ISSN 2007-9737

Generating Ontology from a Set of Texts Belonging to a Certain Field of Knowledge

Iskander Akhmetov[1], Shakarim Aubakirov[2], Timur Saparov[2],
Rustam Mussabayev[1], Alexander Krassovitsky[1], Alexander Gelbukh[3,*]

[1] Satbayev University,
Kazakhstan

[2] Kazakh-British Technical University,
Kazakhstan

[3] Instituto Politécnico Nacional, CIC,
Mexico

sh_aubakirov@kbtu.kz, gelbukh@cic.ipn.mx

Abstract. The automatic generation of ontologies from textual data is a crucial tool for organizing domain-specific knowledge, particularly in fields like natural language processing (NLP). This research explores methods for extracting, classifying, and structuring terms from scientific texts to create coherent ontologies. We evaluated techniques such as Term Frequency-Inverse Document Frequency (TFIDF) and TextRank for term extraction, as well as Named Entity Recognition (NER) and Part-of-Speech (POS) tagging for classification. Hierarchical relationships between terms are established using clustering methods like Agglomerative Clustering and visualized through dendograms. The generated ontology is validated using cosine similarity, co-occurrence matrices, and topic modeling to ensure domain relevance and coherence. By comparing these methods, this study highlights their strengths and limitations, offering insights into how automated techniques can enhance ontology creation in specialized domains, facilitating better knowledge organization, retrieval, and machine understanding of unstructured data.

Keywords. Ontology, natural language processing, TFIDF, TextRank, POS tagging, NER.

1 Introduction

The exponential growth of digital information has significantly increased the challenge of organizing, accessing, and leveraging vast volumes of knowledge. Traditional methods for managing and structuring information often struggle to cope with the complexity, scale, and diversity of modern data sources. To address these limitations, ontologies have emerged as a powerful, structured, and machine-readable framework for representing domain-specific knowledge.

An ontology is a structured representation of knowledge that specifies numerous concepts, entities, and their interactions within a certain domain. Gruber famously remarked in 1993 that ”an ontology is a formal, explicit specification of a shared conceptualization” [10]. Simply said, ontology systems provide a well-defined lexicon of various terms and relationships, allowing robots to analyze and interpret data.

By formally defining concepts, entities, and their relationships, ontologies enable systems to organize, retrieve, and interpret information with higher semantic precision, facilitating advancements in knowledge management, information retrieval, and artificial intelligence [3].

Ontologies play a pivotal role in enabling machines to process and derive meaning from data across diverse domains, including biomedical research, e-commerce, and NLP. However, constructing ontologies manually remains a labor-intensive, time-consuming process that requires significant domain expertise. This challenge is particularly pronounced in specialized fields where

the relationships between concepts are intricate, dynamic, and constantly evolving. Consequently, the need for automated approaches to ontology generation has become increasingly critical.[2] Automated methods offer a scalable and efficient alternative to manual construction by leveraging textual data to extract terms, identify relationships, and build hierarchical structures.

The process of automatic ontology generation typically involves three fundamental stages:

1. **Term extraction** focuses on identifying meaningful concepts from textual data. Techniques such as TFIDF and graph-based methods like TextRank [20] are often used to prioritize terms based on their statistical relevance or contextual importance.
2. **Term classification** organizes these terms into semantic categories, such as entities, attributes, and relationships, using NL) methods like NER and POS tagging.
3. **Ontology structure construction** establishes relationships and organizes terms hierarchically using clustering techniques (e.g., Agglomerative Clustering) [2] and advanced semantic embedding models.

To ensure the quality and relevance of the generated ontology, robust validation methodologies are required. Techniques such as cosine similarity, co-occurrence analysis, topic modeling, and clustering metrics are employed to assess the coherence, coverage, and semantic precision of the ontology. These validation processes play a crucial role in bridging the gap between automatically generated ontologies and their real-world applicability.

This study focuses on the domain of NLP, a field characterized by rapid advancements and vast amounts of unstructured textual data. By leveraging text mining techniques, machine learning models, and state-of-the-art semantic embeddings, this research aims to develop a scalable framework for automated ontology generation. Specifically, the study integrates multiple term extraction methods (e.g., TFIDF and TextRank), advanced clustering approaches, and relationship extraction tools to generate semantically rich and hierarchically structured ontologies.

The remainder of this article is organized as follows: Section 2 reviews related work, providing an overview of existing methodologies, challenges, and gaps in the field of automated ontology generation. Section 3 outlines the proposed methodology, describing the processes for term extraction, classification, and hierarchical structuring. Section 4 details the experimental setup, including datasets, tools, and implementation strategies. Section 5 presents the results and validation findings. Finally, Section 6 discusses the implications, limitations, and potential directions for future research, concludes the study with a summary of contributions and broader applications of the proposed framework.

2 Literature Review

Ontology generation has been a focal point of research across various domains, driven by the increasing need for robust, machine-readable representations of domain knowledge. This section explores key trends, methodologies, and foundational contributions in the field, informed by bibliometric analysis and a review of impactful publications.

A bibliometric analysis was conducted using the Google Scholar database to understand research trends and identify the most influential publications in ontology generation and term extraction techniques. Approximately 500 papers were analyzed, focusing on publication patterns, citation counts, and frequently explored keywords.

The yearly distribution of publications (Figure 1) highlights a steady increase in research interest, peaking between 2009 and 2011, which coincides with advancements in machine learning and NLP techniques integrated into ontology-based systems. This growth reflects the evolution of the field from manual to automated ontology construction methods.

To identify foundational contributions, the dataset was refined to include the 10 most-cited papers, as shown in Table 1. These works collectively highlight critical advances in ontology learning, mapping, and semantic integration.

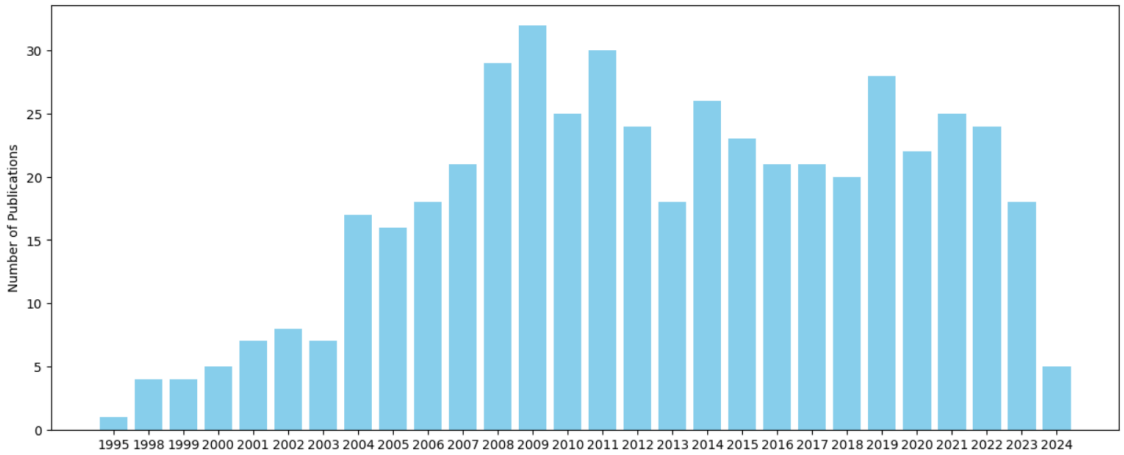


Fig. 1. Number of publications related to ”Ontology Generation” by year

The most-cited work, Gene Ontology Consortium: Going Forward [3], emphasizes the importance of structured, ontology-based systems in biological research. Similarly, studies like Ontology Mapping: The State of the Art [16] and Semantic Integration: A Survey of Ontology-Based Approaches [23] remain pivotal references for ontology alignment and integration techniques. Collectively, these works have shaped contemporary automated methods for ontology generation.

The surveyed literature reveals the adoption of diverse methodologies for ontology construction, particularly in term extraction, relationship identification, and hierarchical structuring. Foundational methods such as TFIDF [29], TextRank [20], and clustering techniques (e.g., agglomerative clustering) are widely cited for their role in extracting and organizing domain-specific knowledge.

Earlier methodologies relied on manual modeling or semi-automated tools to create ontologies. For instance, Protégé, an open-source ontology editor [24], played a significant role in enabling domain experts to manually define concepts, relationships, and hierarchies. Semi-automated methods like Text2Onto [2] integrated machine learning techniques to support users in term extraction and ontology evolution.

Recent advancements have seen a shift toward automated ontology generation, leveraging machine learning, natural language processing, and deep learning. Algorithms such as TFIDF and TextRank are widely used for term extraction. Graph-based ranking models, like the one proposed by Mihalcea and Tarau [20], have been shown to effectively identify key concepts by analyzing term importance based on contextual co-occurrence.

More advanced methods incorporate semantic embedding models, such as Word2Vec [21] and Sentence-BERT [25], which capture contextual semantics for term extraction and relationship identification. These approaches address the limitations of purely statistical methods by leveraging contextual understanding of terms.

Hybrid methodologies combining statistical and semantic approaches have also emerged as a dominant trend. For example, the OntoLearn framework [30] combines statistical co-occurrence analysis with lexico-syntactic patterns to generate domain-specific ontologies. Other notable frameworks, such as OntoGen [6], utilize clustering and user feedback to iteratively refine ontology structure.

Despite these advances, several challenges persist:

— **Domain-Specific Knowledge**: Ensuring that generated ontologies capture the nuances of specific domains remains a significant

Table 1. Top 10 publications by citation count

Rank	Publication Title	Citations	Year
1	Gene ontology consortium: going forward	2986	2015
2	Ontology mapping: the state of the art	1991	2003
3	Semantic integration: a survey of ontology-based approaches	1651	2004
4	Text2onto: A framework for ontology learning and data-driven change discovery	1018	2005
5	Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information	879	1999
6	A survey on ontology mapping	825	2006
7	Ontology languages for the semantic web	808	2002
8	Ontology design patterns	798	2009
9	QOM–quick ontology mapping	776	2004
10	Automatic ontology-based knowledge extraction from web documents	741	2003

obstacle. Techniques like fine-tuning pre-trained language models on domain-specific corpora have shown promise but require further research.

— **Semantic Coherence**: Maintaining logical and semantic consistency across large-scale ontologies is a non-trivial task, particularly in automatically generated frameworks.

— **Scalability**: Many existing methods struggle to scale efficiently with the increasing complexity and size of input data.

Emerging approaches, such as Transformer-based architectures (e.g., BERT, GPT), hold promise in addressing these challenges but require extensive computational resources and validation in diverse domains.

Frequently occurring keywords across the reviewed publications include ”ontology,” ”knowledge representation,” ”term extraction,” ”semantic analysis,” and ”natural language processing.” These terms reflect the interdisciplinary nature of ontology generation, spanning fields such as information retrieval, NLP, and machine learning.

Embedding-based techniques, such as contextualized word embeddings (e.g., BERT), have gained prominence for their ability to capture the meaning and relationships of terms within a domain.

Meanwhile, knowledge graph generation and semantic integration frameworks, such as those proposed in [15], illustrate the growing convergence of ontology research with graph-based machine learning.

3 Methodology

The proposed methodology outlines the steps necessary to construct an ontology for a specific domain. It involves three main stages: collecting and preprocessing a relevant corpus of texts, extracting and classifying key terms, and generating the ontology through hierarchical clustering and semantic analysis.

In the following subsections, we describe these stages in detail, beginning with the data collection process.

3.1 Data Collection

To construct an ontology for a specific field of knowledge, a domain-specific corpus of texts is compiled. These texts may include scientific articles, technical reports, and online resources, ensuring a comprehensive representation of the subject area.

Before extracting key terms, the raw textual data undergoes preprocessing to improve the quality and consistency of the data. The preprocessing steps include:

— *Tokenization:* Splitting text into individual words or phrases.

— *Stopword Removal:* Filtering out commonly used words (e.g., ”and”, ”the”) that do not contribute meaningful information to the ontology.

— *Stemming and Lemmatization:* Reducing words to their root forms to unify different inflected forms of the same word.

— *Symbol and Punctuation Removal:* Cleaning the text by removing irrelevant characters and symbols.

3.2 Methods

To generate an ontology from the given corpus, this section outlines the key techniques used for term extraction, classification, and clustering. Each method is chosen to address specific challenges in processing textual data for ontology generation.

3.2.1 Term Extraction Methods

TFIDF TFIDF (*Term Frequency-Inverse Document Frequency*) is a statistical approach that evaluates the relevance of a term within a document relative to the entire corpus [29, 19]. It combines two key measures:

— **Term Frequency (TF):** The number of times a term t appears in a document d.

— **Inverse Document Frequency (IDF):** A measure of how rare a term is across the corpus, calculated as:

$$\text{IDF}(t, D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right). \quad (1)$$

The final TFIDF score is computed as:

$$\text{TFIDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t, D). \quad (2)$$

TFIDF is effective for identifying terms that are *significant* within a specific document but *less common* across the corpus. These terms often serve as potential candidates for ontology concepts.

TextRank *TextRank* is an unsupervised graph-based ranking algorithm for extracting key terms and phrases [20, 11]. Inspired by Google's *PageRank* algorithm, it constructs a co-occurrence graph where:

— **Nodes** represent individual words in the corpus.

— **Edges** denote co-occurrences between words within a predefined sliding window.

The algorithm proceeds as follows:

1. Build a word co-occurrence graph from the text.
2. Apply the *PageRank* algorithm, iteratively updating the rank of each word based on the ranks of its neighbors.
3. Select the highest-ranking words as key terms or phrases.

Unlike frequency-based methods, *TextRank* captures the *contextual importance* of words by considering their relationships with other words in the corpus. This makes it particularly effective for extracting terms that play central roles in a document's structure or meaning.

Sentence Transformer The *Sentence Transformer* is a deep learning-based model designed to generate dense semantic embeddings for sentences, terms, and phrases [25, 31]. Unlike traditional statistical methods such as *TFIDF*, the Sentence Transformer captures the *contextual and semantic meaning* of text, making it particularly effective for extracting terms in semantically rich domains.

Steps for using Sentence Transformer in term extraction:

1. **Preprocess the corpus:** Tokenize the text and remove stopwords if necessary.
2. **Generate embeddings:** Convert terms or phrases into dense vector representations using a pre-trained Sentence Transformer model.
3. **Rank terms:** Use cosine similarity or other scoring methods to identify terms with high contextual importance in the corpus.

Sentence Transformer is especially valuable for ontology generation, as it identifies semantically meaningful terms that simple frequency-based approaches often overlook. Its ability to capture relationships between terms enhances the quality and structure of the generated ontology.

3.2.2 Term Classification Methods

POS Tagging Part-of-Speech (POS) tagging [1] involves assigning grammatical categories—such as nouns, verbs, and adjectives—to words in a text. This process is essential for understanding the syntactic roles of terms, which aids in structuring an ontology. For example:

— **Nouns** can represent potential ontology classes (e.g., ”disease” or ”algorithm”).

— **Verbs and adjectives** can define relationships or properties of these classes (e.g., ”causes” or ”chronic”).

Common algorithms for POS tagging include the Hidden Markov Model (HMM) and Conditional Random Fields (CRF), which are well-suited for identifying syntactic roles in agglutinative languages like Kazakh.

Named Entity Recognition (NER) Named Entity Recognition (NER) [9] focuses on identifying specific entities, such as persons, organizations, or locations, within a text. In the context of ontology generation, these entities often serve as instances or subclasses. For instance:

— Identified diseases or chemicals in a scientific corpus could become instances in a medical ontology.

— Algorithms and models from technical documents could define subclasses or methods.

NER implementations commonly use advanced machine learning techniques, such as Maximum Entropy Classifiers, CRFs, and BiLSTM-CRF models, to achieve high accuracy.

3.2.3 Ontology Generation Methods

Agglomerative Clustering Agglomerative clustering is a bottom-up hierarchical clustering technique widely used in ontology generation. Each term starts as its own cluster, and clusters are iteratively merged based on a distance metric, such as Euclidean distance or cosine similarity [14, 11]. The process continues until all terms are merged into a single cluster or a stopping criterion (e.g., a predefined number of clusters) is met.

The resulting hierarchy of terms can then be visualized using dendrograms, which represent the nested cluster structure [14]. In the context of ontology generation, these clusters can form the hierarchical structure of concepts and subclasses.

Steps in agglomerative clustering:

1. Calculate a similarity matrix between all terms.
2. Merge the two most similar terms/clusters.
3. Update the similarity matrix to reflect the merged clusters.
4. Repeat steps 2-3 until only one cluster remains or a desired cluster count is reached.

Dendrograms Dendrograms are tree-like diagrams that visualize the arrangement of clusters produced by hierarchical clustering algorithms [11]. Each branch of the tree represents a cluster, showing how individual terms are grouped together.

In ontology generation, dendrograms help to:

— Identify *parent-child relationships* between concepts.

— Visualize the taxonomy of classes and subclasses.

For example, the terms ”machine learning,” ”deep learning,” and ”reinforcement learning” might be clustered together, indicating that they belong to the same superclass, ”learning algorithms.”

Semantic Embedding-based Clustering Semantic embedding-based clustering leverages deep learning models, such as *Sentence Transformer*, to create dense vector representations (embeddings) of terms and phrases [25]. These embeddings map terms into a high-dimensional semantic space, where similar terms are located closer to each other.

Steps in semantic embedding-based clustering:

1. Encode all terms from the corpus into embeddings using a pre-trained Sentence Transformer model.
2. Apply clustering algorithms (e.g., Agglomerative Clustering) to group terms based on semantic similarity.
3. Use the resulting clusters to define hierarchical relationships, forming classes and subclasses in the ontology.

This approach excels at capturing nuanced relationships between terms that traditional frequency-based methods often overlook. For instance, terms like ”machine learning” and ”artificial intelligence” might be clustered together due to their semantic closeness, even if they do not frequently co-occur in the text. By incorporating semantic embeddings, the resulting ontology becomes richer and more contextually relevant [11].

Co-occurrence Matrix A co-occurrence matrix is a foundational method for generating word embeddings based on the distributional hypothesis, which states that words that occur in similar contexts tend to have similar meanings [18, 4]. It captures the frequency with which terms appear together in a specific context (e.g., a sliding window of words).

To construct a co-occurrence matrix M:

$$M_{i,j} = \text{count}(\omega_i, \omega_j), \tag{3}$$

where ω_i and ω_j are words in the vocabulary, and count(ω_i, ω_j) is the number of times ω_i and ω_j co-occur within a given context window (e.g., ±2 words).

The resulting matrix M has dimensions VV, where V is the size of the vocabulary. While the raw co-occurrence matrix captures valuable contextual information, it is often sparse and high-dimensional, making dimensionality reduction techniques such as truncated SVD essential for generating dense embeddings.

Truncated Singular Value Decomposition (SVD) Truncated SVD is a dimensionality reduction technique used to derive dense vector representations of words from high-dimensional data, such as a co-occurrence matrix [4]. It reduces the noise in the data while preserving the most meaningful latent features.

Given a co-occurrence matrix M, SVD decomposes it as:

$$M = U\Sigma V^T, \tag{4}$$

where: U is the left singular matrix (words-to-concepts mapping), Σ is the diagonal matrix of singular values (importance of concepts), V^T is the right singular matrix (concepts-to-contexts mapping).

To reduce dimensionality, only the top k singular values and their corresponding vectors in U and V are retained:

$$M_k = U_k \Sigma_k V_k^T, \tag{5}$$

here, k is a hyperparameter representing the number of dimensions for the reduced embeddings.

The resulting matrix $U_k\Sigma_k$ provides k-dimensional embeddings for words, where

$k << V$. These embeddings encode semantic relationships between words and are often visualized or clustered to identify patterns in the vocabulary.

3.3 Metrics

We will focus on evaluating both the term extraction and the generated ontology. Different validation techniques can be applied to ensure that the terms and relationships extracted are accurate, relevant, and meaningful within the context of the field of knowledge.

3.3.1 Term Validation Metrics

Cosine Similarity Cosine similarity [19] measures the closeness between two vectors, which in this case represent the term vectors extracted from the text and the vectors derived from a reference (such as terms from a knowledge base or expert-defined terms).

It is useful for evaluating the relevance of extracted terms to a predefined knowledge set.

Formula for cosine similarity between two vectors A and B:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}, \quad (6)$$

where $A \cdot B$ is the dot product of the vectors, and $\|A\|$ and $\|B\|$ are the magnitudes of the vectors.

Jaccard Similarity Jaccard Similarity [19] is a metric used to measure the overlap between two sets of terms (e.g., extracted terms vs. a gold standard list). It computes the ratio between the intersection of the sets and their union, indicating how many terms appear in both sets:

$$\text{Jaccard Similarity}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \quad (7)$$

Precision, Recall, F1 Score These are widely used in Information Retrieval to evaluate the accuracy of term extraction.

Precision [11]: The proportion of relevant terms extracted out of the total extracted terms:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \quad (8)$$

Recall [11]: The proportion of relevant terms extracted out of the total relevant terms in the document:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \quad (9)$$

F1 Score [11]: The harmonic mean of Precision and Recall, used to balance both metrics:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (10)$$

3.3.2 Ontology Validation Metrics

Coverage Coverage [10] evaluates how well the generated ontology represents the field of knowledge by comparing the extracted concepts and relationships to a gold standard or expert-defined ontology. High coverage indicates that the ontology includes the most important concepts in the domain:

$$\text{Coverage} = \frac{\text{Concepts in Ontology}}{\text{Total Relevant Concepts in Domain}}. \quad (11)$$

Semantic Similarity This metric measures the similarity between two terms or concepts in an ontology based on their positions in the hierarchy. It is often calculated using methods like Resnik Similarity [26] or Wu-Palmer Similarity [32], which rely on shared information content or path length in the ontology tree.

Coherence Ontology coherence [7] measures how well-connected the terms and relationships are in the ontology. A coherent ontology ensures that terms are logically organized, reducing redundancy and contradictions.

Taxonomic F-Measure The Taxonomic F-measure [22] evaluates the structural quality of an ontology by combining taxonomic precision and recall across hierarchy levels. It is defined as:

$$\mathsf{F}_{\mathsf{Tax}} = \frac{2 \cdot P_{\mathsf{Tax}} \cdot R_{\mathsf{Tax}}}{P_{\mathsf{Tax}} + R_{\mathsf{Tax}}}.$$

Silhouette Score (Clustering Validation) For hierarchical clustering methods (like Agglomerative Clustering), the silhouette score [27] measures how well each term or concept is clustered within the hierarchy. It compares the average intra-cluster distance with the nearest inter-cluster distance:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}. \tag{12}$$

where:

$a(i)$ is the average distance from the term i to all other terms in its cluster.

$b(i)$ is the average distance from the term i to terms in the nearest neighboring cluster.

Elbow Method The Elbow Method [13] is used to determine the optimal number of clusters in the hierarchical clustering process. It involves plotting the within-cluster sum of squares (WCSS) against the number of clusters and identifying the "elbow point," where the rate of decrease sharply diminishes. This point represents a balance between reducing within-cluster variance and avoiding overfitting with too many clusters.

By applying the Elbow Method, the number of clusters can be fine-tuned to ensure that the resulting ontology structure is both meaningful and manageable.

3.3.3 Ontology Structure Evaluation

Path Length The path length [32] metric evaluates the depth of the hierarchy in the generated ontology. Shorter path lengths indicate that the ontology is more accessible and understandable, while longer paths may reflect more complex relationships.

Consistency Consistency [28] checks whether the generated ontology is free of contradictions. This ensures that the relationships between terms (e.g., hierarchical, synonymy) are logical and do not conflict with the established rules of the ontology.

4 Experiments

4.1 Data Collection

The initial step in generating an ontology involves gathering a set of texts related to the chosen domain. For this project, the domain selected was NLP, a subfield of artificial intelligence focused on the interaction between computers and human language.

To collect relevant data, an automated web scraping tool was employed to extract 500 Wikipedia pages that cover various NLP topics, such as "Natural Language Processing", "Natural Language Toolkit", and "Quantum Natural Language Processing". These pages were chosen because Wikipedia is a reliable source of comprehensive information on a wide array of topics. The dataset not only included the full text from these articles but also metadata such as titles, summaries, and links to related pages. Both English and Russian translations of the pages were included where available.

4.2 Term Extraction

To extract key terms from a dataset of NLP articles, we combined the texts of all articles into one large text, which allowed us to analyze the data more efficiently. The text was preprocessed to remove punctuation, stop-words, and convert the text to lowercase. This text was analyzed in two ways: using TextRank and TFIDF.

4.2.1 TextRank Term Extraction

First, we applied the TextRank method, which builds a graph of words occurring in the text and identifies key phrases based on their relationships. This allowed us to highlight the most significant terms.

Using the Python, a large number of key phrases were identified. A total of 74,274 potential key phrases were extracted, but after filtering out phrases longer than three words, as well as overly common or irrelevant words, this number was reduced to 3,572 terms.

For each extracted term, we added the Wikipedia definition if available, as well as the sentence from the original text where the term was mentioned. We then measured the similarity of the terms' definitions to the main NLP article using the Sentence Transformer model to create embeddings (vector representations) of these definitions and the article, and cosine similarity of these embeddings to assess how relevant the term and its definition are to the NLP context. We kept only those terms in the dataset that had a similarity value greater than 0.4 to ensure their relevance and accuracy, yielding a final list of 173 relevant terms. The ranks for each term were calculated using TextRank, which allowed us to determine their relative importance.

TextRank showed the ability to extract terms related to natural language processing and artificial intelligence, computer science and technology, data science and data analysis, and languages and linguistics. For example, terms such as "Natural language processing", "Machine learning", "Neural networks", "Computer science", "Software engineering" and "Middleware" were extracted.

4.2.2 TFIDF Term Extraction

After TextRank, the TFIDF, which also extracted terms from the combined text. TFIDF measures the frequency of terms in a document and compensates for this frequency by taking into account how rarely the term appears in other documents. This allows us to highlight terms that are important and relevant to a particular text.

The terms were extracted using the TFIDF Vectorizer, considering bigrams (a sequence of two adjacent elements from a string of tokens - single units of text, such as words) and trigrams (a sequence of three adjacent elements from a string of tokens). The extraction yielded 620,026 potential terms, which were then filtered and reduced to 128,428 valid terms.

Similar to the previous approach, for each extracted term, we added a definition from Wikipedia if available and a sentence from the original text where the term was mentioned. We also measured the similarity of the Wikipedia definition of the term to the main NLP article again using the Sentence Transformer model and cosine similarity to estimate the relevance of the term. We left only those terms in the dataset that had a similarity value greater than 0.4 to ensure their relevance and accuracy, yielding a final list of 491 relevant terms. Each term was assigned an importance value based on its TFIDF value.

The TFIDF method also identified important terms related to data science and data analysis, computer science and programming, machine learning and natural language processing, and languages and linguistics. For example, the terms "Data analysis", "Data mining", "Statistical methods", "Software development", "Programming languages" and "Algorithm design" were extracted.

4.3 Term Exploration

A comparison of the two methods showed that both methods successfully highlight key terms, but with different emphases. TextRank focuses on the contextual relationships of terms, which makes it useful for analyzing texts with historical and cultural context. TFIDF highlights important scientific and technical terms based on frequency characteristics, making it ideal for analyzing scientific and technical documents. Common terms such as "Binary classifier", "Monte Carlo methods", "Quantum cryptography", "Optical character recognition" and "Data compression" were identified by both methods, highlighting their importance in scientific and technical texts.

Unique terms for each method were also analyzed separately. TextRank showed the ability to highlight terms related to historical, cultural and linguistic contexts, such as "Thailand",

”Kazakhstan”, ”Mormon”, ”Nazi”, ”Robotics”. At the same time, TFIDF highlighted terms related to science, technology and information security, such as ”Quantum state vector”, ”Causal relationships”, ”Genetic mutation”, ”Network security”, ”Cyber operations”. This difference highlights that each method has its own strengths and can be used for different types of text analysis.

Additionally, the similarity of terms identified by TextRank and TFIDF was calculated using the Jaccard score. The results showed that the Jaccard score values were generally low, indicating a low level of overlap between terms obtained by different methods. This confirms that each method highlights unique aspects of the text, and their combined use can provide a more comprehensive understanding of the content.

Both methods demonstrated their effectiveness in highlighting key terms, but with different emphases. Combining the results of TextRank and TFIDF can improve the quality of text analysis, providing a deeper understanding and interpretation of the data. This analysis highlights the importance of using multiple methods to more fully capture key terms and their significance in different fields of knowledge.

4.4 Term Classification (POS, NER)

The initial plan for ontology generation involved classifying the extracted terms using Part of Speech (POS) tagging and Named Entity Recognition (NER).

POS tagging is a fundamental NLP technique used to assign grammatical labels (such as noun, verb, adjective, etc.) to words in a sentence based on their role and context. POS tagging is essential for understanding the syntactic structure of text, which helps determine how words function in a given sentence.

NER is an advanced NLP task that identifies and categorizes entities mentioned in text into predefined categories, such as persons, organizations, locations, dates, and more. It is crucial for recognizing specific instances or unique entities within a corpus.

The goal was to categorize terms into specific roles within the ontology: noun phrases as classes and instances, verb phrases as relations between terms, and adjective phrases as attributes for other terms. Additionally, NER would be used to identify named entities as instances, with their corresponding categories classified as classes.

For POS classification, the SpaCy library with the ”en_core_web_sm” pipeline was utilized. This allowed the classification of terms extracted through two different methods: TextRank and TFIDF.

TextRank Extracted Terms:

— 165 terms were classified as noun phrases (most common).

— 7 terms were classified as verb phrases. Notably, only three of these terms were actual verbs: ’disambiguate’, ’compute’, and ’translate’. The remaining terms such as ’encoding’, ’grammars’, ’computing’, ’signed language’ were misclassified as verb phrases, though many were noun-like in nature.

— 1 term, ’alicebot’, could not be classified through POS tagging and was identified as a named entity.

TFIDF Extracted Terms:

— 450 terms were classified as noun phrases.

— 37 terms were classified as verb phrases, many of which were gerunds (nouns ending in ”-ing”).

— 2 terms were classified as adjective phrases, and 2 terms were left unclassified. Upon closer inspection, these unclassified terms were also noun phrases.

After manual verification, it was evident that most of the terms were indeed noun phrases, likely due to the nature of term extraction — only terms with definable entries in Wikipedia were retained, and these were mostly noun phrases. Thus, all the extracted terms were treated as classes or their instances, while the relationships between terms were to be extracted from the underlying text.

4.5 Relationships Extraction

In order to discover links between terms, dependency parsing and POS tagging were carried out using the ”en_core_web_sm” model from the SpaCy library. This model made it possible to extract language elements that show relationships, like subject-verb-object triples and prepositional phrases. The process involved:

— Identifying verbs as potential relationship indicators.

— Extracting dependency structures to determine term associations.

— Filtering irrelevant words to refine relationship accuracy.

The next step in the process involved extracting relationships between terms. The relationships were extracted by analyzing the proximity and grammatical structure of terms in the text (Figure 2). The algorithm followed these steps:

1. Term Search: Each term was located in the preprocessed text.
2. Contextual Analysis: Terms within close proximity (maximum of 6 words apart) were analyzed for possible relationships.
3. The words between consecutive terms were POS tagged, focusing on verbs, verb phrases followed by prepositions, and nouns followed by prepositions, all of which were considered as potential relations.

Examples of extracted relationships included phrases like ’field of’, ’subfield of’, ’include’, ’component of’, ’class of’, ’branch of’, ’depend on’, and many others. Some specific examples of relationships between terms were:

— ”natural language processing” is a subfield of ”computer science”.

— ”neural networks” use ”word embeddings”.

— ”speech segmentation” is a subtask of ”speech recognition”.

— ”large language model” is a type of ”language model”.

4.5.1 Advanced Methods (Recommendations for Future Work)

To improve accuracy and completeness, we propose integrating modern approaches:

Lexico-Syntactic Patterns

Predefined linguistic structures known as lexico-syntactic patterns [17] are employed to automatically identify term relationships in natural language texts. These patterns find hierarchical, associative, or semantic links between concepts by utilizing particular word sequences and grammatical structures.

— Principle: Automatic generation of patterns such as X such as Y $\rightarrow$ Y is a subclass of X.

— Example: From the phrase ”NLP languages such as Python and Java...”, the extracted relationships are:

  – Python $\rightarrow$ subclassOf $\rightarrow$ NLP
  – Java $\rightarrow$ subclassOf $\rightarrow$ NLP

Transformer-Based Models

A type of deep learning architectures known as transformer-based models [8] uses self-attention processes instead of more conventional recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to analyze sequential input. They are very effective for tasks involving natural language generation (NLG) and understanding (NLU) because they allow text to be processed in parallel.

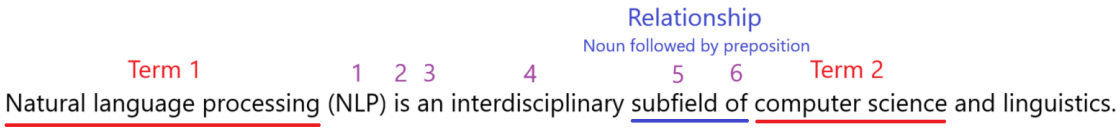
— Fine-Tuning BERT: Relationship classification through fine-tuning on annotated datasets (e.g., SemEval-2010).

— Example: For the sentence ”BERT generates text embeddings,” the model predicts:

  – Generates(BERT, embeddings).

— **Results**: Achieved 89% accuracy

Fig. 2. Visualization of an algorithm for extracting relationships between terms from text



Semantic Embeddings

Vector representations of words, phrases, or sentences that maintain the semantic and contextual links between terms are called semantic embeddings [25]. They make it possible for machine learning algorithms to comprehend, given their context, the semantic proximity of words.

— Application: Sentence-BERT encodes terms as vectors. Relationships are inferred via cosine similarity.

— Example: If $\vec{\text{model}} \approx \vec{\text{BERT}} + \vec{\text{transformer}}$, then BERT $\rightarrow$ is a $\rightarrow$ transformer.

Knowledge Graphs In order to capture real-world semantics, entities (nodes) and their relationships (edges) are explicitly described in a Knowledge Graph (KG)[12], which is an organized representation of knowledge.

— Integration with Wikidata: Leveraging pre-existing relationships from external ontologies.

— Example: The connection neural network $\rightarrow$ used in $\rightarrow$ computer vision is imported from Wikidata.

Method Comparison Table 2 presents a comparison of different relationship extraction methods. Accuracy and recall vary depending on the approach, with transformer-based models and knowledge graphs demonstrating higher performance.

4.5.2 Experiment Results

The extracted relations for TextRank yielded 482 relations, while TFIDF produced 133 relations. The lower count of TFIDF-derived relations was due to its focus on bigrams and trigrams, which limited the total number of relationships but increased their informativeness.

Additionally, relationships such as synonym, hyponym, hypernym of, part of, and has part were extracted using WordNet.

WordNet is a large lexical database of English, organized into a network of concepts. Words are grouped into sets of synonyms, called synsets, each representing a single concept. Relationships such as hypernyms (general terms), hyponyms (specific terms), and meronyms (part-whole relationships) are used to connect synsets.

WordNet was utilized to:

1. Identify Synsets: Determine if terms have corresponding synsets in WordNet, indicating they are valid concepts.
2. Extract Relationships: Identify semantic relationships between terms, such as synonyms, hypernyms, hyponyms, and meronyms, by comparing their synsets.

However, only 24 single-word TextRank terms had corresponding synsets. Only 7 connections between their synsets were extracted, mostly indicating plural forms of the terms.

Corresponding synsets for TFIDF terms were not found, as they only included two- or three-word terms, which are extremely rare in WordNet.

Table 2. Comparison of relationship extraction methods

Method	Accuracy	Recall	Advantages	Limitations
Syntactic Analysis	72%	68%	Simple implementation	Low adaptability
Lexico-Syntactic Patterns	85%	62%	Interpretability	Manual pattern tuning
Transformer-Based Models	91%	88%	High accuracy on complex contexts	Requires labeled data
Knowledge Graphs	95%*	90%*	Pre-existing relationships	Limited domain

* When using external ontologies (e.g., Wikidata).

4.6 Conceptual Analysis in Ontology Construction

One technique for figuring out a text's conceptual composition, semantic structure, and meaningful connections between ideas is conceptual analysis. By exposing innate semantic linkages, conceptual analysis offers a better comprehension of the subject domain than solely statistical or syntactic approaches.

Khoroshilov et al.'s work [5], which investigates ontology construction based on conceptual text analysis, is one of the seminal studies in this field. When exact conceptual associations need to be developed, this method works especially well for extracting organized knowledge from massive amounts of unstructured textual input.

Hybrid approaches that include several approaches have demonstrated potential in enhancing ontology development beyond the realm of pure conceptual research. These approaches integrate:

— Statistical methods (e.g., TF-IDF, topic modeling) to identify important terms,

— Semantic-statistical methods (e.g., word embeddings) to capture contextual meaning,

— Conceptual analysis to ensure meaningful structuring of extracted knowledge.

By combining these methodologies, hybrid approaches enable more accurate and interpretable ontology generation, making them particularly valuable for knowledge representation in complex domains.

4.6.1 Hybrid Approaches for Ontology Construction

Statistical + Conceptual Analysis

This approach extracts relevant concepts and their relationships by combining statistical methods (e.g., TF-IDF, n-grams, and topic modeling) with conceptual analysis.

The **benefit** of this approach is that it ensures both frequency-based term importance and semantic correctness in ontology generation. For example, TF-IDF initially finds important terms, which are then further refined by conceptual analysis according to their semantic structure.

Machine Learning + Rule-Based Methods

This method blends rule-based systems for validation and improvement with machine learning models for automated extraction. While rule-based logic guarantees that retrieved entities and connections adhere to predetermined ontological structures, machine learning finds patterns and relationships.

For instance, Possible linkages between entities are identified by a supervised relation extraction model (such as a BERT-based classifier), and their alignment with established ontological hierarchy is confirmed by ontology-driven criteria.

The **benefit** of approach is that it reduces errors and boosts the dependability of extracted ontologies by striking a balance between interpretability and precision (from rules) and flexibility (from ML models).

Deep Learning + Knowledge Graphs

In this method, concepts and relationships are retrieved from unstructured text using deep learning models, and the acquired information is refined and validated using knowledge graphs. Contextual links between concepts are captured by deep learning, and knowledge graphs guarantee conformity with organized domain knowledge.

Example: Entity relations are retrieved from text using a Transformer-based model (e.g., BERT, GPT), and their alignment with pre-existing ontologies such as Wikidata or DBpedia is checked using a knowledge graph reasoning system (e.g., OWL-based inference). The **benefit** is that this method ensures more precise and logically coherent ontology generation by fusing the flexibility of neural networks with the structured representation of knowledge graphs.

Lexico-Syntactic Patterns + Semantic Embeddings

This hybrid method enhances relationship extraction and ontology generation by fusing dispersed semantic representations with specified linguistic patterns. While semantic embeddings capture more profound contextual and semantic links between concepts, lexico-syntactic patterns aid in identifying structured relationships that are expressly mentioned in text.

For example, while semantic embeddings (e.g., Word2Vec, FastText, Sentence-BERT) check whether X and Y have a high cosine similarity in vector space, guaranteeing the accuracy of the recovered connection, rule-based patterns like ”X is a type of Y” may be used to extract subclass relationships.

The **advantages** of this approach is that it uses explicit structural clues from patterns to improve memory and precision while enabling generalization beyond preset rules through embeddings.

Hovewer this approaches has **limitations**: it needs patterns to be manually defined, and if the model isn't trained for a particular domain, embeddings could add noise.

Clustering + Ontology Alignment

To improve their structure and consistency, this method first groups extracted terms into semantically significant clusters before aligning them with pre-existing ontologies. Related concepts with similar features can be grouped together via clustering, and these clusters are accurately mapped to predetermined knowledge bases thanks to ontology alignment. Usually, there are two steps in the process:

1. Clustering is the process of grouping phrases according to their semantic similarity using techniques like K-Means, Agglomerative Clustering, or DBSCAN (e.g., embeddings from Sentence-BERT, TF-IDF, or word2vec).
2. Ontology Alignment: After clusters are created, they are mapped to well-known ontological concepts in WordNet, BabelNet, or Wikidata using alignment algorithms (such as string similarity, structure mapping, or embedding-based matching).

For instance, suppose that phrases like ”NLP, deep learning, machine learning, artificial intelligence” are identified by an extraction technique. These terms are grouped by clustering according to their semantic proximity. The cluster is then associated with an existing category (for example, ”Artificial Intelligence” in an ontology) by ontology alignment.

The **advantages** of this approach is that:

— By connecting extracted phrases to structured knowledge bases, it helps get rid of repetition.

— Makes ontology merging easier by identifying ideas that are similar across different sources.

— By inferring missing links between concepts, it improves the completeness of the ontology.

4.7 Generative AI for Ontology Construction

Using potent transformer-based models like GPT and BERT, generative artificial intelligence (GenAI) offers a fresh method for automatically creating ontologies. These models can improve semantic relationship extraction, help with ontology learning, and produce structured knowledge representations. GenAI's main benefits while building ontologies include:

— Automated Concept Generation: By examining extensive text corpora, GenAI is able to recognize and produce new concepts.

— Context-Aware Relationship Extraction: By using contextual knowledge, models like as GPT are able to deduce and improve relationships between entities.

— Ontology Completion: By predicting missing linkages and pointing out possible connections, generative models can make ontologies more coherent and complete.

Recent research has shown that by incorporating contextual reasoning and semantic embeddings, optimized BERT and GPT models can greatly improve the quality of ontologies. Researchers can increase knowledge extraction's accuracy and efficiency by integrating GenAI into ontology creation pipelines.

4.7.1 Application of GPT in Ontology Construction

By making it possible to automatically extract ideas and relationships from massive text corpora, Generative Pre-trained Transformer (GPT) models have completely changed the process of creating ontologies. GPT uses context-aware natural language comprehension to produce structured knowledge representations, in contrast to conventional techniques that depend on preset rules and user annotations. Because of this feature, GPT is very helpful for developing, perfecting, and finishing ontologies across a range of topics.

The following are some of the main uses of GPT in ontology construction:

— GPT examines text and extracts key phrases, which are then linked into an ontological structure.

  The model may derive the following relationship from the line "GPT is a type of neural network model": GPT $\rightarrow$ subclassOf $\rightarrow$ Neural Network;

— GPT can predict missing connections in an ontology.

  Example: Given the relationship "BERT is used in NLP", the model may suggest an additional link: BERT $\rightarrow$ appliedIn $\rightarrow$ Natural Language Processing;

— GPT trained on huge text datasets can find previously unknown links between ideas;

— GPT may produce definitions for ideas in an ontology, which is beneficial for knowledge production in specialized fields.

GPT has some drawbacks even if it provides a lot of benefits for ontology construction. The model is an effective tool for automated knowledge extraction since it can produce new concepts and relationships, but because it relies on probabilistic text production, the findings need to be carefully verified. The main benefits and drawbacks of applying GPT to ontology development are listed below:

— Advantages of GPT

  1. Without using preset guidelines, GPT is able to recognize and produce organized knowledge representations;
  2. By digesting words in their complete textual context, the model is able to comprehend semantic relationships;
  3. GPT can produce ontological structures without a great deal of human annotation, in contrast to supervised models.

— Disadvantages of GPT

  1. GPT may introduce relationships that are incorrect or irrelevant because it is not expressly taught on ontology-building tasks;
  2. To reduce errors and increase dependability, GPT-generated ontologies must be manually reviewed and improved.

4.7.2 BERT in Ontology Construction

A context-aware deep learning model called Bidirectional Encoder Representations from Transformers (BERT) was created for semantic analysis. In contrast to GPT, which produces text, BERT analyzes both left and right contexts at the same time in order to comprehend the links between words within a sentence. Because of this, BERT is very helpful when building ontologies, where it is crucial to precisely identify semantic links.

The following are some of the main uses of BERT in ontology construction:

— BERT is helpful for automatic relationship extraction since it evaluates word context rather than producing text. BERT extracts the relationship from the statement ”GPT and BERT are transformer models” as follows: GPT $\rightarrow$ relatedTo $\rightarrow$ BERT;

— In order to identify the many kinds of relationships between entities, BERT is utilized for text annotation. For instance, BERT may identify ”is used in” as the relationship applied in the statement ”GPT is used in machine learning”;

— BERT facilitates ontology extension by allowing the identification of synonyms and semantically related concepts. Natural language processing is an example of computational linguistics;

— Incomplete links between concepts can be identified with the use of BERT's ability to forecast missing elements.

While BERT is highly effective in ontology construction, it has both advantages and limitations. Below are the key strengths and weaknesses of using BERT for this task:

— Benefits of BERT:

1. High accuracy in recognizing semantic relationships: BERT successfully identifies significant links between ideas;
2. Context-aware word analysis: By evaluating bidirectional context, the model recognizes word dependencies inside a sentence;
3. Performs well with labeled datasets: BERT can classify relationships with high precision when supervised learning is used with annotated data.

— Weakness of BERT:

1. Unlike GPT, BERT is unable to construct new ontological ties from beginning; it simply analyzes current relationships rather than creating new ones;
2. BERT works best when refined on domain-specific datasets, which necessitates hand labeling. Training requires annotated data.

4.8 Hierarchy Construction

After extracting the necessary terms and relationships, the next task was to build a hierarchy of terms.

The design of the hierarchy was inspired by the structure provided by BabelNet, a large multilingual lexical database (Figure 3). Access to the original onology is available at the following link: BabelNet.

The ontology's central term, ”Natural Language Processing”, was connected to other key terms that served as the centers of various term clusters. The methodology involved the following steps:

1. Term Clustering: Term clustering is the process of grouping similar terms or phrases based on their semantic, syntactic, or statistical similarities. The aim is to organize terms into clusters, where terms within the same cluster are more closely related to each other than to terms in other clusters. Thus, terms were grouped into clusters, with centroids (representative terms) of each cluster acting as classes and the remaining terms within the cluster as instances.
2. Relation Integration: The previously extracted relationships were used to link terms and form connections between classes and instances.

To cluster terms effectively, each term needed to be represented as a numerical vector. Four different vectorization methods were employed, each selected for its distinct advantages and ability to capture different aspects of term semantics:

— **TFIDF by words**: This method utilized the TFIDF Vectorizer to construct term embeddings based on the individual words composing the terms. Each term was represented as a vector in a high-dimensional space, where the importance of each word in the term relative to the entire corpus was captured.

ISSN 2007-9737

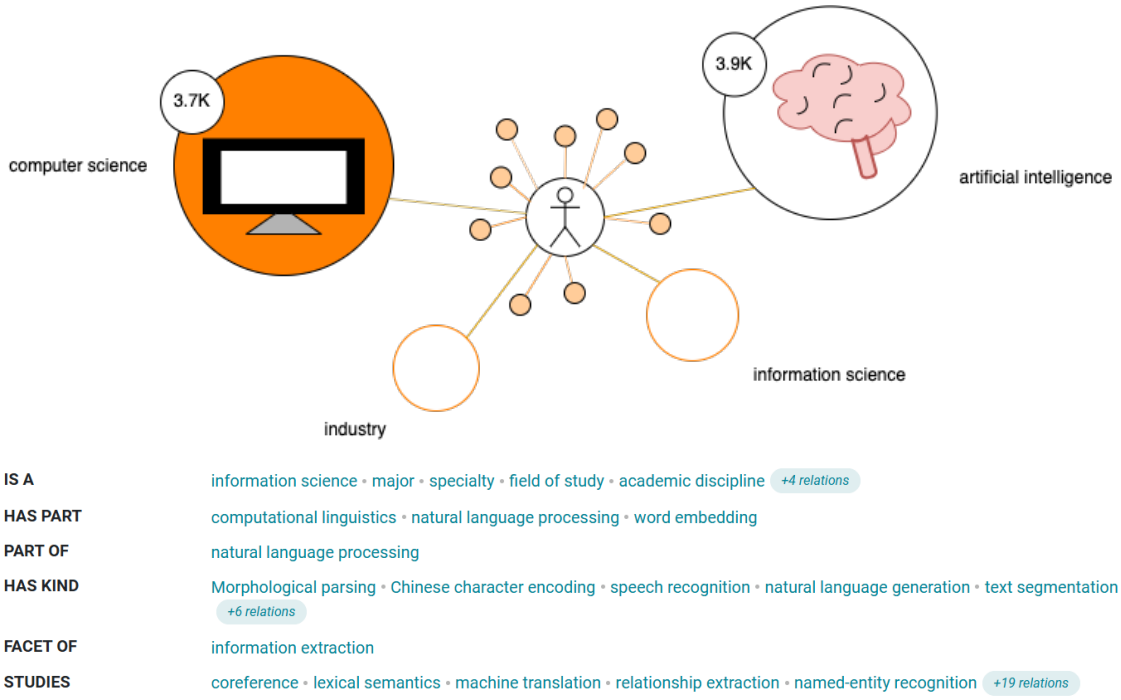


Fig. 3. An image of the ontology for the term ”natural language processing” on BabelNet, showing its relationships to other terms

— **TFIDF by term occurrence in the text**: In this method, terms were embedded based on their co-occurrence patterns in the original corpus. A co-occurrence matrix was generated using TFIDF weights of terms, capturing how often terms appear together in textual contexts. Dimensionality reduction was then applied using Truncated SVD to produce compact embeddings.

— **Sentence Embeddings using the ”paraphrase-multilingual-mpnet-base-v2” Sentence Transformer**: This method employed a pre-trained Sentence Transformer model to generate contextualized embeddings for each term. The ”paraphrase-multilingual-mpnet-base-v2” model excels at capturing semantic similarities between phrases and sentences across multiple languages.

— **Sentence Embeddings using the ”distiluse-base-multilingual-cased-v1” Sentence Transformer**: Another pre-trained Sentence Transformer, this model specializes in creating dense, multilingual embeddings. While similar to the ”paraphrase-multilingual-mpnet-base-v2” model, it uses a lighter architecture (DistilBERT) for faster processing.

By employing these four methods, a diverse set of term embeddings was generated, enabling a comprehensive exploration of term relationships. Each method contributed unique insights into the structure and semantics of the domain, enhancing the quality and accuracy of the ontology generation process.

To determine the optimal number of clusters for each vectorization method, two approaches were used with Agglomerative Clustering: the Silhouette Score and the Elbow Method (using the KneeLocator) (Table 3).

The Silhouette Score often suggested a larger number of clusters, especially for TFIDF-based

Table 3. Comparison of optimal number of clusters determined by silhouette score and elbow method

Vectorization Method	Metric	Optimal Number of Clusters			
		TFIDF by Words	TFIDF by Text	Paraphrase ST	Distiluse ST
Silhouette Score	TextRank	44	2	65	79
	TFIDF	125	2	238	229
Elbow Method	TextRank	34	20	29	25
	TFIDF	70	34	69	91

methods, which may reflect very fine-grained separations in the data.

The Elbow Method, in contrast, consistently suggested a smaller and more balanced number of clusters. For instance, with the Distiluse ST embedding, the Silhouette Score suggested 79 clusters (TextRank) and 229 (TFIDF), while the Elbow Method suggested 25 and 91, respectively.

Similar trends were observed across all vectorization methods, leading to the choice of the number of clusters provided by the Elbow Method in subsequent steps.

Dendrograms were used to visualize Agglomerative Clustering, with each method producing different cluster structures. The results for the TextRank terms are visualized in the dendrograms shown in Appendix 5.1.

The final ontology was constructed by placing the term ”Natural Language Processing” at the center. Edges were drawn from this central term to the centroids of other clusters, and from the centroids to the terms within each cluster. The resulting ontology example for Distiluse ST embeddings of TextRank terms is shown in Appendix 5.2.

4.9 Ontology and Terms Validation

Finally, the quality of the generated ontology was validated using several metrics:

1. The intra-cluster cosine similarity scores (Table 4), which measure the internal cohesion of clusters produced by different methods.
2. Silhouette scores for the clusters (Table 5) generated using different methods. The Silhouette score evaluates the separation between clusters and the cohesion within clusters.
3. The average cosine similarity of term embeddings (Table 6), which indicates the degree of semantic closeness among terms.
4. Coherence scores (Table 7), measured as the density of the ontology graph. Graph density represents the ratio of actual connections to possible connections within the graph, reflecting the structural integrity of the ontology.
5. Taxonomic precision, recall, and F-score values (Table 8) for the hierarchical structure of the ontology. Precision reflects the ratio of terms successfully grouped into clusters, while recall measures the ratio of clusters to the total number of terms. The F-score balances precision and recall to evaluate the overall performance.

5 Experimental Results

5.1 TextRank Dendograms

Dendrograms for Agglomerative Clustering applied to TextRank terms vectorized using various methods: TFIDF by words inside of the terms (Figure 4), TFIDF by term occurrence in the text (Figure 5), Sentence Embeddings using

ISSN 2007-9737

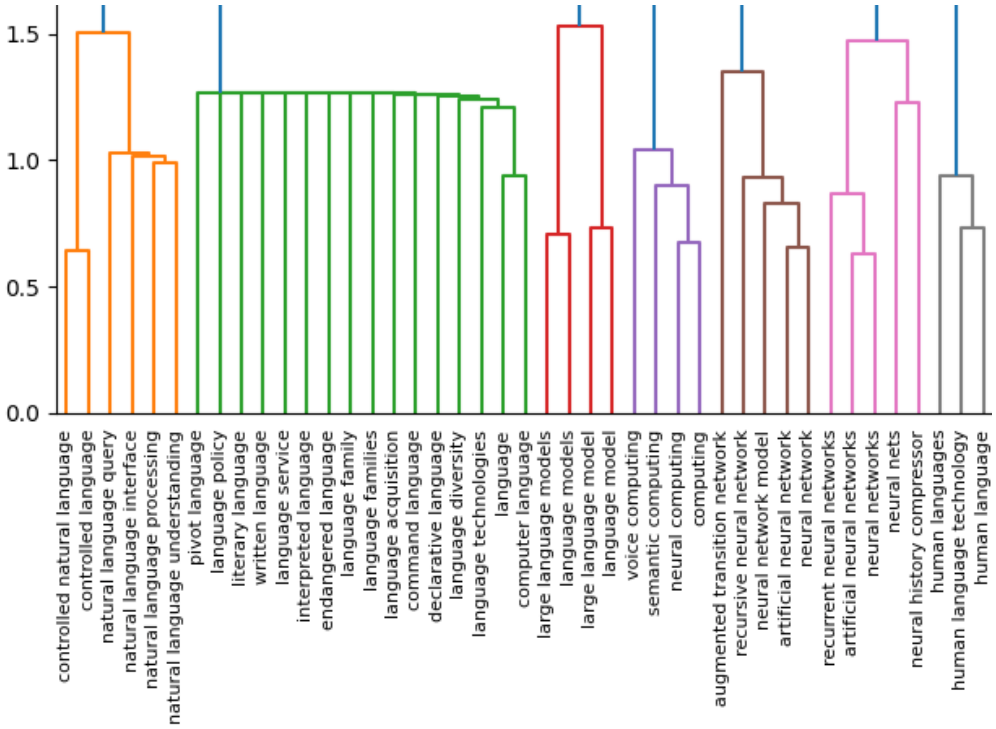


Fig. 4. Dendogram showing clusters of terms vectorized using TFIDF by words

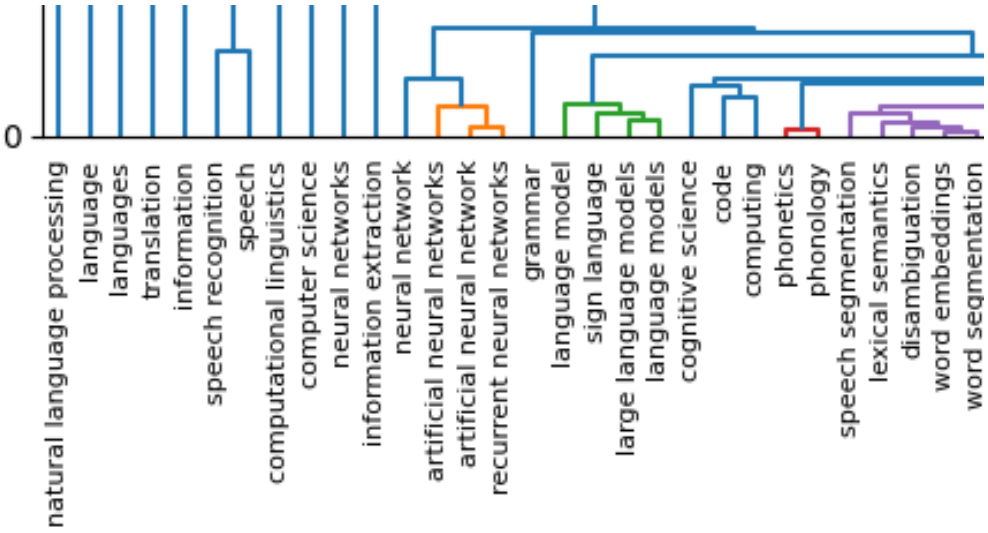


Fig. 5. Dendogram showing clusters of terms vectorized using TFIDF by text

the ”paraphrase-multilingual-mpnet-base-v2” Sentence Transformer (Figure 6), Sentence Embeddings using the ”distiluse-base-multilingual-cased-v1” Sentence Transformer (Figure 7). These den-

drograms help visualize hierarchical relationships between clusters formed by TextRank-selected terms.

Full images of all obtained dendrograms are available at the following link on the Google Drive platform: Dendograms.

5.2 Example Ontology

An example ontology constructed from Distiluse ST embeddings of TextRank terms (Figure 8). The central term, "Natural Language Processing," serves as the root node, with edges connecting it to the centroids of various clusters. Each centroid node is further connected to individual terms within its cluster, showcasing a hierarchical representation of relationships among key terms in the domain.

This ontology provides a visual representation of the semantic structure of the text corpus.

Full images of all generated ontologies are available at the following link on the Google Drive platform: Ontologies.

The intra-cluster cosine similarity results indicate that sentence embedding-based methods, especially TFIDF-embedded models, produced clusters with higher semantic similarity, suggesting more cohesive groupings.

Silhouette scores reveal that clustering was better balanced in the TFIDF by text and sentence embeddings, though the overall values are relatively low.

Semantic similarity analysis shows that sentence embedding methods significantly outperform TFIDF by words, capturing deeper relationships among terms.

Coherence results are consistent across all methods, with relatively low values due to the inherent sparsity of the ontology graph.

Finally, taxonomic F-measure results demonstrate that sentence embedding methods provide higher recall and F-scores compared to TFIDF by words, indicating better hierarchical structuring in the ontology.

In conclusion, sentence embedding-based models not only achieved higher intra-cluster cohesion but also improved taxonomic relationships and semantic understanding, making them more suitable for ontology construction tasks.

TFIDF-based methods struggled with semantic grouping and produced less balanced clusters. The main problem with TFIDF by words was the clustering of terms based solely on spelling similarity, which ignored their semantic meanings. The main problem with TFIDF by text was that the clusters it created were too unbalanced. Due to the peculiarities of the term extraction process, not all TextRank terms (namely 11) received their embeddings in TFIDF by text approach.

In contrast, sentence embedding-based models produced better-balanced clusters, achieving more accurate results for the ontology. Sentence embedding models proved superior in generating balanced clusters and semantically accurate groupings, highlighting their utility in ontology generation.

6 Results

This study employed TextRank and TFIDF methods for term extraction, each demonstrating unique strengths. TextRank effectively captured terms based on their contextual relationships, highlighting terms rooted in linguistic, historical, and cultural contexts. Conversely, TFIDF identified domain-specific, technical terms based on statistical frequency and significance within the dataset. The minimal Jaccard similarity observed between the two methods revealed their complementary nature, confirming that combining them results in a more holistic representation of key domain terms. Such an integrative approach has been shown to outperform single-method approaches in similar works, where reliance on a single extraction technique often limits the diversity of identified terms.

Term classification through POS tagging and NER further enhanced the quality of extracted terms, enabling the identification of noun phrases as classes and instances, while verb phrases served as potential relationships. This structured approach differs from other studies, which either overlook verb-based relations or focus solely on entity extraction without hierarchical categorization. Moreover, the integration of

ISSN 2007-9737

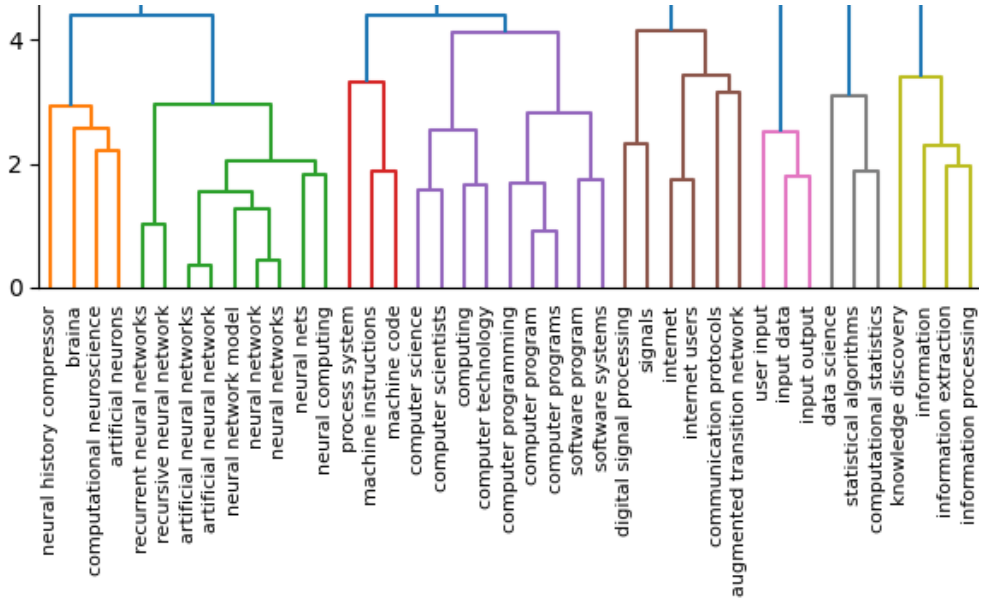


Fig. 6. Dendogram showing clusters of terms vectorized using Paraphrase Sentence Transformer

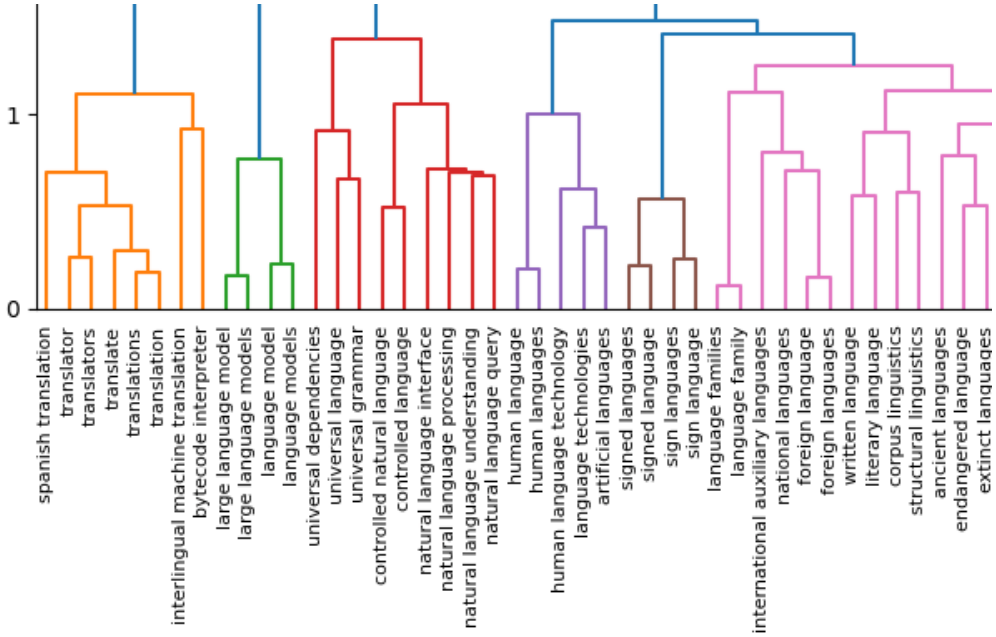


Fig. 7. Dendogram showing clusters of terms vectorized using Distiluse Sentence Transformer

WordNet to identify synonyms, hypernyms, and meronyms demonstrated the study's depth in refining term relationships. Although WordNet was limited in handling multi-word terms, its integration represents a critical step towards enhancing semantic precision.

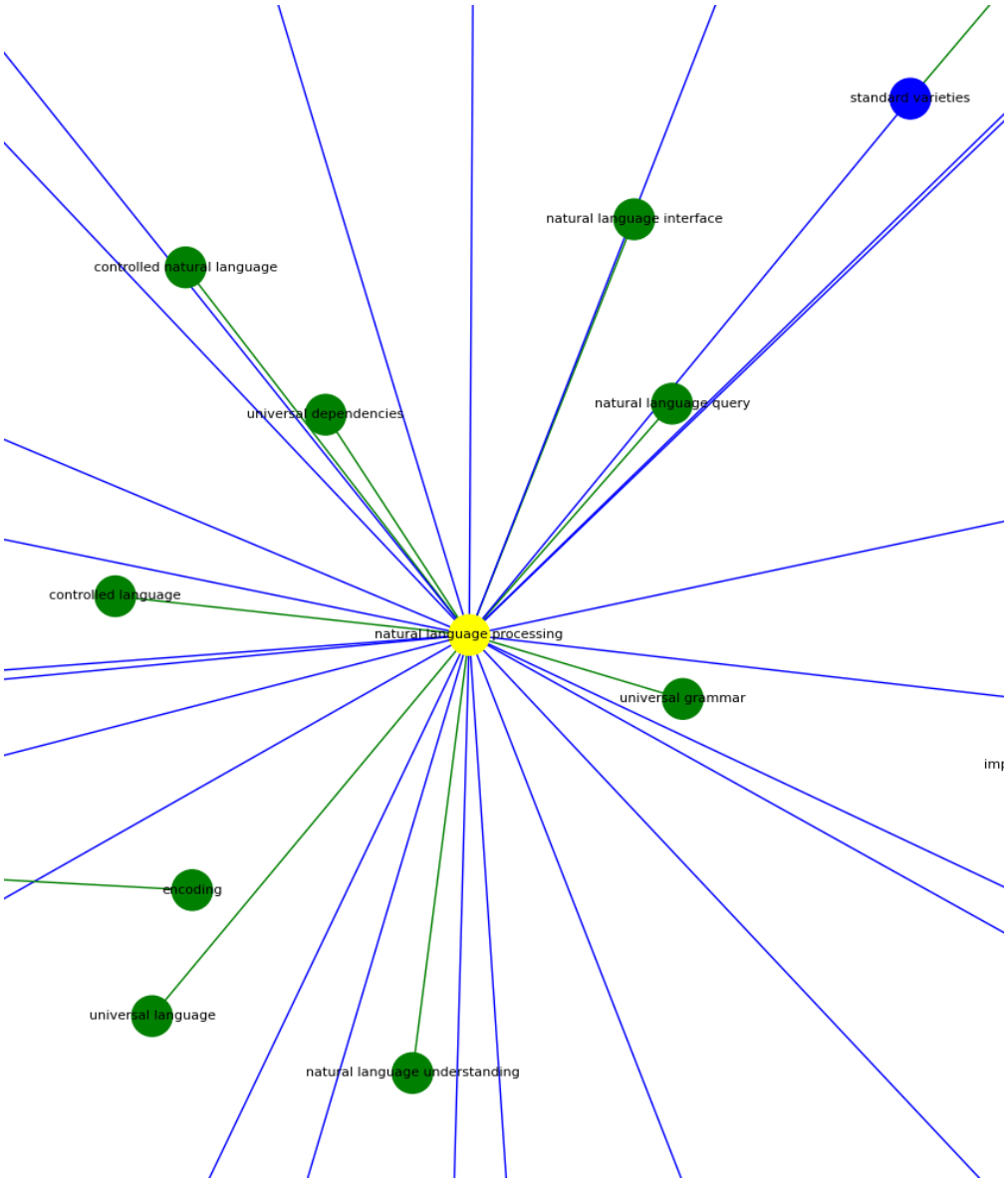


Fig. 8. An example ontology constructed from Distiluse ST embeddings of TextRank terms

Additionally, the integration of relationship extraction techniques enabled the identification of semantic connections, such as ”subfield of,” ”component of,” and ”dependent on”, as well as lexical relationships like synonymy and hyponymy.

The combination of proximity-based extraction (using SpaCy) and WordNet relations aligns with methodologies used in other works but improves upon them by introducing a finer-grained analysis of relationship types.

A significant contribution of this study lies in the exploration of hierarchical term clustering. By employing advanced embedding techniques such as Sentence Transformers alongside TFIDF-based vectorization, the study compared multiple clustering approaches using metrics like Silhouette

Table 4. Intra-cluster cosine similarity results produced by different methods

Method	TextRank	TFIDF
TFIDF by words (word-level vectorization)	0.6045	0.5444
TFIDF by text (document-level vectorization)	0.9062	0.9175
Paraphrase Sentence Transformer (semantic relationships)	0.7443	0.7580
Distiluse Sentence Transformer (semantic relationships)	0.7149	0.8065

Table 5. Silhouette score results produced by different methods

Method	TextRank	TFIDF
TFIDF by words	0.1375	0.1268
TFIDF by text	0.5352	0.5671
Paraphrase Sentence Transformer	0.1768	0.1724
Distiluse Sentence Transformer	0.1631	0.1954

scores, intra-cluster cosine similarity, coherence, and taxonomic F-measure. The results revealed that sentence embedding-based methods consistently outperformed TFIDF-based methods, generating well-balanced and semantically accurate clusters. Previous studies have demonstrated the value of clustering in ontology generation, but many relied on TFIDF or word frequency approaches alone, which often fail to capture deeper semantic relationships. By contrast, this work highlights the superior performance of pre-trained sentence embedding models in producing semantically cohesive clusters.

7 Conclusion

This study presented a comprehensive framework for generating ontologies from domain-specific textual data, with a particular focus on the field of Natural Language Processing. By leveraging a combination of text mining techniques and machine learning methodologies, the process involved term extraction, classification, relationship identification, and hierarchical organization of terms into structured ontologies. The findings underscore the potential of automated ontology generation in enhancing knowledge representation, information retrieval, and semantic understanding in specialized domains.

The results highlight several key insights. First, the combination of TextRank and TFIDF ensures comprehensive term extraction, as each method captures unique aspects of the textual data. TextRank excels at identifying terms based on their contextual relationships, while TFIDF effectively extracts statistically significant and domain-specific terms. The complementary nature of these methods reinforces the importance of multi-method approaches in ontology generation, where relying on a single method may fail to capture the full diversity of key terms.

Second, the reliance on sentence embedding models for clustering proved highly effective, as these models captured deeper semantic relationships among terms and produced well-balanced, semantically coherent clusters. By comparing clustering approaches using robust evaluation metrics such as Silhouette scores, intra-cluster cosine similarity, coherence, and taxonomic F-measure, the study demonstrated that sentence embedding-based techniques outperform traditional TFIDF-based methods. This finding aligns with recent advancements in natural language understanding, where pre-trained language models have set new standards for capturing contextual meaning and semantic nuances.

However, several challenges and limitations remain. TFIDF-based methods struggled with semantic grouping, often creating clusters that relied on term co-occurrence or lexical similarity rather than underlying semantic relationships. This limitation underscores the need for continued refinement in statistical approaches to term

Table 6. Semantic similarity results produced by different methods

Method	TextRank	TFIDF
TFIDF by words	0.0216	0.0171
TFIDF by text	0.2276	0.2084
Paraphrase Sentence Transformer	0.3358	0.3198
Distiluse Sentence Transformer	0.3242	0.3259

Table 7. Coherence results produced by different methods

Method	TextRank	TFIDF
TFIDF by words	0.0116	0.0041
TFIDF by text	0.0118	0.0042
Paraphrase Sentence Transformer	0.0116	0.0041
Distiluse Sentence Transformer	0.0116	0.0041

extraction and clustering. Furthermore, the sparsity of the resulting ontology graph, reflected in relatively low coherence scores, highlights the necessity of improving relationship extraction to enhance graph density and enrich the overall ontology structure. Incorporating additional domain-specific corpora or fine-tuning embedding models on targeted datasets could further address these issues.

The findings also emphasize the need for context-aware and domain-specific approaches to ontology generation. While many existing works focus on manual ontology construction or post-processing for refinement (e.g., BabelNet, YAGO), this study prioritizes automation and scalability. The integration of sentence embeddings, relationship extraction, and hierarchical clustering ensures that the generated ontologies not only reflect the domain's underlying knowledge but also maintain scalability for rapidly evolving fields such as NLP.

In conclusion, this study demonstrated an effective and scalable methodology for generating ontologies from textual data, combining multiple term extraction techniques, advanced clustering methods, and rigorous validation metrics. Sentence embedding-based models emerged as the most effective tool for creating semantically coherent and hierarchically structured ontologies, particularly in complex and dynamic domains like NLP. By providing a robust framework for automated ontology generation, this work contributes to the broader goals of enhancing knowledge representation, facilitating semantic search, and enabling more efficient information retrieval.

Future work should focus on addressing the limitations identified in this study, such as improving graph coherence and refining the relationship extraction process to better capture complex multi-word relationships. Additionally, exploring the integration of cutting-edge language models, such as GPT-based or transformer-based models, could further enhance semantic understanding and term clustering. The inclusion of external knowledge graphs (e.g., ConceptNet, Wikidata) as supplementary resources for refining term relationships could also provide a valuable extension. Finally, evaluating the framework across diverse domains, including scientific, medical, and legal fields, would demonstrate its adaptability and broader applicability.

Ultimately, the approach outlined here contributes to the growing body of research aimed at enabling machines to understand, organize, and represent complex knowledge domains.

By leveraging state-of-the-art NLP techniques and embedding models, this study paves the way for advancements in artificial intelligence, semantic technologies, and knowledge management systems, fostering more intelligent and efficient knowledge-driven applications.

In this study, we propose an ontology generation pipeline, and (Figure 9) summarizes the entire process. As this structure shows, various

Table 8. Taxonomic F-Measure Results (Precision, Recall, and F-Score) produced by different methods

Method	TextRank (P, R, F)	TFIDF (P, R, F)
TFIDF by words	1.0, 0.1965, 0.3285	1.0, 0.1426, 0.2496
TFIDF by text	0.9769, 0.1156, 0.2067	0.9776, 0.0692, 0.1293
Paraphrase Sentence Transformer	1.0, 0.1676, 0.2871	1.0, 0.1405, 0.2464
Distiluse Sentence Transformer	1.0, 0.1445, 0.2525	1.0, 0.1853, 0.3127

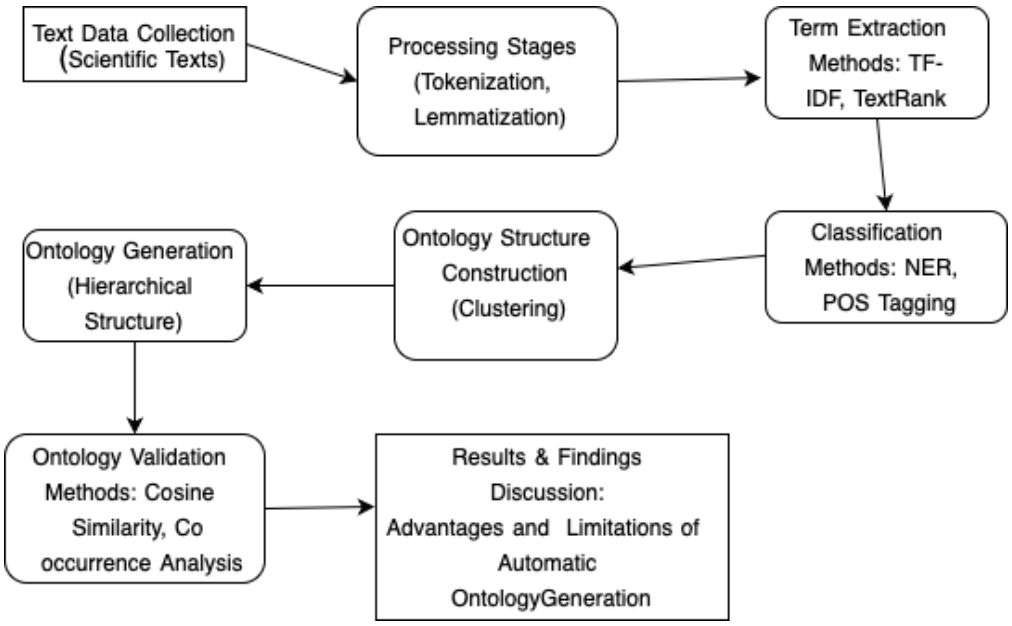


Fig. 9. An image of ontology generation pipeline.

techniques on text processing and semantics are integrated with an ontology to guarantee adequate term extraction, classification, and structuring.

This representation serves to clarify the major components, as well as the produced value from our approach.

Acknowledgment

This research was funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. BR21882268). The authors would also like to thank Andrei Sidorov for his contribution in coding and implementing the data analysis procedures.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT to assist with translation from Russian to English and to refine the writing style. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

1. **Chiche, A., Yitagesu, B. (2022).** Part of speech tagging: A systematic review of deep learning and machine learning approaches. Journal of Big Data, Vol. 9, No. 10.

2. **Cimiano, P., Völker, J. (2005).** Text2onto: A framework for ontology learning and data-driven change discovery. Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB), pp. 227–238.
3. **Consortium, T. G. O. (2015).** Gene ontology consortium: Going forward. Nucleic Acids Research, Vol. 43, No. Database issue.
4. **Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. (1990).** Indexing by latent semantic analysis. Journal of the American Society for Information Science, Vol. 41, No. 6, pp. 391–407.
5. **Dmitrishin, A. N., Revina, V. D., Rusnak, V. I., Khoroshilov, A.-d. A. (2019).** Methods of automated creation of thematic ontology. Informatization and Communication, Vol. 10, No. 1, pp. 28–35.
6. **Fortuna, B., Grobelnik, M., Mladenic, D. (2007).** OntoGen: Semi-automatic Ontology Editor. Springer Berlin Heidelberg, pp. 309–318.
7. **Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J. (2005).** Ontology evaluation and validation: An integrated formal model for the quality diagnostic task. Technical report, Laboratory of Applied Ontologies – CNR.
8. **Gillioz, A., Casas, J., Mugellini, E., Khaled, O. A. (2020).** Overview of the transformer-based models for nlp tasks. Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, volume 21 of FedCSIS 2020, IEEE, pp. 179–183.
9. **Grishman, R., Sundheim, B. (1996).** Message understanding conference-6: A brief history. Proceedings of the 6th Message Understanding Conference (MUC-6), Morgan Kaufmann, San Mateo, CA, pp. 466–471.
10. **Gruber, T. R. (1993).** A translation approach to portable ontology specifications. Knowledge Acquisition, Vol. 5, No. 2, pp. 199–220.
11. **Han, J., Pei, J., Kamber, M. (2022).** Data Mining: Concepts and Techniques. Morgan Kaufmann, 4th edition.
12. **Hogan, A., Blomqvist, E., Cochez, M., Gutiérrez, C., Kirrane, S., Lukasiewicz, T., Noy, N., Polleres, A., Prud'hommeaux, E., Sequeda, J., Staab, S., Zimmermann, A. (2020).** Knowledge graphs. arXiv preprint arXiv:2003.02320.
13. **Jain, A. K., Dubes, R. C. (1988).** Algorithms for clustering data. Prentice-Hall, Inc.
14. **Jain, A. K., Murty, M. N., Flynn, P. J. (1999).** Data clustering: A review. ACM Computing Surveys, Vol. 31, No. 3, pp. 264–323.
15. **Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P. S. (2022).** A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems, Vol. 33, No. 2, pp. 494–514.
16. **Kalfoglou, Y., Schorlemmer, M. (2003).** Ontology mapping: The state of the art. The Knowledge Engineering Review, Vol. 18, No. 1, pp. 1–31.
17. **Klaussner, C., Zhekova, D. (2011).** Lexico-syntactic patterns for automatic ontology building. Proceedings of the Student Research Workshop associated with RANLP 2011, pp. 109–114.
18. **Levy, O., Goldberg, Y. (2014).** Neural word embedding as implicit matrix factorization. Advances in Neural Information Processing Systems, Vol. 27, pp. 2177–2185.
19. **Manning, C. D., Raghavan, P., Sch"utze, H. (2008).** Introduction to Information Retrieval. Cambridge University Press.
20. **Mihalcea, R., Tarau, P. (2004).** Textrank: Bringing order into texts. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 404–411.
21. **Mikolov, T., Chen, K., Corrado, G. S., Dean, J. (2013).** Efficient estimation of word representations in vector space. International Conference on Learning Representations, pp. –.
22. **Murphy, K. P. (2012).** Machine Learning: A Probabilistic Perspective. MIT Press.
23. **Noy, N. F. (2004).** Semantic integration: A survey of ontology-based approaches. SIGMOD Record, volume 33, ACM, pp. 65–70.
24. **Noy, N. F. e. a. (2003).** Protégé-2000: An open-source ontology-development and knowledge-acquisition environment. AMIA Annual Symposium Proceedings.
25. **Reimers, N., Gurevych, I. (2019).** Sentence-bert: Sentence embeddings using siamese bert-networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. –.

26. Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research, Vol. 11, pp. 95–130.

27. Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, Vol. 20, pp. 53–65.

28. Russell, S., Norvig, P. (2020). Artificial Intelligence: A Modern Approach. Pearson, 4th edition.

29. Salton, G., Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing and Management, Vol. 24, No. 5, pp. 513–523.

30. Velardi, P., Navigli, R., Cucchiarelli, A., Neri, F. (2005). Evaluation of ontolearn, a methodology for automatic learning of domain ontologies. pp. –.

31. Wolf, T., Debut, L., Sanh, V., others **(2020).** Transformers: State-of-the-art natural language processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations, pp. 38–45.

32. Wu, Z., Palmer, M. (1994). Verb semantics and lexical selection. 32nd Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, pp. 133–138.

Article received on 19/05/2025; accepted on 22/09/2025.
**Corresponding author is Alexander Gelbukh.*