Universität Zürich[UZH]

Bachelor's thesis
for the degree of
Bachelor of Arts
presented to the Faculty of Arts and Social Sciences
of the University of Zurich

Taxonomy Learning without Labeled Data

Building on TaxoGen

Author: Janis Goldzycher
Student ID: 13-926-357

Examiner: Prof. Martin Volk

Supervisor: Nora Hollenstein

Institute of Computational Linguistics

Submission date: 10.06.2019

Abstract

Taxonomy learning is of great interest for automated knowledge acquisition since Taxonomies not only are a popular way to represent knowledge, but they also enable deductive reasoning and constitute an important step for ontology learning. Taxonomies are made of hypernym relations. Most current methods need labeled data to extract hypernym-relations. TaxoGen is a method for unsupervised learning of topical taxonomies using distributional semantics and a recursive, adaptive clustering process. In this thesis I reimplement TaxoGen, test it with different embedding and clustering techniques, and introduce a new label score.

Zusammenfassung

Das algorithmische Lernen von Taxonomien ist ein zentrales Thema für automatische Wissensanreicherung, weil Taxonomien nicht nur eine leicht zugängliche Art der Wissensrepräsentation darstellen, sondern auch weil sie deduktives Schliessen ermöglichen. Taxonomien bestehen aus Hypernymrelationen. Die meisten aktuellen Methoden benötigen annotierte Daten, um solche Hypernymrelationen zu extrahieren. TaxoGen ist eine unsupervisierte Methode für das Erlernen von Thementaxonomien basierend auf distributioneller Semantik und einem anpassungsfähigen, rekursiven Clusteringprozess. In dieser Arbeit reimplementiere ich TaxoGen, teste die Methode mit verschiedenen Embeddings und Clusteringtechniken und führe einen neuen Labelscore ein.

Acknowledgement

I want to thank my supervisor Nora Hollenstein for her continuous support, for her patience and for getting me interested in the topic of ontology learning and taxonomy learning through her lectures. I would like to thank Martin Volk for his support when I was not sure if this thesis could be finished, Simon Clematide for rescuing me when I thought the server setup would never work and Jochen Leidner for the interesting and illuminating discussion on topical taxonomies. Many thanks to Max Lauber, Nils Mollenhauer and Aneta Zuber for proof reading.

List of Acronyms

CS	Computer Science
DBLP	dblp computer science bibliography
MWE	Multiword Expression
NLP	Natural Language Processing
NP	Noun Phrase
PMI	Pointwise Mutual Information
PPMI	Positive Pointwise Mutual Information
POS	Part-Of-Speech
RNN	Recurrent Neural Network
SPC	Signal Processing Corpus
SVD	Singular Value Decomposition
SVM	Support Vector Machine
UTF-8	Unicode Transformation Format (8-bit)

Contents

Abstract i

Acknowledgement ii

List of Acronyms iii

1 Introduction 1

2 Theoretical background and related work 3

2.1 Pattern-based Hypernym Extraction 4

2.2 Embedding-based Hypernym Extraction 6

2.3 Taxonomy Construction 8

3 TaxoGen 11

3.1 Adaptive Spherical Clustering 11

3.2 Representativeness Score 12

3.3 Local Embeddings 14

3.4 TaxoGen Implementation 15

3.5 Summary and Limits 18

4 Methods - Building on TaxoGen 20

4.1 Embeddings 20

4.1.1 GloVe Embeddings 20

4.1.2 ELMo Embeddings 20

4.1.3 Why Test these Embeddings 21

4.2 Clustering 22

4.3 Label Score 23

4.3.1 Distributional Inclusion Score 23

4.3.2 Hyponym Classification Score 24

5 Experiments and Results 26

5.1 Dataset 26

5.2 Implementation 26
5.3 Evaluation 28
5.3.1 General Procedure 28
5.3.2 Metrics 28
5.3.3 Evaluation Results 30

6 Discussion **34**
6.1 Comparison with the TaxoGen Results 34
6.2 Comparison between Different Versions 35
6.3 About the Evaluation of Topical Taxonomies 35
6.4 Future Work 37

7 Conclusion **39**

References **40**

1 Introduction

Traditionally, knowledge has been stored as written text or, even earlier, has been passed on in oral form. Today, we are looking for and developing ways to store knowledge in a machine-readable way. Knowledge stored as machine-readable data is usually called structured data. In contrast, text, images, video and other media contain information which is not in the same way accessible to machines and is called unstructured data.[1] *Taxonomies* are a way to store a specific kind of structured data, namely hierarchical data. In the field of computer science and computational linguistics the term *taxonomy* mostly refers to domain taxonomies, which only store knowledge about a particular domain. Taxonomies specifically facilitates deductive reasoning which makes them useful for many downstream tasks and applications including search engines, question answering systems, expert systems, virtual assistants and different kinds of autonomous agents. Lastly, taxonomies constitute the backbone on which ontologies are built. Ontologies provide downstream tasks like the ones listed above with even more knowledge. *Taxonomy learning* refers to techniques that automatically extract, learn, or induce a taxonomy from large corpora of raw text. It can be seen as a specific mapping of unstructured to structured data. Taxonomy learning thereby makes the extracted knowledge accessible to all kinds of software systems that can then further process and use this knowledge for downstream tasks.

In this thesis I focus on TaxoGen, a method to learn topical taxonomies in an unsupervised fashion. A topical taxonomy is a kind of taxonomy which contains hierarchical relations between entire topics, not just between concepts or words. TaxoGen mainly relies on clustering, local embeddings and a score to measure how representative a term is for a topic. I reimplement and advance the method in this thesis. Specifically, my contributions are as follows:

1. I reimplement TaxoGen to reproduce the results reported in its publication and formalize parts of the original TaxoGen implementation that are not described in its publication or differ from the descriptions in the publication.

[1] Of course, these types of data all have rich structures too. Their structures are just, for now, not easily accessible to computers.

2. I motivate and implement the usage of new embedding and clustering algorithms for TaxoGen.

3. I propose a label score for the ranking of terms within topics to enhance the coherency of the taxonomy.

4. I conduct extensive testing of the different TaxoGen configurations proposed in this thesis with several metrics and discuss the usefulness of these metrics.

The thesis is structured as follows: In Chapter 2 I briefly explain the necessary linguistic background and then review the existing rule-based and machine learning-based approaches for hypernym extraction and taxonomy construction. The Chapter 3 is devoted to explaining TaxoGen and its implementation in depth and discuss its current limits. In Chapter 4, I describe my own methods and contributions to enhance TaxoGen. In Chapter 5, I describe the experimental setup, the evaluation procedure and the results. In Chapter 6, I discuss the results, the evaluation metrics used and I sketch ideas for further developments. Finally, I conclude the thesis with Chapter 7.

2 Theoretical background and related work

Taxonomies are a hierarchical division of things into categories and subcategories in a tree-like manner. For information technology, taxonomies and ontologies are ways to store and represent knowledge in a structured, machine readable way. They only represent hypernym relations, also called is_a-relation. To understand what is meant by such a relation consider the sentence: `Red is a color.` We can describe this relationship between `red` and `color` as `is_a(red, color)`. In this hypernym relationship, `color` is *the* hypernym of `red` and `red` is *a* hyponym of `color`. Note that a hypernym can have multiple hyponyms, but a hyponym can only have one (direct) hypernym. From here on, I will only use the term *hypernym relation* to denote such a relation. In CS and NLP taxonomies are often represented as directed acyclic graphs, where each node stands for a concept or word and each edge stands for a hypernym relation. In the case of topical taxonomies, a hierarchical organization of topics, each node stands for a topic instead of a concept or word.

Taxonomy learning generally consists of two main steps: Hypernym extraction and taxonomy construction. Hypernym extraction is the process of extracting all tuples of terms `(a, b)` for which hold `is_a(a, b)` from a given term set [Buitelaar et al., 2005]. Taxonomy construction uses these extracted relations to organize the terms into a taxonomy. Some methods assume a given set of terms, other methods start only with the raw text [1]. The latter methods need an additional prior step to identify relevant terms, which is called term extraction. In clustering-based methods, as we will later see, hypernym extraction and taxonomy construction fall together into one step.

Approaches for hypernym extraction can be divided into pattern-based approaches and approaches based on distributed word representations. Furthermore pattern-based approaches can be divided into rule-based approaches and statistical ap-

[1] Methods assuming a given set of terms: de Mantaras and Saitia [2004], Yang and Callan [2009], Liu et al. [2012]. Methods starting only with the raw text: Fountain and Lapata [2012], Anh et al. [2014]. This list is not exhaustive and just points to some examples.

proaches where hypernym extraction is typically formulated as a classification problem: *Given two terms, predict a boolean variable indicating if there is a hypernym relation between the two terms.* Some approaches based on distributed word representations use a classification approach too. More recent methods, though, prefer linear projections combined with a nearest neighbor search. Lastly, approaches based on distributed word representations can also use clustering to infer hypernym relations. Methods like classification and linear projections belong to the category of supervised methods, which means that they need labeled training data. Clustering-based methods on the other hand do not need labeled data and hence belong to the class of unsupervised learning methods. TaxoGen, the method I am building on, uses clustering, thus is an unsupervised method. However, as I will describe in Chapter 4, I am expanding TaxoGen to not only make use of clustering- but also classification-based methods. This new method will still not be in need of manually labeled data, since the labels are extracted in an automatic pattern based way.

2.1 Pattern-based Hypernym Extraction

Hearst [1992] proposes the lexico-syntactic patterns now called Hearst Patterns. She collects these using an initial seed of patterns and bootstrapping. All patterns extracted through this process are filtered out if they do not meet the following three conditions:

1. They occur frequently and in many text genres.
2. They (almost) always indicate the relation of interest.
3. They can be recognized with little or no pre-encoded knowledge.

Using these conditions she collects the following 6 patterns, which have gained a lot of attention and are frequently used today:

```
1. NP such as {NP,}* {or|and} NP
2. such NP as {NP,}* {or|an)} NP
3. NP{, NP}*{,} or other NP
4. NP{, NP}*{,} and other NP
5. NP{,} including {NP,}* {(or|and)} NP
6. NP{,} especially {NP,}* {or|and} NP
```

Alfarone and Davis [2015] build on Hearst Patterns by first using these to extract initial hypernym relations. They further search for more hypernym relations by using the NPs extracted through Hearst Patterns and what is known as the string

subsumption method. The string subsumption method can discover hypernym relations encoded in a single NP. Consider the term `machine learning`. We can remove the modifier `machine` such that we get the more general term `learning`. Using this evidence we can infer the hypernym relation `is_a(machine learning, learning)`. String subsumption extracts hypernym relations by isolating the heads of NPs or MWEs from their modifiers. Alfarone and Davis filter all extracted relations based on the frequency of their occurrence to improve precision. They then perform open relation extraction to calculate the similarities between the previously extracted NPs based on the number of their shared relations. Finally using this similarity score to recursively cluster the NPs, they correct wrong relations in the taxonomy and infer new ones.[2]

Snow et al. [2005] use a semi-supervised approach. They collect hypernym relations from WordNet and look for sentences containing both terms of a relation in a corpus. These sentences are assumed to express the hypernym relation between the two terms. For all of these sentences they extract features based on the dependency path between hypernym and hyponym and use it to train a hypernym classifier greatly increasing recall over the before discussed purely rule-based methods.

Navigli and Velardi [2010] automatically extract definitional sentences with a classifier. They then generalize these definitional sentences to star patterns. Consider the following example: `In *, a <TARGET> is a *` Here the first star denotes any domain and the second star denotes the hypernym of `<TARGET>`. The extracted definitional sentences are then grouped into clusters according to the star pattern they belong to. Finally, for each star pattern a Finite State Automaton, which is able to process all sentences in it's group, is constructed. A new tuple of two candidate terms then gets classified as a hypernym relation if one of the Finite State Automata is able to process it.

Grefenstette [2015] trains a hypernym classifier using features based on patterns extracted with the string subsumption method and a number of frequency measures. These include term frequency, document frequency, the frequency of terms occurring in the same sentence and the frequency of terms occurring in the same document. With such a seemingly simple approach he won the SemEval 2015 Task 17 Bordea et al. [2015] where one has to structure a given set of terms into a taxonomy.

[2] I mainly discuss clustering-based methods in section 2.3. This approach appears under the pattern-based methods because the focus here lies on the string subsumption method.

2.2 Embedding-based Hypernym Extraction

A distributed representation of a word, also called word embedding, is a vector of real numbers encoding syntactic and semantic properties of the word. Although the idea for distributed word representations has been known for several decades, it has gained most of its popularity since [Bengio et al., 2003] and even more so since [Mikolov et al., 2013]. Today distributed word representations are a tool so popular, that they are used by most approaches in almost all areas in NLP. I will describe some specific methods to generate such representations later in section 4.1. Most approaches for hypernym extraction today are embedding-based. In the following, I will discuss some of the most important embedding-based approaches.

[Yu et al., 2015] train embeddings specifically for hypernym extraction. They define a hypernym and a hyponym embedding for each term in their vocabulary. The loss function of the neural net generating the embeddings then enforces the following three properties:
Let `x` *be a hypernym of* `y` *and* `z`*. Let* `x1` *and* `x2` *be additional hypernyms.*

1. The hypernym embedding of `x` is similar the to hyponym embedding of `y`.
2. The hyponym embeddings of `y` and `z` are similar.
3. If `x1` and `x2` share hyponyms their hypernym embedding should be similar.

Yu et al. then train a SVM that takes as input the concatenated embeddings of two terms and the distance between them to predict if this tuple of terms is in a hypernym relation.

The approach of Kruszewski et al. [2015] is based on the inclusion hypothesis, according to which a hyponym tends to only appear its hypernyms contexts. Building on this hypothesis they derive boolean vectors, vectors containing 0s and 1s, from word embeddings by mapping all values in a vector that are close to 1 to 1 and all values that are close to 0 to 0. The idea being that if a value in the hyponym's vector is active, thus close 1, the vector of its hypernym must also be active at this place, thus, at that place, there must also be a 1. They then train a SVM that takes the concatenated boolean vectors of two terms as input and outputs a prediction if there is a hypernym relation between them.

During the last years, more and more methods relying on linear projections have been proposed. Constructing a linear projection means abandoning the classification approach and instead reformulating the problem as a regression problem. Given a word vector, find a transition matrix that projects the word to where its hypernym

most likely is in the vector space. After the projection has been performed, find the best match with a nearest neighbour approach using the Euclidean distance. Fu et al. [2014] pioneered this approach by proposing a piecewise linear projection. A hypernym relation between hypernym x and hyponym y is represented by their offset x-y. Based on this representation they cluster all given hypernym relations into groups and then train a linear projection for each of the clusters.

Yamane et al. [2016] further develop this approach by jointly learning the clusters and projections. During training, each new pair of terms first gets assigned to a cluster using a similarity score, that measures how good the current projection matrix of the cluster predicts the hypernym of the new pair, when given the hyponym of the new pair. The cluster with the highest similarity score gets the new training pair and the cluster's projection matrix gets updated. If the similarity score remains below a certain threshold for all the clusters, then a new cluster is created for the pair. While Fu et al. [2014] used no negative samples for the projection learning, Yamane et al. [2016] generate negative samples during each learning process. They define a loss function for projection learning that penalizes projections near the negative samples. Ustalov et al. [2017] further explore negative sampling for projection learning by making use of reversed hypernym relations and synonyms, which are by definition guaranteed to be negative samples.

At the SemEval-2018 Task 9, where goal is to extract hypernym relations, Bernier-Colborne and Barriere [2018] won by combining a pattern-based approach with linear projections. For the pattern-based part they used Hearst patterns and the string subsumption method. To increase the recall they used a new set of patterns designed to identify cohyponyms. Each extracted hypernym relation then got a frequency-based confidence score. For the projection learning they used a fixed number of projections (24) and soft clustering instead of hard clustering, meaning that a relation doesn't belong to exactly one cluster but instead has a distribution over clusters it belongs to. The confidence scores of both of their subsystems are then fed into a weighting function to compute a final confidence score.

There have also been approaches to hypernym extraction that not only embed words but the relation as a whole. Anh et al. [2016] generate a hypernym embedding with the following method: They take examples of hypernym relations from WordNet and extract sentences expressing these relations from the Wikipedia corpus. For each of these sentences they create a triple consisting of the hyponym, the hypernym and the text in-between. The hyponym, hypernym and the words in the text in-between are all encoded with one hot encoding[3]. A neural network gets the hyponym and the

[3]A one hot encoding is a vector of 0s and 1s with the dimension of the input texts length. For a

text in-between as input and has to predict the hypernym. Anh et al. then extract the weights of the hidden layer and use them to embed hypernym relations. In the final setting, a SVM takes as input the embedded hyponym, hypernym and their difference, to account for the context in-between in order to predict if a hypernym relation exists for this tuple.

There has been doubt if classifiers taking distributional word representations as input really learn inference relations or if they just memorize typical hypernyms. Levy et al. [2015] test this hypothesis systematically over several distributed word representations and classifier inputs based thereon. Specifically, they test embeddings using the popular Skip-Gram model as well as PPMI and SVD representations. As classifiers they test a SVM and a logistic regression classifier with 4 different inputs: the concatenation of hypernym and hyponym; the difference between hypernym and hyponym; only the hyponym as input; and only the hypernym as input. The performance of only feeding the hypernym to the classifiers is shown to be almost as good as using both hypernym and hyponym as input for the classifiers. This shows that when using such a setup, the classifier does not look much at the hyponym information and mostly just memorizes hypernyms. Chen et al. [2018] use an approach specifically for finding hypernym relations not mentioned in the text. For a hyponym x and a hypernym y they train an autoencoder taking the input x and producing the output x-y. They avoid learning a mapping directly to the hypernym to increase generalization and by that avoid the above described criticism.

2.3 Taxonomy Construction

If hypernym extraction and taxonomy construction are separate processing steps and thus hypernym relations have already been extracted, then taxonomy construction is typically formulated as a minimum cost flow problem. A graph is constructed by interpreting extracted hypernyms and hyponyms as nodes and linking them with edges. Each edge gets a score indicating the system's confidence in the hypernym relation. Then, a search for the graph with the overall minimum cost is conducted, where low confidence is interpreted as high cost and high confidence is interpreted as low cost. For more detailed descriptions and different variations of this approach see Kumar et al. [2001], Woon and Madnick [2009] and Fountain and Lapata [2012]. Going forward I will focus on methods based on clustering.

Clustering-based methods for taxonomy learning typically merge hypernym extrac-

given word it has 1s at every place or dimension where that word appears.

tion and taxonomy construction into one step. Hierarchical or recursive clustering generates a taxonomy structure containing hierarchical, thus hypernymic, relations without extracting them explicitly in a separate step.

de Mantaras and Saitia [2004] compare conceptual clustering, agglomerative clustering and bi-section-k-means clustering for taxonomy construction. In conceptual clustering a set of attributes is extracted for each term. The task then is to find terms whose attributes are included in the set of attributes of other terms. This implies that the second term is more general and higher up in the taxonomic hierarchy. Hierarchical agglomerative clustering is a bottom-up clustering technique. Every term starts in its own cluster. Then in each step the two most similar clusters are merged. For a more elaborate explanation of hierarchical agglomerative clustering, see section 4.2. In bi-section-k-means clustering, all data points are first clustered using standard k-means. Then the cluster with the highest variance gets selected and is split into two clusters by k-means. This process repeats until the desired number of clusters is reached. It can thus be seen as a form of divisive hierarchical clustering. In the evaluation of de Mantaras and Saitia [2004], formal concept analysis produces the best results but the term representations do not have the quality of today's embeddings. So which clustering algorithm performs best today might be quite different.

Yang and Callan [2009] perform incremental clustering using a wide range of information to generate the feature vector. They use the co-occurrences of words measured by PMI, features based on syntactic dependency paths, features based on lexico-syntactic patterns and other features, even including Google search results. Using the resulting feature vectors Yang and Callan follow an incremental clustering approach. Given a term to start with, the method adds terms one by one to the taxonomy, each time creating a new partial taxonomy and choosing the relation that maximizes two criteria, minimal evolution and abstractiveness, in a joint model. According to the criterion of minimal evolution the best partial taxonomy that includes the new term is the one with minimal added information. The criterion of abstractiveness says that objects on the same taxonomy level have a similar amount of abstractiveness, meaning they occupy a similar place on a spectrum between purely physical and more non-physical objects. Terms on the same level thus need to have similar characteristics. The method finishes when all terms have been added and the partial taxonomy thus becomes a full taxonomy.

Liu et al. [2012] use an agglomerative clustering method called bayesian rose tree. In this method, a tree gets constructed using the three operations `join`, `absorb` and `collapse`. They start out, however, not with a corpus, but with a given set of

terms. To construct a context for each given term, they thus search a knowledge base for relevant concepts and use the top results of search engine queries. Using frequency- and distribution-based measures, a weight score is calculated for each word in the constructed context. These word scores then are used to produce the feature vector, which is turn fed to the clustering algorithm.

3 TaxoGen

This thesis focuses on improving TaxoGen, which also relies on hierarchical clustering. In the following I will describe this method in detail. Zhang et al. [2018] developed TaxoGen to construct topical taxonomies. This means that each node in the taxonomy tree represents a topic instead of a single word. A topic in this context is a set of semantically coherent terms. For a given root node and a given number of clusters per level, TaxoGen generates the taxonomy in a recursive top down process. It stops when either no unassigned terms are left or when a maximum depth, specified beforehand, is reached. The algorithm is made up of two main components: adaptive spherical clustering and local embeddings. In the following I will describe these two main components and how they work together.

3.1 Adaptive Spherical Clustering

For a given topic C, the goal is to divide it into k semantically coherent subtopics S_1 to S_k. The k-means algorithm from Hartigan and Wong [1979] provides an easy way to accomplish this. For a given set of data points and the number of clusters k it divides the data points x_1 to x_n into k clusters[1] such that the sum of the cluster's variances is minimized. This is equivalent to minimizing the sum of squared Euclidean distances from the cluster center. A cluster center μ is defined as the arithmetic mean of all data points belonging that cluster. k-means thus implicitly measures distances of data points with a Euclidean metric. To quantify semantic similarities in the embedding space, however, the cosine distance has proven to be much more useful than the Euclidean distance for NLP tasks. Zhang et al. thus choose *spherical k-means* clustering introduced by Dhillon and Modha [2001] which relies on cosine similarities instead of Euclidean distances.

[1] In the context of the TaxoGen-method a cluster represents a subtopic. Therefore, I will treat the two words interchangeably.

Formally we can define the objective function of the k-means algorithm as

$$\arg\min \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3.1}$$

where $\|x - \mu\|^2$ denotes the squared Euclidean distance, also called L2-norm, for the data point x and the cluster center μ. Accordingly the objective function of the spherical k-means algorithm is defined as

$$\arg\min \sum_{i=1}^{k} \sum_{x \in S_i} \text{cos-dist}(x, S_i) \tag{3.2}$$

where cos-dist(x, S) denotes the cosine distance between the data point x and the mean direction of the cluster S. The mathematical notation here shows us how the objective functions of the two algorithms only differ in what metric they use to calculate distances.

Not all terms in C necessarily belong to a subtopic. Some terms probably belong to the topic C itself. Thus, in TaxoGen, terms are assigned a representativeness score for the subtopic they were clustered into. Those terms which are not representative of the topic are assumed to be too general and are *pushed up* to the parent topic. That is how the topic C and every other topic gets its members: not by getting terms assigned directly but by child topics pushing up terms to the parent. Pushing up such general terms has the additional benefit of clearing the boundaries between the subtopics since such terms might fall in-between different topics and thereby blur the topic boundaries. Thus, the *adaptive* part of the *adaptive spherical clustering* specifies the iterative process in which a set of terms is clustered into subtopics, followed by pushing up general terms. The clustering and pushing-up process then starts again with the non-general (and thus, not-pushed up) terms. This is repeated until no more general terms are found. Iterative clustering and pushing up general terms is a self-reinforcing process. Clearer topic boundaries lead to a better partitioning of the topics, which increases the quality of the topic representativeness scores, which in turn again leads to clearer boundaries by pushing up non-representative terms.

3.2 Representativeness Score

The central problem now becomes how to measure the representativeness of a term t for a subtopic S_k. Zhang et al. assume that a term, representative of S_k, appears frequently in S_k but rarely in its sibling topics $S \backslash \{S_k\}$. Based on this assumption

they consider two factors making up representativeness:

Popularity: If t is representative of S_k then it appears frequently in S_k.

Concentration: If t is representative of S_k then it is more relevant to S_k than to its sibling topics $S \backslash \{S_k\}$.

Notice that to describe the relevancy of a term to a document in a document collection there already are scores like TF-IDF and BM25. They additionally assume that these two factors have conjunctive conditions, and thus define representativeness as

$$r(t, S_k) = \sqrt{\text{pop}(t, S_k) * \text{con}(t, S_k)} \tag{3.3}$$

where $r(t, S_k)$ denotes the representativeness of t for S_k, $\text{pop}(t, S_k)$ the popularity of t in S_k and $\text{con}(t, S_k)$ the concentration of t in S_k. To measure these two factors they create a subcorpus for each subtopic. The entire corpus is denoted by the document collection D. A subcorpus that belongs to the subtopic S_k is denoted by D_k accordingly. The membership of a document is determined by summing up all IDF scores per subtopic for each document and then choosing the subtopic for which the document has the highest sum of IDF scores. The popularity of t in S_k, $\text{pop}(t, S_k)$, then is calculated as the smoothed and normalized frequency of t in the documents D_k.

$$\text{pop}(t, S_k) = \frac{\log\left(tf(t, D_k) + 1\right)}{\log tf(D_k)} \tag{3.4}$$

$tf(t, D_k)$ denotes the frequency of t in D_k and $tf(D_k)$ denotes the total number of tokens in D_k. The concentration score of t in S_k is calculated as the terms normalized relevance to D_k.

$$\text{con}(t, S_k) = \frac{\exp\left(\text{rel}(t, D_k)\right)}{1 + \sum_{j=1}^{K} \exp\left(\text{rel}(t, D_j)\right)} \tag{3.5}$$

We interpret all documents in D_k as one concatenated pseudo-document. $\text{rel}(t, D_k)$ then denotes the relevance of term t to the pseudo-document D_k. This enables us to use the already existing relevance scores for terms in documents. Zhang et al. choose the Okapi BM25 score from Robertson et al. [1995]. The score describes the relevance of words in a query vector q to a document D_k. We can view the Okapi BM25 score as a version of TFIDF where the term frequency is additionally normalized to account for different document lengths. It is computed as follows:

$$\text{BM25}(q_i, D_k) = \sum_{i=1}^{n} \text{IDF}(q_i) * \frac{tf(q_i, D_k) * (k_1 + 1)}{tf(q_i, D_k) + k_1 * (1 - b + b * \frac{|D_k|}{\text{avgdl}})} \tag{3.6}$$

q_i is the i-th word of the query vector q. IDF(q_i) is the inverse document frequency of q_i in D, $|D_k|$ is the length of D_k in words and avgdl is the average document length in D. k_1 and b are free parameters.[2] When applying this score to TaxoGen we interpret the query vector q as the term whose relevance for the document D_k we want to calculate. Since in the context of TaxoGen multiword expressions are concatenated to one single term, the query vector only consists of a single term. Thus we can simplify the equation to

$$\text{BM25}_{TaxoGen}(t, D_k) = \text{IDF}(t) * \frac{tf(t, D_k) * (k_1 + 1)}{tf(t, D_k) + k_1 * (1 - b + b * \frac{|D_k|}{\text{avgdl}})} \tag{3.7}$$

where t denotes the term in question.

3.3 Local Embeddings

Zhang et al. use the Skip-Gram algorithm from Mikolov et al. [2013] to compute embeddings for terms. The Skip-Gram model employs a neural network with one hidden layer to predict the most likely context words for a given input word. Specifically, the given word is represented with a one-hot-encoding (a binary array, unique for each word in the vocabulary). For each position in the given context window, which is typically 2 (the two words before and the two words after the given word) the neural network produces a probability distribution. This probability distribution encodes how likely it is for each word of the vocabulary to appear at that position in the context window. During training this context window is slid over each sentence and at each position of the window the network has to predict the context words for the given center word. To get the actual word vectors after the training, the weights between input layer and hidden layer are extracted and the dot product with each word's one hot vector representation is calculated. In short, the embedding vector, a word gets, when using the Skip-Gram algorithm, encodes in which local contexts the word typically occurs.

Just using one globally trained embedding model has the disadvantage that at lower levels of the taxonomy, terms are very close to each other in the vector space and it becomes difficult to further divide term clusters into meaningful subclusters. For this reason Zhang et al. propose to train a local embedding model for each topic or node respectively on a subcorpus extracted for that topic. This helps to get fine grained representations of terms that reflect minor differences in usage from each

[2] k_1 is normally given a value in the interval $[1.2, 2.0]$ and b is usually set to 0.75.

other at the lower levels of the taxonomy. The clustering is then performed on the local embedding model.

The only question left now is how to get the set of documents D_k^e [2] that make up the subcorpus of the topic S_k. Zhang et al. employ two methods: *clustering-based* and *retrieval-based*. For the clustering based method Zhang et al. simply sum up all TFIDF scores of terms in a document per cluster, to compute a cluster score per document. They then assign each document to the cluster, which has the highest score in that document. For the retrieval based method, Zhang et al. compute a document embedding using a TF-IDF weighted average of the document's terms. Let T denote the set of terms $\{t_1...t_n\}$ in the document D_k^e. Then $E(D_k)$, the embedding of document D_k, is calculated as

$$E(D_k^e) = \frac{\sum_{1 \leq j \leq n} \text{TF-IDF}(t_j, D_k^e) * E(t_j)}{n} \tag{3.8}$$

where TF-IDF(t_j, D_k) denotes the TF-IDF-score of t_j for D_k^e and $E(t_j)$ denotes the embedding vector of t_j. This means that the TF-IDF-score is used as a scalar by which the term embeddings is multiplied. They then compute the average direction of each topic by averaging the direction of its cluster members. Finally Zhang et al. collect the top m documents with the most similar direction to the topic's direction. The retrieval-based method is only used when the clustering-based method does not produce enough documents for a cluster. Note two important points: (1) For the extraction of documents belonging to a topic, the global embeddings (not the local embeddings) are used. (2) The search for documents belonging to a topic is always done globally and not only in the corpus of its parent topic.

3.4 TaxoGen Implementation

When reimplementing the TaxoGen-Method, I found that the exact reimplementation of the method described above actually yields results quite different from the ones reported in the paper. Some differences can probably be explained through implementation-specific details. For example, choosing a different implementation of the Skip-Gram algorithm can lead to slightly different embeddings, which influences how the terms are clustered into topics. Different topic clusters lead to different subcorpora extracted for the topics which in turn lead to different representativeness scores. Thus, rather small differences can trigger a snowball effect. However, I

[2] I use the superscript e here to differentiate the subcorpus extracted for the local embedding training from the subcorpus extracted to calculate the representativeness score.

think there is at least some degree of similarity to expect between the results of the different implementations.

Therefore, when seeing these differences in the results, I started looking into the source code of the original TaxoGen implementation and found significant discrepancies between the descriptions in the publication and the implementation. The following is my own interpretation and formalization of those parts I found in the source code which are different from the descriptions in the publication or do not appear in it.

In the implementation it is not the term frequency, that is used to calculate the popularity score. Instead the document frequency is used. Additionally, no normalization is applied to the popularity score as described in the paper. Thus, the popularity for a given term t belonging to a topic S_k is just calculated as

$$\text{pop}(t, S_k) = \log\left(df(t, D_k) + 1\right) \tag{3.9}$$

where $df(t, D_k)$ denotes the document frequency of t in the corpus D_k.

The implementation of the BM25 score also differs from the "textbook defintion". Normally, for the document length, the number of tokens in a document is counted. In the implementation, the number of terms appearing in a document is used instead of the document length. Additionally, Zhang et al. replace the IDF in the BM25 score with a normalization of the document frequency. Specifically, they calculate, what they call in the source code, a df-factor as

$$\text{df-factor}(t, S_k) = \frac{\log(1 + df(t, S_k))}{1 + df_{\max}(t)} \tag{3.10}$$

where $df_{\max}(t)$ denotes the maximum document frequency of t in any of the pseudo documents D_1 to D_k. Finally, to get the BM25 scores into a more useful range they multiply the resulting score by 3. Therefore, the BM25 score is implemented as

$$\text{BM25}_{Imp}(t, D_k) = 3 * \text{df-factor}(t, S_k) * \frac{tf(t, D_k) * (k_1 + 1)}{tf(t, D_k) + k_1 * (1 - b + b * \frac{|D_{k_{\text{terms}}}|}{\text{avgdl}_{\text{terms}}})} \tag{3.11}$$

where $|D_{k_{\text{terms}}}|$ and $\text{avgdl}_{\text{terms}}$ denotes the document length and average document length, where only terms and not all tokens are counted.

When the entire representativeness score has been calculated, it is normalized by dividing the score for term t in topic S_k by the sum of the representativeness scores

for its sibling topics.

$$r_{\text{norm}}(t, S_k) = \frac{r(t, S_k)}{\sum_{j=1}^{K} r(t, S_j)} \tag{3.12}$$

Note that adding this normalization is not equal to leaving out the normalization of the popularity score, since: (a) the normalization of the popularity score in (3.4) is done by dividing by the total number of tokens in D_k whereas the normalization in (3.12) is done by dividing by the sum of representativeness scores for S. (b) This means that the already normalized concentration score is normalized again. This normalized representativeness score r_{norm} is calculated for the topic the term belongs to *and* for all its sibling topics.

As a next step, the Kullback-Leibler divergence is applied. The Kullback-Leibler divergence, also called relative entropy, is a measure expressing how two probability distributions differ from each other. Normally, it is used to compare an empirically measured distribution against a reference distribution, often the a priori expected distribution. For discrete probability distributions, the Kullback-Leibler divergence is defined as follows:

$$D_{\text{KL}}(P\|Q) = \sum_{x \in \chi} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{3.13}$$

P and Q are discrete probability distributions over χ, the set of observations. The $\|$ in this case is simply used to emphasize that the order of the arguments matter, because the Kullback-Leibler divergence is an asymmetric measure. When this measure is applied to compare an empirically measured distribution against an expected distribution, it can be interpreted as the expected sum of logarithmic differences between P and Q.

Zhang et al. use the Kullback-Leibler divergence to measure how much the normalized representativeness scores r_{norm} differ from the scores that would result from a totally random distribution of terms over documents. They define the expected distribution as $Q(x) = \frac{1}{|S|}$ where $|S|$ is the number of clusters, which in this case is always 5. This means that $Q(x)$ is always 0.2. Therefore, for the specific case of TaxoGen, the Kullback-Leibler divergence can be rewritten as

$$D_{\text{KL}}(r_{\text{norm}}\|Q) = \sum_{x \in \chi} r_{\text{norm}}(x) \log \left(\frac{r_{\text{norm}}(x)}{0.2} \right) \tag{3.14}$$

Finally, this resulting divergence is used as a score to decide which terms are general and should be pushed up to the parent topic. Note, that in the paper the criterion to pushing up a term is its representativeness score. In the implementation, the

criterion is how much the representativeness score of a term in a topic positively deviates from its sibling topics.

Zhang et al. describe a retrieval-based method and a clustering-based method in the paper to extract a subcorpus for local embedding training. I could not find an implementation of those methods in the source code. Instead they employed the following method to extract a subcorpus: For the m terms closest to the cluster center (using cosine distance), all documents in which those terms appear are collected.

In the following chapters of the thesis, when I use the term *TaxoGen*, I refer to the implemented version of TaxoGen, not the version described in the publication, since the implemented version is the one that actually produced the results reported.

3.5 Summary and Limits

In summary, TaxoGen is an unsupervised taxonomy learning algorithm. It creates the taxonomy recursively in a top-down manner. On each level it clusters a term set into a given number of topics and extracts a relevant subcorpus for each topic. Terms are scored based on popularity and concentration. Terms with a representativeness score lower than a given threshold are assumed to be too general for their topic and are pushed up to the parent topic. The clustering, scoring and pushing up of general terms repeats until no more general terms are found. For each topic, separate local embeddings are trained, so that even in subtopics of the lower levels of the taxonomy, fine distinctions between different concepts can be identified. However, TaxoGen has limitations and there is room for improvements on several fronts:

1. No lemmatization of candidate terms is performed, which leads to terms often appearing in their singular form and in their plural form in the taxonomy.

2. No domain term extraction is performed. The absence of a filter for domain terms shows in the results of TaxoGen. Although one could argue that pushing up general terms on the top level has a similar effect as filtering non domain terms.[3]

3. The number of subtopics for each topic is predefined, but a flexible number of topics would allow to better approximate the underlying semantic structure of how the terms are hierarchically related.

4. TaxoGen uses the Skip-Gram model to produce word embeddings. While

[3]Probably, non-domain terms are evenly distributed across topics. This means they are the first to get pushed up into a top level node that is not considered as a part of the taxonomy.

this technique has become very popular for finding and extracting semantic similarities, more sophisticated embeddings techniques have been developed since its publication.

5. It relies on spherical k-means clustering, which like all clustering techniques based on k-means tends to produce even cluster sizes. While this behaviour might be desirable for a balanced taxonomy tree there is no reason to assume that candidate terms are evenly distributed across topics and that similar cluster sizes should be produced.

6. TaxoGen is susceptible to terms that often appear with each other but have a different meaning from each other. For example, error analysis shows that the terms *anonymity* and *cryptography* often are the top candidates as labels for the same cluster. However, the best label for a subtopic of *anonymity* is different from the best label for the subtopic of *cryptography*.

4 Methods - Building on TaxoGen

I propose modifications to TaxoGen on three different levels, specifically on the embeddings-level, the clustering-level and the choice of labels for topics in the taxonomy. In the following sections I motivate and explain these changes.

4.1 Embeddings

On the embedding level I propose to test different embeddings, namely GloVe and ELMo embeddings.

4.1.1 GloVe Embeddings

GloVe, introduced by Pennington et al. [2014], is a technique to produce word embeddings using global contexts. First, a global co-occurrence table, which captures for each word how often it occurs in the context of each of the other words of the vocabulary, is constructed. This matrix is converted to a matrix of co-occurrence probabilities expressing how probable it is for a word to occur in the context of a given other word. Using this matrix, GloVe trains the word vectors in such a way that the dot product of the embeddings of two words is as close as possible to the logarithmic probability of the words occurring in each others context. GloVe embeddings differ from embeddings created using the Skip-Gram algorithm in that they explicitly model global co-occurrences, and thus global contexts, as opposed to Skip-Gram modeling local contexts.

4.1.2 ELMo Embeddings

ELMo, introduced by Peters et al. [2018], is a method to produce context-dependent embeddings. In context-dependent embeddings, a word or string of characters does not just have one fixed vector representation. Instead, a word gets a different vector representation depending on the context it appears in. Formulated differently, in

traditional embeddings, the vector representation is a function of only the given input word. In context dependent embeddings, the vector representation is a function of the given input word and the context it appears in. ELMo considers a sentence as the word's context. These word embeddings are produced using a bidirectional language model. A bidirectional language model is made up of two RNNs that walk through text (in this case a sentence) in opposite directions and predict the most probable next word based on previous evidence. The previous evidence - in this case - is constituted by the words already walked through. For each word the hidden state after processing that word is extracted as its vector representation. For ELMo, two bidirectional language models are stacked upon each other. The overall embedding of a word is calculated as a weighted sum of the embeddings in the language models. The standard version of the ELMo embeddings, which I use for this thesis, has 1024 dimensions.

Since calculating ELMo embeddings is computationally much more costly than calculating Skip-Gram or GloVe embeddings, it is infeasible to recursively train new ELMo embeddings on every subcorpus extracted in TaxoGen. Therefore, I employ a different method to get ELMo embeddings fine-tuned to specific subcorpora. When using ELMo embeddings, for each instance of a term a different embedding is calculated. To collapse all of the embeddings for the instances of a term into one embedding that can be used for adaptive spherical clustering I calculate the average embedding of all occurrences of the term in the current subcorpus. Formally, let $E(t, S_k)$ denote the embedding of term t in subcorpus, S_k. Let t_{j,S_k} denote the j-th instance of t in S_k. Then $E(t, S_k)$ is given by:

$$E(t, S_k) = \frac{\sum_{j=1}^{n} E(t_{j,S_k})}{n} \tag{4.1}$$

4.1.3 Why Test these Embeddings

Skip-Gram, GloVe and ELMo embeddings are all fundamentally different from each other. The first relies on a shallow neural network to predict contexts of words the second on global co-occurrence statistics and the third on two stacked language models. The use of global co-occurrence statistics by the GloVe embeddings is attractive in the case of hypernym identification. The main reason for this being, the distributional inclusion hypothesis introduced by Geffet and Dagan [2005]. It states that the contexts in which a hypernym appears tends to include the contexts in which its hyponyms appear. Since such context relations are global, they are more probable to be captured by GloVe embeddings than by Skip-Gram embeddings. The

use of ELMo embeddings is just motivated by their superior performance in other tasks including the task to identify textual entailment, which is closely related to identifying hypernymy [Peters et al., 2018].

4.2 Clustering

As an alternative to spherical k-means clustering, I test two agglomerative clustering techniques, which better allow for varying cluster sizes.

All agglomerative clustering techniques follow the same basic method: In the beginning, each data point is viewed as its own cluster. The distances between each pair of clusters is calculated. Then, the two clusters closest to each other are merged. This calculation of cluster distances and merging of the two closest clusters is repeated until only one cluster containing all data points is left. This procedure creates a hierarchy of clusters, with the top cluster containing all data points at the top, the two clusters that were merged into the top cluster right beneath, the four clusters that were merged into the two clusters one level deeper and so on. To get the desired number of clusters, one simply traverses down the hierarchy to the level where the desired number of clusters exists.

The question that remains and gives rise to different versions of agglomerative clustering is how to measure cluster similarity. There are two parameters to be chosen: affinity and linkage. Affinity defines the distance metric. This can be any kind of distance like the Manhattan distance, the Euclidean distance or the cosine distance. Linkage defines which data points of a cluster are used to calculate similarity. Minimum or single linkage calculates clusters distances between pairs of clusters by using only the two data points (one from each cluster) closest to each other. Complete linkage on the other hand calculates cluster distances by using only the two data points (one from each cluster) furthest apart. Average linkage calculates the distance from each data point of one cluster with each data point of the other cluster and defines their similarity using the average distance. Ward clustering is a special case of agglomerative clustering. It does not minimize the distance between two clusters. Instead, it minimizes the within-cluster variance, like the k-means algorithm. In other words: they have the same objective function.

After testing all possible combinations of linkage and affinity on a small sample data set I chose to test complete linkage with cosine similarity as affinity and Ward clustering on TaxoGen.

4.3 Label Score

To find a label for a topic in TaxoGen, the mean direction of the topic is calculated and its terms are sorted according to their cosine similarity to the topic's mean direction. To better ensure that topic labels are hyponyms of each other, I instead propose a label score that uses information from embeddings as well as from word distributions over documents. It is made up from three elements: The cosine similarity between the term and the topic center[1], a hyponym classification score and a hyponym distribution score. The term with the highest score is chosen as the cluster label. For term t in topic S_k with parent topic C the Label-score $\mathrm{L}(t, S_k, C)$ is given by:

$$\mathrm{L}(t, S_k, C) = \sqrt{\text{cos-sim}(t, S_k) * \text{hypoclf}(t, C) * \text{inclusion}(t, C)} \tag{4.2}$$

4.3.1 Distributional Inclusion Score

The distributional inclusion score aims to capture hyponymy relations and is based on the distributional inclusion hypothesis from Geffet and Dagan [2005], explained in section 3.5. A hyponym h of term t tends to occur in a subset of t's contexts. When applying this hypothesis on word distributions over documents, one can assume that h, a hyponym of t, tends to occur in a subset of the documents in which t occurs. Thus, for a given hypernym t and a candidate hyponym h, one can assume the following: The higher the ratio between documents containing both t and h and all documents containing h, the more likely it is for h to be a hyponym of t. Formally this relation can be expressed as

$$\text{inclusion}(t, h) = \frac{df(h, t)}{df(h) + 1} \tag{4.3}$$

where $df(h, t)$ denotes all documents containing both h and t and $df(h)$ denotes the document frequency of h.

However, in the scenario of a topical taxonomy like TaxoGen, an entire parent topic is given, not only a parent term. Thus, the inclusion score should be between the parent topic C and the hyponym h, which is to be scored. This opens up the question of how inclusion on a document level of a term by a topic should be measured. A naive approach could consist of simply collecting all documents in which a term of cluster C appears in. Such an approach would be susceptible to terms, that barely made it into the cluster. These terms could enlarge the topic document collection in

[1]The cosine similarity is mapped to the interval [0,1] for this score.

an non-useful manner. Thus, I choose to only include the documents in which the top n terms of a cluster (by similarity to the cluster center) occur. The inclusion score inclusion(C, h) for a given hyponym h and its parent topic C can then simply be expressed as:

$$\text{inclusion}(C, h) = \frac{df(C, h)}{df(h) + 1} \tag{4.4}$$

$df(C, h)$ denotes the number of documents which contain h and at least one of the top n terms in C.

4.3.2 Hyponym Classification Score

The hyponym score indicates how likely it is that a term qualifies as a hyponym of the parent topic using classification. It has a range from 0 to 1, with 1 denoting certainty that the term in question qualifies as a hyponym. The hyponym score is calculated with a SVM using a radial basis function kernel. Although SVMs are normally used for binary output, some implementations also make a technique, introduced by Platt et al. [1999], available which maps the SVM output to a probability via a sigmoid function. The SVM is trained with hypernym-hyponym pairs. To get corpus-specific training data without needing to manually create a training set I employed the following method:

1. Create positive examples by extracting hypernym-hyponym pairs from the entire corpus using the Hearst Patterns from Hearst [1992], described in section 2.1.

2. Create negative examples by (a) inverting the hypernym-hyponym pairs as done by Ustalov et al. [2017] and (b) extracting random pairs of NPs from the corpus and checking if they are not present in the positive examples. Inverting a hypernym-hyponym pair guarantees a negative example because hypernymy is an asymmetrical relationship. Extracting two random NPs could erroneously result in an unseen positive example, but this is very unlikely.

3. Embed the positive and negative samples using the embeddings trained on the entire corpus (global embeddings).

4. For each example subtract the embedded hyponym from the embedded hypernym to get their distance. This distance represents the sample and constitutes the feature vector fed to the SVM.

5. Train the SVM on the positive and negative samples.

The reason for calculating the difference of the hyponym and the hypernym embedding is to avoid the criticism from Levy et al. [2015] that hypernym classifiers just memorize typical hypernyms. I described this criticism in section 2.2.

To calculate a hyponym score for a new candidate term, the difference between the embedding of the parent topic and the embedding of the candidate term has to be computed and to be given to the SVM.

5 Experiments and Results

5.1 Dataset

To keep the results of this thesis comparable to those of the original TaxoGen paper, it is desirable to use the same corpus as has been used for the testing of TaxoGen. TaxoGen has been tested with two corpora: The DBLP corpus[1] and the SP corpus[2]. Since the authors of TaxoGen published their preprocessed version of the DBLP corpus[3], I used their preprocessed DBLP corpus for evaluation. The DBLP corpus is a large collection of publications in computer science and neighbouring fields. It contains about 4.3 Mio. publications. The authors of TaxoGen reduced the corpus to about 1.8 Mio. articles.

5.2 Implementation

The reimplementation is first and foremost an implementation of how TaxoGen is described in its publication (Zhang et al. [2018]). However, the differences between publication and implementation and my proposed modifications are included as modules that can be switched on and off using a configuration file. Since TaxoGen needs large corpora as input to work properly, the implementation has to be efficient regarding space and time complexity. For this implementation, I chose to move as much processing steps as possible into the preprocessing, since the preprocessing only is run once while the taxonomy generation is run many times. The preprocessing includes extracting noun phrases as term candidates, lemmatization, extracting hypernym relations with Hearst patterns, indexing the corpus, calculating term frequencies per document, calculating document frequencies, calculating

[1]The website can be found here: `https://dblp.uni-trier.de/`; Last visited: 24.05.2019

[2]The website can be found here: `https://signalprocessingsociety.org/publications-resources`; Last visited: 24.05.2019

[3]The preprocessed corpus can be downloaded via a link on the GitHub site, on which the implementation of TaxoGen is hosted: `https://github.com/franticnerd/taxogen`; Last visited: 24.05.2019

document lengths, training global embeddings, calculating document embeddings and training the hypernym classifier. This means that during the actual generation of the taxonomy no text has to be processed, which makes the taxonomy generation less space and time consuming.

Two distinct preprocessing pipelines are set up: one starting with the initial raw corpus downloaded from the DBLP website and one starting with the preprocessed corpus provided by the authors of TaxoGen. All results in this thesis are produced with the preprocessing relying on the already preprocessed corpus from the authors of TaxoGen to enable better comparability. However, for further development it might be attractive to switch to the preprocessing pipeline starting from the raw corpus, since it allows to additionally lemmatize terms (including MWEs) and is simple to expand.

The integration of the ELMo embeddings was not possible as planned and described in section 4.1.2. The averaging of a term's embeddings of all occurrences in a subcorpus took up either too much RAM when loading all necessary embeddings at once or was too slow when loading one embedding at a time. Thus, for the evaluation of TaxoGen with ELMo embeddings I did not use any local embeddings, instead I always used the global embeddings, even at the deeper levels of the taxonomy.

There are several hyper-parameters that need to be set for TaxoGen. The TaxoGen implementation contains some additional restrictions, also implemented through parameters. All of those parameter-settings are described in table 3.

Name	Description	Value
`n_clusters`	The number of clusters per level.	5
`threshold`	The threshold to push term up.	0.25/0.35
`max_iter`	The maximum number of spherical clustering iterations per level.	5
`max_depth`	The maximum depth of the taxonomy.	4
`n_expand`	How many terms near the cluster center should be used to extract documents for embedding training as described at the end of section 3.4.	200
`n_incl`	How many terms near the cluster center should be used to collect documents for the topic C in equation 4.4.	10

Table 1: Parameter settings.

The two values for the threshold parameter stand for a threshold of 0.25 for the top level of the taxonomy and a threshold of 0.35 for all levels below.

The code for the reimplementation, building on top of TaxoGen and for the evaluation can be found here: `https://github.com/jagol/BA_Thesis`[4]

5.3 Evaluation

5.3.1 General Procedure

To separate the effects of the different changes proposed, testing all different combinations of the proposed changes would be desirable. However, this would result in over 20 versions to evaluate. To limit the number of generated taxonomies, which have to be evaluated I conducted the following procedure: I first tested TaxoGen with different embedding versions. I then tested the different clustering algorithms only with those embeddings that performed best. Lastly, I tested the labeling score only with the embeddings and clustering algorithm that performed best.

5.3.2 Metrics

It is not an easy question how a topical taxonomy should be evaluated. For this evaluation, I tried to follow TaxoGen's evaluation as closely as possible to ensure comparability. Zhang et al. report a relation accuracy[5]. The relation accuracy "aims at measuring the portions of the true positive parent-child relations in a given taxonomy."[6]. But, there are three open questions with this metric:[7]

1. How do Zang et al. get the information to compute a relation accuracy? Accuracy in the context of binary classification is defined as:

$$\text{ACC} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}} \tag{5.1}$$

   However, when automatically constructing a taxonomy without having a gold taxonomy, we do not know the number of True Negatives and the number of False Negatives. It is therefore impossible to calculate a relation accuracy in this context. I assume that Zhang et al. instead computed the relation

[4]Last visited: 24.05.2019

[5]They also report a cluster quality and term coherency. However, I focus on relation accuracy in this thesis as it the most important metric to measure the quality of taxonomy.

[6]Zhang et al. [2018]

[7]I contacted the authors to ask them these questions about their evaluation, but they did not respond.

precision, which is defined as follows:

$$\text{PRE} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5.2}$$

Relation precision only needs the number of True Positives and False positives, which can be obtained by extracting relations out of the taxonomy and annotating them.

2. Are these parent-child relations between topics or between terms? A parent-child relation could denote a relation between a parent topic and a child topic. Or it could denote a relation between a term in the parent topic and a term in the child topic. For example, it could denote the relation between the term with the highest score in the parent topic and the term with the highest score in the child topic.

3. What are the necessary conditions for a valid parent-child relation? In a narrow sense, a parent-child relation could denote a hypernym-hyponym relation. In a wider sense it could mean, that the child is in some way a subtoptic of the parent.

To give consideration to all possible interpretations of how TaxoGen was evaluated, I employ three different metrics:

Hypernym relation precision: This is a term-level metric. For this metric I view a parent-child relation as a relation between the highest scoring term in the parent topic and the highest scoring term in the child-topic. The relation is valid, if the child term is a hyponym of the parent term.

Is-subtopic relation precision: This is a term-level metric too. For this metric I view a parent-child relation as a relation between the highest scoring term in the parent topic and the highest scoring term in the child-topic. The relation is valid, if the child term can be viewed as a subtopic of the parent term. A subtopic-relation can be a hypernym relation, a meronym-relation or anything that comes up immediately when the parent topic is discussed.

Is-subtopic relation precision - topic level: This is a topic-level metric. For this metric I view a parent-child relation as a relation between two entire topics. Each topic is a set of words. The relation is valid if most terms in the child topic generally can be viewed as a subtopic of the terms in the parent topic.

In section 6.3 I further discuss these metrics. For each generated taxonomy I extracted its relations and sampled 100 random relations per metric. For the case of topic-level relation precision, the taxonomy often contained less than 100 relations.

This can happen if there are nodes to whom no terms have been pushed up, because the no representativeness scores are below the threshold for pushing up terms. If this was the case, I just used all relations in the taxonomy.

5.3.3 Evaluation Results

Table 2 contains the obtained results and table 3 contains explanatory descriptions.

	Hyp-RP				Is-sub-RP				Is-sub-RP-TL			
	TP	FP	Total	RP	TP	FP	Total	RP	TP	FP	Total	RP
TaxoGen	-	-	-	0.775 ?	-	-	-	0.775 ?	-	-	-	0.775 ?
W2V+SPKm (ReImpPub)	2	98	100	0.02	9	91	100	0.09	5	6	14	0.36
W2V+SPKm (ReImpImp)	2	98	100	0.2	35	65	100	0.35	35	16	51	0.69
GloVe+SPKm	7	93	100	**0.07**	40	60	100	0.4	18	15	33	0.55
ELMo+SPKm	2	98	100	0.02	19	81	100	0.19	14	86	100	0.14
W2V+Ward	6	94	100	0.06	23	77	100	0.23	15	17	32	0.47
W2V+CompL	2	98	100	0.02	18	82	100	0.18	12	18	30	0.40
W2V+SPKm+Hyp	3	97	100	0.03	43	57	100	**0.43**	37	14	51	0.73
W2V+SPKm+Incl	1	99	100	0.01	39	61	100	0.39	39	12	51	**0.76**
W2V+SPKm+Hyp+Incl	1	99	100	0.01	40	60	100	0.4	39	12	51	**0.76**

Table 2:
Evaluation Results.

Short name	Description
TaxoGen	The results reported in the TaxoGen publication [Zhang et al., 2018]. It is not clear which interpretation of a parent-topic relation was deployed by Zhang et al. Therefore, the result is listed in every column, but with an question mark.
ReImpPub	The reimplementation of TaxoGen as it is described in its publication. The threshold to push terms up is set to 0.0005, since the representativeness scores are generally much lower if the score is implemented as described in the publication.
ReImpImp	The reimplementation of how TaxoGen including the differences found.
W2V	TaxoGen with Skip-Gram embeddings (named after the alternative name *Word2Vec*).
GloVe	TaxoGen with GloVe embeddings.
ELMo	TaxoGen with ELMo embeddings.
SPKm	TaxoGen with Spherical K means clustering.
Ward	TaxoGen with Ward clustering.
AvgL	TaxoGen with average linkage agglomerative clustering.
CompL	TaxoGen with complete linkage agglomerative clustering.
Hyp	TaxoGen with the label score only considering center similarity and hyponym score.
Incl	TaxoGen with the label score only considering center similarity and inclusion score.
Hyp+Incl	TaxoGen with the full label score as described in section 4.3.

Table 3: Short name descriptions

As table 2 shows, according to the hypernym relation precision the configuration using GloVe embeddings (GloVe+SPKm) performed best. According to the `is-subtopic` relation precision the configuration using only the hypernym classification score (W2V+SPKm+Hyp) performed best. And according to the `is-subtopic-topic-level` relation precision the configuration using the full label score and the configuration using only the distributional inclusion score (W2V+SPKm+Incl and W2V+SPKm+Hyp+Incl) performed best. Comparing the relation precisions of TaxoGen and its reimplementation including the differences found (ReImpImp) suggests that the score of 0.775 reported by Zhang et al. actually refers to a topic level relation precision. Overall the reimplementation scores lower than the original implementation but the additional labeling score closes the gap to the original implementation. Possible reasons for these differences in the results are discussed in section 6.1.

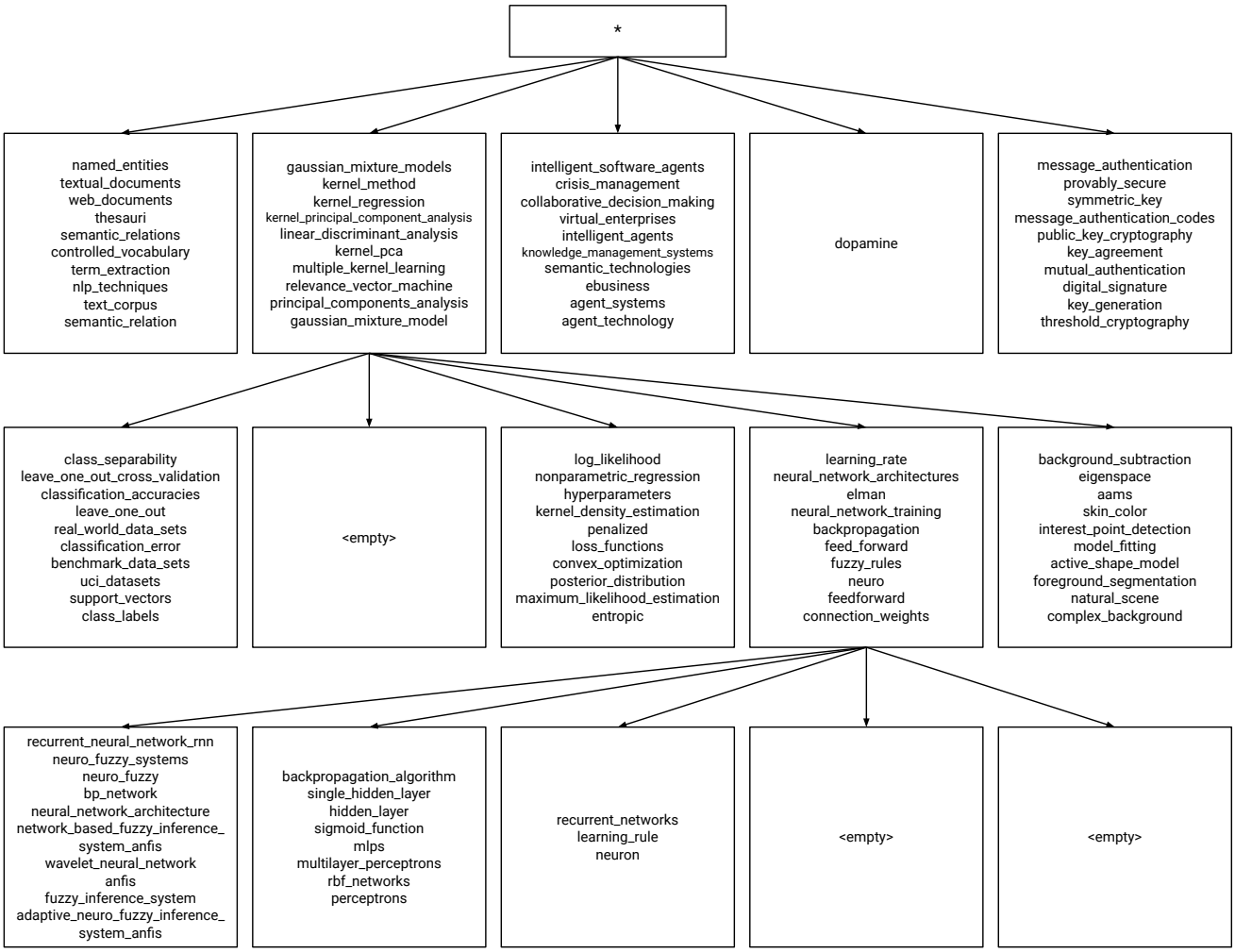

Figure 1:
A part of the taxonomy generated by Skip-Gram embeddings, spherical-k-means clustering and the full label score (W2V+SPKm+Hyp+Incl).

Figure 1 shows the taxonomy generated by using Skip-Gram embeddings, spherical k-means clustering and the full labeling score (W2V+SPKm+Hyp+Incl). The top node is divided into five topics. The obvious odd one out is the fourth topic with only one term *dopamine*. This term cannot be seen as general in any way and is an obvious error. The other four topics on the top level revolve around text processing, kernel methods and machine learning, intelligent agents and cryptography related topics. However, there is noise in the topics. For example the topic of intelligent agents is mixed with e-commerce terms and contains the term *crisis management*. The hierarchy of the topics is not as clear as would be desirable. The second topic on the top level, revolving around machine learning, already contains specific machine learning methods and not more general terms like *machine learning* or *learning algorithms*. Examining the subtopics of the machine learning topic shows that the

terms in these subtopics are semantically related but not necessarily more specific. However, a comparison with the worst performing configuration (ELMo+SPKm) makes clear how the topics above are coherent to quite a high degree.

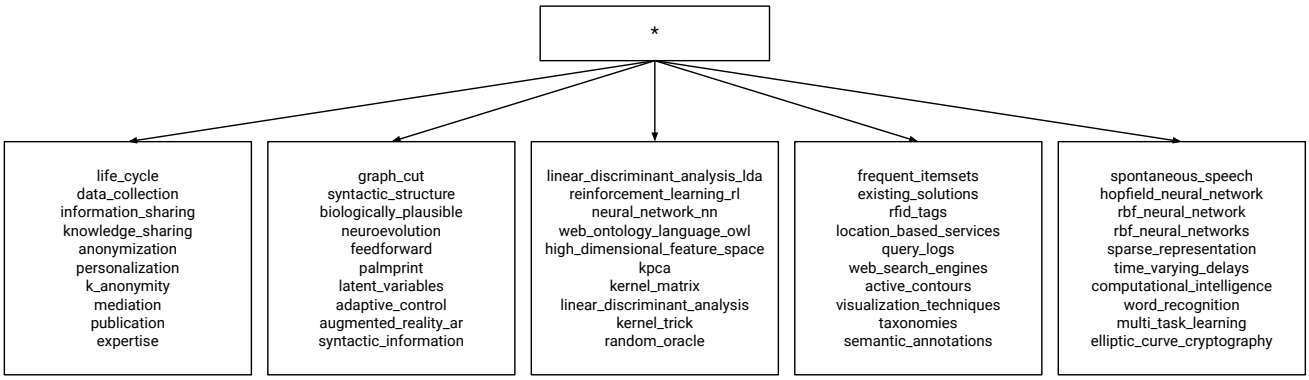

Figure 2:
The top level of the taxonomy generated using ELMo embeddings (ELMo+SPKm).

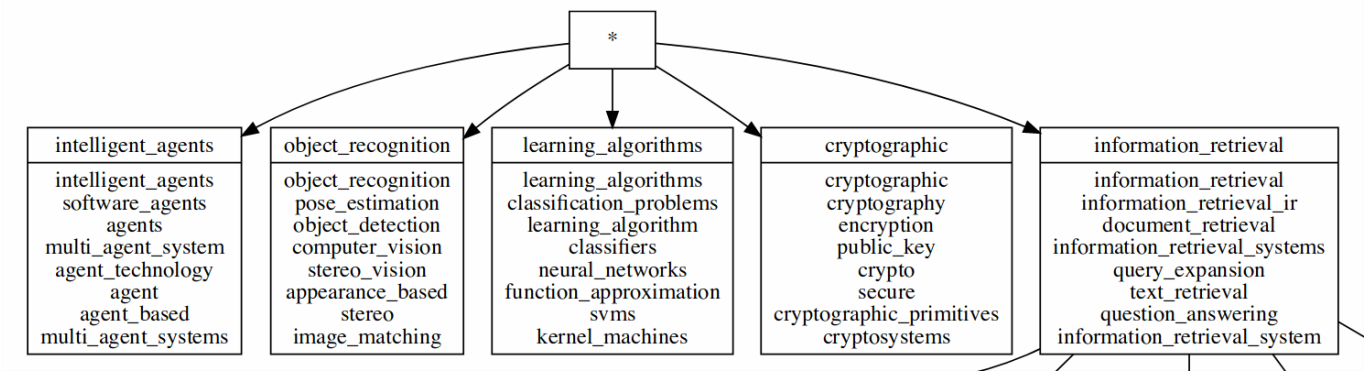

Figure 3:
The top level of the taxonomy generated by the original TaxoGen implementation. The figure is taken from Zhang et al. [2018].

6 Discussion

6.1 Comparison with the TaxoGen Results

The results of the reimplementation of TaxoGen considerably differ from the results in the original implementation. The TaxoGen reimplementation which includes the differences found (ReImpImp) clearly yields better results than the TaxoGen reimplementation that strictly follows the publication's descriptions (ReImpPub). Assuming that the relation precision reported in the TaxoGen publication refers to topic-level evaluation, the reimplementation obtains results that are slightly worse. Manually comparing the coherency of the individual topics in figures 1 and 3 shows that the results from the reimplementation (ReImpImp) do clearly not have the same quality as the results from the original TaxoGen implementation.

This difference in results could have a number of possible reasons: Small implementation differences of the Skip-Gram algorithm could have a snowball effect like described in section 3.4. Other implementation-specific details, like the exclusion of terms in a cluster if they are not contained in the local embedding model, might also play a role in the generation of different results between the implementations.[1] However, it is not likely that such small differences in the implementation would produce such a different outcome. It is also possible that there are hidden errors in the code of the reimplementation. However, the key parts of the system are unit-tested and therefore probably do not contain errors. A last possibility is that there are more differences between the original implementation of TaxoGen and how it is described in the paper. My analysis of the implementation concentrated on the adaptive spherical clustering, local embedding training and calculation of the representativeness score of TaxoGen but not, for example, on how data is loaded or how it is passed between the different system modules. Future efforts to improve the reimplementation could focus on these other parts of the system, which are not at the core of the proposed system but may nevertheless be essential for the kind of results produced.

[1] It is possible that a term does not have a local embedding. This happens if the corpus extracted for local embedding training does not contain the term in question.

6.2 Comparison between Different Versions

On the embedding level, the Skip-Gram embeddings performed best. The GloVe embeddings performed better according to `hyponym` and `is-subtopic` relation precision but worse in the topic-level evaluation. The ELMo embeddings performed by far the worst showing that the higher dimensionality could not make up for the lack of local embeddings. On the clustering level, spherical k-means clustering performed better than the agglomerative clustering techniques. This shows that enforcing similar cluster sizes, as done by k-means and spherical k-means clustering, is not a problem. Instead it has a positive effect. From the agglomerative clustering techniques ward clustering performed better than complete linkage. These results are surprising. The cosine distance has proven to be a superior distance measure than the euclidean distance for many NLP tasks. Since complete linkage clustering is in this case configured to use the cosine distance and ward clustering is only possible with the euclidean distance, one would expect complete linkage clustering to have an advantage over ward clustering. However, ward as a linkage criterion outweighs the disadvantage of using the euclidean distance.

The proposed label score clearly improves the results. However, the full label score (W2V+SPKm+Hyp+Incl) and the configuration without the hyponym classification score (W2V+SPKm+Incl) perform equally good. Thus, the inclusion score is probably the deciding factor, not the hyponym classification score. For further analysis it would be interesting to test the hyponym classifier as a binary classifier in isolation. A low performing hyponym classifier could be explained through false positives in the Hearst pattern extraction which would act as noise in the training data.

6.3 About the Evaluation of Topical Taxonomies

Evaluating a topical taxonomy comes with several difficulties, since it is not clear what a topical gold taxonomy should look like. Two main questions emerge: (1) What kind of relations should the topical taxonomy actually depict? (2) If the relations are only vaguely defined, how should one decide if a relation is valid or not?

Regarding the first question: Simple, non-topical, taxonomies only contain hypernym relations. A hypernym relation can only exist between two single terms. It cannot exist between two topics. So, the question becomes what relation is depicted

instead. Two possible answers come to mind: The first one consists in saying that a parent and a child node should be in a relation where the child is a subtopic to the parent. I assume that this is what Zhang et al. where thinking of, when they evaluated their taxonomy. However, such a relation is vaguely defined and in different contexts different topics could be considered sub- or child topics of a parent topic. For example in the context of cyber security `decryption` could constitute a valid subtopic of `risk`, since `decryption` is a `risk` for, for example, classified digital information. But in the context of finance or politics, `decryption` would hardly qualify as a valid subtopic for `risk`. The second possible answer consists in stating that a hypernym relation should hold between the cluster label of the parent topic and the cluster label of the child topic. Additionally, each label has terms in its topic which are closely related to it. This could be named the taxonomy-plus view of topical taxonomies, since it views a topical taxonomy as a normal taxonomy, where each term in the taxonomy is additionally surrounded by other closely related terms.

Regarding the second question: If a topical taxonomy is interpreted as depicting `is-subtopic` relations, it is difficult to draw a line what a valid relation is, since the notion of a subtopic has no clear definition. One possible solution consists in developing further criteria a relation has to meet to be counted as valid. For the evaluation in this thesis, for the `is-subtopic-topic-level`-relation, I counted the relation in question as valid if at least one of the following three questions could be answered with yes:

1. Are some terms in the child topic in a hypernym relation with a term in the parent topic?
2. Are some terms in the child topic in a meronym relation with a term in the parent topic?
3. If there is a detailed discussion of the parent topic, is it highly likely that the child topic comes up?

A positive example for question 3 would be: "If the parent topic `deep learning` comes up, is it highly likely that the child topic `recursive neural network` comes up?" A negative example for question 3 would be: "If the parent topic `deep learning` comes up, is it highly likely that the child topic `support vector machine` comes up?"

Question 3. helps to cover cases which are not covered by hypernymy or meronymy, but where the child topic still should be counted as a subtopic. To clarify this using the example above: There is neither a hypernym nor a meronym relation

between `deep learning` and `recursive neural network`. But `recursive neural network` still should count as a subtopic of `deep learning`. This is made possible by question 3. However, there is still vagueness in this criterion as there are cases where it is debatable if question 3 can be answered with yes. How much disagreement there is about how these questions should be answered and if a different or expanded set of questions would help to further make the relation boundaries clear could be measured with an inter-annotator agreement.

6.4 Future Work

What future work should be done depends on the current limitations of TaxoGen. Thus, this section is closely related to section 3.5, where some of the limitations of TaxoGen are described. An obvious next step is to further investigate why the two implementations differ in their results. But apart from that, there are several ideas for improvements:

Lemmatization: TaxoGen does no lemmatization. Although lemmatization is not part of the configurations evaluated in this thesis, it is already implemented and can easily be added to the pipeline.

Number of subtopics: The number of subtopics or clusters per level in the taxonomy is set to 5. However, it would be much more desirable for the number of clusters to depend on the input data points. There exist several methods, like knee analysis, to automatically find the number of clusters which group the data points in an optimal way. However, these methods come with a drawback of having to cluster multiple times to compare the clustering outcomes. This, in combination with the already iterative process of adaptive spherical clustering would make TaxoGen very time consuming. It is an open question how finding the best number of clusters can be made compatible with time constraints.

Threshold to push terms up: The threshold to push terms up is static; meaning that it does not change depending on the current distribution of representativeness scores. This can lead to empty topics, in the sense that they contain no terms. A threshold that dynamically moves up or down based on an analysis of the current distribution of representativeness scores could solve this problem.

Taxonomy coherency: Even if the terms in a taxonomy have been pushed into the correct topics and subtopics, the taxonomy can appear as incoherent if the topic labels are not related to each other in the same way. For example, if most or all relations between a parent topic and its child topics can be characterized as hypernym relations, this would be a coherent taxonomy. However, if the relations in a taxonomy between the labels of parent topics and child-topics are a mix of hypernymy, meronymy and other relations which would just be characterized as `issubtopic`, then this taxonomy would not be as coherent. The labeling score introduced in section 4.3 constitutes a first step to ensure coherency, since it makes the labeling of child topics dependent of the top scoring terms (and thus also the label) of the parent topic. However, topic labels in this score are chosen using only a *local* context by considering the parent label. In future work, this approach could be extended to a taxonomy-wide, thus *global*, optimization for label coherency. The proposed labeling score could serve as a starting point for initial labeling of the topics. Then, global optimization could be deployed by systematically switching out topic labels to maximize a global coherency score.

7 Conclusion

The goal of this thesis was to reproduce the results of TaxoGen and enhance it with better embeddings, clustering algorithms and a new label score. I showed that to reproduce TaxoGen's results, techniques different from the descriptions in the publication and additional techniques not mentioned in the publication are needed. However, I was not able to exactly reproduce TaxoGen's results and it is not entirely clear where the reason lies. Testing TaxoGen with GloVe and ELMo embeddings did not lead to an improvement, showing that Skip-Gram-based embeddings are good enough for this purpose. However, I was not able to train local ELMo embeddings. If this was possible, the results for TaxoGen with ELMo embeddings probably would have been better. Testing TaxoGen with agglomerative clustering techniques showed that although they allow better for different cluster sizes, the quality of the taxonomy did not improve. I also proposed a label score which combines cosine similarity, a hyponym classification score and a distributional inclusion score. This label score significantly improved the relation precision. The improvement seems mostly due to the inclusion score, whereas the classification score does not have the same impact. Overall there still remains a lot of room for improvement. The topics are often noisy and the hierarchy of topics does not seem conclusive. In the last chapter I discussed how topical taxonomies can be evaluated in general and sketched several ideas how TaxoGen could be improved in the future. But it is not only necessary to improve and optimize the taxonomy generation techniques, it is also necessary to think about what exactly a topical taxonomy should be optimized for. I think that one of those optimization criteria is coherence. I proposed the label score as a first step into this direction.

References

D. Alfarone and J. Davis. Unsupervised learning of an is-a taxonomy from a limited domain-specific corpus. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

T. L. Anh, J.-j. Kim, and S. K. Ng. Taxonomy construction using syntactic contextual evidence. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 810–819, 2014.

T. L. Anh, Y. Tay, S. C. Hui, and S. K. Ng. Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 403–413, 2016.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

G. Bernier-Colborne and C. Barriere. Crim at semeval-2018 task 9: A hybrid approach to hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 725–731, 2018.

G. Bordea, P. Buitelaar, S. Faralli, and R. Navigli. Semeval-2015 task 17: Taxonomy extraction evaluation (texeval). In *Proceedings of the 9th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2015.

P. Buitelaar, P. Cimiano, and B. Magnini. *Ontology learning from text: methods, evaluation and applications*, volume 123. IOS press, 2005.

H.-Y. Chen, C.-S. Lee, K.-T. Liao, et al. Word relation autoencoder for unseen hypernym extraction using word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4834–4839, 2018.

R. L. de Mantaras and L. Saitia. Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. In *16th European*

Conference on Artificial Intelligence Conference Proceedings, volume 110, page 435, 2004.

I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1-2):143–175, 2001.

T. Fountain and M. Lapata. Taxonomy induction using hierarchical random graphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 466–476. Association for Computational Linguistics, 2012.

R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1199–1209, 2014.

M. Geffet and I. Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics, 2005.

G. Grefenstette. Inriasac: Simple hypernym extraction methods. *arXiv preprint arXiv:1502.01271*, 2015.

J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

G. Kruszewski, D. Paperno, and M. Baroni. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388, 2015.

R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. On semi-automated web taxonomy construction. In *WebDB*, pages 91–96, 2001.

O. Levy, S. Remus, C. Biemann, and I. Dagan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, 2015.

X. Liu, Y. Song, S. Liu, and H. Wang. Automatic taxonomy construction from keywords. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1433–1441. ACM, 2012.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

R. Navigli and P. Velardi. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1318–1327. Association for Computational Linguistics, 2010.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3): 61–74, 1999.

S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*, pages 1297–1304, 2005.

D. Ustalov, N. Arefyev, C. Biemann, and A. Panchenko. Negative sampling improves hypernymy extraction based on projection learning. *arXiv preprint arXiv:1707.03903*, 2017.

W. L. Woon and S. Madnick. Asymmetric information distances for automated taxonomy construction. *Knowledge and information systems*, 21(1):91–111, 2009.

J. Yamane, T. Takatani, H. Yamada, M. Miwa, and Y. Sasaki. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1871–1879, 2016.

H. Yang and J. Callan. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 271–279. Association for Computational Linguistics, 2009.

Z. Yu, H. Wang, X. Lin, and M. Wang. Learning term embeddings for hypernymy identification. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, and J. Han. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2701–2709. ACM, 2018.