DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Automated Refinement of an Ontology of NLP Research Concepts

Karim Arabi

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Automated Refinement of an Ontology of NLP Research Concepts

Automatisierte Erweiterung einer Ontologie von NLP-Forschungsbegriffen

Author:	Karim Arabi
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Tim Schopf, M. Sc.
Submission Date:	June 15, 2023

I confirm that this Master's Thesis in Informatics report is my own work and I have documented all sources and material used.

Munich, June 15, 2023 Karim Arabi

Acknowledgments

I want to show my deepest appreciation and gratitude to my advisor, Tim Schopf, for providing the invaluable resources, continuous feedback, and unwavering support whenever I needed it. With his help, I held a firm grasp on my objective on a week-to-week basis. Moreover, I'd like to give my sincere thanks to my supervisor, Prof. Dr. Florian Matthes, for allowing me to undertake this thesis at his chair. I extend special thanks to all the researchers who generously gave their time to participate in my evaluation studies. I would also like to add a final heartfelt thank you to my friends and family who stood by me and offered their support and encouragement at every step.

Abstract

As the volume of academic publications across all disciplines continues to grow, researchers face the challenge of effectively exploring literature relevant to their fields. Addressing this, we propose a semi-unsupervised method to classify papers into a structured ontology, enabling easier navigation and more efficient topic investigation. Our process involves four key steps: 1) key phrase extraction from a corpus of academic papers; 2) merging similar key phrases; 3) developing taxonomic relations among the merged concepts; and 4) establishing non-taxonomic relations to augment the conceptual graph. Our study extends an existing key phrase extraction technique and introduces novel strategies for concept merging and hierarchy development. The final phase incorporates a modified version of the SciCero method for developing non-taxonomic relationships between concepts, thereby enhancing the overall effectiveness and utility of our proposed system.

Kurzfassung

Da das Volumen wissenschaftlicher Veröffentlichungen in allen Disziplinen weiterhin wächst, stehen Forscher vor der Herausforderung, relevante Literatur in ihren Fachbereichen effektiv zu erkunden. Um diesem Problem entgegenzuwirken, schlagen wir eine teilweise unüberwachte Methode zur Klassifizierung von Papieren in eine strukturierte Ontologie vor, um eine einfachere Navigation und effizientere Themenuntersuchungen zu ermöglichen. Unser Prozess umfasst vier wesentliche Schritte: 1) Extrahierung von Schlüsselphrasen aus einem Korpus wissenschaftlicher Arbeiten, 2) Zusammenführung ähnlicher Schlüsselphrasen, 3) Entwicklung taxonomischer Beziehungen zwischen den zusammengeführten Konzepten und 4) Herstellung nicht-taxonomischer Beziehungen zur Erweiterung des konzeptionellen Graphen. Unsere Studie erweitert eine bestehende Technik zur Extrahierung von Schlüsselphrasen und führt neuartige Strategien zur Konzeptzusammenführung und Hierarchieentwicklung ein. Die abschließende Phase beinhaltet eine modifizierte Version der SciCero-Methode zur Entwicklung nicht-taxonomischer Beziehungen zwischen Konzepten, wodurch die Gesamtwirksamkeit und Nützlichkeit unseres vorgeschlagenen Systems verbessert wird.

Contents

1 Introduction

According to a report generated in 2018 by International Association of Scientific, Technical, and Medical Publishers (STM), there exists a 4% yearly growth rate in articles and a 5% yearly growth in journals were observed. More current data demonstrates that these growth rates continue to increase [1].

With the ever-expanding purview of available research studies and documents becoming available, the discoverability of such papers has become challenging. As the rate of scientific publications is increasing with time, many publications with relevant topics could be omitted from a simple search due to a difference in terminology.

A better means of sorting through and finding relevant research topics rather than through traditional keyword searches would be a helpful tool for researchers to utilize. A domain-specific ontology would satisfy this issue, providing a search through semantic understanding of a requested topic. Researchers could also utilize this ontology to explore direct relations to the queried topic, as well as discover new avenues for research.

One advantage is the idea of employing 'concepts' in the ontology, which are an aggregate of different topics in Natural Language Processing (NLP) that have the same semantic meaning. Users could search for their requested topic without having to know exact keywords. Instead, if a user is searching for 'emotion recognition' or 'sentiment analysis', they are routed to the same topic, and can gain an understanding of the different variations of a concept that can exist in various publications.

Another advantage is the hierarchical nature of the ontology. Users don't need an in-depth understanding of a topic in order to see under which umbrella of NLP it exists. Instead, they only need to traverse up the hierarchy. And similarly (flipside), if a user is unsure what detailed concepts exist under different areas of NLP, they can simply explore the taxonomic relations of the ontology to retrieve ancestors or descendants of a concept.

And finally, an ontology offers more than just the hypernym-hyponym relations one finds in a taxonomy. An ontology allows the user to explore a multitude of ways that a concept affects any other concept in the hierarchy. For a topic of their interest, they can peruse which concepts utilize it, replace it, affect it, and a plethora of other possible information about the ripple effects of a concept in the field of NLP.

Unfortunately, if one were to try and construct an ontology of NLP topics by hand, it

would take a considerable amount of time and repetitive labor. In addition, such a task could only be feasibly done by those with sufficient knowledge of the field such as NLP experts and researchers, and it would require a large number of them too. Therefore, we propose a semi-unsupervised approach to learn the ontology automatically.

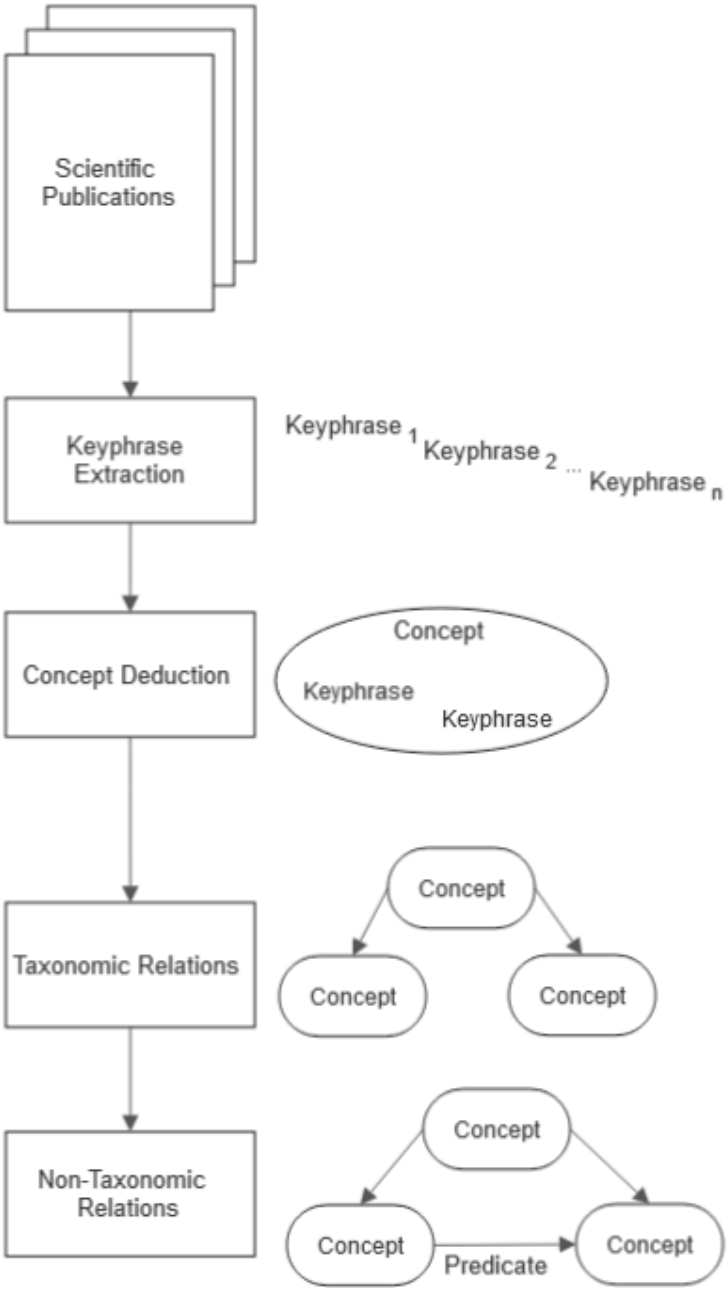

Figure 1.1: Flowchart showing the step by step process of building an ontology out of a corpus of scientific publication, as inspired by Asim et al. [2].

We aim to build off the work of a previous master's thesis to expand on what has been defined [3]. The aforementioned thesis provides a solid foundation for keyphrase extraction and filtering from a corpus of research documents about Natural Language Processing. It also presents methods for concept clustering and the formation of hierarchical relationships between these concepts. This paper continues to improve on the ideas presented to build a high quality automated ontology of NLP concepts.

The main task is this: How do we deepen the hierarchical relationships between research concepts? To answer, we delve into more specific challenges that need investigating:

- How to use manual refinement to improve top-level navigation for users?
- How to enhance the existing concepts and relations through automated refinement approaches?
- How to transition from a taxonomy to an ontology with more complex relations?

Our thesis aims to answer these questions by investigating the different subtasks that, together, create a pipeline for constructing an ontology from a corpus of abstracts and titles. The idea of layering the different steps for ontology creation from unstructured text is described by Asim et al. [2], who provide a helpful overview of the ontology creation pipeline.

This thesis is structured in the following way: Chapter 2 delves into the technical foundations of this research by providing some background knowledge on different terms or concepts that will be referred to and explored throughout the other chapters. Chapter 3 provides an overview of different available literature that delve into a very similar topic to this one, but that aren't necessarily utilized or referenced for our own implementation of this process. Chapter 4 explores the different methodologies employed by this thesis and explains our implementation in detail. Chapter 5 presents the results of our methodologies after evaluating our processes. Chapter 6 summarizes the work and discusses its results, and Chapter 7 showcases avenues for improvement and for future work that can be continued following this thesis.

2 Technical Foundation

The upcoming sections will provide an overview of key technical terms used in this thesis. Additionally, a brief discussion will be provided on the selected Part-of-Speech tagger.

2.1 Terms in Text Processing

WordNet is a widely utilized lexical database and semantic network in the fields of natural language processing and computational linguistics. Its purpose is to organize words and their meanings into synsets, which are sets of synonymous words [4]. With its expansive collection of word relationships, including hypernymy, hyponymy, meronymy, and entailment, WordNet facilitates the exploration of semantic connections between words. It serves as a valuable resource for tasks such as word sense disambiguation, information retrieval, and ontology construction [4]. Researchers and developers heavily rely on WordNet due to its comprehensive coverage and its significant contributions to various language-related applications.

Natural Language Toolkit (NLTK) is a widely used open-source Python library for Natural Language Processing (NLP) tasks [5]. It provides a broad range of features, such as tokenization, Part-of-Speech tagging, and syntactic parsing. With its extensive collection of corpora, lexicons, and algorithms, NLTK serves as a valuable resource for NLP research, experimentation, and development [5]. Its intuitive interface and detailed documentation make it a preferred choice for both novice and experienced NLP practitioners and researchers. Additionally, NLTK's active community ensures ongoing support and updates for the library.

SpaCy is a popular Python library for natural language processing. It is known for its efficiency, simplicity, and extensive features [6]. With its pre-trained models, SpaCy enables tasks such as tokenization, Part-of-Speech tagging, named entity recognition, and dependency parsing. It also supports various language models and provides word vector representations. SpaCy's user-friendly interface and extensive documentation make it a preferred choice for NLP practitioners and researchers. It offers seamless integration with popular deep learning frameworks like TensorFlow and PyTorch, allowing for easy incorporation of custom models. It provides efficient pipeline processing, allowing users to apply multiple NLP tasks in a single pass. The library also includes powerful visualization capabilities for analyzing linguistic features and dependencies [6]. SpaCy's open-source nature encourages community contributions and enables continuous improvement. It has gained widespread adoption in both academia and industry, serving as a reliable tool for various NLP applications and

research projects.

Taxonomy is a scientific discipline that classifies and categorizes entities into hierarchical structures [7]. Taxonomy plays a significant role in NLP by providing a structured framework for organizing and categorizing linguistic elements. In NLP, taxonomies can be used to classify and categorize text data based on various criteria, such as topic, sentiment, or intent. By leveraging taxonomy, NLP models can achieve better accuracy and efficiency in tasks like text classification, information retrieval, and entity recognition [8]. Additionally, taxonomies can facilitate the development of ontologies, which aid in knowledge representation and semantic understanding within NLP systems. Overall, taxonomy serves as a valuable tool in NLP for organizing and analyzing textual information, enabling more effective language processing and understanding.

Ontology in NLP involves creating a structured representation of knowledge in a domain, clarifying concepts, and enabling effective communication [9]. It enhances system interoperability, offering benefits like reusability and reliability . Ontologies can be formal or informal, encompassing elements such as concepts, relations, axioms, and instances. Constructing an ontology involves analyzing the domain, defining terms, and establishing conceptual connections. It combines contributions from philosophical and AI ontologists for foundational aspects and domain experts for domain-specific knowledge. However, populating the ontology with specific concepts while maintaining consistency can present challenges [9]. Overall, ontology in NLP plays a crucial role in reducing conceptual confusion, fostering shared understanding, and facilitating information exchange and collaboration.

Part-of-Speech (PoS) tagging is a key NLP task that assigns grammatical labels to words, aiding in sentence understanding and enabling diverse NLP applications [10]. It helps extract information, recognize named entities, translate text, perform sentiment analysis, and more. PoS tagging provides essential linguistic details for disambiguation and determining word roles. It enhances NLP models' ability to analyze and process text effectively. For the purposes of this thesis, we use the Stanford CoreNLP [10, 11] NLP toolkit which boasts a comprehensive suite of tools for various tasks like PoS, named entity recognition, dependency parsing, and sentiment analysis. It integrates state-of-the-art models and offers seamless integration of multiple NLP tasks [10, 11].

Dependency parsing is an NLP technique that analyzes sentence structure by identifying syntactic relationships between words [12]. It represents these relationships as labeled edges in a parse tree, capturing grammatical roles and dependencies. It aids tasks like information extraction and machine translation by disambiguating word meanings and capturing grammatical relationships. Various algorithms and machine learning models are used for parsing, trained on annotated data to optimize accuracy. Dependency parsing is vital for understanding word relationships and enabling deeper linguistic analysis in NLP applications [12].

2.2 Mathematical Implementations of Text Processing

2.2.1 Word Embeddings

Word embeddings are compact representations of words that capture their semantic and syntactic features. Derived from large text datasets, they enable machines to understand and process language by capturing word relationships. Widely applied in natural language processing, word embeddings enhance tasks such as language modeling and sentiment analysis. According to authors Liu et al., a text corpus, represented as a sequence S of tokens $(t_1, t_2, ..., t_N)$, is considered [13]. Each token t_i is associated with a dense feature vector h_{t_i}, forming distributed representations of words. On the one hand, global word embedding matrix $E \in \mathbb{R}^{V \times d}$, is learned in *traditional* word embedding techniques, where each row e_i corresponds to the global embedding of a word type i in the vocabulary V. Two well known models that follow traditional word embedding techniques are Word2vec [14] and GloVe [15].

On the other hand, *contextual* embedding methods involve associating each token t_i with a representation h_{t_i} that depends on the entire input sequence S, expressed as $h_{t_i} = f(e_{t_1}, e_{t_2}, ..., e_{t_N})$, where e_{t_j} represents the non-contextualized representation of the input token t_j, and f denotes an aggregation function. Context-dependent representations, which capture sequence-level semantics effectively and handle polysemy, are obtained [13].

2.2.2 Cosine Similarity

Cosine similarity is a widely adopted metric in NLP that enables the comparison of similarity between two vectors [16]. By calculating the cosine of the angle formed between the vectors, cosine similarity captures the alignment of their directions. This metric is derived from the dot product, also known as the inner product, which serves as a similarity measure in linear algebra. The dot product acts as a similarity metric because it tends to yield higher values when the two vectors exhibit large values in the same dimensions [16]. Conversely, if vectors have zeros in different dimensions or are orthogonal, the dot product evaluates to 0, indicating their significant dissimilarity. Suppose our two vectors a and b are N-dimensional, their dot product would then be:

$$\text{dot product}(a, b) = a \cdot b = \sum_{i=1}^{N} a_i b_i = a_1 b_1 + a_2 b_2 + ... + a_N b_N \tag{2.1}$$

However, an inherent limitation of the raw dot product arises from its bias towards longer vectors. Vector length, defined as the magnitude of a vector 2.2, influences the dot product. Longer vectors, often associated with more frequent words, tend to have higher values in each dimension due to their increased co-occurrence with other words and higher co-occurrence values [16].

$$\text{magnitude}(a) = |a| = \sqrt{\sum_{i=1}^{N} a_i^2} \tag{2.2}$$

To address this issue, a normalization step is introduced to the dot product to account for vector length. This normalization involves dividing the dot product by the lengths of the two vectors being compared 2.3. By normalizing for vector length, cosine similarity offers a more balanced and frequency-independent measure of similarity, allowing for more accurate comparisons between words or vectors in NLP applications [16].

$$a \cdot b = |a||b|cos\theta$$
$$\frac{a \cdot b}{|a||b|} = cos\theta = \text{cosine similarity} \tag{2.3}$$

3 Related Work

Although there exist methods that are related to our research, we may not necessarily incorporate them into our method. These methods have similarities to our ideas but were deemed incompatible or deviated too much from our approach. However, they are still valuable for readers to gain insights into different methodologies and research in the field. In the subsequent sections, we will provide a more detailed description of some of these methods. This will allow readers to delve deeper into the methods that are related to our research, despite not being directly integrated into our own methodology.

3.1 Non-Taxonomic Relation Extraction

Authors Nabila et al. [17] developed a method that revolves around identifying and extracting non-taxonomic relations from domain texts, and they achieve this through a series of steps as follows:

1. **Concept Extraction:** The process of concept extraction comprises several key tasks. Firstly, text preprocessing involves performing PoS tagging to assign word types and eliminating stop words using Brill's Rule-based PoS tagger [18]. Secondly, morphological analysis is employed to normalize word forms and reduce dissimilarity. Thirdly, relevant terms are identified using the tf-idf metric, which assesses their importance within the text domain. Moreover, dependency triples generated by MINIPAR shallow parser [19] help determine the subjects and objects of sentences. By matching the identified subjects and objects with the relevant terms, the concepts for the text domain are established.

2. **Predicate Identification:** Text documents undergo preprocessing steps, followed by the elimination of non-domain-related verbs (is, do, has, have,...etc) using a stop word list. The remaining verbs are considered as predicates, which are grouped based on similar meanings using WordNet. Sentences are represented as "predicate (subject, object)" or $p(s, o)$, where p represents a predicate, s represents a subject concept, and o represents an object concept. The extracted rules are organized into the PS set and PO set, containing information about predicate subjects and predicate objects, respectively.

3. **Concept Pair Identification:** In the concept pair identification step, concept pairs are generated based on the appearance of predicates. Each group of predicates in P is used to extract rules from PS and PO. The concepts extracted from PS are collected in S^{new}, while the concepts from PO are collected in O^{new}. The most frequent concepts in S^{new} and O^{new} are selected as concept pairs. The set of concept pairs, q, is therefore defined as the Cartesian product of S^{new} and O^{new}: $q = S^{new} \times O^{new}$.

4. **Relation Extraction and Labeling:** In this step, a list of concept pairs is generated based on groups of predicates with similar meanings. Each concept pair may have multiple predicates associated with it. The goal is to select the most suitable predicate to label the semantic relationship between each concept pair. The suitability is determined by the predicate's degree of support count, which is defined as:

$$\begin{aligned} sprt(q \rightarrow p) &= sprt(S \rightarrow p) + sprt(O \rightarrow p) \\ &= \frac{|s \cup p|}{R} + \frac{|o \cup p|}{R} \end{aligned}$$

Where q is the concept pair containing Subject S and Object O, p is the predicates extracted from rules in PS and PO, s and o are elements that appear in the Subject set S and Object set O respectively, and R is the total rules count.
The predicate(s) with the highest degree of support count are considered as suitable relationships for the concept pair. If multiple predicates have the same highest degree of support count, they are all considered as suitable relationships for that concept pair.

3.2 OntoGain: Unsupervised Ontology Acquisition from Text

Authors Drymonas et al. introduce the OntoGain system [20], which aims to acquire ontologies from unstructured text in an unsupervised manner.

The methodology of OntoGain involves several key steps. The first step is preprocessing, where the OpenNLP suite of tools [21] is used for tokenization, PoS tagging, and shallow parsing. Additionally, the WordNet Java Library is utilized for acquiring word lemma information.

The next step is concept extraction, where OntoGain employs the C/NC-value method for extracting multi-word and nested terms. This approach initially selects noun phrases through linguistic filtering and then uses statistical measures such as C-value and NC-value to determine the candidate noun phrase termhood. These measures are defined as follows:

- **C-value:** C-value is calculated as the ratio of the cumulative frequency of a word sequence in the text to the frequency of occurrence of this sequence as part of larger proposed terms in the same text. It captures the nested nature of multi-word terms.
- **NC-value:** NC-value refines C-value by assigning additional weights to candidate terms that tend to co-occur with specific context words. It takes into account the co-occurrence patterns of terms.

Once the concept extraction is completed, the taxonomy construction phase begins. OntoGain employs two methods for unsupervised taxonomic relation acquisition: **agglomerative hierarchical clustering** and **formal concept analysis (FCA)**. Hierarchical clustering involves merging similar clusters based on term similarity. The similarity between two clusters is typically computed using a similarity measure such as the group average method [22], which

calculates the average similarity across all pairs of concepts within the two clusters.

FCA [23] is another approach used for taxonomy construction [24, 25]. It associates objects (extracted multi-word terms) with their attributes (associated verbs) based on the syntactic dependencies analysis. This relationship is represented using a formal contexts matrix, which serves as input to the FCA algorithm.

The final step is non-taxonomic relation acquisition, where OntoGain explores two approaches: **association rules** and a **probabilistic algorithm**. Association rules are generated using the generalised association rules algorithm, enhanced with more general terms based on the built taxonomy [26]. The predictive apriori algorithm implementation is used to evaluate each rule based on predictive accuracy, which considers support and confidence.

The probabilistic algorithm, on the other hand, relies on conditional probability measures to select the most appropriate non-taxonomic relationships. Conditional probability is estimated by considering the frequency of a dependency relation and propagating it through the respective super concepts in the taxonomy [25].

3.3 Keyphrase Matching and Coreference Resolution

Authors Cattan et al. propose a novel system for evaluating and modeling cross-document coreference resolution [27]. The system addresses the inconsistencies and limitations of existing evaluation protocols and proposes a practical evaluation methodology.

The presented evaluation methodology focuses on raw text input and excludes gold mention annotations and singleton clusters. Instead of relying on gold mentions, the system evaluates coreference clustering based on predicted mentions [27]. This approach provides a more realistic and challenging evaluation, simulating real-world scenarios. Singleton clusters, which are not relevant for coreference resolution's downstream purposes, are omitted from the evaluation to avoid biasing the results towards models that excel at detecting mentions rather than linking them.

To standardize the evaluation process, the methodology introduces topic and corpus level evaluations. The corpus level evaluation assesses sets of documents without specific topic information, making it suitable for datasets lacking document categorization. In the topic level evaluation, each gold topic is evaluated separately, including challenging cases with lexical ambiguity. This setting reflects real-world scenarios where documents are initially collected at the topic level.

To establish baseline results, the paper presents an end-to-end cross-document coreference model inspired by the successful e2e-coref model used for single-document coreference [28]. The model integrates mention detection and coreference link prediction and addresses the

challenge of ordering in the multiple-document setting using an agglomerative clustering-based approach [29, 30, 31, 32].

During training, the model encodes each document separately using $RoBERTa_{LARGE}$ [33]. Long documents are divided into segments, and mention scoring is employed to select the highest-scoring spans. The mention scorer is pre-trained on gold mention spans. The model compares mentions across all documents using a pairwise scorer, training the system using binary cross-entropy loss to optimize the likelihood of correct antecedents.

The paper further expands on the implementation details by introducing topic-level processing, which is performed separately for each topic. It utilizes gold topic segmentation during training and employs a clustering approach during inference when the number of topics is unknown.

In addition to the aforementioned details, the paper presents experimental results demonstrating the effectiveness of the proposed evaluation methodology and the end-to-end coreference model. The experiments showcase the system's capability to handle cross-document coreference resolution challenges and provide insights into its strengths and limitations.

3.4 Taxonomy Construction & TaxoGen

The TaxoGen method [34] is introduced by authors Zhang et al. for unsupervised topic taxonomy construction, which is the process of organizing terms into a hierarchical structure that represents conceptual topics. The method aims to address the limitations of existing pattern-based approaches by considering the topical proximity and semantic correlations among terms.

The overall process of TaxoGen involves embedding concept terms into a latent space and using these embeddings to recursively build the taxonomy. The construction starts with a root node representing the most general topic for the given text corpus. Fine-grained topics are generated level by level through top-down spherical clustering. The construction process continues until a maximum number of levels is reached.

The adaptive spherical clustering module in TaxoGen is responsible for splitting a coarse topic into fine-grained ones. It utilizes the spherical K-means algorithm[35], which groups term embeddings into clusters based on their similarity in embedding directions. The center direction of a topic acts as a semantic focus, and the terms fall around it to represent coherent semantic meaning.

However, two challenges need to be addressed during the recursive construction process. First, not all terms in a topic should be allocated to child topics, as general terms should remain in the parent topic. Second, global term embeddings learned on the entire corpus

may not capture subtle term semantics as the construction moves to lower levels. To tackle these challenges, TaxoGen introduces the adaptive clustering and local embedding modules.

The adaptive clustering module iteratively identifies general terms and refines sub-topics by excluding general terms from the clustering process. This improves the clarity of sub-topic boundaries and enables the detection of additional general terms. The representativeness of a term for a sub-topic is measured based on its popularity (frequency within the sub-topic's documents) and concentration (relevance to the sub-topic compared to sibling topics).

The local embedding module aims to enhance the discriminative power of term embeddings at lower levels of the taxonomy. It learns local term embeddings specific to each topic being split. Two strategies are employed to create a sub-corpus relevant to a topic: clustering-based and retrieval-based. The SkipGram model [14] is then applied to the sub-corpus to obtain tailored term embeddings for splitting the topic.

By combining adaptive spherical clustering and local embedding, TaxoGen constructs topic taxonomies that capture topical proximity and semantic correlations among terms. Experimental results on real datasets demonstrate the effectiveness of TaxoGen compared to baseline methods.

In summary, TaxoGen is an unsupervised method that leverages term embeddings and hierarchical clustering to construct topic taxonomies. It addresses the challenges of term allocation and discriminative power in the recursive construction process, resulting in improved taxonomy quality.

3.5 Ontology Construction

3.5.1 The RENT Algorithm

The RENT algorithm as described by authors Kaushik et al. is designed for automatic term extraction [36]. This algorithm focuses on extracting relevant terms from agricultural text using domain-specific patterns expressed as regular expressions.

To initiate the process, twenty carefully selected patterns are employed based on the analysis of over 1000 pages of agricultural text. These patterns have been curated by experts in the field and were derived from various agricultural handbooks and websites.

The algorithm first extracts candidate terms by applying the regular expressions to the text. The extracted terms are then assigned weights according to certain assumptions. Nouns are given preference over other words that satisfy the same patterns, high-frequency non-stop words are considered significant, and words appearing with multiple patterns receive higher weights.

Although the initial set of candidate terms may include irrelevant terms, a manual inspection process is conducted to remove them. Due to the absence of a suitable threshold value for automated segregation, manual intervention is necessary to ensure the accuracy of the extracted terms.

The algorithm then expands the obtained vocabulary by extracting composite terms, which consist of multiple words (up to three words in length). The decision to set the length threshold at three was made through manual inspection and reference to AGROVOC, an agricultural thesaurus [37].

To extract composite terms, the algorithm employs linguistic filters based on combinations of parts-of-speech. These filters include various noun combinations (NNP, NNP), (NNP, NNS), (NNP, NN), (NNS, NNS), (NN, NN), (NN, NN, NNS), and (NN, NN, NN). Additionally, adjective-noun combinations (JJ, NNP), (JJ, NN), and (JJ, NN) are used. Here, N, J, P, and S stand for noun, adjective, preposition, and subordinating conjunction, respectively.

Composite terms that include at least one constituent word present in the list of candidate terms are considered valid and included in the final list of terms generated by the algorithm.

3.5.2 HCHIRSIM Algorithm

The HCHIRSIM (Hybrid Chir-Statistic and Similarity) algorithm [38] aims to construct an ontology by identifying representative concepts and relevant websites for a specific domain.

According to B. Frikh and A. S. Djaanfar, the algorithm begins by analyzing a large number of websites through a k-means clustering algorithm to create initial clusters. This helps in organizing the data based on similarities. Next, an initial keyword is chosen to represent the domain, along with predefined parameters that guide the search and concept selection process [39].

Candidate concepts are extracted by examining the neighborhood of the initial keyword. Words appearing before and after the keyword are considered as potential concepts. For each candidate concept, a score $S(w)$ is calculated using a specific measure, which helps in assessing its importance or relevance. The candidate concepts are then sorted in descending order based on their $S(w)$. The top l concepts with the highest scores are selected from the sorted list.

These selected concepts are incorporated as classes or instances into the ontology. Additionally, the URLs of the websites from which these concepts were extracted are recorded for reference. For each concept incorporated into the ontology, a new keyword is generated by combining it with the initial keyword. This process is recursively repeated until a desired

depth level is reached or no further results are found.

Finally, a refinement process is performed to improve the taxonomy's structure, ensuring a more compact representation and eliminating redundancy.

4 Methodology

In this chapter, we discuss how we create our ontology of NLP concepts out of a dataset of research papers and based on the foundations set by Klimek's work [3]. The process is described in four parts. First, we describe how we choose the hand-picked concepts and relations that belong in the upper levels of our hierarchy. Next, we explain how we validate our keyphrases before further processing. After that, we delve into the ways we can combine similar keyphrases. Further, we explain how we infer hierarchical relations between our concepts. Finally, we show how we infer more complex relations between these concepts.

4.1 Manual Taxonomy Construction

The top layers of a taxonomy are a user's first point of interaction with the navigation of a complex hierarchy. An unclear high-level layer can have ripple effects into subsequent interactions and cause confusion and miscommunication, as each concept in a taxonomy is dependent on its predecessors located in previous layers.

Therefore, In order to ensure the quality of navigation for the users of our ontology, we decide to manually define the top levels of the hierarchy, and only use automated methods to extract relations in the deeper levels of the hierarchy. This approach is inspired by [40], who manually defined the top two levels of concepts in their hierarchy.

The results of the previously built taxonomy [3] show that while using these automated tools for hierarchy construction is convenient, they can generate faulty or less-than-ideal relations between concepts. Therefore, the manually defined layers are there to ensure correctness and reliability in the taxonomy's critical entry point.

The challenge in this method is finding the ideal candidates from a broad spectrum of NLP topics to be chosen as our top-level concepts. We comb through the following sources of information to find these topics:

- The Computer Science Ontology (CSO) is a comprehensive research categorization system in the field of Computer Science. It was created using the Klink-2 algorithm, which analyzed a dataset of 16 million publications. CSO combines semantic technologies, machine learning, and external knowledge sources. Experts also manually refined some relationships during the preparation of surveys. CSO covers various research areas,

including its primary root in Computer Science, as well as secondary roots like Linguistics and Geometry. It offers benefits over manual categorizations by characterizing higher-level research areas and allowing easy updates with new publications [41].

- The Association for Computational Linguistics (ACL) conferences are prominent events in natural language processing and computational linguistics. They bring together researchers, practitioners, and industry professionals to present advancements and discuss applications in language-related research. With research papers, keynote speeches, workshops, and tutorials, ACL covers topics like machine translation, sentiment analysis, information extraction, and dialogue systems. These conferences foster collaboration, disseminate knowledge, and drive innovation in computational linguistics. The topics covered in previous ACL conferences (for instance Dialog and Interactive Systems in ACL 2021) were used as our top-level concepts [42].

- By examining multiple papers (e.g. [43, 44, 45]), a comprehensive collection of NLP-related terms is gathered, which aids in understanding the breadth and depth of the field. These terms may include various NLP tasks such as sentiment analysis, machine translation, named entity recognition, and more. Additionally, they could cover different methodologies, such as neural networks, statistical modeling, and rule-based systems. Overall, the collected terms provide a valuable resource for exploring and categorizing the multidimensional landscape of NLP research.

Following this, we build a prototype taxonomy with two to three layers beneath the root node of 'NLP'. Since this step primarily depends on conjecture, additional manipulations need to be made to the taxonomy in order for it to fit the ideal standard. Therefore, we pass the taxonomy through several qualitative testing processes with researchers in the NLP domain to ensure the terms are approved by experts in the field (the full description of this process can be found in 5.1).

Figure 4.1: First prototype of the top-levels taxonomy.

Eventually, we reach a final prototype of the taxonomy that satisfies the largest common denominator of the researchers' expectations.

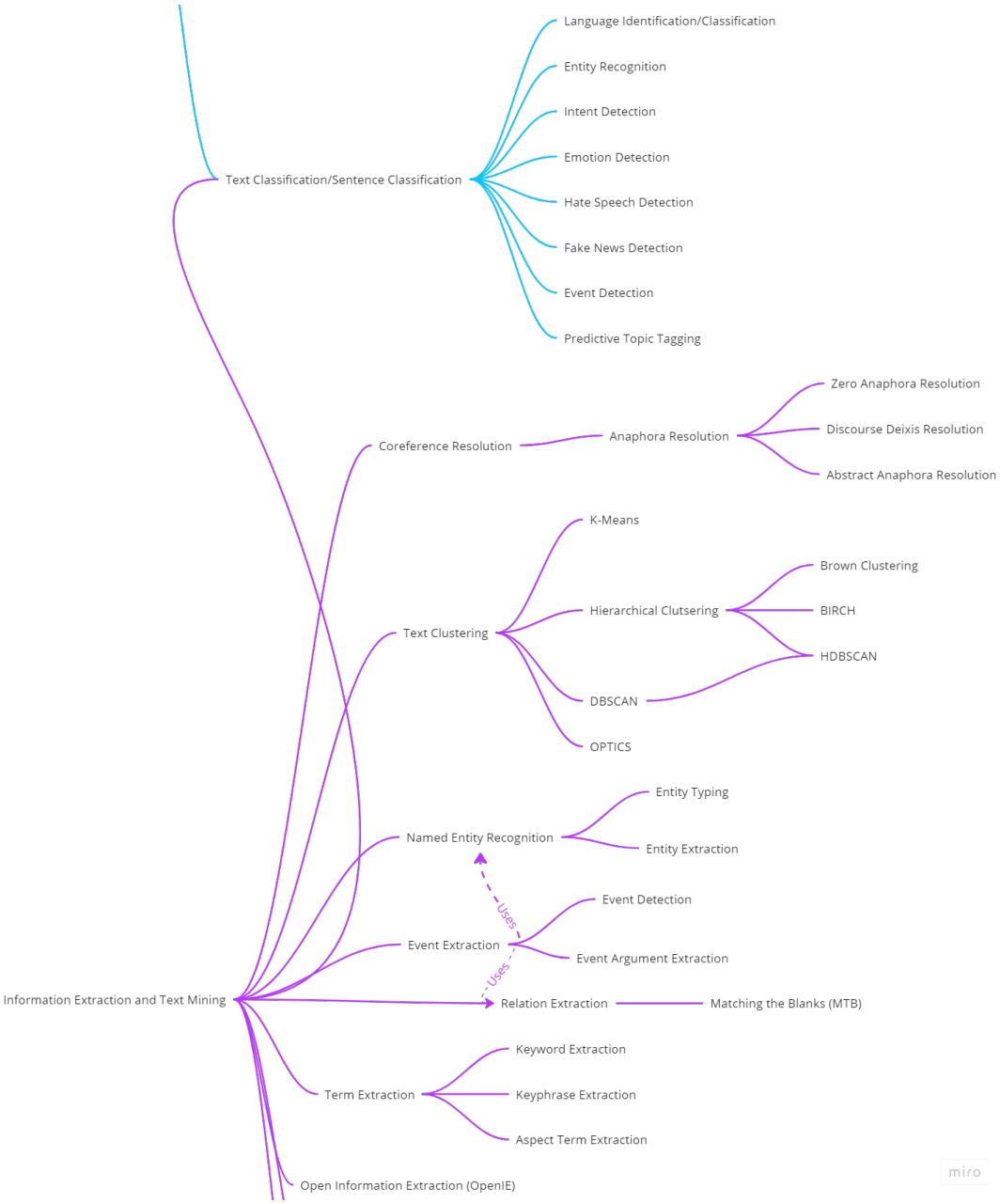

Figure 4.2: Snippet from the final prototype of the top-levels taxonomy.

4.2 Keyphrase Validation

For our intents and purposes, whereas a keyword consists of one token, a keyphrase may consist of several. Keyphrases represent important or relevant terms in a paper, and they are used as the basis from which we build our ontology [3]. We rely on the existing work regarding keyphrase extraction. Instead, our work aims to add a post-processing step in the keyphrase extraction module. This new module is inspired by the work from [46], and aims to ensure that the keyphrases we process are of higher quality and facilitate further processing. We call the 'Keyphrase Validator Component', and comprises of three functions:

- ***First,*** it trims all keyphrases that include acronyms (e.g: 'machine learning (ml)' becomes 'machine learning') and that end or begin with any of the following punctuation marks: !#$%*+,.:;<=>?@[]^|()/{}\~.
- ***Second,*** it discards keyphrases that still contain any of those same punctuation marks that remain after the first step. We also discard keyphrases that start with a number or contain only one character.
- ***Third***, for keyphrases that start or end with a hyphen, it extends the keyphrase to encapsulate the missing attached tokens if they exist, and discards them otherwise.
- ***Fourth*** and last, the information content (IC) score of each keyphrase is retrieved. This score is representative of how specific or generic a keyphrase is. We retrieve our score based on Wordnet's SemCor corpus, and discard any keyphrase which is deemed too generic and falls below our threshold of 10 [46].

4.3 Concept Deduction from Keyphrases

With a list of keyphrases that have been extracted, cleaned, and filtered, we may still have keyphrases that are similar in nature and refer to the same meaning or concept. Thus, we need a reliable method to merge these keyphrases so that they are embodied by the same concept.

We cannot use simple means to compare semantic similarities of keyphrases in isolation with the use of tools such as WordNet similarity measures, since they consider only word-level semantics and relationships, and fail to capture the nuances of the keyphrase. For example, the keyphrases 'bank' and 'branch' could be interpreted in the domain of banking institutes and locations, or they could refer to riverbanks and tributaries.

Though there already exists a concept merging module based on the BERT-based Lexical Substitution (BERT-LS) method [3], the results can vary in coherence. Therefore, we present three additional approaches for deducing concepts.

4.3.1 BART-Based Lexical Substituion

As mentioned, the BERT-LS method currently implemented has its shortcomings. It specifically struggles when it comes to multi-token keyphrases in two ways. First, some words could be represented with more than one token, and since BERT operates on a word-piece basis, only the largest word-piece is considered. The other problem lies with the processing of multi-word keyphrases. In such cases, the current implementation applies BERT-LS to generate alternative tokens for each word, and ranks the mean scores of the various combinations of tokens generated for the final substitutes. Therefore, if a keyphrase consisted of 3 tokens (such as Lexical Sentiment Analysis), it could only generate substitutes that also consisted of 3 tokens [3, 47].

The BART-LS (Bidirectional Autoregressive Transformer for Lexical Substitution) approach is an alternative method for generating substitutes for keyphrases. It can be viewed as a more generalized version of BERT-LS. BART is a denoising autoencoder built with a sequence-to-sequence model. It pre-trains the model combining Bidirectional and Auto-regressive Transformers. This pre-training task consists of corrupting an original document with random noise and then the model is trained to reconstruct the original text [48].

BART offers flexibility by allowing arbitrary transformations on the input text, including changes in length. Different noising approaches were tested and evaluated, finding the best performance by randomly shuffling sentence order and using a novel in-filling scheme where random lengths of a span of text are replaced with a single mask token. This approach improves the model's understanding of sentence length and enables it to make longer-range transformations [48].

BART demonstrates impressive performance in text generation tasks and shows strong performance in comprehension tasks as well. It achieves comparable results to RoBERTa when trained with similar resources on widely-used benchmarks such as GLUE and SQuAD. Furthermore, BART sets new standards in abstractive dialogue, question answering, and summarization tasks, surpassing previous approaches by a margin of 3.5 ROUGE on the XSum dataset, which measures summarization quality [48].

In the context of concept deduction, BART-LS utilizes the ability of BART to generate plausible alternative phrases or substitutions for a given keyphrase. Given a sentence with a target keyphrase, we replace the keyphrase with a <mask> token and use BART to fill in and try to replicate the original text, similarly to how the model was trained. Unfortunately, in this scenario, BART has no point of reference for the original semantics of the masked token. Therefore, inspired by the works of M. Pogoda, we insert the original keyphrase at the beginning of the input text to provide BART with further context [49].

E.g: We explore different Machine Translation approaches.
The input to the BART model becomes:
[Machine Translation] We explore different <mask> approaches.

However, since BART is trained on noising all the input, it can sometimes predict and change parts of the sentence that go beyond just the masked portion. Therefore we filter the outputs generated as such:

- We place a limit of up to five newly generated tokens.
- We discard newly generated keyphrases that fail the validation check as specified in 4.2.
- We discard generated outputs that made any changes to the input beyond just the <mask> token.

Both the existing BERT-LS and new BART-LS methods rely on generating synonyms or substitutes to the keyphrases. The amount of overlap between the substitutes of different keyphrases determines whether they are considered the same concept. In the previous implementation, this percentage was chosen to be 5%, as higher percentages proved ineffective in merging any keyphrases [3]. However, as our method generates more substitutes for each keyphrase on top of the synonyms already generated with BERT-LS, we increase this value to 10% after experimenting with different percentage thresholds.

4.3.2 SciConceptMiner Approach

In this approach, we leverage the use of online documentation of NLP topics to determine whether different keyphrases should refer to the same concept. We adapt an approach detailed by Shen et al. that links concepts together based on their web search relevance. The Bing Web Search API is utilized to retrieve the search results [50, 51].

The approach examines the overlap in the top search result URLs between pairs of keyphrases. If there is a substantial enough overlap, it indicates that the candidates are synonymous with the same concept. For our purposes, we retrieve the top 30 results, and place an overlap threshold of 50%.

In order to ensure the quality of the results, an allowed-list is employed [50]. Only the results originating from reputable academic domains listed in the allowed-list are accepted. In addition, we amend each search query to include the terms 'in NLP' appended at the end of the keyphrase to focus the webpage results. This ensures that keyphrases won't have false-positives overlapping in domains unrelated to NLP or from online dictionaries that we find unsuitable. The list of allowed domains can be seen as follows:

- Wikipedia.org
- Kaggle.com
- GitHub.com
- Spacy.io
- Nlp.johnsnowlabs.com
- Medium.com
- Towardsdatascience.com
- Pub.towardsai.net
- Kdnuggets.com
- Geeksforgeeks.org
- Datascience.stackexchange.com
- Paperswithcode.com
- Pytorch.org
- Arxiv.org
- Researchgate.net
- Aclanthology.org
- Huggingface.co
- Microsoft.com/en-us/research
- Ai.facebook.com
- Ai.google
- Allenai.org
- Amazon.science
- Dl.acm.org
- Ieeexplore.ieee.org
- Stanfordnlp.github.io/CoreNLP
- Nltk.org

However, a major drawback of this approach is the limitations of the Bing Web Search API. For research purposes, we use API's free subscription tier which limits search requests to 3 per second, and up to 1000 requests a month. In our implementation, we utilize this method for only as far as the specified API key is valid before it terminates [52].

4.3.3 Sentence Transformers Merging Approach

Considering that our keyphrases are extracted from a corpus of papers related to the field of NLP, if two keyphrases share a token in common and their embeddings have high cosine-similarity, even in isolation of their context, then we hypothesize that the likelihood of them being connected to the same concept is high.

We base this principle on the works of Dessí et al. We construct an index of tokens that link together all keyphrases containing that same token. Any two keyphrases are compared if they share at least one token in common. We do this by encoding the keyphrases using the paraphrase-distilroberta-base-v2 model from SentenceTransormers framework [46].

The paraphrase-distilroberta-base-v2 transformer model is a pre-trained language model specifically designed for generating paraphrases of input sentences and is based on the distilroberta-base architecture. The distilroberta-base model is a smaller and faster version of the RoBERTa model, which itself is a robust transformer-based architecture that has been trained on a large corpus of text data [53, 54].

If the cosine similarity of these keyphrases is greater than a specified threshold (we choose this similarity threshold to be ≥ 0.9 as it yielded the best results, and aligns with the findings of Dessí et al.), then these keyphrases are merged [46].

As an example, if we have the keyphrases Emotion Detection and Emotion Recognition, then these values will be linked to the key 'Emotion', and since their cosine-similarity is at least 0.9, these keyphrases are merged. However, if we also have the keyphrase Sentiment Detection, then it is linked to Emotion Detection with the key 'Detection', and subsequently merged if their cosine-similarity meets the threshold. Thus, all three keyphrases will be merged into one concept.

4.4 Construction of Concept Hierarchy

With our keyphrases merged into concepts, our next step is to construct taxonomic relations between these concepts in order to form our hierarchy. The output from this step is an acyclic directed graph where each vertex V represents a concept and each edge $E(V_1, V_2)$ represents a taxonomic relation between two vertices, V_1 and V_2, where V_1 is a parent of V_2. A taxonomic relation in a hierarchy denotes one concept as broader or encompassing of another. We choose our graph to be acyclic because we want the user to delve deeper into a topic as they explore

our hierarchy and avoid falling into repeating loops.

As specified in Klimek's paper [3], we consider two types of taxonomic relations:

- **A Hypernym-Hyponym relation**, where one concept, the hyponym, is a type or instance of another, the hypernym. As an example, the concept 'Cat' is a hyponym of 'Animal'.
- **A Holonym-Meronym relation**, where one concept, the meronym, represents a part or component of the meronym. As an example, the concept 'Leaf' is a meronym of 'Tree'.

By incorporating these taxonomic relations into our hierarchy, we create a comprehensive framework for navigating and exploring the interconnected concepts in a logical and organized manner.

4.4.1 String Inclusion Approach

This is a straightforward method proposed by Tuan et al. for determining taxonomic relations, and it involves testing string inclusion. This means that when one term is included in another longer or more specific term, we can make an educated estimation about a parent-child relation existing between the two. For instance, if the term "Machine Translation" is a substring of "Neural Machine Translation", we can infer that the former is a hypernym of the latter [55].

We use Tuan et al.'s enhancement of this idea, the SIWN algorithm (String Inclusion with WordNet). This method leverages WordNet and synsets to expand on the string inclusion idea. (refer to section 2.1 for an explanation of synsets) [55].

- We define $W_1 \gg W_2$ to mean that W_2 is a direct or inherited hyponym of W_1, according to WordNet. This is determined if a synset of W_1 is a direct or inherited hypernym of W_2.
- We define $W_1 \approx W_2$ to mean that W_1 and W_2 are considered similar according to WordNet. This is determined if W_1 and W_2 contain a synset in common.

Given concepts C_1 and C_2, we denote $K_1 \in C_1$ to be any keyphrase that has been merged to form C_1, and similarly, $K_2 \in C_2$. For any combination of K_1 and K_2, we examine each word W_1 in K_1 from left to right. Our algorithm checks for W_1 if there is a corresponding word W_2 in K_2 such that $W_1 \gg W_2$ or $W_1 \approx W_2$. We conclude that K_1 is a hypernym of K_2 (and therefore C_1 is a hypernym of C_2) if every word in K_1 has a corresponding word in K_2 such that there exists at least one $\gg$ relation.

4.4.2 Weighted Ensemble Approach

Looking at Klimek's hierarchy construction results, we see that the Lexical Syntactic Method bears some opportunity for improvement. We take inspiration from Tuan et al.'s Lexical-syntactic Pattern method and insert new patterns into our existing implementation to capture more instances of hierarchical relationships [55]. The new rules are as follow:

1. `KEYPHRASE is (a|an) KEYPHRASE`
2. `KEYPHRASE is a (kind|type) of KEYPHRASE`

However, since we value a high quality resulting ontology, we attempt to supersede the best results presented in the current implementation by combining our three hierarchy construction approaches in an ensemble. We place equal weight on each of our methods and only pick the taxonomic relations that have been extracted by at least two out of our three methods: Subsumption method, Lexical Syntactic Method, and String Inclusion Method.

4.5 Extraction of Ontological Relations

At this point, we have a taxonomic representation of NLP concepts, containing simple parent-child relationships between them. The graph specified in section 4.4 captures the hierarchical structure of concepts. However, to enrich our ontology further, we recognize the need to incorporate non-taxonomic relations. These relations are denoted by triples (S, P, O) belonging to the set T where T is a subset of E, S represents the subject belonging to the set V, P signifies the predicate, and O denotes the object, also belonging to the set V. By incorporating these non-taxonomic relations, we can capture a broader range of semantic associations and dependencies within the NLP domain.

In this context, a predicate refers to the verb or relational term that connects the subject and object within a triple. It represents the action or relationship expressed by the triple. For example, in the triple (Semantic Search, uses, Text Representation), "uses" acts as the predicate, indicating the usage relationship between the subject "Semantic Search" and the object "Text Representation".

We employ two main methods for extracting such triples, the Dependency Tree Paths Based Approach, and the Part-of-Speech Based Approach. Both rely on the use of the Stanford CoreNLP module, which is a widely used JVM-based toolkit in core natural language analysis developed by Stanford University. It provides a wide variety of text processing functionalities such as PoS, Named-Entity Recognition (NER), Syntactic Parsing, Coreference Resolution, Sentiment Analysis, Dependency Parsing and more [10, 46].

4.5.1 Dependency Tree Paths Based Approach

This approach is based on the Stanford CoreNLP Dependency Parser. It aims to extract meaningful triples from sentences by leveraging pre-defined paths on dependency trees.

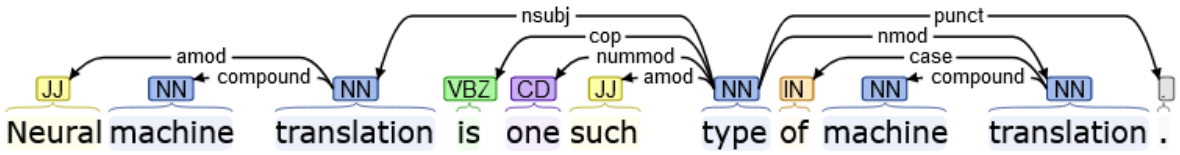

Figure 4.3: Example of Stanford CoreNLP Dependency Parser.

To identify frequently occurring and high-quality paths, a representative sample of scientific papers was utilized by Dessí et al. This sample was employed to construct a collection of dependency trees (DT) along with associated concepts for each tree. By examining the shortest paths within the dependency trees, which included at least one verb between the tokens of a concept pair, the authors sought to identify relationships between these concepts. The 50 most frequently occurring shortest paths were considered as potential candidates for meaningful paths, capable of capturing text fragments that could be transformed into triples for integration into an ontology or knowledge graph (KG) [46].

In order to assess the validity of the candidate paths, four researchers specializing in Computer Science manually annotated 20 triples for each path. Through a meticulous evaluation process, the correctness of the extracted triples was determined. Paths that yielded a correctness rate exceeding 60% were deemed as strong contenders, resulting in a final selection of 12 paths [46].

The twelve dependency tree paths deduced by Dessí [46] to be 'good paths' are:

1. **'nsubj', 'obj':** The subject of the sentence is connected to the direct object through a verb.
2. **'acl:relcl', 'obj':** An adjectival clause modifies the object of the main clause.
3. **'nsubj', 'obj', 'conj':** The subject and the direct object are connected through coordination, indicating multiple subjects or objects in the sentence.
4. **'conj', 'obl', 'nsubj:pass':** Two elements are connected through coordination, with one element being an oblique argument and the other being the passive subject of the verb.
5. **'acl', 'obj':** An adjectival clause modifies the object of the main clause.
6. **'nmod', 'nsubj', 'obj':** A nominal modifier connects a noun phrase to both the subject and the direct object of the verb.
7. **'obl', 'nsubj:pass':** An oblique argument is connected to the passive subject of the verb.
8. **'nsubj', 'obj', 'nmod':** The subject and the direct object are connected to a nominal modifier indicating a relationship with another noun phrase.
9. **'acl:relcl', 'obl':** An adjectival clause modifies an oblique argument.

10. **'obl', 'acl':** An oblique argument is connected to an adjectival clause.
11. **'nmod', 'obj', 'acl':** A nominal modifier connects a noun phrase to the object of the verb, and an adjectival clause further modifies the object.
12. **'acl', 'obj', 'nmod':** An adjectival clause modifies the object of the main clause, and a nominal modifier connects another noun phrase to the modified object.

For our use case, when we input the sentence containing at least two keyphrases, each belonging to a concept, the module builds the dependency tree and references the twelve 'good paths' to extract the relationships between any pair of keyphrases. We leverage frequent paths on the dependency trees, and thus, identify prominent textual patterns to construct our triples.

4.5.2 PoS Tag-Based Relationships Extraction Approach

This approach builds upon the work of Dessí et al., who introduced a simple but broader method for triple extraction utilizing the Stanford CoreNLP PoS Tagger. The PoS Tagger is a tool employed to assign Part-of-Speech (PoS) tags to each term in a sentence that contains at least two keyphrases associated with different concepts. By leveraging PoS tags, this method identifies verbs situated between pairs of concepts within a sentence.

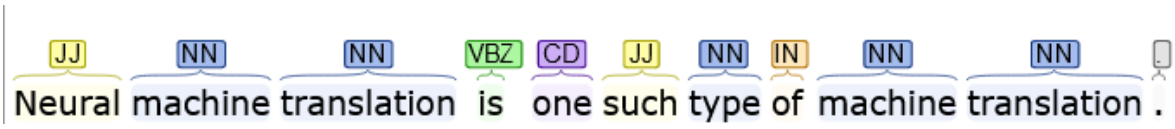

Figure 4.4: Example of Stanford CoreNLP Part-of-Speech.

Specifically, given a sentence and the set of concepts C, the verbs $V = v_0, ..., v_z$ occurring between each pair of keyphrases $(K_1, K_2)|K_1 \in C_1, K_2 \in C_2, C_1 \neq C_2$ are captured. These verbs are then utilized to form triples $< C_1, v, C_2 >$, where $v \in V$. To mitigate potential noise, this approach restricts the processing only to verbs between concept pairs that have a maximum distance of 10 tokens between them. The output of this module is a collection of extracted triples [46].

4.5.3 Relation Mapping

The previous two methods produce a wide variety of triples with predicates that can range greatly in meaning. However, it's also possible to have triples with similar meaning predicates that can cause confusion in the ontology should they exist together. For example, the verbs Apply, Employ, and Use all hold similar meanings and redundantly serve the same purpose. In addition, it would be beneficial for the comprehension of the user if the ontological relations present in the graph were focused to a limited set of verbs with clear and distinct meanings.

Therefore, we employ the solution proposed by Dessí et al. and choose to map all the

verbs we encounter during parsing to a set of 38 accepted representative verbs [46]. These verbs are:

uses, produces, provides, supports, proposes, base, improves, includes, identify, acquires, adapts, analyzes, links, matches, manages, interacts, queries, guides, automates, lacks, limits, affects, processes, contributes, causes, classifies, annotates, visualizes, predicts, standardizes, learns, executes, outperforms, extracts, highlights, transfers, solves, discusses.

The original list of predicates also includes the verb "is", but we discard it since it represents a hypernym-hyponym relation which we have already extracted in the previous modules. We also discard any triple containing a verb that doesn't appear as one of the mapped 464 verbs. The verb mapping (of 194 verbs) is originally based on a list compiled by Dessí et al. who constructed it by employing a hierarchical clustering algorithm on the Word2Vec embeddings of the verbs [56]. Within each cluster, the representative verb was determined by selecting the verb closest to the centroid. This representative verb was then assigned to all other verbs within the same cluster [56].

It was then extended with an additional mapping of 270 verbs using VerbNet. VerbNet is a lexicon that organizes verbs into hierarchical classes that consist of a set of verbs based on their syntactic and semantic information [46, 57].

5 Evaluation

In this chapter, we present our results in evaluating the different methods introduced throughout our ontology-producing pipeline. First, we look at the evaluation performed for the manual taxonomy construction. Then, we investigate the new methods for concept merging and how they compare to the existing implementation. Moreover, we do the same with the new methods for hierarchy construction and how they compare to the existing implementation. And finally, we take a look at the newly added ontological relations extraction methods.

For all of our evaluations, we relied on user studies. Our participants consisted of seven researchers that work within the field of NLP at Technical University of Munich (TUM).

5.1 Analysing the Manual Taxonomy Creation

The creation of the manual taxonomy was evaluated in two parts. First, it underwent an iterative and qualitative evaluation process. Then, we performed a quantitative test to evaluate the correctness and reliability of the final built prototype.

5.1.1 Qualitative Manual Taxonomy Construction Evaluation

As the hierarchy presented by us is ultimately meant to be utilized for research purposes, our participants proved to be ideal candidates for influencing the construction of the manual taxonomy. Therefore, when we were in the process of gathering requirements from our users to be represented in our taxonomy, we held iterative tests, where changes were applied to the taxonomy in between sessions.

These tests were semi-structured and qualitative in nature. Instead of collecting a strict set of information, we held free-flowing discussions with guided questions and accompanied by a presentation. The participants were asked about their general thoughts and opinions on the taxonomy, as well as more detailed questions regarding topics they felt were missing, misrepresented, superfluous, too vague, or too specific.

5.1.2 Quantitative Manual Taxonomy Construction Evaluation

Once we had formed our final prototype of the manual taxonomy, we interviewed each participant once more. The participants were requested beforehand to prepare a list of 5 or more concepts in the domain of NLP. We confirmed and assured the presence of these concepts in our taxonomy, and then met with the participant to observe how they navigated

our graph to find each of their specified concepts. This made sure that they were familiar with the topics they were looking for as they had requested them themselves. In our taxonomic graph, the children of any concept vertex were collapsed, and the participant had to click to expand and see the child concepts of the parent concept they visited. Every 'visit' or 'expansion' is considered one step for the purposes of our evaluation.

In this way, we were able to record the amount of steps it took for each participant to reach the target concept in the taxonomy. In addition, upon finding the topic, the participants were asked if they found the location of the concept correct, i.e. if they thought the concept's relation to the hierarchical structure was valid.

Table 5.1: Table comparing our resulting relation accuracy against that of previously used methods [3].

Approach	Relation Accuracy
Manual Taxonomy Creation	**0.988**
Subsumption Method, SCOPUS	0.860
TaxoGen, DBLP	0.775

As expected from a manually constructed taxonomy, the relation accuracy is near perfect, especially when compared to the relation accuracy scores of automated implementations. Though the metric used to measure the relation accuracy of the manual taxonomy is not the exact same as the one used to measure the other automatic methods due to the nature of our manual tests, they are close enough that we can make a fair comparison to decide that the manually constructed taxonomy performs the best by far in terms of correctness.

We also note another score, the MAPE (Mean Absolute Percentage Error) score. MAPE is a metric we use to note the reliability of our taxonomy by measuring the percentage of error or 'stumbles' the user takes when navigating the taxonomy to reach the target concept. The MAPE score is calculated using the following equation:

$$\text{MAPE} = \frac{1}{n} \sum \frac{|\text{Total Steps Taken} - \text{Ideal \# Steps}|}{\text{Ideal \# Steps}} \tag{5.1}$$

Where n denotes the amount of concept-finding tasks that were performed. The results can be seen in figure 5.1.

Our MAPE score is **0.478**, and we set this as a benchmark for future iterations of this manual taxonomy to be compared to. If a manual taxonomy manages a lower score, then their user is, on average, able to find and reach their target concept in a more reliable manner.

Interviewee	Topic	# of Steps	Ideal Steps	Correct
#1	semantic parsing	2	2	1
	data-to-text generation	2	2	1
	conversational QA	3	3	1
	dialogue policy learning	5	5	1
	entity linking	12	6	1
	differential privacy	5	4	1
	chatGPT	4	4	1
#2	fact verification	6	2	1
	relation extraction	2	2	1
	sentiment analysis	1	1	1
	generative question answering	2	2	1
	knowledge graph embedding	7	7	1
	DeBERTa	4	4	1
	word edit distance	3	3	1
	green NLP	2	2	1
#3	Text Summarization	2	2	1
	Seq2Seq models	8	3	1
	Generative Question Answering	2	2	1
	Transformers	3	3	1
	Efficient Transformers	6	4	1
	Topic Modeling	2	2	1
#4	Information retrieval	1	1	1
	Natural Language Generation	1	1	1
	Dependency Parsing	3	3	1
	Spoken Language Understanding	6	3	1
	Event Extraction	3	2	1
	Coreference Resolution	9	2	1
	Question Answering	2	2	1
	Entity Linking	12	6	1
	Entity Resolution	6	6	1
#5	Interpretable/Explainable/Tansparent NLP	2	2	1
	Trustworthy NLP	1	1	0
	Sarcasm/Irony Detection	4	3	1
	Intent Detection	2	2	1
	Sentence Classification	6	2	1
#6	Irony and sarcasm detection	15	3	0
	Authorship obfuscation	4	4	1
	Trustworthy NLP	1	1	1
	Video captioning	3	3	1
	String matching	3	3	1
	Slang detection and identification	3	3	1
#7	Text summarization	2	2	1
	Transformers	7	3	1
	LLMs	3	3	1
	Sentiment Analysis	1	1	1
	Question Answering	8	2	1
	ROUGE	3	3	1
	Few Shot Learning	7	3	1
	NER	2	2	1

Figure 5.1: Results of the quantitative manual taxonomy evaluation.

5.2 Analysing Concept Merging Coherence

We evaluate our new methods for Concept Merging. As the existing implementation already utilizes its own Concept Merging technique, we employ the same evaluation method as the one used by Klimek [3].

For the user studies, it is explained that an evaluation method is used that was introduced by Zhang et al. [34] called the term intrusion user study. In this test, we measure the coherence of the concepts that we generate. We do this by sampling 30 random concepts generated by a method. We then introduce an 'intruder' keyphrase picked from a disconnected concept to muddy the test data for every concept [3, 34].

Term 0	Term 1	Term 2	Term 3	Term 4	Term 5	Real Intruder Index (0 - 5)	Evaluator Intruder Index (0 - 5)
use convolutional neural	convolutional neural network	convolutional neural	text classification			3	3
processing	embeddings and weights	processing performed	processing works			1	1
nlp literature	word embeddings freely	word embeddings	word embedding	new word embedding		0	0
natural language questions	artificial neural	natural languages	natural language answer	natural language	natural language question	1	1
semantic similarity	automatically obtained synonyms	semantic similarities	semantic feature similarity			1	1
...							
Correct guesses (%)							**98.10%**

Figure 5.2: Snippet of the sentence transformer results, with the final concept coherence at the bottom (in %).

We set a task for our participants: for every concept, they must identify the intruder keyphrase from a list of keyphrases that represent the same concept. If they are able to do so, then it proves that the concept has a clear identity, and the correct keyphrases cohere to the same meaning.

Table 5.2: Table comparing the concept coherence resulting from our different methods against previously used ones [3]. TaxoGen results also included [34].

Approach	Concept Coherence
BART-LS and BERT-LS, SCOPUS	**0.816**
Sentence Transformers, SCOPUS	**0.981**
SciConceptMiner, SCOPUS	**0.988**
BERT-LS, SCOPUS	0.747
TaxoGen, DBLP	0.728

Looking at the results presented in table 5.2, we can see that sentence transformers method achieved a coherence score of **0.981**, the BART-LS and BERT-LS achieved a coherence score

of **0.816**, and the SciConceptMiner achieved a coherence score of **0.988**. These new methods all outperformed existing solutions including the BERT-LS method with a score of 0.747 and TaxoGen with a score of 0.728.

As a final note, Klimek acknowledges that while the TaxoGen method is run on a separate dataset, we can still use its score values for comparison considering that SCOPUS and DBLP have overlap in the computer science domain [3].

5.3 Analysing Taxonomic Relation Construction

We evaluate our new methods for the construction of Taxonomic relations. As the existing implementation already utilizes its own Concept Merging technique, we employ the same evaluation method as the one used by Klimek [3].

For the user studies, it is once again explained that an evaluation method is used that was introduced by Zhang et al. [34]. In this test, we measure the accuracy of the relations that we construct. We do this by sampling 30 random taxonomic relations generated by a method [3, 34].

Parent	Child	Interviewee #1	Interviewee #2	Interviewee #3	Interviewee #4	Interviewee #5	Majority
semantic	capturing crucial semantic	1	0	0	0	1	0
Term Memory	term memory network	1	1	1	1	1	1
language processing	natural language processing	1	1	1	1	1	1
neural network	convolutional neural network	1	1	1	1	1	1
semantic	semantic web reasoning	1	1	1	1	1	1
...							
Correctness (%)							**90.00%**

Figure 5.3: Snippet of the weighted ensemble results, with the final relation accuracy at the bottom (in %).

Each participant described whether the relation was considered correct or false. With this evaluation, we had five participants in total, and therefore, we chose a relation to be correct if the majority voted as such.

Table 5.3: Table comparing the relation accuracy resulting form our different methods against previously used ones [3]. TaxoGen results also included[34].

Approach	Relation Accuracy
String Inclusion, SCOPUS	**0.667**
Weighted Ensemble, SCOPUS	**0.900**
Lexical Syntactic Method, SCOPUS	0.440
Subsumption, SCOPUS	0.860
TaxoGen, DBLP	0.775

Looking at the results presented in table 5.3, we can see that the string inclusion method achieved a relation accuracy score of **0.667** and the weighted ensemble method achieved a relation accuracy score of **0.900**. While the string inclusion method performed okay, it did not surpass the existing subsumption method score. On the other hand, the weighted ensemble method outperformed the existing solutions including the subsumption method with a score of 0.860, the lexical syntactic method with a score of 0.440, and TaxoGen with a score of 0.775.

The note in section 5.2 regarding the differing dataset of TaxoGen still applies.

5.4 Analysing Ontological Relation Extraction

We evaluate our new methods for the extraction of the Ontological relations. This step has no existing counterpart in Klimek's implementation, but we employ the same evaluation method as the one used in 5.3 due to the similarity of the nature of the evaluation [3].

Subject	Predicate/ Verb	Object	Interviewee #1	Interviewee #2	Interviewee #3	Interviewee #4	Interviewee #5	Majority
indexed journal articles	matches	keywords	0	1	1	1	0	1
GANCoder	produces	programming language codes	1	1	1	1	1	1
deep bidirectional transformers	uses	Improved prediction	0	0	0	1	1	0
lexicon matching	includes	character classifier	0	1	1	1	0	1
base WordNet	affects	replace rare words	0	0	0	0	0	0
...								
Correctness (%)								**53.33%**

Figure 5.4: Snippet of the dependency tree paths results, with the final relation accuracy at the bottom (in %).

We sample 30 random ontological relations generated by a method. Each participant described whether the relation was considered correct or false. With this evaluation, we once again had five participants in total, and therefore, we chose a relation to be correct if the majority voted as such. We compare our results to the precision accuracy scores achieved by the OntoGain paper which was explored in section 3.2 [20].

Table 5.4: Table showing the relation accuracy resulting from two of our methods. OntoGain results also included [20].

Approach	Relation Accuracy
Dependency Tree Paths, SCOPUS	**0.533**
PoS Parsing, SCOPUS	**0.300**
Association Rules Algorithm, CS Corpus	0.728
Probabilistic Algorithm, CS Corpus	0.617

Looking at the results presented in table 5.4, we can see that the Dependency Tree Paths based method achieved a relation accuracy score of **0.533** and the PoS parsing method achieved a relation accuracy score of **0.3**. Both scores are lower than the benchmark set by OntoGain.

The note in the section 5.2 regarding the differing dataset also applies to OntoGain.

6 Conclusion

Looking back at our initial objectives, we can consider the primary task of the thesis "How do we deepen the hierarchical relationships between research concepts?" to be fulfilled. This was done in several subtasks, which first involved using manual refinement to improve top-level navigation for users. Our manual taxonomy construction showed to have a great amount of success with our user studies, amounting to a 98.8% relation accuracy score. As mentioned in section 5.1.2, this high score is expected when compared to automated methods, but our score could also be influenced by another factor. Our taxonomy was evaluated by the same participants that influenced the manipulation of the taxonomy throughout the iterative qualitative testing process as described in section 5.1.1, and therefore, the score is biased to tend higher.

The second subtask revolved around enhancing the existing concepts and relations through automated refinement approaches. Our newly adapted concept merging methods are the Sentence Transformers approach with a concept coherence score of 98.10%, the SciConceptMiner approach with a concept coherence score of 98.89%, and the BERT-LS and BART-LS approach with a concept coherence score of 81.67%. All three approaches yielded higher score results for concept coherence than the existing implementation's best score (which stemmed from the pure BERT-LS approach, with a concept coherence score of x). The Sentence Transformers and SciConceptMiner approaches proved to work with exceptionally high result scores, though this comes with a caveat for each: the Sentence Transformers approach is only applied on concepts that share at least one token in common, so if two concepts share the same meaning semantically but have no common token, they are skipped in processing. On the other hand, the SciConceptMiner approach shows great promise but is limited by the number of free API calls allowed per month. Therefore, we recommend and use both the Sentence Transformers approach and BERT-LS and BART-LS approach to merge concepts, and limit the SciConceptMiner based on the API key utilized.

The other end of this subtask involves our newly developed taxonomic relation construction methods. Our string inclusion approach achieved a relation accuracy score of 66.67%, while our weighted ensemble approach achieved a relation accuracy score of 90%. The existing implementation's highest score achieved was with the subsumption method, with a relation accuracy score of 86%. Our weighted ensemble approach surpasses this method, which is expected, as it only includes relations that were detected through more than one method.

Our third and final subtask is the transition from a taxonomy to an ontology with more complex relations. We introduce two methods for extracting non-taxonomic relations, the

dependency tree paths approach which achieved a relation accuracy score of 53.33% and the PoS parsing approach which achieved a relation accuracy score of 30%. We compare these values to the highest score reached by the OntoGain system, which was 72.85%. Since the PoS parsing approach did not perform well according to our expectations, we discard its use and rely on the dependency tree paths approach. However, this still falls short of existing implementations. This can be attributed to the fact that our system does not perform extra validation steps on the extracted triples to check their correctness. Nevertheless, we are still able to, on average, extract these triples to form the missing relations and complete our ontology.

7 Future Work

In this thesis, we provided a second step towards building an automated ontology from a corpus of scientific publications. Whereas Klimek showed a first step towards this process [3], we improved upon the areas of that paper that needed further investigation and implementation (namely the concept deduction and taxonomic relation construction modules) and added new implementations (the manual taxonomy and non-taxonomic relation extraction modules). In turn, our work needs to be further expanded on to fully realize the goal of the project and create an automated high quality ontology of NLP concepts.

For our corpus of texts, we primarily relied on Klimek's collection of papers from the SCOPUS dataset [paper]. This proved to perform well for the collection of keyphrases, though topics of the papers are slightly skewed towards the application of NLP topics in other non-NLP domains and fields. This introduces a lot of unwanted keyphrases that sometimes slip through the keyphrase filtering process. Therefore, it would be of interest to investigate alternative datasets that are more focused on the studies of NLP topics themselves.

In addition, the quantitative manual taxonomy should be re-evaluated with a new set of participants that are disconnected from the participants forming the qualitative evaluation process. This would produce less biased results that could portray a more accurate reading of the manual taxonomy's reliability and correctness.

The biggest room for improvement lies in the non-taxonomic relations extraction step. Due to time restraints, we only retrieved the triples of ontological relations and mapped them, but did not perform any further validation to ensure their correctness, as suggested by Dessí et al. [46]. This validation step could take into consideration the amount of times a triple was extracted, and the number of methods that were able to extract it.

List of Figures

List of Tables

Acronyms

ACL Association for Computational Linguistics 16

AI Artificial Intelligence 5

API Application Programming Interface 21, 23

BART-LS BART-Based Lexical Substitution 20, 21, 32, 36

BERT-LS BERT-Based Lexical Substitution 19–21, 32, 33, 36

CS Computer Science 35

CSO Computer Science Ontology 15

DT Dependency Trees 26

KG Knowledge Graphs 26

MAPE Mean Absolute Percentage Error 30

NER Named-Entity Recognition 25

NLP Natural Language Processing 1–7, 15, 16, 21, 23, 25, 29, 38

NLTK Natural Language Toolkit 4

PoS Part-of-Speech 4, 5, 8, 9, 25, 27, 35

STM International Association of Scientific, Technical, and Medical Publishers 1

tf-idf term frequency-inverse document frequency 8

TUM Technical University of Munich 29

URL Uniform Resource Locator 13, 21

Bibliography

[1] T. International Association of Scientific and M. Publishers. *STM global brief 2021 - economics market size. 2021*.

[2] M. N. Asim, M. Wasim, M. U. G. Khan, W. Mahmood, and H. M. Abbasi. "A survey of ontology learning techniques and applications". In: *Database* 2018 (2018).

[3] S. Klimek. *Learning Hierarchical Relations between Research Concepts from Abstracts and Titles of NLP Publications*. Aug. 2022.

[4] C. Fellbaum. "WordNet". In: *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.

[5] E. Loper and S. Bird. "Nltk: The natural language toolkit". In: *arXiv preprint cs/0205028* (2002).

[6] B. Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[7] J. M. Padial, A. Miralles, I. De la Riva, and M. Vences. "The integrative future of taxonomy". In: *Frontiers in zoology* 7.1 (2010), pp. 1–14.

[8] D. Hupkes, M. Giulianelli, V. Dankers, M. Artetxe, Y. Elazar, T. Pimentel, C. Christodoulopoulos, K. Lasri, N. Saphra, A. Sinclair, et al. "State-of-the-art generalisation research in NLP: a taxonomy and review". In: *arXiv preprint arXiv:2210.03050* (2022).

[9] R. Navigli, P. Velardi, and A. Gangemi. "Ontology learning and its application to automated terminology translation". In: *IEEE Intelligent systems* 18.1 (2003), pp. 22–31.

[10] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. "The Stanford CoreNLP natural language processing toolkit". In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.

[11] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. "Feature-rich part-of-speech tagging with a cyclic dependency network". In: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 2003, pp. 252–259.

[12] D. Chen and C. D. Manning. "A fast and accurate dependency parser using neural networks". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 740–750.

[13] Q. Liu, M. J. Kusner, and P. Blunsom. "A survey on contextual embeddings". In: *arXiv preprint arXiv:2003.07278* (2020).

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

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

[16] D. Jurafsky and J. H. Martin. *Speech and language processing*. 3rd Edition Draft. Pearson, 2022.

[17] N. Nabila, A. Mamat, M. Azmi-Murad, and N. Mustapha. "Enriching non-taxonomic relations extracted from domain texts". In: *2011 International Conference on Semantic Technology and Information Retrieval*. IEEE. 2011, pp. 99–105.

[18] E. Brill. *A simple rule-based part of speech tagger*. Tech. rep. Pennsylvania Univ Philadelphia Dept of Computer and Information Science, 1992.

[19] D. Lin. "Using collocation statistics in information extraction". In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*. 1998.

[20] E. Drymonas, K. Zervanou, and E. G. Petrakis. "Unsupervised ontology acquisition from plain texts: the OntoGain system". In: *Natural Language Processing and Information Systems: 15th International Conference on Applications of Natural Language to Information Systems, NLDB 2010, Cardiff, UK, June 23-25, 2010. Proceedings 15*. Springer. 2010, pp. 277–287.

[21] X. Schmitt, S. Kubler, J. Robert, M. Papadakis, and Y. LeTraon. "A replicable comparison study of NER software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate". In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE. 2019, pp. 338–343.

[22] M. Kavalec, A. Maedche, and V. Svátek. "Discovery of lexical entries for non-taxonomic relations in ontology learning". In: *SOFSEM 2004: Theory and Practice of Computer Science: 30th Conference on Current Trends in Theory and Practice of Computer Science Měřın, Czech Republic, January 24-30, 2004 Proceedings 30*. Springer. 2004, pp. 249–256.

[23] R. Ganter and R. Wille. "Formal concept analysis: Mathematical foundations Springer-Verlag Berlin Germany". In: (1999).

[24] H.-M. Haav. "An application of inductive concept analysis to construction of domain-specific ontologies". In: *Brandenburg University of Technology at Cottbus* (2003), pp. 63–67.

[25] P. Cimiano, A. Hotho, and S. Staab. "Learning concept hierarchies from text corpora using formal concept analysis". In: *Journal of artificial intelligence research* 24 (2005), pp. 305–339.

[26] R. Srikant and R. Agrawal. *Mining generalized association rules*. 1995.

[27] A. Cattan, A. Eirew, G. Stanovsky, M. Joshi, and I. Dagan. "Streamlining cross-document coreference resolution: Evaluation and modeling". In: *arXiv preprint arXiv:2009.11032* (2020).

[28] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. "End-to-end neural coreference resolution". In: *arXiv preprint arXiv:1707.07045* (2017).

[29] B. Yang, C. Cardie, and P. Frazier. "A hierarchical distance-dependent bayesian model for event coreference resolution". In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 517–528.

[30] P. K. Choubey and R. Huang. "Event coreference resolution by iteratively unfolding inter-dependencies among events". In: *arXiv preprint arXiv:1707.07344* (2017).

[31] K. Kenyon-Dean, J. C. K. Cheung, and D. Precup. "Resolving event coreference with supervised representation learning and clustering-oriented regularization". In: *arXiv preprint arXiv:1805.10985* (2018).

[32] S. Barhom, V. Shwartz, A. Eirew, M. Bugert, N. Reimers, and I. Dagan. "Revisiting joint modeling of cross-document entity and event coreference resolution". In: *arXiv preprint arXiv:1906.01753* (2019).

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[34] 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*. 2018, pp. 2701–2709.

[35] I. S. Dhillon and D. S. Modha. "Concept decompositions for large sparse text data using clustering". In: *Machine learning* 42 (2001), pp. 143–175.

[36] N. Kaushik and N. Chatterjee. "Automatic relationship extraction from agricultural text for ontology construction". In: *Information processing in agriculture* 5.1 (2018), pp. 60–73.

[37] *Agricultural Thesaurus*. URL: `https://www.fao.org/agrovoc/`.

[38] B. Frikh, A. S. Djaanfar, and B. Ouhbi. "A Hybrid Method for Domain Ontology Construction from the Web." In: *KEOD*. 2011, pp. 285–292.

[39] A. S. Djaanfar, B. Frikh, and B. Ouhbi. "A domain ontology learning from web documents". In: *Intelligent Distributed Computing IV: Proceedings of the 4th International Symposium on Intelligent Distributed Computing-IDC 2010, Tangier, Morocco, September 2010*. Springer. 2010, pp. 201–208.

[40] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia. "Microsoft academic graph: When experts are not enough". In: *Quantitative Science Studies* 1.1 (2020), pp. 396–413.

[41] *The Open University: Computer Science Ontology*. URL: `https://cso.kmi.open.ac.uk/`.

[42] *The Association for Computational Linguistics*. URL: http://www.aclweb.org/.

[43] S. Bandyopadhyay. "Emerging Applications of Natural Language Processing: Concepts and New Research: Concepts and New Research". In: (2012).

[44] E. Cambria and B. White. "Jumping NLP curves: A review of natural language processing research". In: *IEEE Computational intelligence magazine* 9.2 (2014), pp. 48–57.

[45] N. Ranjan, K. Mundada, K. Phaltane, and S. Ahmad. "A Survey on Techniques in NLP". In: *International Journal of Computer Applications* 134.8 (2016), pp. 6–9.

[46] D. Dessı, F. Osborne, D. R. Recupero, D. Buscaldi, and E. Motta. "SCICERO: A deep learning and NLP approach for generating scientific knowledge graphs in the computer science domain". In: *Knowledge-Based Systems* 258 (2022), p. 109945.

[47] W. Zhou, T. Ge, K. Xu, F. Wei, and M. Zhou. "BERT-based lexical substitution". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3368–3373.

[48] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *arXiv preprint arXiv:1910.13461* (2019).

[49] M. Pogoda, K. Gawron, N. Ropiak, M. Swędrowski, and J. Kocoń. "Deep Neural Sequence to Sequence Lexical Substitution for the Polish Language". In: *Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part I*. Springer. 2022, pp. 692–705.

[50] Z. Shen, C.-H. Wu, L. Ma, C.-P. Chen, and K. Wang. "SciConceptMiner: A system for large-scale scientific concept discovery". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. 2021, pp. 48–54.

[51] *Bing Search APIs*. URL: https://www.microsoft.com/en-us/bing/apis.

[52] *Bing Search API Pricing*. URL: https://www.microsoft.com/en-us/bing/apis/pricing.

[53] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. "Transformers: State-of-the-art natural language processing". In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 2020, pp. 38–45.

[54] N. Reimers and I. Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. eprint: arXiv:1908.10084.

[55] A. T. Luu, 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)*. 2014, pp. 810–819.

[56] D. Dessı, F. Osborne, D. Reforgiato Recupero, D. Buscaldi, E. Motta, and H. Sack. "Ai-kg: an automatically generated knowledge graph of artificial intelligence". In: *The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II 19*. Springer. 2020, pp. 127–143.

[57] K. K. Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania, 2005.