeration»
BabelDomain
DatasetSearchMask
stringanalysis
«Enumeration»
Language
use
javax.json
Database
«Interface»
ConceptCreator
ConceptId
import
JsonLdFile
dataset
Configuration
CategorizerSettings
ConceptFeatures


LanguageDetector
ConceptText
use
«Interface»
ConceptDetector
Babelfyer
import
use
1
1
create
«Enumeration»
Language
1
*
create
import
«Interface»
Concept
BabelConcept
babelfy
ConceptId
1
1
import
1
create
use
create
babelnet
1
«Enumeration»
BabelDomain
ConceptScores
«Interface»
ConceptCreator
BabelConceptCreator