Politecnico di Torino

Politecnico di Torino

Master's Degree in Computer Engineering

Academic Year 2024/2025

Graduation Session July 2025

Benchmarking Synonym Extraction Methods in Domain-Specific Contexts

Supervisors:

Luca Cagliero

Luca Gioacchini

Irene Benedetto

Candidate:

Shayan Taghinezhad Roudbaraki

Abstract

Accurate identification of synonyms is crucial for several Natural Language Processing tasks and it presents significant challenges when done in a specialized domain. These problems arise because of unique vocabularies, domain jargon, semantic shift of words when used in non-general domains and limited domain-specific resources for synonym detection. This thesis analyzes different methods for synonym extraction in domain-specific contexts by evaluating a subset of techniques on a multi-domain dataset which includes terms, their usage contexts and ground truth synsets in different domains such as agriculture, automotive, economy, geography, legal, medical and technology.

The analysis include synonym extraction using traditional lexical resources like WordNet, various available forms of distributional semantic models like fastText, domain-specific corpus training and fine-tuning, and contextual embedding models like BERT. Clustering algorithms are also investigated when applied to combined term and definition representations. For a more thorough analysis, Name Entity Recognition for term identification is explored and compared with information extraction models and LLMs for the same task. Additionally, capabilities of large language models (LLMs) for definition generation and synonym grouping is explored. Evaluation of experiments is done by using standard Precision, Recall, F1-score metrics specifically adapted for synset recovery and recall for term identification.

The research concludes that currently the proposed multi-step approach is most effective in synset creation which consists of: term identification and definition generation by an LLM, unsupervised clustering, and additionally refining the clusters by an LLM.

Keywords: Synonym Extraction, Synset Induction

Table of Contents

List of Tables

Glossary

Contextualized Word Embedding A word embedding where the vector representation for a word is dynamically generated based on the surrounding text, allowing the model to capture different meanings of polysemous words (e.g., BERT).

Synset A group of one or more synonyms that are interchangeable in a specific context, representing a single distinct concept. In this thesis, synsets serve as the ground truth for evaluation and the target output of the extraction methods.

Target Term The specific word or multi-word expression within a text chunk for which synonyms are to be identified. It serves as the primary input for the synonym extraction process.

Text Chunk A segment of domain-specific text that contains a target term. The chunk provides the necessary linguistic context to disambiguate the meaning of the target term and identify its appropriate synonyms.

Word Embedding A dense vector representation of a word in a multi-dimensional space. The geometric relationships between vectors (e.g., distance) are intended to capture the semantic relationships between the corresponding words.

Acronyms

K-GMA k-nearest-neighbor Greedy Modularity Algorithm

LLM Large Language Model

MWE Multi-word Expression

NER Named Entity Recognition

NLP Natural Language Processing

OOV Out Of Vocabulary

Chapter 1

Introduction

A fundamental part of understanding human language is understanding the relationship between words. In this regard, semantic relationships play an important role. Synonymy is the relationship between words with close and similar meaning and it's needed for many NLP tasks which includes information retrieval [1], question answering, text summarization [2] and machine translation [3] among others. When synonyms are identified, systems can detect that some words convey the same idea and therefore will be better able to to process and generate natural language. The ability of an intelligence system to understand that 'USA' and 'United States' refer to the same entity, or that 'crabby' and 'irritable' can be used interchangeably in specific contexts, is necessary for accurate processing and response to user queries and understanding complex texts [4].

One of the traditional resources for capturing and keeping track of synonyms were dictionaries and thesauri. These are important resources but cannot keep pace with the dynamic developments in language and specifically the terms that emerge and are specifically used in certain domains. Another challenge is that the meaning and usage of a word can vary based on the domain of discourse. A term like 'cell' have hugely different meaning in biology, telecommunication and prison system. On the other hand, words that are close together in general language, might have distinct meanings in a technical field or conversely terms that have different meaning in general language can have similar meaning in a specific domain.

It's because of all of the mentioned features of language that domain-specific language makes synonym extraction challenging. Experts within a field often use existing words with altered or narrowed meanings, create new terms or use jargon and acronyms. General-purpose resources or models that have captured the word relationships based on a broad corpora may fail to capture these delicate nuances.

The increasing volume of digital text data found in specialist domains such as scientific papers, legal documents, medical histories, technical manuals, financial statements, etc is both an opportunity and a challenge. The information contains

the implicit knowledge of domain-specific words and their relations, like synonymy. The challenge is converted into an opportunity through the use of the information to automatically extract domain-specific synonyms, thereby building or extending domain-specific lexical resources. Another challenge is in developing computational methods that can successfully learn these relationships from unstructured text, beyond the reach of general-purpose tools and the inherent vagueness of language.

Benchmarking different synonym extraction techniques in specialized domains, therefore, becomes essential. It allows both researchers and practitioners to learn about the advantages and limitations of certain methods when applied to specialized vocabulary. Such benchmarks can guide the selection of appropriate methods to build applications for special domains, point out areas where current methods fall short, and stimulate the development of new, better algorithms for domain adaptation.

Finding accurate and effective synonyms for terms in specialized domains and from unstructured text data is the main issue this thesis attempts to solve. The complex and specialized semantic relationships common in domain-specific language are usually missed by general-purpose synonym extraction techniques, which frequently rely on broad lexical resources or models trained on general language corpora. This failure shows up in a number of ways:

- **Inadequate Coverage**: Many domain-specific terms and their synonyms might not be adequately captured in general resources and models [5].
- **Problems with polysemy**: Domain-specific terms may have multiple meanings and general approaches might not be able to clearly distinguish the intended domain-specific meaning, resulting in inaccurate synonyms [6].

1.1 Research Questions

The following research questions are addressed in this thesis:

1. How well do traditional lexical resources, distributional semantic models and contextual embedding models perform in identifying domain-specific synonyms?
2. Can the accuracy of synonym extraction be increased by fine-tuning, training distributional semantic models or contextual embedding models on domain-specific corpora?
3. How well do clustering algorithms group synonyms when they are applied to vector representations of domain-specific terms enhanced with definitions?

4. Does a combination of clustering and refinement by LLMs perform good on synonym extraction?

To answer these question, this thesis focuses on a comparison of computational methods for synonym extraction in several domains by evaluating different off-the-shelf available models used for this purpose. First, Named Entity Recognition (NuExtract [7], BERT, NLTK [8], SpaCy [9], variants of OpenAI GPT-4 [10]) are used for extracting key terms from the text corpora. Then to answer the first two questions, a comparison is made between available methods to evaluate the effectiveness of these methods on the specified task. Another analysis goes more in-depth by fine-tuning some of the models on domain-specific corpora. Methods that use lexical resources (WordNet [11]), static word embeddings (GloVe [12], fastText [13], Word2Vec [14]), contextual word embeddings (BERT [15]) are evaluated to understand how do they perform for extracting synonyms.

Finally to answer the third and fourth question on effectiveness of clustering and LLM refinement for synonym extraction, a new approach is proposed based on synonym clustering and a final refinement by an LLM. Clustering techniques (K-Means, Agglomerative, DBSCAN, K-GMA) are applied to Sentence-BERT [16] embeddings of terms and definitions to find the semantically similar clusters. And finally the clusters are refined with variants of GPT-4 [10] to get more detailed synsets. All of the evaluations are based on a dataset that contains target terms, their domain-specific contexts (text chunks) and ground-truth Synsets. The main evaluation metrics are Precision, Recall, and F1-score [17] calculated at the word level and averaged, as well as recall for term extraction.

The thesis is divided into the following seven chapters: 'Chapter 1: Introduction' introduces the problem of domain-specific synonym extraction, along with its significance and difficulties. The research questions covered in the thesis are also outlined. 'Chapter 2: Background' covers the theoretical and computational underpinnings of synonym extraction. These include named entity recognition, semantic relationships, conventional lexical resources, different word embedding models (static and contextualized), clustering techniques, and the function of Large Language Models (LLMs) in NLP tasks. 'Chapter 3: Related Work' reviews the research on synonym discovery, which also identifies the gaps in the literature that this thesis seeks to fill and highlights recent developments.

'Chapter 4: Methodology' explains the multi-stage methodology, which includes LLM-based refinement, baseline methods and the suggested hybrid approach involving LLM-generated definitions, Sentence-BERT embeddings, and clustering, as well as term identification (using NER, IE models, and LLMs). 'Chapter 5: Dataset and Experimental Setup' offers a thorough explanation of the experimental setup, including particular models, libraries, and the hardware/software environment, as well as the custom-made, multi-domain dataset that was used. 'Chapter 6: Results'

presents and discusses the experimental results for term extraction, baseline synonym extraction techniques, and the suggested clustering with GPT refinement approach which includes an error analysis. 'Chapter 7: Conclusion' highlights the main conclusions, describes the thesis's contributions, talks about its shortcomings, and recommends directions for further study.

Chapter 2

Background

2.1 Understanding Synonymy and Semantic Relationships

This chapter is an in-depth discussion of theory and current computational techniques available for synonym extraction methods in general and specialized domains. The linguistic notion of synonymy, different techniques for computational representation of word meaning, techniques for determining key terms in text data, and clustering techniques involved in grouping semantically similar items are investigated. Particular emphasis is placed on dealing with challenges when applying these techniques to a domain-specific context.

Synonymy indicates the relation between words that convey closely related meanings. Yet, the task of describing and determining true synonyms is difficult both linguistically and computationally. According to philosophers such as Quine [18] and Goodman [19], true synonymy is impossible because one can't define it. The majority of words identified as synonymous are actually near-synonyms or share overlapping but not the same meaning, frequently differing in subtle aspects.

Also it should be mentioned that generally the degree of semantic similarity between two words is associated with the frequency in which they stand in association with the same words (i.e., the similarity of their contexts) as mentioned in the famous quote by John Rupert Firth saying: *You shall know a word by the company it keeps* [20]. This notion underscores the importance of context in understanding the word meaning and remained a central concept in computational linguistics.

Further research confirmed that human judgment about semantic similarity correlates strongly with the degree to which words share contexts [21]. This could be the basis for the distributional hypothesis which posits that words appearing in similar context tend to have similar meanings.

In the field of lexical semantics, the meaning of words and the relationships between them are studied. Other important semantic relationships beyond synonymy include:

- **Antonymy**: Words that have opposite meanings (e.g., hot/cold).
- **Hypernymy/Hyponymy**: One word representing a general category (hypernym) and the other a specific instance (hyponym) which create a hierarchical relationship (e.g., animal/dog, vehicle/car). This is mostly mentioned as the "is-a" relationship.
- **Meronymy/Holonymy**: A part-whole relationship (e.g., wheel/car, finger/-hand).
- **Relatedness**: A more general concept which includes any type of semantic connection between words, such as functional relationships, association, or participation in the same topic (e.g., doctor/hospital, coffee/cup).

Synonymy is a specific type of semantic similarity, but the terms 'semantics similarity' and 'semantic relatedness' are mostly used interchangeably in computational linguistics. Semantic similarity generally refers to the degree to which words and terms can be replaced by one another which is often based on their position in semantic hierarchy (like hypernymy). Semantic relatedness is a broader concept and includes similarity but also other relationships like antonymy and functional associations. For synonym extraction task, semantic similarity is the main focus but methods that capture more general relatedness might still be useful if they can distinguish near-synonyms from other related terms.

Different studies has explored the distinction between similarity and relatedness [22, 23, 24]. Datasets like WordSim-353 [25] and SimLex-999 [26] have been curated to analyze computational models on these differences in semantic meaning. WordSim-353 consists of pairs that have a relationship (synonymy, antonymy, association) and SimLex-999 focuses on human judgment of similarity and not mere relatedness between word pairs. These datasets were used to compute the correlation between human relatedness judgment and cosine similarities of model-produced word embeddings. SimVerb-3500 [27] extends this to verbs.

In domain-specific text, detection of semantic relationships becomes more challenging. Words can be highly related in one domain and unrelated in another. For example 'stock' and 'bond' have close relatedness in economy domain, however they are less related in general language or other domains. In addition, domain-specific jargon is generally a multi-word term or compound noun (e.g., 'electronic control unit', 'precision agriculture') and it's more complex to identify the synonyms of these multi-word expressions.

The dataset that is used in this thesis provides 'synsets' for target terms in each domain. A synset is a concept mentioned in resources like WordNet, which represents a group of words that can be interchangeable in some contexts. In the context of this thesis, these synsets are the ground-truth for domain-specific synonymy and define the target groups of words to be extracted by different methods. The task definition is to identify these domain-specific synsets by using the provided text chunks.

2.2 Traditional Lexical Resources

Historically, manually constructed lexical resources have been the main source for information about word relationships and meaning. These resources are created by linguists who study language usage and provide the resulting semantic information in these resources [28, 29].

2.2.1 WordNet

WordNet [11] can be mentioned as the most prominent of these lexical resources in English language. This large lexical resource was developed at Princeton University and includes English nouns, verbs, adjectives and adverbs that are grouped into synsets. Each of these synsets represent a distinct concept. Synsets are interconnected by different semantic relationships including hypernymy/hyponymy (is-a), meronymy/holonymy (part-whole), antonymy, and others which creates a comprehensive semantic network.

WordNet has been used in Natural Language Processing (NLP) for tasks that require semantic information such as word sense disambiguation, information retrieval and measurement of semantic similarity and relatedness. There are several methods that use the hierarchical structure in WordNet for computing semantic similarity [30, 31, 32, 33, 34]. These methods measure the distance between synsets in the WordNet hierarchy, mostly by incorporation of the notion of Information Content (IC), which measures the specificity of concepts based on their frequency in a corpus [30, 35, 36, 37, 38]. When concepts are close in the hierarchy or they share a highly specific and common ancestor, they are considered more similar. There are other methods that use the glosses (definitions) of synsets and measure the overlap between the words that are present in the glosses [39].

It can be said the although WordNet has extensive coverage and is rich in semantic information, it has its own limitations, in particular when it's applied to domain-specific language:

- **General Vocabulary**: WordNet covers the general English vocabulary without a comprehensive coverage for specialized terms, multi-word terms or jargon

that is used in professional or technical domains.

- **Static Nature**: WordNet is not updated in a manner to keep pace with the dynamic evolution seen in the language. It's mainly a static resource and can't keep up with rapid introduction of new terms in fast-changing domains such as technology.

- **Limited Context Sensitivity**: Without the use of other external methods like Word Sense Disambiguation (WSD), WordNet cannot provide a mechanism to detect the most relevant sense of a word in any specific domain context.

- **Subjectivity**: WordNet is constructed based on human judgment of words and their meaning, which introduces subjectivity and possible inconsistencies.

It's for this reason that using WordNet for domain-specific synonym extraction, as done in the baseline experiments, mostly produces limited results because many domain-specific words and their synonyms are not present or correctly linked int he WordNet structure.

2.2.2 Other Lexical and Knowledge Resources

Other knowledge bases or lexical resources are developed and some of them try to address the problems of general resources or use information that is provided from multiple sources.

Thesauri like Roget's Thesaurus provide lists of synonyms and related words by grouping words based on ideas and concepts. They are less structured than WordNet and also are made based on the general language. They have been used for measurement of semantic similarity [40].

Domain-Specific Terminologies and Ontologies are specific structured resources that have been made in some domains. Unified Medical Language System (UMLS) [41] can be named as one of these resources in the biomedical domain. While invaluable, they're generally domain-specific and might not be available for all domains.

Wikipedia and Other Structured Knowledge Bases have been developed using large collaboration are also used for creating computational resources. For example, words or texts in Explicit Semantic Analysis (ESA) [42] are represented as vectors of Wikipedia concepts, measuring similarity based on the overlap and weighting of different concepts. Other structured knowledge sources like DBpedia [43] and YAGO [44] can be mentioned here which are extracted from Wikipedia and other sources which represent huge number of entities and relationships that are useful for semantic tasks [45]. Wikify! [46] is a system designed to link text to Wikipedia concepts. These resources provide broader coverage than WordNet

and can include domain-specific entities but accurate mapping of text terms to concepts is still a challenge.

BabelNet [47] combines information from WordNet, Wikipedia and other resources to create a multilingual and large semantic network. The aim is to provide a more complete coverage and associate concepts across different languages. BabelNet has been useful in different semantic tasks such as measuring semantic similarity [48].

Paraphrase Databases like 'The Paraphrase Database' [49] are created for collection of paraphrases and semantically equivalent phrases which might include synonyms. These entities are mostly extracted in an automatic manner from parallel corpora.

These resources contain valuable information, but they still cannot present a full coverage for domain-specific synonymy, especially when novel terms or context-dependent meanings are mentioned. It can be said that because domain language is dynamic in its nature, it's necessary to use methods that can directly learn from the domain-specific text data.

2.3 Distributional Semantics

The distributional hypothesis is famously summarized by Firth's quote "You shall know a word by the company it keeps," and forms the basis for distributional semantics. According to this paradigm, words with similar meanings appear in relatively similar linguistic contexts. There are computational models that are developed based on this hypothesis. These models analyze the word co-occurrences in large text corpora and therefore learn and create vector representations (embeddings) for each word. Semantic properties are captured in these vectors and an approximate measure of semantic similarity can be computed by calculating the geometric distance or similarity of these vectors (e.g., cosine similarity).

Models using distributional semantic had a huge impact in NLP and provided a powerful and data-driven representations of word meaning. By using these methods on raw text, word relationships can be automatically learned which is far more useful in many cases compared to static and manually crafted resources, especially for domain-specific use-cases where there is a lack of resources with full coverage of the terms.

2.3.1 Count-Based Models

Early distributional models generally used count-based methods by constructing a co-occurrence matrix with rows being the target words and columns being context words. The cells in this matrix represent the number of times that a target word appears in relative proximity of a context word in a defined window. For ignoring

less informative and common words and emphasizing the informative co-occurrences, weighting schemes (e.g., Pointwise Mutual Information - PMI) are applied to the raw counts.

Latent Semantic Analysis (LSA) [50, 51] can be mentioned as a prominent count-based method that applied Singular Value Decomposition (SVD) for dimensionality-reduction of term-document or term-context matrix. Lower-dimensional vectors that are produced, form the word embeddings. In LSA words are similar if they appear beside the same context words but it also considers higher-order co-occurrences and therefore if they appear with context words that themselves share similar contexts, they're considered more similar. It's been shown that LSA can capture semantic similarity [52].

Hyperspace Analogue to Language (HAL) model [53] in another early approach that uses a count-based method to create word-word co-occurrence matrix within a sliding window. Also some approaches tried to scale distributional similarity methods to large corpora [54].

2.3.2 Prediction-Based Models

Prediction-based models, in particular those based on neural networks, are a more recent development in creating word embeddings. They don't count the co-occurrences and instead they train a neural network with the task of prediction of context words given a target word and vice versa. The hidden layer in these neural networks learn the word embeddings.

Word2Vec [14, 55] is a well-known prediction-based model with two main architectures: Continuous Bag-of-Words (CBOW) for prediction of a target word from the surrounding context word, and Skip-gram to predict context words from a target word. There is popular variant called Skip-gram with negative sampling (SGNS) for optimizing the training of the model to distinguish between the true context words and randomly sampled 'negative' words. Word2Vec models are have efficient performance and can be easily trained on large corpora to produce dense, low-dimensional vectors.

GloVe (Global Vectors for Word Representation) [12] is another one of the popular models for learning word embeddings. It's not a predictive model like Word2Vec and instead it's based on word-word co-occurrence statistics in a corpus. It factorizes the logarithm of the corpus's word co-occurrence matrix. It learns the word vectors by trying to make the dot product of two words equal to the logarithm of their co-occurrence probability. Glove embeddings perform sufficiently well on word analogy tasks and other semantic tasks.

fastText [13, 56] represents each word as a bag of character n-grams. Vector representations are learned for each character n-gram. For calculating a word embeddings, the n-gram vectors of the character n-grams of the word are summed

together. fastText shows strong performance especially in languages with rich morphology and when a task includes rare words. fastText is able to:

1. Generate vector representations for out-of-vocabulary (OOV) words by summing the embeddings of the character n-grams in that word.

2. Understand morphological information: Words with similar morphemes (e.g., 'national,' 'nation,' 'nationality') share character n-grams and therefore their embeddings are related.

In general, prediction-based models (like Word2Vec's Skip-gram) outperform traditional count-based distributional models on different semantic tasks [57]. However, as GloVe showed later, count-based methods can also perform well.

It's necessary to evaluate unsupervised word embeddings. One study [58] mentions two different kinds of evaluations, one being intrinsic evaluation (e.g., analogy task or word similarity) and the other one being extrinsic evaluation (performance on downstream NLP tasks). Another reproducible survey [59] compares word embeddings and ontology-based methods used for word similarity and finds that linear combinations generally outperform individual state-of-the-art methods by a large margin.

2.4 Contextualized Word Embeddings

Traditional word embeddings like GloVe, Word2Vec and fastText are static or type-level embeddings: there is only one single vector representation for each word type regardless of its context. This is a huge limitations because many words are polysemous which means that they have multiple meanings. In contextualized word embedding models this is addressed by the fact that the model generates different embeddings for words depending on the context in which the word appears in (token-level embeddings).

ELMo (Embeddings from Language Models) [60] uses a bidirectional Long Short-Term Memory (LSTM) network trained as a language model and generates deep contextualized word representations. Each word embeddings is a function of the internal states of the BiLSTM which captures information from preceding and succeeding words. These embeddings can be considered deep because they're calculated as a weighted sum of hidden states from all of the layers in the BiLSTM.

Transformer Architecture and Attention [61] made a huge impact on sequence transduction tasks, becoming the foundation for most of the contextualized embedding models that were introduced later on. Instead of recurrence that was generally used in transduction tasks, it entirely relies on attention mechanism to understand global dependencies between input and output. With self-attention, the model can weigh the importance of words in a sequence to compute the

representation of a word. Therefore contextualized models can differentiate between word senses based on context which is highly relevant for synonym extraction because a word's synonyms can change significantly with its meaning.

The core of the attention mechanism in Transformer models is the Scaled Dot-Product Attention, defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where:

Q is the Query matrix
K is the Key matrix
V is the Value matrix
d_k is the dimension of the keys (and queries)

This mechanism allows the model to weigh the importance of different parts of the input sequence when processing each element.

For Multi-Head Attention, the process is extended by performing several attention operations in parallel, each with its own learned linear projections for Q, K, and V. The results from these "heads" are then concatenated and linearly transformed to produce the final output:

$$\begin{aligned}\text{head}_i &= \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)\\ \text{MultiHeadAttention}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O\end{aligned}$$

where:

W_Q^i, W_K^i, W_V^i are weight matrices for the i-th head
W^O is the output weight matrix
h is the number of attention heads

Multi-Head Attention allows the model to jointly attend to information from different representation subspaces at different positions.

BERT (Bidirectional Encoder Representations from Transformers) [15] uses the Transformer encoder architecture and it's trained on massive corpora on two unsupervised tasks:

- **Masked Language Model (MLM)**: A portion of input tokens are randomly chosen and masked and the model is trained for prediction of the original masked tokens based on available unmasked context which results in BERT learning deep bidirectional representations.

- **Next Sentence Prediction (NSP)**: Pairs of sentences are given to the model and it has to predict whether the second sentence is the next sentence that follows the first one in the original text. BERT understands sentence relationships in this way. BERT embeddings are used in combination with fine-tuning for different NLP tasks and have achieved state-of-the-art performance.

BERT Variants are also proposed to improve its efficiency, performance or robustness:

- **RoBERTa (Robustly Optimized BERT Pretraining Approach)** [62]: Showed that BERT was under-trained and introduced another improved pre-training strategy (e.g., training on more data for a longer period with dynamic masking and removing NSP).

- **ALBERT (A Lite BERT for Self-supervised Learning)** [63]: Used parameter-reduction techniques to reduce BERT's parameter size. It maintained performance while making the model more memory-efficient by using factorized embedding parameterization and cross-layer parameter sharing.

- **SciBERT** [64]: It's a BERT model that is pre-trained on a large corpus consisting of scientific text from computer science and biomedicine which improve the mode's performance on scientific NLP tasks.

- **DistilBERT** [65] and **TinyBERT** [66]: Smaller and faster variants of BERT that were created by knowledge distillation which are more suitable for resource-constrained environments.

- **ERNIE 2.0** [67]: It introduces novel pre-training tasks in addition to MLM and NSP as a continual pre-training framework.

- **XLNet** [68]: Combines the advantages of auto-regressive models (like GPT) and auto-encoding models (like BERT) with a permutation language modeling objective.

Sentence Embeddings (Sentence-BERT) [16] is proposed to provide more accurate embedding for sentences. BERT produces token-level embeddings that have excellent performance on many NLP tasks, however the performance is lower when it's used to generate fixed-size sentence embeddings by using [CLS] token output or calculating the average of token embeddings. Sentence-BERT (SBERT) uses a Siamese or triplet network architecture to fine-tune the pre-trained BERT (or RoBERTa, XLNet). The aim is to be able to generate meaningful sentence embeddings and to be able to use cosine similarity for the comparison of sentence embeddings. This thesis uses this model for embedding term definitions.

While the BERT model is trained on next-sentence prediction and masked-language prediction and can capture useful semantic information in the special CLS token, it's not fine-tuned in a way for similar sentences to have a close CLS token embedding. Sentence-BERT changes this by applying a pooling operation (such as mean pooling) on top of BERT's final contextualized token embeddings. During training, the new Siamese or triplet network structures, change the original BERT weights in a way so that similar sentences have closer embeddings. This training is done on NLI dataset so that the sentences can be classified correctly and also on triplet loss so that sentences with similar meaning have a closer distance than sentences with dissimilar meaning.

2.5 Fine-tuning Embeddings for Specific Tasks and Domains

Static and contextualized pre-trained embeddings are trained on large and general-domain corpora (e.g., Wikipedia, Google News, Common Crawl). They capture broad semantic knowledge, however in specialized domains or tasks their performance can be suboptimal [5]. In fine-tuning, given a pre-trained model, the training continues on a smaller and domain-specific or task-specific dataset. This way, the model can adapts its learned representations to the specific requirements in the target task or domain.

To do fine-tuning for static embeddings like Word2Vec or fastText, the model can be initialized with pre-trained vectors and then the training can be continued on domain-specific corpus. This way the vectors can get updated to be able to better reflect the semantic meanings in the new domain.

In contextualized models like BERT, fine-tuning can be done by adding a task-specific layer to the pre-trained Transformer stack and then continuing to train the entire model (or parts of it) on a supervised task dataset (e.g., question answering, text classification, NER). After fine-tuning, even if it's a different task like domain classification, it can be expected that the underlying representations become more domain-aware and benefit tasks like synset extraction.

2.6 Named Entity Recognition (NER) and Term Extraction

Before doing synonym extraction, domain-specific glossary or a list of key terms should be extracted as a preliminary step. Named Entity Recognition (NER) can be described as identification and classification of named entities in text and assigning them to pre-defined categories such as organizations, persons, locations, dates, etc.

NER focuses on general categories and therefore there is a need for an extension of the underlying techniques to be able to extract domain-specific terms.

- **Traditional NER**: These early NER systems relied on handcrafted rules, lists of known entities and machine learning models.
- **Neural NER**: Recent approaches use neural networks, in particular Recurrent Neural Networks (RNNs) such as LSTMs [69] and a combination of BiLSTMs with a Conditional Random Field layer [70] to achieve the best results. These NER models now commonly use pre-trained contextual embeddings like BERT as input features which further boosts their performance [71].
- **Models for Information Extraction**: The NuExtract models are used in this thesis and are examples of transformer-based models that are used to extract information based on templates provided by user. These models can be adapted for term extraction with the correct template.
- **LLMs for Term Extraction**: Large Language Models like GPTs are also another solution to extract key terms or entities from the text. These models generally show impressive zero-shot or few-shot performance.

The quality of the input glossary which is used for synonym extraction depends on the chosen term extraction method.

2.7 Clustering for Grouping Similar Terms

Clustering can be defined as an unsupervised machine learning task which involves grouping some objects (e.g., terms represented by their embeddings) in a way that objects in the same group (called a cluster) have more similarity with themselves in comparison with those in other groups. If semantic similarity is captured in term embeddings, it can be expected to get groups of synonyms or closely related terms after clustering.

Several clustering algorithms are used in this thesis such as **K-Means Clustering** which is an iterative algorithm that creates k clusters for the provided observations. Each observation is assigned to the cluster with the nearest cluster centroid (cluster mean). Number of clusters k is pre-specified. It's computationally efficient method but might be sensitive to initialization and the assumed cluster shapes are spherical. **Agglomerative Hierarchical Clustering** uses a bottom-up method and each observation starts in its own cluster and by moving up the hierarchy, pairs of clusters are merged. Dendrogram cat be cut at a certain level to get the resulting clusters and also the desired number of clusters can be specified for the algorithm to stop there. Common methods to measure the distance between clusters include:

- Single linkage: distance between the closest members in two clusters are measured.
- Complete linkage: distance between the farthest members in the two clusters are measured.
- Average linkage: average distance between all pairs of members from the two clusters are measured (which is used in this thesis with cosine metric that for pre-computed distances).
- Ward's linkage: minimizes the variance in each cluster

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based algorithm for clustering which groups together points that are close together and have many nearby neighbors and marks the points that lie alone or show up in low-density regions as outliers. It does not need the a specification for the number of clusters and it needs two parameters which include: `eps` as the maximum distance of two samples for them to be considered neighbors and `minimum number of samples` as the minimum number of samples in a neighborhood so that the point can be considered a core point. In **Graph-based Clustering (Louvain Method, k-GMA)**, data is represented as a graph in these methods with nodes being data points and edges representing similarity or proximity. To find densely connected subgraphs as clusters, community detection algorithms are applied.

- **The Louvain methods** [72]: A greedy algorithm that is used for community detection when dealing with large networks. It works by iteratively improving a metric called modularity which is a measure for density of links inside communities compared to the number of links between them.
- **K-GMA (k-Nearest-neighbor Graph-based Modularity Algorithm)** [73]: It's the algorithm that is used in this thesis and is defined as a graph-based Louvain algorithm that uses k-nearest neighbors. The `resolution` parameter changes the granularity of the detected communities.

2.8 Large Language Models (LLMs) for NLP Tasks

Large Language Models (LLMs), which are deep learning models with billions of parameters, are pre-trained on massive amount of text data. These models such as OpenAI's GPT series (Generative Pre-trained Transformer) [74, 75, 76], Google's T5 [77], PaLM [78], and Meta's LLaMA [79] are able to demonstrate powerful

understanding and generating capabilities for human-like text across a range of NLP tasks.

The rapid advancements in LLMs are continuously opening new possibilities for tackling complex NLP problems, including those related to lexical semantics and domain-specific language understanding. Their ability to process instructions and generate structured output makes them a versatile tool in the NLP toolkit.

Chapter 3

Related Work

Considering the challenges present in synonym discovery and the need for a better solution for this task, a range of techniques have been proposed in research, which includes lexical resource linking [80, 47, 81], the use of lexicosyntactic patterns [82], word sense induction [32, 83, 84, 85, 86, 87, 88], clustering [89, 90, 86, 84, 91], graph-based models [92, 93, 94, 95] and distributional semantics [96, 97, 98] or a combination of them as mentioned by several surveys [99, 100, 101].

Recent research analyzes different methods for domain-specific synonym extraction and addresses challenges like dictionary sparsity [102] and multi-word terms [103]. The challenge with dictionary sparsity is that some edges (synonyms) are missing in the input resource. Also, automatic acquisition of synonyms for multi-word terms is difficult due to their non-compositional nature, data sparsity, and especially the variable length of their synonyms. Approaches generally use word embeddings [104, 105] or graph-based clustering [106, 102] tailored for domain data, including Chinese medical [107] in which combines semantic context from word embeddings, cross-lingual evidence from English translations, and unique Chinese linguistic features like character radicals and pronunciation, and knowledge base entities [104] in which integrates broad, corpus-level distributional statistics with precise, sentence-level linguistic patterns and uses knowledge-base synonyms as distant supervision for training. These studies highlight the value of integrating diverse signals and systematic evaluation in specialized domains.

One study proposes SurfCon [108] and focuses on synonym discovery in clinical data, for situations where raw text is not available due to privacy concerns and only the medical terms and their co-occurrence statistics are given. Their framework makes use of surface form of terms and also global context information from co-occurrence graphs. Another study proposes SynSetMine [4] which works by distant supervision from knowledge bases to perform mining entity synonym sets from raw text. They use a neural set-instance classifier and combine it with a set generation algorithm which demonstrates effectiveness in their experiments.

SynSetExpan [109] hypothesizes that Entity Set Expansion (ESE) and Entity Synonym Discovery (ESD) mutually enhance each other and contrary to previous research, they can be jointly performed. In each iteration, the framework performs ESE to output a ranked list of potential candidates for set expansion. It uses this list, in combination with the current synonym discovery model's predictions, to generate pseudo-training data for fine-tuning the synonym discovery model. The resulting fine-tuned model then provides new scores used to refine the ESE ranking which is finally used for set expansion. They demonstrate improved performance on both tasks.

Another framework is proposed for synonym discovery to address the vocabulary gap in e-commerce search [110]. It is based on an unsupervised two-phase method that uses dictionaries and query logs for candidate extraction. It then filters them based on co-occurrence transition probability of products and queries and BERT-based semantic similarity. They use these synonyms by adding them to product indices and report improved search performance.

To address some other challenges in entity synonyms discovery in specific domain, KGSynNet [111] proposes a framework to mitigate out-of-vocabulary terms, hidden connections and rare appearance of terms in text corpus. It uses domain-specific sub-word embeddings to capture the meaning of domain-specific terms. It then uses a joint TransC-TransE knowledge graph model which combines hierarchical and general relation modeling and learns rich entity representations. Then a fusion gate is used to combine semantic and knowledge features into a unified entity representation which is then passed through a classifier based on similarity matching.

KEML [112] is proposed for Lexical Relation Classification (LRC) which also includes synonymy extraction. In their Knowledge-Enriched Meta-Learning framework, they first create relation-aware concept embeddings with LKB-BERT which is a BERT model fine-tuned on text corpora and lexical knowledge bases. KEML then goes through meta-learning by auxiliary tasks which are each designed to distinguish each relation type from random pairs, followed by supervised fine-tuning for the final multi-way classification. It results in robust relation recognition. KEML achieves noteworthy performance on LRC benchmarks.

One of the most recent studies working on synonym discovery is ProSyno [113]. It is introduced as a context-free prompt learning method for synonym discovery that works without any need for contexts or knowledge graphs by using Wikitionary descriptions as a semantic source. Given a pair of terms, it uses a hierarchical semantic encoder to extract semantic representations from word descriptions by using a dynamic matching mechanism so that when multiple descriptions for a word are available, the more relevant one will have more weight. It then uses a pre-trained language model and prompt learning to calculate the synonym probability of the pair of terms. It shows competitive accuracy on different datasets including the

domain-specific ones.

Another recent study proposes EnSynFields [114] for generation of synonym sets. It leverages relevant sentence contexts as flexible perceptual fields and multi-layer contextual information across entity, set, and sentence levels in a three-layer interaction network and a dynamic-weight-based algorithm for balanced set construction. It is evaluated on three real-world datasets, including the domain-specific PubMed and shows efficacy.

3.1 Current Gap

According to a thorough review of related work, from basic distributional techniques to the use of more advanced contextual language models, there are still several opportunities and gaps that remain in automatic synonym extraction, especially when it comes to domain-specific synonymy, which this thesis seeks to fill:

Comprehensive analysis in several domains. Although numerous approaches have been suggested, there is still a gap concerning the comparative study of conventional, embedding-based, and modern LM-based methods tailored or tested specifically for domain-specific synonym extraction across multiple disparate domains. Most of the existing works concentrate on one method or one domain. This thesis bridges this gap by creating an analysis for seven different domains.

Analysis of static vs. contextual embeddings for domain synonymy. The transition towards contextual embeddings has been striking. Still, it is useful to examine the performance of different forms of static (non-contextual) embeddings (pretrained or fine-tuned on domain data) against various contextual embedding types (pretrained or fine-tuned BERT) for domain-specific synonym extraction.

Evaluation of term extraction methods and term extraction by LLMs. The effectiveness of synonym extraction heavily relies on the quality of the initial synonyms being searched. Although NER is a typical prerequisite, systematically evaluating various approaches to term extraction such as traditional NER, template-based LM extraction, and generative LLMs in specific domains is worthwhile.

Integration of LLM capabilities for synonym extraction. Most studies on automatic synonym extraction have focused on utilizing the LMs' embeddings or probing them for candidates. The application of LLMs, during more interactive stages, such as curating or filtering synonym sets produced through other methods is not explored in considerable depth.

Chapter 4

Methodology

Synonym extraction techniques at domain-specific levels will be measured in the context of this thesis. In particular, different methods are evaluated which include baseline strategies, corpus-centric methods, and a new pipeline of embedding-based clustering followed by LLM refinement. This chapter describes the approaches used to develop candidate terms for the domains and the subsequent synonym extraction processes to create synsets.

4.1 Overview of Methodological Stages

The proposed methodology is composed by different steps:

1. **Term Identification**: The first step involves the identification of domain-specific relevant terms for which synonyms are to be obtained. This step is crucial since the quality of input terms greatly influences the success of the synonym extraction process that follows. A number of approaches were investigated for this purpose:

    - **Named Entity Recognition (NER) techniques**: Utilizing various NER models to identify candidate terms from domain-specific text segments. These methods use statistical or neural models trained to label sequences of text based on token types.
    - **Information Extraction Models**: Variants of an information extraction model were used to evaluate their performance in domain-specific term extraction. Specifically NuExtract variants were used which are fine-tuned on schema-based extraction tasks.
    - **Proposed LLM-based Term Extraction and Definition Generation**: Using LLMs facilitates easier term extraction for various domains in addition to definition generation based on context. The glossary of

terms and definitions was created with a general-purpose LLM and showed improved performance in recalling terms.

2. **Synset Creation**: After coming up with a list of target words, this stage focuses on the determination of synonyms related to these words in order to form synsets. This stage involved diverse types of methodologies:
   - **Baseline Methods**: Utilizing established lexical resources such as WordNet, alongside pre-trained general-purpose embeddings including GloVe, fastText, Word2Vec, and BERT.
   - **Corpus-based Embedding Techniques**: Fine-tuning or training embedding models (fastText, Word2Vec, BERT) on the provided domain-specific corpus in order to acquire more specialized semantic nuances.
   - **Proposed Hybrid Approach**: This new approach encompasses:
     – Creating dense embeddings for the terms in the glossary (from an LLM, along with their definitions) with the Sentence-BERT.
     – Employing a range of clustering algorithms, including K-Means, Agglomerative Clustering, DBSCAN, and K-GMA, to categorize semantically analogous terms according to these embeddings.
     – The clusters that were produced by the most successful clustering algorithm, K-GMA, were enriched using prompting techniques for identifying the ultimate synsets.
3. **Post-processing and Merging of Synsets**: The same post-processing pipeline was employed for numerous synonym extraction methods, most significantly the baseline and corpus-adapted embedding approaches, to transform sets of candidate synonyms into more organized synsets for evaluation.

In the following sections each of these stages and components are investigated.

4.2 Term Identification

A necessary step in domain-specific synonym extraction is the extraction of relevant term in the domain. Then the synonym sets are found within these extracted terms. In this thesis, several methods are evaluated to perform this task and their results are compared with the ground truth terms available in the dataset.

4.2.1 NER and Information Extraction Models

In Named Entity Recognition, some predefined categories are specified such as person, names, organizations, locations, etc and names entities are identified and

classified into one of these categories. Therefore, it is one of the methods that are evaluated in this thesis for extraction of domain-specific terms from the text chunks. The general workflow involved:

1. Providing each text chunk available in the dataset to the NER model
2. Creating a collection of all the terms extracted by the model
3. Entities were converted to lowercase for better comparison
4. For evaluation, these terms were compared against the ground-truth terms available for each domain in the dataset

The following NER techniques and models are evaluated:

BERT-based NER (Standard Token Classification). This was done with a pre-trained BERT model fine-tuned for NER. In this standard approach, the pipeline processes each chunk and identifies the entities in each domain.

NLTK-based NER. The Natural Language Toolkit (NLTK) also provides NER functionalities which are based on part-of-speech (POS) tagging and chunking rules with a pre-trained classifier inside. This process consists of tokenizing, POS tagging and the final step to identify named entities. It's generally effective for common entity types but might not be suited for novel domain terms.

SpaCy-based NER. Another popular NLP library is SpaCy that offers accurate and efficient models for NER. In its default models, it uses convolutional neural networks (CNNs). For term extraction, a SpaCy language model is loaded and text chunks are processed. As SpaCy models are trained on general-domain corpora, they might be able to detect more entity types.

NuExtract is a transformer-based model designed for information extraction using templates. Therefore, a template was provided to guide the model in the extraction process. The model is specifically instructed to look for the terms in the provided domain.

This NER stage was done to identify if minimally guided and off-the-shelf NER tools would be able to identify the target terms present in the dataset which are considered domain-specific vocabulary.

4.2.2 Proposed: LLMs for Term Extraction and Definition Generation

As an alternative method, Large Language Models are used for term extraction. This approach leverages LLMs' semantic reasoning to follow term extraction instructions and harnesses their generative power to produce domain-specific definitions.

LLM was prompted to extract the domain-specific key terms from the text chunks and generate a related definition for each term based on the context. The

output is two columns as 'term' and the related 'definition' generated by LLM. For example, a row might contain 'aerial application' as the term and a GPT-generated definition like 'The process of applying substances such as pesticides or fertilizers to crops from an aircraft.'

This LLM-generated glossary has two main purposes:

1. It provides a complete list of extracted domain-specific terms.

2. The definitions generated by LLM give us rich semantic context for each term which is later used in the hybrid synset creation method by embedding the whole 'term: definition' string for a more rich contextualized vector representation of each term.

4.3 Synset Creation

The GPT-generated extracted terms are used for synset creation in the proposed method.

4.3.1 Baselines for Synonym Candidate Generation

Different methods are evaluated, from traditional lexical lookup to more sophisticated embedding-based techniques, to create the synonym sets.

WordNet (NLTK). All the synonyms for each distinct target term in the dataset were retrieved. Using a widely recognized and manually-created lexical resource serves as a baseline to show how good can general-purpose thesauri perform for synonym extraction in specific domains.

Pre-trained Static Embeddings. Several pre-trained word embedding models are analyzed that were trained on large and general-purpose corpora. Synonyms of each target term are identified by looking for the closest terms in the embeddings space (e.g., by cosine similarity).

GloVe. Pre-trained GloVe embeddings are loaded from pretrained GloVe model which was trained on 42 billion tokens from the Common Crawl dataset that resulted in creating 1.9 million vocabularies and their 300-dimensional vectors.

fastText. Also pre-trained fastText embeddings are utilized. This model was trained on Wikipedia articles and News data and provided a vocabulary size of 1 million words with their 300-dimensional embeddings vectors. A potential advantage of fastText is that it can grab sub-word information and therefore generate embeddings for out-of-vocabulary (OOV) words. This analysis provides another baseline for a model with sub-word information capability.

Word2Vec. Pre-trained Word2Vec model was trained on 100 billion tokens, and provides 300-dimensional word embeddings with a vocabulary size of 3 million

words. This add another widely used pre-trained embedding model to the baselines. For target terms with multiple words (e.g., 'aerial application'), the vectors of its words ('aerial', 'application') were averaged to create one vector (mean-pooling). Therefore the effectiveness of popular pre-trained static embedding models in finding domain-specific synonyms are assessed.

Fine-tuned Static Embeddings

Using several models, word embeddings are adapted to each domain corpus in the dataset to try to capture more relevant semantic nuances.

fastText (fine-tuned on domain classification task). fastText was trained on a task for domain classification. First, labeled sentences are created from the text chunks with domain as label for each sentence. Then load the embedding vectors from a pre-trained model. After that, the fine-tuning for domain classification is done and word vectors are extracted. The hypothesis is that fine-tuning for domain classification makes the embeddings vectors present the words in the domain more accurately.

Word2Vec (fine-tuned from pre-trained). This approach starts by loading the pre-trained Word2Vec embeddings and further train them on domain-specific corpus in several versions. For words that are present in the corpus, all the embedding vectors are copied from the pre-trained model as initialization values for vocabularies and then training starts on domain sentences. In this way the model can leverage the general semantic knowledge accumulated by training on a large dataset while also adapting it to the domain-specific corpus.

Contextual Embeddings (BERT)

BERT was used as the main model representing a powerful contextual embedding model.

BERT Pre-trained. First, the terms were tokenized using BERT's tokenizer. If a term (e.g., 'deblossoming') was split into multiples sub-word tokens (e.g., 'de', 'bl', 'oss', 'omi', 'ng'), the embeddings for these sub-word tokens (from the model's embedding layer) are averaged. Therefore the capabilities of the embedding layer of a powerful contextual LM for the synonym extraction task is evaluated.

BERT Fine-tuned (on domain classification task). The pre-trained model was fine-tuned for domain-classification task on labeled sentences from the text chunks. Embeddings for vocabulary tokens were then extracted from this fine-tuned model. The hypothesis is that fine-tuning even for domain-classification task can help the embedding vector to be more domain-aware.

For all of the embedding-based methods evaluated above (GloVe, fastText, Word2Vec, BERT), the general synonym extraction process involves representing

each target term as a vector (by using mean-pooling for multi-word terms) and then finding the nearest k neighbors in the embeddings space.

4.3.2 Proposed: Clustering and LLM-based Refinement

Clustering Algorithms

Different clustering algorithm are used on embeddings of 'term: definition' strings. The goal is to group terms that are good synonymy candidates and has close semantic values.

K-Means. A grid search is performed for the optimal number of clusters. `random state` is set to zero for reproducibility. Euclidean distance is used as the metric.

Agglomerative Clustering. Cosine metric and the precomputed cosine distance matrix are used. The same values as K-Means for the number of clusters are used for a grid search.

DBSCAN. Different configurations are explored such as Euclidean metric and cosine metric with different number of minimum samples (the minimum density required for a region to be considered a cluster).

K-GMA. A customized script is utilized for community detection on a k-NN graph. A search is performed over `resolution` parameter to control cluster granularity. The K-GMA algorithm first builds a k-NN graph from the input embeddings and then applies the Louvain algorithm.

In the resulting clusters, each cluster is treated as a synset for evaluation purposes.

LLM-based Synset Refinement

In this step, output of the most promising clustering algorithm is passed to the LLM to do a refinement on these clusters and create more coherent synsets.

Clusters generated by the best-performing unsupervised clustering method are used as the basis for this experiment. Each cluster consists of a set of terms which are treated as batch of potentially related terms. For each of these clusters, an LLM is prompted for refinement to extract the groups of synonyms from the provided cluster. Each term from the original K-GMA clusters is assigned a new cluster ID by the LLM which effectively refines the initial groups.

In this step, commonsense reasoning and vast knowledge base of LLMs is utilized to correct some of the errors made by purely geometric clustering (e.g., splitting clusters that are too big, merging clusters that were fragmented incorrectly or removing outliers) to ensure that synonym extraction is coherent within each domain. The output of this stage also provides us with synsets which are directly used for evaluation purposes.

4.4 Synset Post-processing and Merging

In the baselines, for synonym extraction methods that produce a list of synonyms for each target term (i.e., WordNet, GloVe, fastText, Word2Vec, BERT), a standardized post-processing pipeline was applied to create and evaluate synsets.

1. **Initial Synset Formation**: For each target word 'T', if a method extracts synonyms $\{S_1, S_2, \ldots, S_n\}$, an initial synset is formed as $\{T, S_1, S_2, \ldots, S_n\}$. A limit on 'n' is applied (set to 5 here). Extracted synonyms are lower-cased.

2. **Synset Merging**: Synsets that share common terms are merged together. The synset merging process works by iteratively combining any synonym lists that share at least one common word. It repeatedly scans all lists, merging any two that have an overlap, and continues this cycle until a full pass occurs with no new merges. This ensures that all related synsets are transitively joined together, transforming many small, overlapping lists into a final set of larger, distinct, and non-overlapping synonym clusters.

3. **Handling Unassigned Terms**: If no synonym is found for a target term, either because its word or sub-word embedding is missing from the model's embedding space or it it not found in the lexical resource (e.g., WordNet), is it assigned to a unique singleton synset. This ensures that all target terms are accounted for evaluation purposes. Each such target term forms a synset containing only itself.

4. **Final Structure for Evaluation**: The result is a number of disjoint synsets.

This approach ensures that the results of different synonym generation methods are transformed into a standard format to be evaluated. The clustering-based methods and the final LLM-refined clusters inherently produce disjoint synsets and therefore bypass this post-processing step and are directly compared against the ground-truth synsets.

4.5 Potential Benefits of the Proposed Method

The novel methodology that is proposed in this thesis aims to leverage the semantic richness of domain-specific term definitions provided by an LLM and combine it with best clustering methods and a final LLM refinement to achieve the best results. Several benefits can be mentioned for the proposed method:

Avoiding context limit. One of the main limitations when working with LLMs is that they have a maximum input size that they can accept. For smaller models, this context size could be smaller, but for recent most-powerful models it

gets to 1 million tokens. While this might seem like a big enough context size for most use-cases, the longer inputs to a LLM could introduce other challenges that is mentioned in the next points.

Avoid lost-in-the-middle effect. A well-known effect called lost in the middle [115] can happen when a large input is given to an large context LLM. When some models are prompted with large input, about the size of their context limit, the performance significantly drops when they have to access information from the middle parts of their input context.

LLM can focus on fine-grained clustering. Giving a more focused problem to a LLM to solve, could result in a more fine-grained answer.

Usage of smaller models and less processing power. When the problem is simple enough, smaller LLMs could be used to solve the task which results in using less processing power and processing time and also in the case of using LLM APIs, it results in more budget-friendly solutions.

Chapter 5

Dataset and Experimental Setup

This chapter provides a detailed description of the dataset that is used for experiments, different baseline methods that were evaluated, the evaluation metrics used and the hardware and software environment in which the research was conducted. A clear review of these components is necessary for the correct interpretation and possible reproducing of the findings in this thesis.

5.1 Dataset

It's crucial to understand the main characteristics of the dataset used for the experiments. A private, custom-made dataset was provided for this thesis. Unlike other naturally occurring corpora, this dataset was engineered to facilitate the evaluation of domain-specific term and synonym extraction, focusing on interchangeability of synonyms within specific contexts.

5.1.1 Origin and Nature

The dataset used in this thesis was not a derivation of raw text gathered from different sources but was rather synthetically constructed through a multi-stage process that was designed to create a suitable dataset for evaluating synonymy. This process involved three main steps:

Glossary Acquisition from Curated Sources

The initial step was gathering glossaries specific to each domain. For this purpose some curated online sources were chosen. Following this path, it can be said with

high confidence that the terms are already recognized as relevant in their respective fields. The sources included:

Wikipedia Glossary Pages. For general domain like 'geography', glossaries such as 'Glossary of geography terms' [116] from Wikipedia were scraped. These pages list terms and their definitions which provide a structured starting point for the dataset creation.

Governmental and Institutional Glossaries. For specialized domain like 'legal', resources from official bodies were utilized, such as 'Legal Terms Glossary' [117] from U.S. Department of Justice. These sources can be relied on for providing precise, legally defined terms.

Academic and Expert-Curated Dictionaries. For fields like 'medical', highly reputable academic sources were used such as 'Medical Dictionary of Health Terms' [118] which is curated by Harvard. In this dictionaries, expert-validated terminology and definitions are provided.

Manual Synset Creation from Acquired Glossaries

Following the acquisition of domain-specific glossaries, the next step is to create the ground-truth synsets. A human annotator reviewed the scraped terms and their definitions. The objective was to group terms that could be considered true synonyms or highly interchangeable in the context of their specific domain.

This process is vital because synonym detection is often ambiguous and context-dependent. Manually doing this process ensures a high-quality and human-validated ground-truth for synonymy. Also, some purely automatic methods might make mistakes in identifying true synonyms from terms with semantic relatedness. It ensures the precise semantic boundaries for each synset which serves as a gold standard for evaluation.

GPT-based Text Chunk Generation with Synonym Interchangeability

The final step involved asking GPT-4o-mini to generate contextual text chunks. It was prompted to create text chunks in such a way that any synonym within that synset could replace the target term in the chunk without altering the meaning or grammatical correctness of the sentence.

5.1.2 Size and Structure

The dataset contains 3296 rows. Each row represents a unique text chunk. The main columns used in this thesis are:

- **target**: The reference term that is present in the 'chunk'.
- **chunk**: A segment of text that contains the 'target' term.

- **domain**: The specific domain to which the 'target' term and 'chunk' belong.

- **synset**: A label which represent the ground-truth synset to which the 'target' term belongs.

5.1.3 Content Characteristics and In-depth Analysis

An in-depth analysis shows several observations about the dataset:

Chunks. The average word count per chunk is approximately 60 words. Each chunk contains about 3 sentences and each sentence is about 20 words long.

Domains. The dataset spans 7 unique domains: 'agriculture', 'automotive', 'economy', 'geography', 'legal', 'medical', and 'technology'. Because of this diversity, the robustness of different synonym extraction methods across different domain-specific vocabularies can be tested.

Target Terms: There are 1099 unique target terms which are distributed across different domains. These terms are the primary focus in synonym extraction. According to Table 5.1 About 55% of target terms are one-word terms and the rest are multi-word terms of mostly 2 to 4 words long.

Synsets. The dataset defines 428 unique ground-truth synsets. Each 'target' term is associated with one of these 'synset' labels. Synsets include between 4 to 18 terms.

Chunk and Target Term Relationships. According to Table 5.2 each 'target' term appears in 2 to 5 different 'chunk' entries which provides multiple contextual examples for each term. 57% of chunks contain exactly one mention of their target term while others mostly mention the target term 2 to 4 times. 20.3% of chunks include plural form of the target term which in almost all cases is accompanied with a singular form of the term in the same chunk.

Number of Words	Percentage
1	55.78%
2	36.49%
3	6.64%
4	0.91%
5	0%
6	0.18%

Table 5.1: Term Length Distribution by Word Count

Count of Term in Chunk	Percentage
0	1.7%
1	57.43%
2	32.89%
3	6.74%
4	1.03%
5	0.21%

Table 5.2: Target Term Count Distribution

5.1.4 Errors in the dataset

The dataset includes 46 chunks that don't include an exact match of their target term in plural or singular form. They don't mention the target term at all or change it in a way that make an exact match impossible (by altering spaces, grammatical signs, etc). These chunks in total affect 19 unique target terms.

The dataset includes 2 target terms which are each assigned to two different domains. This is in contradiction to each term being assigned to one domain.

5.1.5 Pre-processing

Two main pre-processing stages was done on the data during different experiments.

Lower-casing. For experiments involving Word2Vec training, all entries of 'chunk' column were converted to lower-case. This can be considered as a standard pre-processing step for many NLP models that are case-sensitive.

Stop-word Removal and Stemming. For specific variants of Word2Vec, stop-words were removed using English stop-word list provided by NLTK and stemming provided by SpaCy. This was performed to evaluate the impact of these common text normalization techniques.

5.1.6 Unique Characteristics and Considerations

The synthetic and controlled character of this dataset offers exclusive advantages to performing analysis on synset creation methods:

Controlled Synonymy. The systematic design for synonym exchange between chunks provides clear, strong contextual hints to synonymy, which is ideal for testing any model's ability to capture this specific semantic relation.

High-Quality Ground Truth. The careful choice of synsets from credible glossaries creates a reliable standard for evaluation, thus minimizing vagueness in the ground truth itself.

Domain Specificity. The information is inherently domain-specific, so direct comparison of performance of techniques on specialized domains can be done.

It is necessary to mention that the regulated characteristics also suggest specific restrictions:

Possible dissimilarity to noisy real-word text. The data may not fully capture the noise, vagueness, and more implicit semantic relationships found in truly raw, uncurated, real-world domain-specific text.

Possible unnatural linguistic style. The produced chunks may carry a particular linguistic style or simplicity that is not found in naturally occurring expert speech.

Despite these aspects, the dataset concerned presents a solid and targeted environment for the extensive analysis of synonym extraction methods across specialized domains, particularly in their ability to identify equivalent terms.

5.2 Experimental Setup

A wide range of models and checkpoints were used for evaluation of different methods that will follow.

5.2.1 Term Extraction / NER Models

These models were used for evaluation of term extraction methods:

NuExtract. This method deals with extracting domain-related words from chunks of text according to the NuExtract models. The dataset is read and for each text chunk in the dataset, a specific extraction template is constructed dynamically based on the domain of the chunk. The template format is `{"words_related_to_DOMAIN_domain": []}` (e.g., `{"words_related_to_agriculture_domain": []}`). The `AutoModelForCausalLM` and `AutoTokenizer` classes of the transformers library are used to load either NuExtract-1.5-tiny or NuExtract-1.5. The model is executed on a GPU.

Batches of text segments are processed along with their corresponding templates. The prompt required for an extraction is created by joining an input marker, the JSON template, the text, and an output marker. The model then produces new tokens to finish the output. The output is decoded as JSON, from which the list of terms pertaining to the provided extraction field (e.g., `words_related_to_agriculture_domain`) is taken.

NLTK-NER. To find possible domain-specific terms, this approach makes use of NLTK's integrated Named Entity Recognition features. The dataset is loaded by the script. It starts by making sure the required NLTK resources are downloaded. The following actions are taken for every text segment in the dataset: `nltk.sent_tokenize` and `nltk.word_tokenize` are used to tokenize the sentences that

make up the chunk. `nltk.pos_tag` is used to apply part-of-speech (POS) tagging to the tokenized sentences. Lastly, `nltk.ne_chunk` is used to identify and label named entities through Named Entity Recognition. These named entities' text content is extracted and gathered.

SpaCy-NER. This technique extracts domain-specific terms using SpaCy's pre-trained statistical model for Named Entity Recognition. The SpaCy language model's `en_core_web_sm` is loaded with the dataset. The loaded SpaCy NLP pipeline processes each text chunk in the dataset. Next, `doc.ents` is used to extract named entities from the processed document.

Term Extraction using an LLM. Each chunk of text is separately given to the GPT-4o-mini model with a prompt specifying the LLM to look for domain-specific terms in the chunk.

Post-processing

All terms that are extracted with each technique are lower-cased. Only those extracted terms that occur in the set of all unique ground-truth target terms in the dataset are considered for the purpose of evaluation. The recall of the extracted terms is then calculated with respect to the ground-truth target terms, both overall and domain-wise.

5.2.2 Synonym Extraction Baselines

The main methods used for baselines are reviewed and also the proposed framework is described. Several experiments were done to perform a through analysis of different methods that could be used for synset creation. The experimental setup for each of these methods in described below.

WordNet. This approach uses an existing lexical database to retrieve synonyms. Every distinct target term in the dataset is iterated through by the script. It retrieves all synsets related to each target term by executing a query against the `nltk.corpus.wordnet` module. All lemma names are taken out of these synsets and gathered as potential synonyms.

Glove Pre-trained. This approach finds synonyms based on vector similarity using pre-trained GloVe embeddings. The 300-dimensional GloVe embeddings are first loaded into memory. These loaded embeddings are used to build an FAISS (Facebook AI Similarity Search) index, which normalizes the vectors and generates a `IndexFlatIP` for inner product (cosine) similarity in order to enable effective similarity search. The distinct target terms from the dataset are then iterated through by the script.

The similarity search is carried out for every target term. The vector representation of a target term that is a multi-word expression (such as 'aerial application') is

calculated by mean-pooling the embeddings of its individual words. The function then uses cosine similarity to query the FAISS index and retrieve the top nearest neighbors (apart from the target term itself). For every target term, these potential synonyms are gathered together with their similarity scores.

fastText Pre-trained. Using a vector similarity approach similar to GloVe, this method uses pre-trained fastText embeddings for synonym extraction. The embeddings are loaded in the memory. A FAISS index is constructed for the loaded embeddings. The distinct target terms from the dataset are then iterated through by the script. The top nearest neighbors are identified for every target term. Mean-pooling of constituent word embeddings is used for target terms with multiple words.

fastText Fine-tuned. Using the given corpus, this approach makes use of fastText embeddings that have been optimized on a domain classification task. The first step in the process is preparation of training data, in which labeled sentences are created by extracting sentences form the text chunks in the dataset and labeling them with their corresponding domain. Labeled sentences are divided into training and validation sets. The model is then trained by the script.

For training, the fastText model is initialized with `dim=300`, `lr=0.1`, and `wordNgrams` set to 2 and the word embeddings are initialized using `wiki-news-300d-1M` as `pretrainedVectors`. The epoch that produces the best validation precision is chosen for the final model after the model has been trained for a variable number of epochs. After being extracted, word vectors from the trained model are used to extract synonyms. The procedure is the same as the 'fastText Pre-trained' method: A FAISS index is constructed and each target word's nearest neighbors are found through a search.

BERT Pre-trained. This approach uses embeddings from the `bert-base-uncased` pre-trained model. Before searching for synonyms, there are two primary steps in the process: embedding extraction and synonym search. The first step is to extract the static input embeddings for every token in BERT's vocabulary from `model.bert.embeddings.word_embeddings`.

Then, the same pre-trained BERT model is loaded. The script uses `BertTokenizer` to tokenize each distinct target term. In order to generate a single 768-dimensional vector representation for the target term, it then retrieves the static input embeddings for each sub-word tokens and calculates their mean. Lastly, synonym extraction is performed. A FAISS index (with `size=768`) is constructed on the vocabulary embeddings. The top nearest neighbors are identified for every target term.

BERT Fine-tuned. This approach makes use of embeddings from a `bert-base-uncased` model that has been refined on a domain classification task. To serve as training data, the dataset's sentences are labeled with their domain. `transformers.Trainer` is used to fine-tune the model with `fp16=True` enabled, 10

epochs, a batch size of 16, and a learning rate of `1e-4`. Based on evaluation loss, the optimal model is selected. Following fine-tuning, all vocabulary tokens' static input embeddings are extracted. After that, the script loads the fine-tuned model to calculate mean-pooled static input embeddings for every distinct target term. Although it makes use of these refined vocabulary and target word embeddings, the synonym extraction procedure is the same as the 'BERT Pre-trained' method.

Word2Vec Pre-trained. The Google News dataset's pre-trained Word2Vec embeddings are used in this technique. `gensim.downloader.load` is used to load the 300-dimensional embedding vectors. For effective search, a FAISS index is constructed. After that, the script iterates through each of the distinct target terms, performing mean-pooling for each one and identifying its closest neighbors.

Word2Vec Fine-tuned. Using a tokenized and lower-cased version of the domain corpus, this technique refines the pre-trained Word2Vec (Google News) model. This corpus is prepared by the script using `nltk.word_tokenize` to tokenize sentences and lowercase all text chunks. It starts a Word2Vec model with `vector_size=300`, `window=5`, `min_count=1`, and `workers=4` for fine-tuning. It builds the starting vocabulary from the corpus. Then, it loads the pre-trained `word2vec-google-news-300` vectors and copies the overlapping word vectors into the newly initialized model. `model.train` is then used to further train the model on the corpus for five epochs. In order to extract synonyms, word vectors are extracted from this refined model, a FAISS index is constructed, and for each target term, the nearest neighbors are found using mean-pooling. Three different experiments were done on Word2Vec models:

- **pri**. Is the default experiment that is explained above.
- **bis**. For creating the corpus in this version, stop-words are removed. The rest of the process is the same.
- **tri**. For creating the corpus in this version, words as stemmed. The rest of the process is the same.

Post-processing

A standard post-processing step is done after synset creation by each method, which consists of converting all extracted synonyms to lowercase, eliminating any synonyms that are not included in the dataset's set of unique target terms, and combining any synsets that share terms. Lastly, distinctive singleton synsets are created for any extracted terms that were not allocated to an extracted synset.

5.2.3 Proposed Method: Clustering of Term Definitions and GPT Refinement

This methodology, which is the foundation of the suggested approach, aims to use Large Language Models' (LLMs) reasoning powers and the semantic richness of term definitions to extract high-quality synsets. The three primary steps of the process are glossary preparation, unsupervised clustering and embedding LLM-based refinement.

Embeddings of Glossaries

The glossary of domain-specific terms and their definitions, produced by the GPT-4o-mini model, serves as the pipeline's first input. A list of terms with detailed definitions is provided in this glossary. The term and its definition are combined into a single string for every entry in this glossary, using the format 'term: definition' (for example, 'aerial application: The process of applying substances such as pesticides or fertilizers to crops from an aircraft.'). In contrast to using just the term itself, this concatenation gives each term a richer semantic context, which enables the subsequent embedding model to capture more nuanced meaning.

Dense vector representations are then produced by feeding these concatenated strings into a Sentence-BERT model, specifically `SentenceTransformer("all-MiniLM-L6-v2")`. These embeddings serve as the input for the clustering algorithms since they capture the semantic meaning of the term as well as its definition. These Sentence-BERT embeddings are then subjected to a variety of unsupervised clustering algorithms in the following step. As the goal, semantically related terms are grouped according to their rich, definition-informed vector representations. The application and assessment of these algorithms are carried out by the script.

Clustering

Different clustering methods are used to group the terms as the first step of the proposed hybrid approach.

k-means. The Sentence-BERT embeddings were subjected to the `sklearn.cluster.KMeans` algorithm. In order to minimize the within-cluster sum of squares, this algorithm divides the data into a predetermined number of clusters. The ideal number of clusters `n_clusters` was found using a grid search, with values ranging from 7, 50, 100, 200, 300, 400, and 500. For reproducibility, the `random_state` parameter was set to 0. For clustering, the standard Euclidean distance metric was applied. To ascertain the cluster assignments, the embedding matrix was subjected to the `fit` method of the `KMeans` class.

Agglomerative Clustering. The `sklearn.cluster.AgglomerativeClustering` algorithm was used, which iteratively merges clusters to accomplish hierarchical

clustering. The `metric` was set to `precomputed` for this method, and the `linkage` criterion was set to `average`. The Sentence-BERT embeddings' pairwise cosine distance matrix had to be pre-calculated using `sklearn.metrics.pairwise.cosine_distances`. The `AgglomerativeClustering` model was then fed this precomputed distance matrix as input. A grid search was performed for the `n_clusters` parameter, similar to KMeans, and values of 7, 50, 100, 200, 300, 400, and 500 were investigated. The clustering was done using the `fit` method.

DBSCAN. The `sklearn.cluster.DBSCAN` (Density-Based Spatial Clustering of Applications with Noise) algorithm was also tested. DBSCAN labels points that are isolated in low-density areas as outliers and groups together points that are densely packed together. The number of clusters does not need to be predetermined when using this method. Two distinct distance metrics, `euclidean` and `cosine`, were used in the experiments. Two values for `min_samples` (the number of samples in a neighborhood for a point to be considered a core point) were tested for each metric: 2 and 5. Throughout all DBSCAN experiments, the `eps` parameter (the maximum distance between two samples for one to be regarded as being in the neighborhood of the other) was set to 0.1. Cluster labels were obtained using the `fit_predict` method.

K-GMA. Additionally, a customized clustering method named K-GMA (k-nearest neighbors Graph-based Louvain Algorithm) was made available and used on data points. This algorithm uses the Louvain community detection algorithm to find clusters after first creating a k-nearest neighbors graph from the input embeddings (by default, `n_neighbors=3` and `metric='cosine'`). The embeddings are subjected to the `fit` method of the `kGMA` class. Over the Louvain algorithm's `resolution` parameter, a grid search was conducted, looking at values of 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0 (default), 1.2, and 1.4. For every resolution, cluster assignments are obtained using the `predict` method of the `kGMA` class.

The resulting clusters from each clustering experiment were transformed into a dictionary format, with each key denoting a cluster ID (e.g., `EXTRACTED_SYNSET_1`) and the value being a list of terms associated with that cluster. In these clusters, only terms that appeared in the original ground-truth were kept. The average F1-score, Precision and Recall (as described in Section 5.3) for each target term were used to assess each clustering configuration's performance.

GPT Refinement

Lastly, an extra GPT-based synset refinement step was applied to the most promising clustering results, namely those produced by K-GMA with a resolution of 0.2. The goal of this innovative step is to further enhance the quality of the machine-generated clusters by utilizing the semantic comprehension and generative powers of LLMs. The terms within each cluster that was determined by K-GMA (resolution

0.2) were supplied as input to the GPT-4.1-nano or GPT-4o-mini models. From this set of terms, the LLM was asked to 'extract the group of synonyms related to a specific domain.'

In order to bring the terms closer to true domain-specific synonymy, this process essentially asks the LLM to re-evaluate and possibly re-group them, using its extensive knowledge base to screen out noise or split excessively broad clusters. Following this procedure, the corresponding GPT model assigns a new cluster ID to each term. For these GPT-refined synsets, the per-domain F1-score and recall are computed and compared to the ground truth.

5.3 Evaluation Metrics

5.3.1 For Term Extraction / NER

Primarily **recall** is used as the percentage of ground-truth target terms that were correctly extracted by NER/LLM method. this is calculated overall and per domain.

$$\text{Recall} = \frac{|\text{Extracted Terms} \cap \text{Ground Truth Target Terms}|}{|\text{Ground Truth Target Terms}|}$$

5.3.2 For Synset Extraction

The primary reported metrics in this section are adapted from information retrieval and focus on the quality of synonyms found for each target term. The evaluation is performed on distinct target terms. For each distinct target word w in a domain (or overall):

1. $GT(w)$: The set of ground-truth synonyms for w excluding w itself.
2. $E(w)$: The set of extracted synonyms for w from the synset generated by the method being evaluated, excluding w itself. (For baseline methods, initially, up to 5 candidate synonyms are considered before merging; for clustering methods, $E(w)$ is the set of other terms in the same cluster as w).
3. $TP(w) = |E(w) \cap GT(w)|$ (True Positives: correctly identified synonyms)
4. $FP(w) = |E(w) \setminus GT(w)|$ (False Positives: incorrectly identified synonyms)
5. $FN(w) = |GT(w) \setminus E(w)|$ (False Negatives: missed ground-truth synonyms)
6. $$\text{Precision}(w) = \begin{cases} 1, & \text{if } E(w) = \emptyset \text{ and } GT(w) = \emptyset \\ 0, & \text{if } TP(w) + FP(w) = 0 \text{ and } GT(w) \neq \emptyset \\ \frac{TP(w)}{TP(w)+FP(w)}, & \text{otherwise} \end{cases}$$

7.

$$\text{Recall}(w) = \begin{cases} 1, & \text{if } GT(w) = \emptyset \text{ and } E(w) = \emptyset \\ 0, & \text{if } TP(w) + FN(w) = 0 \text{ and } E(w) \neq \emptyset \\ \frac{TP(w)}{TP(w)+FN(w)}, & \text{otherwise} \end{cases}$$

8.

$$\text{F1-score}(w) = \begin{cases} 0, & \text{if Precision}(w) + \text{Recall}(w) = 0 \\ \frac{2 \cdot \text{Precision}(w) \cdot \text{Recall}(w)}{\text{Precision}(w)+\text{Recall}(w)}, & \text{otherwise} \end{cases}$$

The final Precision, Recall and F1-score (overall and per domain) is reported as the average of these metrics over all distinct target words w in the respective scope.

Chapter 6

Results

This chapter focuses on presenting experimental results obtained by benchmarking several synonym extraction methods in domain-specific contexts. The results are organized based on stages of methodology: term extraction/NER, baseline and corpus-adapted synonym extraction, and the proposed clustering with GPT refinement approach. For each set of experiments, results are presented in tabular format followed by a discussion about performance, trend and implications.

6.1 Term Extraction Results

In the first stage of the experimental pipeline, domain-specific terms are identified. The effectiveness of various NER and LLM-based methods was calculated by their ability to recall the ground-truth target terms present in the dataset and are presented in Table 6.1.

Method	Overall	Agriculture	Automotive	Economy	Geography	Legal	Medical	Technology
NuExtract-1.5	0.84	0.85	0.91	0.75	0.78	0.84	**0.90**	0.87
NuExtract-tiny	0.74	0.67	0.90	0.75	0.65	0.60	0.79	0.71
GPT-4o-mini	**0.88**	**0.91**	0.90	0.87	**0.80**	0.87	**0.90**	**0.88**
GPT-4.1-nano	**0.88**	0.88	**0.92**	**0.89**	**0.80**	**0.89**	**0.90**	0.86

Table 6.1: Recall of Ground-Truth Target Terms by NER/LLM Term Extraction Methods (Overall and Per Domain).

As shown in Table 6.1, LLM-based approaches (GPT-4o-mini and GPT-4.1-nano) and the larger NuExtract-1.5 model demonstrated strong performance in extraction of target terms.

NuExtract-1.5 also had an acceptable performance with an overall recall of 0.84. The smaller NuExtract-tiny model achieved a lower overall recall of 0.74. This indicates that model size and capacity can play a significant role in the NuExtract

framework's ability to generalize and perform an accurate term extraction based on the provided template.

GPT models models have the highest overall recall at 0.88. GPT-4o-mini has slightly better performance in 'agriculture' and 'technology' domains and GPT-4.1-nano had a slight edge in 'automotive', 'economy' and 'legal' domains. Their strong performance shows that LLMs, when prompted appropriately, can perform effective term extraction on a wide range of domain-specific vocabulary.

The experimental results for traditional NER tools like NLTK, a standard BERT-based NER pipeline, and SpaCy showed no success in identifying the specific target terms in the dataset. They show negligible recall of close to zero.

Although traditional methods can adequately determine common entity types (like Person, Location, Organization), they are not well-suited for extraction of more complex and often multi-word technical terms that are present in different domains.

6.2 Baseline Synonym Extraction Results

This section provides the results of different synonym extraction methods that relied on pre-trained embeddings vectors for this task or used updated word vectors from models that were fine-tuned on the corpus. Precision, Recall and F1-Score was calculated for each method and are presented in in Table 6.2.

Overall, the pre-trained static embeddings, particularly 'fastText Pre-trained' and 'Word2Vec Pre-trained', provide the best performing, though still low, F1-scores (around 0.12) among these methods. WordNet shows a slightly lower F1-Score. Fine-tuning fastText for a classification task improves recall but not F1-score due to a drop in precision. Also fine-tuning pre-trained Word2Vec models lead to bad results with extremely high recall but no precision. BERT-based methods also perform poorly.

Since most embeddings, especially static ones like fastText and Word2Vec, conflate semantic similarity with broader relatedness, retrieving related but non-synonymous words, the low F1-scores across methods indicate that nearest neighbor search in embedding space struggles to capture true synonymy. By overfitting and increasing recall at the expense of precision, fine-tuning worsens this. WordNet lacks inclusion of all the terms, and BERT-based embeddings, although context-aware, are inappropriate for isolated word comparison because they are context-dependent. In general, the fine-grained semantic alignment required for accurate synonym detection is absent from the embedding spaces.

Method	Overall	Agriculture	Automotive	Economy	Geography	Legal	Medical	Technology
				F1-Score				
WordNet	0.08	0.11	0.07	0.01	0.14	0.03	0.13	0.02
Glove Pre-trained	0.07	0.07	0.01	0.02	0.08	0.08	0.15	0.09
fastText Pre-trained	**0.12**	0.13	0.03	0.04	0.15	0.04	0.19	0.23
fastText Fine-tuned	0.11	0.15	0.03	0.04	0.14	0.06	0.19	0.12
BERT Pre-trained	0.03	0.02	0.05	0.01	0.04	0.05	0.02	0.07
BERT Fine-tuned	0.03	0.02	0.05	0.01	0.04	0.05	0.02	0.07
Word2Vec Pre-trained	**0.12**	0.16	0.11	0.04	0.11	0.10	0.18	0.12
Word2Vec Fine-tuned pri	0.01	0.0	0.0	0.0	0.01	0.02	0.01	0.0
Word2Vec Fine-tuned bis	0.00	0.02	0.0	0.0	0.0	0.0	0.0	0.0
Word2Vec Fine-tuned tri	0.00	0.01	0.0	0.0	0.0	0.0	0.0	0.0
				Recall				
WordNet	0.07	0.10	0.07	0.01	0.13	0.02	0.12	0.02
Glove Pre-trained	0.34	0.10	0.55	0.15	0.25	0.33	0.55	0.56
fastText Pre-trained	0.30	0.19	0.38	0.10	0.33	0.42	0.39	0.43
fastText Fine-tuned	0.46	0.33	0.59	0.35	0.46	0.48	0.49	0.67
BERT Pre-trained	0.10	0.02	0.21	0.04	0.10	0.29	0.06	0.11
BERT Fine-tuned	0.10	0.02	0.21	0.04	0.10	0.29	0.06	0.11
Word2Vec Pre-trained	0.17	0.17	0.24	0.05	0.15	0.23	0.25	0.17
Word2Vec Fine-tuned pri	0.81	0.73	0.96	0.68	0.79	0.81	0.92	0.82
Word2Vec Fine-tuned bis	**0.86**	0.82	0.97	0.76	0.81	0.85	0.92	0.95
Word2Vec Fine-tuned tri	0.72	0.69	0.66	0.52	0.83	0.87	0.78	0.83
				Precision				
WordNet	0.10	0.15	0.08	0.01	0.17	0.05	0.16	0.04
Glove Pre-trained	0.08	0.08	0.01	0.03	0.09	0.09	0.14	0.10
fastText Pre-trained	0.11	0.13	0.02	0.04	0.14	0.04	0.19	0.22
fastText Fine-tuned	0.11	0.15	0.03	0.05	0.12	0.05	0.18	0.12
BERT Pre-trained	0.03	0.02	0.04	0.01	0.04	0.04	0.01	0.07
BERT Fine-tuned	0.03	0.02	0.04	0.01	0.04	0.04	0.01	0.07
Word2Vec Pre-trained	**0.12**	0.17	0.10	0.04	0.11	0.08	0.18	0.12
Word2Vec Fine-tuned pri	0.01	0.0	0.0	0.0	0.01	0.03	0.01	0.0
Word2Vec Fine-tuned bis	0.00	0.01	0.0	0.0	0.0	0.0	0.0	0.0
Word2Vec Fine-tuned tri	0.00	0.01	0.0	0.0	0.0	0.0	0.0	0.0

Table 6.2: Per-domain basis F1-Score, Recall, and Precision for Baseline and Corpus-Adapted Synonym Extraction Methods. The best values for each metric are highlighted.

6.3 Clustering and GPT Refinement Results

This section evaluates the performance of the proposed multi-stage methodology, which involves:

1. Generating embeddings for 'term: definition' pairs (from the GPT-4o-mini glossary) using Sentence-BERT.

2. Clustering these rich semantic embeddings using various algorithms.

3. Refining the best clusters using GPT models.

6.3.1 Clustering Algorithm Performance

In the firs step, different clustering algorithms and their parameters are evaluated on the Sentence-BERT embeddings. The overall F1-score, precision and recall are averaged across all distinct terms and reported in Table A.1.

The results shown in Table A.1 show that using embeddings of 'term: definition' pairs from GPT-4o-mini (via Sentence-BERT) leads to much better F1-score compared to baseline methods.

K-Means. Performance of this method improves with increasing the number of clusters. The best F1-score of 0.36 is achieved with number of clusters set to 500 being close to 428 ground-truth synsets which could be a reason for the best performance.

Agglomerative Clustering. Similar to K-Means, F1-score increases as the number of clusters increase. It peaks with an F1-score of 0.31 with number of clusters set to 400 which is again close to the original number of ground truth synsets.

DBSCAN. This method could not perform adequately with a high recall but very low precision and F1-score. This indicated that DBSCAN groups many of the data points together into large clusters or considers many of the data points as noise.

K-GMA. This method shows strong performance, in particular in lower resolution values. The best performance in shown with an F1-score of 0.36 with resolution being set to 0.2 or 0.4. Bigger resolutions lead to a decrease in the number of clusters detected. It's the only clustering method that shows strong performance without needing to force a specific number of clusters which shows the suitability of this clustering method for this task in combination with rich semantic information stored in term-definitions vectors that can from district groups that K-GMA can identify.

6.3.2 GPT Refinement Performance

The candidates from clustering methods were selected and along with the GPT-refined results of K-GMA clusters, were evaluated on a per-domain basis. It's also noteworthy to mention that the number of clusters after GPT refinement increases form the initial 414 clusters in K-GMA (res=0.2) to 432 in GPT-4o-mini and 460 in GPT-4.1-nano.

The most notable result is the significant improvement in F1-score after applying GPT-based refinement to K-GMA clusters. It achieved a substantial jump from 0.36 F1-score of K-GMA alone and surpasses all baselines and corpus-based methods, achieving an F1-score of 0.51 in GPT-4o-mini and 0.53 in GPT-4.1-nano. The GPT models effectively re-structure the clusters. They improve precision significantly which implies that these models are doing good at filtering non-synonymous terms

Method	Overall	Agriculture	Automotive	Economy	Geography	Legal	Medical	Technology
F1-Score								
K-GMA (resolution=0.2)	0.36	0.38	0.44	0.25	0.27	0.28	0.45	0.49
Agglomerative (n_clusters=400)	0.31	0.35	0.37	0.19	0.27	0.16	0.45	0.38
KMeans (n_clusters=500)	0.36	0.41	0.41	0.23	0.32	0.32	0.46	0.50
DBSCAN (metric=_cosine, min_samples=5	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.03
K-GMA (resolution=0.2) + gpt-4o-mini	**0.51**	**0.52**	**0.61**	**0.31**	**0.49**	**0.37**	**0.64**	**0.62**
K-GMA (resolution=0.2) + gpt-4.1-nano	**0.53**	**0.54**	**0.63**	**0.37**	**0.51**	**0.42**	**0.64**	**0.61**
Recall								
K-GMA (resolution=0.2)	0.59	0.61	0.69	0.47	0.52	0.48	0.72	0.66
Agglomerative (n_clusters=400)	0.66	0.75	0.76	0.53	0.59	0.54	0.72	0.71
KMeans (n_clusters=500)	0.55	0.59	0.64	0.39	0.51	0.43	0.70	0.63
DBSCAN (metric=_cosine, min_samples=5	0.79	0.83	0.85	0.75	0.71	0.80	0.85	0.81
K-GMA (resolution=0.2) + gpt-4o-mini	0.50	0.53	0.58	0.29	0.47	0.35	0.67	0.61
K-GMA (resolution=0.2) + gpt-4.1-nano	0.51	0.54	0.60	0.35	0.50	0.40	0.62	0.57
Precision								
K-GMA (resolution=0.2)	0.31	0.32	0.38	0.23	0.23	0.24	0.38	0.45
Agglomerative (n_clusters=400)	0.26	0.29	0.31	0.13	0.22	0.13	0.40	0.31
KMeans (n_clusters=500)	0.32	0.38	0.36	0.18	0.26	0.30	0.39	0.45
DBSCAN (metric=_cosine, min_samples=5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
K-GMA (resolution=0.2) + gpt-4o-mini	0.55	0.55	0.68	0.37	0.54	0.45	0.66	0.67
K-GMA (resolution=0.2) + gpt-4.1-nano	0.58	0.57	0.71	0.42	0.57	0.50	0.67	0.69

Table 6.3: Per-domain basis F1-Score, Precision and Recall for Selected Clustering Methods and GPT Refinement. The best F1-score in each family of methods is highlighted.

and spilling overly broad clusters from K-GMA into more semantically coherent synsets that are more in-line with the ground-truth.

GPT-4.1-nano outperforms GPT-4o-mini in F1-score across most domains. This suggests that the capabilities or knowledge base of this model could be better suited for this synonym refinement task. The GPT-refined methods achieve the highest F1-score in every domain. Domains like 'automotive', 'medical', and 'technology' see the highest F1-scores with GPT refinement (often >0.60), which suggests that a combination of clustering and GPT-refinement could be particular effective for identification of synonyms in technical vocabularies in these domains.

Interestingly, the recall for the GPT-refined methods (0.50-0.51) is lower than the recall for K-GMA alone (0.59) or Agglomerative clustering (0.66). This suggests that GPT refinement process might be conservative and potentially discarding some true synonyms if it prioritizes high confidence. This is a precision-recall tradeoff, where GPT refinement goes for higher precision.

The proposed method, particularly 'K-GMA (resolution=0.2)' clustering of Sentence-BERT embeddings of 'term: definition' pairs, followed by refinement with GPT-4.1-nano, stands out as the top-performing approach. It achieves an overall F1-score of 0.53, significantly outperforming all other benchmarked methods. This highlights the power of combining rich semantic embeddings derived from LLM-generated definitions with the explicit reasoning and knowledge refinement capabilities of advanced LLMs.

The proposed clustering and GPT refinement shows several strong points:

Rich Semantic Input. Utilizing Sentence-BERT on strings formatted as 'term: definition' yields a significantly more comprehensive semantic representation for each term in comparison to relying solely on the embeddings of the term itself. The definition serves to explicitly contextualize the term.

Successful Clustering. K-GMA on these dense embeddings was successful in clustering semantically similar terms, with F1-scores of approximately 0.36, which is considerably higher than most baselines.

Power of LLM Refinement. The most significant improvement was obtained by using GPT models to refine the K-GMA clusters. This demonstrates that LLMs can also act as effective 'curators' or 'reasoners' over machine-generated candidate clusters and leverage their vast knowledge to correct errors. They likely help in:

- **Disambiguation**: When a cluster contains words with slightly different senses, GPT can separate them.
- **Noise Reduction**: Elimination of words that bear a very weak relation.
- **Enforcing Domain Consistency**: Taking note of domain-specific synonyms for the given domain implied by the cluster's content.

Common Types of Errors and Challenges

There are several common errors and challenges:

Multi-Word Expressions (MWEs). Processing MWEs is still an open problem. Mean-pooling MWE embeddings is ad-hoc and not necessarily accurate in their compositional meaning. Solutions that can process MWEs more globally are needed. The approach proposed approach exploits the fact that GPT is able to generate definitions of MWEs, which can be embedded by Sentence-BERT.

Granularity of Meaning. True synonyms are difficult to separate from close relatives (e.g., co-hyponyms, hypernyms/hyponyms). An F1-score of 0.53, while the highest, still indicates room for improvement.

Domain Specificity. Certain words have different synonyms in different domains. While the NER methods and definitions produced by GPT attempt to introduce domain context, it is still challenging to keep the end synsets strictly domain-based. The GPT refinement phase asks for domain-relevant synonyms, which seems to help.

Data Sparsity across Domains. Even with a specialized corpus, some domain-specific terms will be infrequent, making it difficult for data-driven models by themselves to learn their meaning comprehensively. LLMs, with their more comprehensive pre-training, can help mitigate this.

Answering Key Research Questions

Domain classification fine-tuning of FastText helped recall modestly but not F1-score. Fine-tuning pre-trained Word2Vec was also problematic. This shows that naive fine-tuning of static embeddings in small domain corpora needs to be done with caution. The approach suggested, generating definitions via GPT (using its general knowledge) and then clustering, seems a more judicious use of domain-relevance.

GPT-4o-mini (LLM-based) and NuExtract-1.5 (template-based) term extraction outperformed the traditional NER tools by a great extent in identifying the target terms exactly in this dataset. It advocates for utilizing guided or sophisticated methods for developing the initial domain glossary.

6.4 Error Analysis and Further Discussion

The mentioned results provide a quantitative comparison, but a more in-depth analysis is needed to understand the strengths, weaknesses and characteristic errors in each of the different approaches discussed. The following challenges can be named for baseline methods:

Low Precision. Many of the embedding-based methods, including those that achieve high recall (e.g., GloVe, Word2Vec Fine-tuned), suffer from low precision. This is because the detected synsets in these cases, are large and including tens if not hundred of terms.

Insufficiency of Small Domain Corpora. Fine-tuned models performed poorly, which might indicate that the provided domain corpus (3296 chunks, 200k words assuming 60 words/chunk) is small for optimal fine-tuning of these models.

BERT's untapped potential. The poor performance of BERT-based methods might suggest that using BERT embeddings in this way might not be the optimal way to use its power.

Chapter 7

Conclusion

In this master's thesis, a thorough comparison of different synonym extraction techniques in domain-specific settings was conducted. The need for precise synonyms to support advanced Natural Language Processing applications in a variety of industries, including technology, agriculture, automotive, economy, geography, law, and medicine, served as the driving force behind the study. Traditional methods, based on static and contextual word embeddings (both pre-trained and tailored to domain corpora) were investigated and a novel hybrid methodology was proposed that combines Large Language Model (LLM) term extraction and definition generation, Sentence-BERT embeddings, clustering techniques, and subsequent LLM-based synset refinement.

Synonym extraction process as a whole is greatly impacted by the first step of finding domain-specific terms for which synonyms are sought. The poor performance of conventional NER tools (NLTK, standard BERT-NER, and SpaCy) suggests that the target 'terms' in this dataset are frequently particular key phrases or ideas that are not adequately covered by general NER models. Guided extraction with NuExtract-1.5 showed great performance with a recall of 84%. High recall (up to 88%) of ground-truth target terms was shown by proposed LLM-based term extraction (using GPT-4o-mini).

For the second step of synset creation, traditional Lexical Resources (WordNet) produced low overall F1-scores (about 0.08), despite offering some exact synonyms. This was due to its limited coverage of domain-specific terms. Pre-trained Static Embeddings (fastText, GloVe, and Word2Vec) produced modest F1-scores when combined with nearest-neighbor search and mean-pooling for multi-word expressions; the best performing pre-trained fastText and pre-trained Word2Vec were in this category (F1 approximately 0.12).

Using corpus Adaptation of Static Embeddings to fine-tune pre-trained Word2Vec models produced degenerate behavior with nearly zero precision and incredibly high recall. Recall was raised by fine-tuning fastText word vectors using a domain

classification task, but the F1-score remained unchanged from its pre-trained counterpart. For nearest-neighbor synonym search using contextualized Word Embeddings (BERT), mean-pooled embeddings from pre-trained BERT or BERT fine-tuned on a domain classification task produced extremely low F1-scores (0.03), indicating that this is not the best way to take advantage of BERT's capabilities for this task.

The proposed combination of clustering and LLM-based refinement worked best for this purpose. Sentence-BERT's embedding of 'term: definition' strings gave semantic clustering a solid base. Compared to baselines, K-GMA and K-Means clustering algorithms on the definition-rich embeddings produced the best standalone clustering F1-scores (0.36). Using LLMs to refine the K-GMA clusters resulted in the biggest performance improvement. With the highest overall F1-score of 0.53, a combination of K-GMA clusters and LLM refinement via GPT-4.1-nano demonstrated strong performance across a number of technical domains (e.g., 0.63 in automotive, 0.64 in medical). This demonstrates how effective LLMs are at organizing and selecting semantic data. The process of refinement significantly improved precision with a slight decrease in recall.

All approaches were greatly outperformed by the innovative combination of LLM-generated definitions, Sentence-BERT embeddings, clustering, and LLM-refinement. This thesis advances the field of automatic synonym extraction in a number of ways, especially when it comes to domain-specific contexts:

- **Comprehensive Analysis**. Using a common dataset and evaluation framework, it offers a thorough comparative analysis of a broad range of synonym extraction techniques, from conventional methods to cutting-edge LLM-based approaches, evaluated methodically across multiple distinct domains. This provides insightful information about the relative advantages and disadvantages of these approaches for tasks specific to a given domain.

- **Novel Hybrid Methodology**. The suggested pipeline is a novel and efficient approach that combines (1) LLM-based term extraction and definition generation, (2) Sentence-BERT-based rich semantic embedding of these term-definition pairs, (3) unsupervised clustering of these embeddings, and (4) LLM-based refinement of the resulting clusters. For a challenging lexical semantic task, this synergy makes use of the advantages of various AI technologies.

- **Demonstrated Efficacy of LLM-based Refinement**. The significant performance improvement attained by refining machine-generated clusters using LLMs (GPT-4o-mini and GPT-4.1-nano) is a significant finding. This demonstrates a promising path for employing LLMs as powerful tools for organizing, verifying, and curating data produced by other AI systems in addition to serving as end-to-end solution providers.

- **Insights into Term Extraction for Specialized Vocabularies**. The assessment of different NER and LLM-based term extraction techniques clarifies their applicability for locating terminology unique to a given domain. It emphasizes that advanced LLM-based or template-guided extraction works better than general-purpose NER for non-standard, technical 'terms' or key phrases.
- **Empirical Evidence on Corpus Adaptation for Embeddings**. The effects of fine-tuning pre-trained embeddings versus training static embeddings from scratch on small domain corpora are demonstrated in the thesis. The results highlight the need for careful strategy when adapting embeddings to specialized domains, advising caution as naive application on limited data can result in suboptimal or even degenerate results.

7.1 Future Work

Based on the current research, a number of interesting directions exist for further investigation:

- **Expansion to More Domains**. The results would be further validated and new challenges would be revealed by applying and assessing the benchmarked methods and particularly the suggested hybrid approach, on a larger, more naturally occurring domain-specific corpora and a wider range of domains.
- **Downstream Task Evaluation**. An extrinsic measure of the extracted domain-specific synsets' quality would be provided by evaluating their usefulness in downstream NLP applications (such as domain-specific information retrieval, question answering, and machine translation).
- **Exploring Other Lexical Relations**. Building richer domain-specific ontologies would be facilitated by expanding the methodology to extract additional lexical-semantic relations that are pertinent to domain understanding, such as meronymy/holonymy (part-whole relationships) or hyponymy/hypernymy (is-a relationships).

In summary, this thesis has shown that although domain-specific synonym extraction is still a difficult task, it can be significantly improved over more conventional and straightforward embedding-based techniques by using a hybrid approach that carefully blends the advantages of LLM-generated semantic context, rich embeddings, unsupervised clustering, and LLM-based refinement. A clear picture of the current situation is given by the thorough analysis, and the suggested methodology presents a viable way forward for automatically creating useful lexical

resources for specialized domains. The ability of NLP systems to comprehend and process the complex language of expert fields will continue to be improved by additional research in the directions mentioned.

Appendix A

Appendix

A.1 Prompts

A.1.1 Delimiters

```
CONTEXT_BASE = dict(
    tuple_delimiter="<|>",
    record_delimiter="##",
    completion_delimiter="<|COMPLETE|>"
)
```

A.1.2 LLM-based term extractiono and definition generation prompt

```
'''
--Task--
Given a text and a domain, extract the terms that are strictly
   related to the specified domain.
The extracted terms will form a technical glossary related to the
   domain.
Important! Return only the specific terms and concepts *strictly*
   related to the domain.

--Rules--
1. Provide an output in Italian as a single list of all identified
    terms.
2. Use **{record_delimiter}** as the list separator.
3. When you're done, use {completion_delimiter}

Use the following output format:
(("term"{tuple_delimiter}"TERM1"{tuple_delimiter}"Description"){
   record_delimiter})
```

```
14
15 ######################
16 Example
17 Domain: Sport
18 Text:
19 In the final minutes of the match, the famous midfielder launched
      the leather ball with a through pass.
20
21 Output:
22 ("term"{tuple_delimiter}"MATCH"{tuple_delimiter}"Sports encounter
      between two teams with a competitive goal."){record_delimiter}
23 ("term"{tuple_delimiter}"MIDFIELDER"{tuple_delimiter}"Player who
      occupies the center of the field and is responsible for
      distributing the play."){record_delimiter}
24 ("term"{tuple_delimiter}"BALL"{tuple_delimiter}"Spherical object
      used in football."){record_delimiter}
25 ("term"{tuple_delimiter}"THROUGH_PASS"{tuple_delimiter}"Type of
      pass that bypasses the opposing defensive line to serve a
      teammate in a favorable position."){record_delimiter}
26 {completion_delimiter}
27
28 ######################
29 Domain: {domain}
30 Text: {input_text}
31
32 Output:
33 '''
```

A.1.3 LLM-based sysnset refinement prompt

```
1 '''
2 I will give you a synset, i.e., a group of synonymous terms, and a
      reference domain.
3
4 The synset can be of two types:
5 Type A. Uniform, where the terms are indeed all synonyms
6 Type B. Non-uniform, where the terms can be split into better
      synsets
7
8 Your task is:
9 1. Split Type B synsets into a list of Type A sub-synsets
10 2. Return the Type A synsets as they are
11
12 Note: The synsets will be used in a search engine based on Apache
      Lucene
13
14 Important! Never delete synsets!
15
```

```
16 Use the following output format:
17 {{'synset':["TERM1", "TERM2", "TERM3", ...],
18 'description':Description of the synset}}
19
20 #### Real Data ####
21 Domain: {domain}
22 Input synset: {synset}
23 '''
```

A.2 Tables

Configuration	F1-score	Precision	Recall	Num. Clusters
KMeans				
n_clusters=7	0.02	0.01	0.72	7
n_clusters=50	0.10	0.06	0.62	50
n_clusters=100	0.15	0.10	0.60	100
n_clusters=200	0.22	0.16	0.58	200
n_clusters=300	0.28	0.22	0.55	300
n_clusters=400	0.35	0.30	0.55	400
n_clusters=500	**0.36**	0.32	0.55	500
Agglomerative				
n_clusters=7	0.01	0.01	0.77	7
n_clusters=50	0.08	0.05	0.73	50
n_clusters=100	0.13	0.09	0.71	100
n_clusters=200	0.20	0.15	0.68	200
n_clusters=300	0.26	0.20	0.66	300
n_clusters=400	0.31	0.26	0.66	400
n_clusters=500	**0.35**	0.30	0.64	500
DBSCAN				
metric=euclidean, min_samples=2	0.00	0.00	0.79	1
metric=euclidean, min_samples=5	0.00	0.00	0.79	1
metric=cosine, min_samples=2	**0.03**	0.04	0.65	578
metric=cosine, min_samples=5	0.01	0.00	0.79	4
K-GMA				
resolution=0.2	**0.36**	0.31	0.59	414
resolution=0.4	**0.36**	0.32	0.59	418
resolution=0.5	0.17	0.12	0.65	100
resolution=0.6	0.16	0.11	0.65	87
resolution=0.7	0.17	0.12	0.64	96
resolution=0.8	0.16	0.11	0.62	98
resolution=1.0	0.13	0.09	0.64	71
resolution=1.2	0.15	0.10	0.63	83
resolution=1.4	0.16	0.11	0.63	92

Table A.1: F1-Score, Precision and Recall for Clustering Methods. The best F1-score in each family of methods is highlighted.

A.3 Implementation Details

Programming Language: Python (version 3.11.11).

Key Libraries and Versions

- **transformers**:
  - 4.50.3 (mainly used)
  - 4.47.1 (for NuExtract because of incompatibility with higher versions)
- **torch** (core deep learning library):
  - 2.6.0+cpu (mainly used)
  - 2.6.0+cu124 (for BERT fine-tuning and NuExtract on GPU using CUDA)
- **nltk**: 3.9.1 (WordNet, tokenization, stopword removal, NLTK-NER)
- **gensim**: 4.3.3 (Word2Vec, loading some embedding formats)
- **fasttext**: 0.9.3 (fastText model training and usage)
- **scikit-learn**: 1.6.1 (KMeans, AgglomerativeClustering, DBSCAN, evaluation metrics)
- **sentence-transformers**: 4.1.0 (Sentence-BERT for definition embeddings)
- **pandas**: 2.2.3 (data manipulation)
- **numpy**: 1.26.4 (numerical operations)
- **faiss-cpu**: 1.9.0 (efficient similarity search for embeddings)
- **spacy**: 3.8.4 (stemming, SpaCy-NER)
- **SpaCy model**: `en_core_web_sm` (version 3.8.0)
- **python-louvain**: 0.16 (used by the K-GMA clustering algorithm)

Specific Model Checkpoints/Files and Models

- **GloVe**: Pre-trained word vectors trained on Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors). Checkpoint name: `glove.42B.300d`
- **fastText (pretrained)**: 1 million word vectors trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset (16B tokens) were loaded using the fastText library. Checkpoint name: `wiki-news-300d-1M`
- **Word2Vec (pretrained)**: Pre-trained vectors trained on a part of the Google News dataset. Checkpoint name: `word2vec-google-news-300`
- **BERT (pretrained)**: `bert-base-uncased`
- **BERT-NER**: Standard fine-tuned BERT for NER. Model name: `dbmdz/bert-large-cased-finetuned-conll03-english`
- **Sentence-BERT**: `all-MiniLM-L6-v2`
- **NuExtract**: `numind/NuExtract-1.5-tiny`, `numind/NuExtract-1.5`
- **GPT Models**: `GPT-4o-mini`, `GPT-4.1-nano`

Code Structure

The project is organized into directories corresponding to different methods (e.g., `bert/`, `fasttext/`, `clustering/`, `word2vec/`) and utilities (`utils/`). Jupyter notebooks (`.ipynb` files) were used for experimentation and some data processing steps, while `.py` scripts handle more systematic training, extraction, and evaluation.

A.4 Hardware and Software Environment

Software

- **Operating System**:
  - Ubuntu 22.04.4 LTS (main)
  - Ubuntu 24.04.2 LTS (virtual machine used for GPU experiments)
- **Python**: Version 3.11.11
- **Conda**: Used for environment management

Hardware

- **CPU**: Most processing, especially for non-deep learning tasks and FAISS
- **GPU**: Used two T4 GPUs simultaneously for BERT fine-tuning and a P100 GPU for term extraction using NuExtract.

This comprehensive experimental setup in combination with the domain-specific dataset and an array of methods and tools allowed us to perform robust analysis of synonym extraction techniques.

Bibliography

[1] Sun Kim, Nicolas Fiorini, W John Wilbur, and Zhiyong Lu. «Bridging the gap: Incorporating a semantic similarity measure for effectively mapping PubMed queries to documents». In: *Journal of biomedical informatics* 75 (2017), pp. 122–127 (cit. on p. 1).

[2] Muhidin Mohamed and Mourad Oussalah. «SRL-ESA-TextSum: A text summarization approach based on semantic role labeling and explicit semantic analysis». In: *Information Processing & Management* 56.4 (2019), pp. 1356–1372 (cit. on p. 1).

[3] Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. «Bilingual word embeddings for phrase-based machine translation». In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1393–1398 (cit. on p. 1).

[4] Jiaming Shen, Ruiliang Lyu, Xiang Ren, Michelle Vanni, Brian Sadler, and Jiawei Han. «Mining entity synonyms with efficient neural set generation». In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 249–256 (cit. on pp. 1, 18).

[5] Yixuan Tang and Yi Yang. «Do We Need Domain-Specific Embedding Models? An Empirical Investigation». In: *arXiv preprint arXiv:2409.18511* (2024) (cit. on pp. 2, 14).

[6] Marie-Claude L'Homme. «Managing polysemy in terminological resources». In: *Terminology* 30.2 (2024), pp. 216–249 (cit. on p. 2).

[7] NuMind. *NuExtract-1.5 model on Hugging Face*. Available online at `https://huggingface.co/numind/NuExtract-1.5`. Accessed on 2025-4-24. URL: `https://huggingface.co/numind/NuExtract-1.5` (cit. on p. 3).

[8] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009 (cit. on p. 3).

[9] Matthew Honnibal and Ines Johnson. «spaCy: Industrial-strength Natural Language Processing in Python». In: *arXiv preprint arXiv:1712.07612* (2017) (cit. on p. 3).

[10] OpenAI. «GPT-4 Technical Report». In: *arXiv preprint arXiv:2303.08774* (2023). URL: https://arxiv.org/abs/2303.08774 (cit. on p. 3).

[11] George A Miller. «WordNet: a lexical database for English». In: *Communications of the ACM* 38.11 (1995), pp. 39–41 (cit. on pp. 3, 7).

[12] Jeffrey Pennington, Richard Socher, and Christopher 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 (cit. on pp. 3, 10).

[13] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. «Enriching word vectors with subword information». In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146 (cit. on pp. 3, 10).

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. «Efficient estimation of word representations in vector space». In: *arXiv preprint arXiv:1301.3781* (2013) (cit. on pp. 3, 10).

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186 (cit. on pp. 3, 12).

[16] Nils Reimers and Iryna Gurevych. «Sentence-bert: Sentence embeddings using siamese bert-networks». In: *arXiv preprint arXiv:1908.10084* (2019) (cit. on pp. 3, 13).

[17] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge university press, 2008 (cit. on p. 3).

[18] Willard Van Orman Quine. «Two dogmas of empiricism». In: *Perspectives in the Philosophy of Language* (2000), pp. 189–210 (cit. on p. 5).

[19] Leonard Linsky. *Semantics and the philosophy of language: a collection of readings*. Vol. 63. University of Illinois Press, 1952 (cit. on p. 5).

[20] John Firth. «A synopsis of linguistic theory, 1930-1955». In: *Studies in linguistic analysis* (1957), pp. 10–32 (cit. on p. 5).

[21] George A Miller and Walter G Charles. «Contextual correlates of semantic similarity». In: *Language and cognitive processes* 6.1 (1991), pp. 1–28 (cit. on p. 5).

[22] Eneko Agirre, Oier Lopez De Lacalle, Christiane Fellbaum, Shu-Kai Hsieh, Maurizio Tesconi, Monica Monachini, Piek Vossen, and Roxane Segers. «Semeval-2010 task 17: All-words word sense disambiguation on a specific domain». In: *Proceedings of the 5th international workshop on semantic evaluation*. 2010, pp. 75–80 (cit. on p. 6).

[23] Saif M Mohammad and Graeme Hirst. «Distributional measures of semantic distance: A survey». In: *arXiv preprint arXiv:1203.1858* (2012) (cit. on p. 6).

[24] Mohamed Ali Hadj Taieb, Torsten Zesch, and Mohamed Ben Aouicha. «A survey of semantic relatedness evaluation datasets and procedures». In: *Artificial Intelligence Review* 53.6 (2020), pp. 4407–4448 (cit. on p. 6).

[25] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. «Placing search in context: The concept revisited». In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 406–414 (cit. on p. 6).

[26] Felix Hill, Roi Reichart, and Anna Korhonen. «Simlex-999: Evaluating semantic models with (genuine) similarity estimation». In: *Computational Linguistics* 41.4 (2015), pp. 665–695 (cit. on p. 6).

[27] Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. «Simverb-3500: A large-scale evaluation set of verb similarity». In: *arXiv preprint arXiv:1608.00869* (2016) (cit. on p. 6).

[28] Martha Palmer, Daniel Gildea, and Paul Kingsbury. «The proposition bank: An annotated corpus of semantic roles». In: *Computational linguistics* 31.1 (2005), pp. 71–106 (cit. on p. 7).

[29] Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. «Background to Framenet». In: *International Journal of Lexicography* 16.3 (Sept. 2003), pp. 235–250. ISSN: 0950-3846. DOI: 10.1093/ijl/16.3.235. eprint: https://academic.oup.com/ijl/article-pdf/16/3/235/2761222/160235.pdf. URL: https://doi.org/10.1093/ijl/16.3.235 (cit. on p. 7).

[30] Philip Resnik. «Using information content to evaluate semantic similarity in a taxonomy». In: *arXiv preprint cmp-lg/9511007* (1995) (cit. on p. 7).

[31] Jay J Jiang and David W Conrath. «Semantic similarity based on corpus statistics and lexical taxonomy». In: *arXiv preprint cmp-lg/9709008* (1997) (cit. on p. 7).

[32] Dekang Lin et al. «An information-theoretic definition of similarity.» In: *Icml*. Vol. 98. 1998. 1998, pp. 296–304 (cit. on pp. 7, 18).

[33] Yuhua Li, Zuhair A Bandar, and David McLean. «An approach for measuring semantic similarity between words using multiple information sources». In: *IEEE Transactions on knowledge and data engineering* 15.4 (2003), pp. 871–882 (cit. on p. 7).

[34] Jian-Bo Gao, Bao-Wen Zhang, and Xiao-Hua Chen. «A WordNet-based semantic similarity measurement combining edge-counting and information content theory». In: *Engineering Applications of Artificial Intelligence* 39 (2015), pp. 80–88 (cit. on p. 7).

[35] David Sánchez, Montserrat Batet, and David Isern. «Ontology-based information content computation». In: *Knowledge-based systems* 24.2 (2011), pp. 297–303 (cit. on p. 7).

[36] David Sánchez, Montserrat Batet, David Isern, and Aida Valls. «Ontology-based semantic similarity: A new feature-based approach». In: *Expert systems with applications* 39.9 (2012), pp. 7718–7728 (cit. on p. 7).

[37] David Sánchez and Montserrat Batet. «A semantic similarity method based on information content exploiting multiple ontologies». In: *Expert Systems with Applications* 40.4 (2013), pp. 1393–1399 (cit. on p. 7).

[38] Yuncheng Jiang, Wen Bai, Xiaopei Zhang, and Jiaojiao Hu. «Wikipedia-based information content and semantic similarity computation». In: *Information Processing & Management* 53.1 (2017), pp. 248–265 (cit. on p. 7).

[39] Satanjeev Banerjee, Ted Pedersen, et al. «Extended gloss overlaps as a measure of semantic relatedness». In: *Ijcai*. Vol. 3. 2003, pp. 805–810 (cit. on p. 7).

[40] Michael Mc Hale. «A comparison of WordNet and Roget's taxonomy for measuring semantic similarity». In: *arXiv preprint cmp-lg/9809003* (1998) (cit. on p. 8).

[41] Donald AB Lindberg, Betsy L Humphreys, and Alexa T McCray. «The unified medical language system». In: *Yearbook of medical informatics* 2.01 (1993), pp. 41–51 (cit. on p. 8).

[42] Evgeniy Gabrilovich, Shaul Markovitch, et al. «Computing semantic relatedness using Wikipedia-based explicit semantic analysis.» In: *IJcAI*. Vol. 7. 1. 2007, pp. 1606–1611 (cit. on p. 8).

[43] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. «Dbpedia-a crystallization point for the web of data». In: *Journal of web semantics* 7.3 (2009), pp. 154–165 (cit. on p. 8).

[44] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. «YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia». In: *Artificial intelligence* 194 (2013), pp. 28–61 (cit. on p. 8).

[45] Ganggao Zhu and Carlos A Iglesias. «Computing semantic similarity of concepts in knowledge graphs». In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2016), pp. 72–85 (cit. on p. 8).

[46] Rada Mihalcea and Andras Csomai. «Wikify! Linking documents to encyclopedic knowledge». In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 2007, pp. 233–242 (cit. on p. 8).

[47] Roberto Navigli and Simone Paolo Ponzetto. «BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network». In: *Artificial intelligence* 193 (2012), pp. 217–250 (cit. on pp. 9, 18).

[48] Mohammad Taher Pilehvar and Roberto Navigli. «From senses to texts: An all-in-one graph-based approach for measuring semantic similarity». In: *Artificial Intelligence* 228 (2015), pp. 95–128 (cit. on p. 9).

[49] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. «PPDB: The paraphrase database». In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013, pp. 758–764 (cit. on p. 9).

[50] Thomas K Landauer and Susan T Dumais. «A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.» In: *Psychological review* 104.2 (1997), p. 211 (cit. on p. 10).

[51] Thomas K Landauer, Peter W Foltz, and Darrell Laham. «An introduction to latent semantic analysis». In: *Discourse processes* 25.2-3 (1998), pp. 259–284 (cit. on p. 10).

[52] Carlo Schwarz. «lsemantica: A command for text similarity based on latent semantic analysis». In: *The Stata Journal* 19.1 (2019), pp. 129–142 (cit. on p. 10).

[53] Kevin Lund and Curt Burgess. «Producing high-dimensional semantic spaces from lexical co-occurrence». In: *Behavior research methods, instruments, & computers* 28.2 (1996), pp. 203–208 (cit. on p. 10).

[54] James Gorman and James R Curran. «Scaling distributional similarity to large corpora». In: *Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the association for computational linguistics*. 2006, pp. 361–368 (cit. on p. 10).

[55] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. «Linguistic regularities in continuous space word representations». In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013, pp. 746–751 (cit. on p. 10).

[56] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. «Bag of tricks for efficient text classification». In: *arXiv preprint arXiv:1607.01759* (2016) (cit. on p. 10).

[57] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. «Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors». In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014, pp. 238–247 (cit. on p. 11).

[58] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. «Evaluation methods for unsupervised word embeddings». In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 298–307 (cit. on p. 11).

[59] Juan J Lastra-Dı'az, Josu Goikoetxea, Mohamed Ali Hadj Taieb, Ana Garcı'a-Serrano, Mohamed Ben Aouicha, and Eneko Agirre. «A reproducible survey on word embeddings and ontology-based methods for word similarity: Linear combinations outperform the state of the art». In: *Engineering Applications of Artificial Intelligence* 85 (2019), pp. 645–665 (cit. on p. 11).

[60] Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. «Dissecting contextual word embeddings: Architecture and representation». In: *arXiv preprint arXiv:1808.08949* (2018) (cit. on p. 11).

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 11).

[62] Yinhan Liu et al. «Roberta: A robustly optimized bert pretraining approach». In: *arXiv preprint arXiv:1907.11692* (2019) (cit. on p. 13).

[63] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. «Albert: A lite bert for self-supervised learning of language representations». In: *arXiv preprint arXiv:1909.11942* (2019) (cit. on p. 13).

[64] Iz Beltagy, Kyle Lo, and Arman Cohan. «SciBERT: A pretrained language model for scientific text». In: *arXiv preprint arXiv:1903.10676* (2019) (cit. on p. 13).

[65] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. «DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter». In: *arXiv preprint arXiv:1910.01108* (2019) (cit. on p. 13).

[66] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. «Tinybert: Distilling bert for natural language understanding». In: *arXiv preprint arXiv:1909.10351* (2019) (cit. on p. 13).

[67] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. «Ernie 2.0: A continual pre-training framework for language understanding». In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05. 2020, pp. 8968–8975 (cit. on p. 13).

[68] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. «Xlnet: Generalized autoregressive pretraining for language understanding». In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 13).

[69] Xuezhe Ma and Eduard Hovy. «End-to-end sequence labeling via bi-directional lstm-cnns-crf». In: *arXiv preprint arXiv:1603.01354* (2016) (cit. on p. 15).

[70] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. «Neural architectures for named entity recognition». In: *arXiv preprint arXiv:1603.01360* (2016) (cit. on p. 15).

[71] Zihan Liu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. «NER-BERT: a pre-trained model for low-resource entity tagging». In: *arXiv preprint arXiv:2112.00405* (2021) (cit. on p. 15).

[72] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. «Fast unfolding of communities in large networks». In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008 (cit. on p. 16).

[73] Luca Gioacchini. *kGMA - ML Toolbox*. `https://github.com/lucagioacchini/ml-toolbox/blob/master/mltoolbox/clustering/k_gma.py`. Accessed: 2025-4-24. 2024 (cit. on p. 16).

[74] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. «Improving language understanding by generative pre-training». In: (2018) (cit. on p. 16).

[75] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. «Language models are unsupervised multitask learners». In: *OpenAI blog* 1.8 (2019), p. 9 (cit. on p. 16).

[76] Tom Brown et al. «Language models are few-shot learners». In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901 (cit. on p. 16).

[77] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. «Exploring the limits of transfer learning with a unified text-to-text transformer». In: *Journal of machine learning research* 21.140 (2020), pp. 1–67 (cit. on p. 16).

[78] Aakanksha Chowdhery et al. «Palm: Scaling language modeling with pathways». In: *Journal of Machine Learning Research* 24.240 (2023), pp. 1–113 (cit. on p. 16).

[79] Hugo Touvron et al. «Llama: Open and efficient foundation language models». In: *arXiv preprint arXiv:2302.13971* (2023) (cit. on p. 16).

[80] Iryna Gurevych, Judith Eckle-Kohler, and Michael Matuschek. «Linked Lexical Knowledge Bases». In: *Linked Lexical Knowledge Bases: Foundations and Applications*. Springer, 2016, pp. 21–28 (cit. on p. 18).

[81] Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M Meyer, and Christian Wirth. «UBY-a large-scale unified lexical-semantic resource based on LMF». In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. 2012, pp. 580–590 (cit. on p. 18).

[82] Marti A Hearst. «Automatic acquisition of hyponyms from large text corpora». In: *COLING 1992 volume 2: The 14th international conference on computational linguistics*. 1992 (cit. on p. 18).

[83] Patrick Pantel and Dekang Lin. «Discovering word senses from text». In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 613–619 (cit. on p. 18).

[84] Beate Dorow and Dominic Widdows. «Discovering corpus-specific word senses». In: *10th Conference of the European Chapter of the Association for Computational Linguistics*. 2003 (cit. on p. 18).

[85] Jean Véronis. «Hyperlex: lexical cartography for information retrieval». In: *Computer Speech & Language* 18.3 (2004), pp. 223–252 (cit. on p. 18).

[86] Marija Brkić, Sanja Seljan, and Tomislav Vičić. «Automatic and human evaluation on English-Croatian legislative test set». In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer. 2013, pp. 311–317 (cit. on p. 18).

[87] Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. «Making sense of word embeddings». In: *arXiv preprint arXiv:1708.03390* (2017) (cit. on p. 18).

[88] Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone Paolo Ponzetto, and Chris Biemann. «Unsupervised does not mean uninterpretable: The case for word sense induction and disambiguation». In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 2017, pp. 86–98 (cit. on p. 18).

[89] Peter F Brown. «Class-based n-gram models of natural language». In: *Comput. Linguist.* 18 (1990), p. 18 (cit. on p. 18).

[90] Hugo Gonçalo Oliveira and Paulo Gomes. «ECO and Onto. PT: a flexible approach for creating a Portuguese wordnet automatically». In: *Language resources and evaluation* 48 (2014), pp. 373–393 (cit. on p. 18).

[91] Chris Biemann. «Chinese whispers-an efficient graph clustering algorithm and its application to natural language processing problems». In: *Proceedings of TextGraphs: the first workshop on graph based methods for natural language processing*. 2006, pp. 73–80 (cit. on p. 18).

[92] Vincent D Blondel, Anahı' Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. «A measure of similarity between graph vertices: Applications to synonym extraction and web searching». In: *SIAM review* 46.4 (2004), pp. 647–666 (cit. on p. 18).

[93] Kotaro Nakayama, Takahiro Hara, and Shojiro Nishio. «Wikipedia mining for an association web thesaurus construction». In: *International Conference on Web Information Systems Engineering*. Springer. 2007, pp. 322–334 (cit. on p. 18).

[94] Tong Wang and Graeme Hirst. «Extracting synonyms from dictionary definitions». In: *Proceedings of the International Conference RANLP-2009*. 2009, pp. 471–477 (cit. on p. 18).

[95] Einat Minkov and William Cohen. «Learning graph walk based similarity measures for parsed text». In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 2008, pp. 907–916 (cit. on p. 18).

[96] Aron Henriksson, Hans Moen, Maria Skeppstedt, Vidas Daudaravičius, and Martin Duneld. «Synonym extraction and abbreviation expansion with ensembles of semantic spaces». In: *Journal of biomedical semantics* 5 (2014), pp. 1–25 (cit. on p. 18).

[97] Aron Henriksson, Mike Conway, Martin Duneld, and Wendy W Chapman. «Identifying synonymy between SNOMED clinical terms of varying length using distributional analysis of electronic health records». In: *AMIA Annual Symposium Proceedings*. Vol. 2013. 2013, p. 600 (cit. on p. 18).

[98] Qing T Zeng, Doug Redd, Thomas Rindflesch, and Jonathan Nebeker. «Synonym, topic model and predicate-based query expansion for retrieving clinical documents». In: *AMIA Annual Symposium Proceedings*. Vol. 2012. 2012, p. 1050 (cit. on p. 18).

[99] Dhivya Chandrasekaran and Vijay Mago. «Evolution of semantic similarity—a survey». In: *Acm Computing Surveys (Csur)* 54.2 (2021), pp. 1–37 (cit. on p. 18).

[100] Chengyu Wang, Xiaofeng He, and Aoying Zhou. «A short survey on taxonomy learning from text corpora: Issues, resources and recent advances». In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 1190–1203 (cit. on p. 18).

[101] Kartik Detroja, CK Bhensdadia, and Brijesh S Bhatt. «A survey on relation extraction». In: *Intelligent Systems with Applications* 19 (2023), p. 200244 (cit. on p. 18).

[102] Dmitry Ustalov, Mikhail Chernoskutov, Chris Biemann, and Alexander Panchenko. «Fighting with the sparsity of synonymy dictionaries for automatic synset induction». In: *Analysis of Images, Social Networks and Texts: 6th International Conference, AIST 2017, Moscow, Russia, July 27–29, 2017, Revised Selected Papers 6*. Springer. 2018, pp. 94–105 (cit. on p. 18).

[103] Amir Hazem and Béatrice Daille. «Word embedding approach for synonym extraction of multi-word terms». In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018 (cit. on p. 18).

[104] Meng Qu, Xiang Ren, and Jiawei Han. «Automatic synonym discovery with knowledge bases». In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 997–1005 (cit. on p. 18).

[105] Li Zhang, Jun Li, and Chao Wang. «Automatic synonym extraction using Word2Vec and spectral clustering». In: *2017 36th Chinese Control Conference (CCC)*. IEEE. 2017, pp. 5629–5632 (cit. on p. 18).

[106] Dmitry Ustalov, Alexander Panchenko, and Chris Biemann. «Watset: Automatic induction of synsets from a graph of synonyms». In: *arXiv preprint arXiv:1704.07157* (2017) (cit. on p. 18).

[107] Kai Lei, Shangchun Si, Desi Wen, and Ying Shen. «An enhanced computational feature selection method for medical synonym identification via bilingualism and multi-corpus training». In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE. 2017, pp. 909–914 (cit. on p. 18).

[108] Zhen Wang, Xiang Yue, Soheil Moosavinasab, Yungui Huang, Simon Lin, and Huan Sun. «Surfcon: Synonym discovery on privacy-aware clinical data». In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 1578–1586 (cit. on p. 18).

[109] Jiaming Shen, Wenda Qiu, Jingbo Shang, Michelle Vanni, Xiang Ren, and Jiawei Han. «SynSetExpan: An iterative framework for joint entity set expansion and synonym discovery». In: *arXiv preprint arXiv:2009.13827* (2020) (cit. on p. 19).

[110] Hanqing Lu, Yunwen Xu, Qingyu Yin, Tianyu Cao, Boris Aleksandrovsky, Yiwei Song, Xianlong Fan, and Bing Yin. «Unsupervised synonym extraction for document enhancement in e-commerce search». In: (2021) (cit. on p. 19).

[111] Yiying Yang, Xi Yin, Haiqin Yang, Xingjian Fei, Hao Peng, Kaijie Zhou, Kunfeng Lai, and Jianping Shen. «KGSynNet: A novel entity synonyms discovery framework with knowledge graph». In: *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part I 26*. Springer. 2021, pp. 174–190 (cit. on p. 19).

[112] Chengyu Wang, Minghui Qiu, Jun Huang, and Xiaofeng He. «Keml: A knowledge-enriched meta-learning framework for lexical relation classification». In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 15. 2021, pp. 13924–13932 (cit. on p. 19).

[113] Song Zhang et al. «ProSyno: context-free prompt learning for synonym discovery». In: *Frontiers of Computer Science* 19.6 (2025), pp. 1–9 (cit. on p. 19).

[114] Subin Huang, Daoyu Li, Chengzhen Yu, Junjie Chen, Qing Zhou, and Sanmin Liu. «Empowering entity synonym set generation using flexible perceptual field and multi-layer contextual information». In: *PloS one* 20.4 (2025), e0321381 (cit. on p. 20).

[115] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. «Lost in the middle: How language models use long contexts». In: *arXiv preprint arXiv:2307.03172* (2023) (cit. on p. 28).

[116] Wikipedia contributors. *Glossary of geography terms*. `https://en.wikipedia.org/wiki/Glossary_of_geography_terms`. Accessed: 2025-3-1. 2025 (cit. on p. 30).

[117] U.S. Department of Justice. *Legal Terms Glossary*. `https://www.justice.gov/usao/justice-101/glossary`. Accessed: 2025-3-1. 2014 (cit. on p. 30).

[118] Harvard Medical School. *Medical Dictionary of Health Terms*. `https://www.health.harvard.edu/a-through-c`. Accessed: 2025-3-1. 2011 (cit. on p. 30).