Semi-Supervised Ontology Learning from Limited Datasets: A Novel Approach Leveraging Large Language Models and Core Concepts

Aurélien Boissière[1], Mounira Harzallah[1], Margo Bernelin[2], and Patricia Serrano-Alvarado[1]

[1] LS2N, University of Nantes, Rue Christian Pauc, Nantes, 44300, France
[2] DCS, Nantes University of Law, Chemin de la censive du tertre, 44300, France

Abstract. The need for accurate and efficient ontology learning, especially in the context of artificial intelligence (AI) regulations, is essential. Traditional ontology learning methods, which are largely manual, are fraught with challenges - they are time-consuming, error-prone, and difficult to maintain. This paper proposes an innovative approach to ontology learning that uses a large language model, semi-supervised dimension reduction, semi-supervised clustering, and Core Concepts as prior knowledge, specifically designed for use with limited datasets. Our novel method outperforms existing models and is especially effective in dealing with imbalanced classes. Our experiments indicate that the iterative loop incorporated in our model significantly enhances its performance. The choice of Core Concepts also surfaced as a crucial aspect affecting model performance. More generic Core Concepts tend to perform better due to their high probability of occurrence in the corpus and accurate embeddings. This research contributes a fresh perspective to the field of ontology learning and offers a robust, scalable, and adaptable solution for term clustering around Core Concepts, with specific applicability to the interpretation and application of AI regulation laws.

Keywords: Ontology learning · Large Language Model · Model combination · AI regulation

1 Introduction

Artificial intelligence technology have been extensively applied to almost every area of our life during the past ten years, from smartphones and vehicles to houses and robots. A significant portion of these technologies use applications that necessitate natural language processing, information retrieval, and similar techniques, all of which require reference knowledge bases to enable a conceptual understanding of the world and, consequently, a semantic comprehension of the environment. Ontologies, in this context, have emerged as an invaluable tool across numerous domains, playing a pivotal role in the semantic web. They provide a formal method to define concepts and semantic relations between them, enabling reasoning and the derivation of facts.

Given their role as reference knowledge bases, ontologies are a crucial component in any system, with any design flaws having serious implications for the entire system. Therefore, it is imperative that they are constructed in a manner that accurately reflects the actual environment concepts. Unfortunately, the conventional method of manual construction is tedious to maintain, time-consuming, prone to mistakes, and riddled with difficulties. To mitigate these issues, learning techniques have been proposed to automate or semi-automate the process of ontology construction, giving rise to the field of Ontology Learning (OL), a subset of the broader ontology engineering domain.

Contrary to most studies in the field that make use of a large corpus, in this paper we are limited to only a small dataset.With this constraint in mind, we propose a novel approach that leverages a large language model, semi-supervised dimension reduction, semi-supervised clustering, and Core Concepts as prior knowledge. This innovative combination aims to enhance the efficiency and reliability of ontology learning, even when working with a limited dataset.

The implications of our approach extend beyond ontology learning, with a significant application in the field of AI regulation. Our methodology is particularly suited to the analysis and interpretation of AI regulation laws, a critical area given the rapid advancement and widespread application of AI technologies. By applying our methodology to this domain, we aim to contribute to the ongoing efforts to understand, interpret, and apply AI regulation laws more effectively.

The remainder of this paper is organized as follows: Section 2 delves into the related work, providing a comprehensive review of existing methodologies and highlighting the gaps our research aims to fill. In Section 3, we detail our approach, focusing on the integration of semi-supervised dimension reduction, semi-supervised clustering, and Core Concepts as prior knowledge. Section 4 presents the results of our experiments, offering empirical evidence of the performance of our approach. In Section 5, we address the critical aspect of reproducibility, comparing our model to others from the literacy. Finally, Section 6 concludes the paper, summarizing our findings and suggesting potential avenues for future exploration.

2 Related Work

The genesis of taxonomic relation extraction can be traced back to lexico-syntactic patterns, one of the earliest approaches developed for this purpose. Hearst patterns [6] helped the identification of hyponymic relations by parsing text and detecting specific patterns like the following, where Y_i are hyponyms of X.

$$Y_1\{,\ Y_2\} * \{,\}\ \{and \mid or\}\ other\ X$$

However, this approach was typically confined to domain-specific tasks due to the requirement for words to match the pattern.

Deep learning approaches become more popular as a result of the development of machine learning and the increase in computing capacity. A significant milestone was the introduction of Word2Vec [12], one of the first models to propose unsupervised word embedding. This approach, which involves representing words in an embedded space, allows for the application of vectorial equations to extract meaning. This concept was further advanced with the development of transformer-based large language models like Bert [5] and GPT [3], which leverage the attention mechanism and encoder-decoder structure of Transformer [13].

However, embedding alone is insufficient for ontology extraction from unstructured text. A common method involves using the K-means algorithm to create word clusters, although this approach has limitations, particularly in terms of cluster labelling. More advanced clustering techniques, such as hierarchical clustering used by Felix Martel et al. [9], offer the advantage of extracting relations between terms.

A downside of the embedding is that the number of dimension is very high. As [1] show, the notion of distance starts loosing meaning. To compensate this issue, we usually reduce the dimension of our data. Commonly used techniques generally fall into two categories; those that preserve the pairwise distance structure, such as such as PCA [10] and those who preserve more local distances like t-SNE [8].

But the overall performance of models based on those techniques are relatively low. As stated, the clustering results are far from optimal. To compensate for this, Leland McInnes et al. [11] proposed a new approach for dimension reduction that can be use in a semi-supervised manner. Their model, UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction, learn a manifold representation of the data and most importantly tries to keep the local and global structure of the data.

Despite these advancements, the performance of models based on these techniques remains suboptimal, particularly when dealing with small datasets. To address this, various strategies have been developed, including the use of prior knowledge, as demonstrated by Hao Huang et al. [7], and ensemble learning techniques, such as Bootstrap Aggregating (Bagging) proposed by Leo Breiman [2], and the method proposed by David H. Wolpert [14].

3 Approach

The approach we propose addresses the challenge of ontology learning from a small corpus through few-shot learning. It integrates a pattern-based approach, dimension reduction, and semi-supervised clustering, with the latter two fine-tuned by Core Concepts in an iterative manner. The objective is to enhance semi-supervised clustering based on representation learning with prior knowledge.

Four types of prior knowledge are utilized in our approach:

- Pretrained Language Model (PLM)
- Corpus for the fine-tuning of PLM

- Hypernym patterns for the extraction of Core Concept hyponyms
- Core concepts and their hyponyms for fine-tuning of the dimension reduction and for deriving the semi-supervised clustering approach.

As depicted in Figure 1, our approach comprises the following main steps:

1. Fine-tuning of PLM with the domain-specific corpus
2. Pre-processing of the corpus, extraction of terms and Core Concept hyponyms
3. Term embedding with PLM
4. Fine-tuning of dimension reduction with Core Concepts and their hyponyms
5. Term filtering based on the results of dimension reduction
6. Derivation of semi-supervised clustering by Core Concepts and their hyponyms

To enhance the fine-tuning of the dimension and the performance of semi-supervised clustering, an iterative loop over steps 4 to 6 is added.

Fine-Tuning: This step aims to enhance the performance of the embedding model by training a portion of the model on our domain-specific data. This should increase the accuracy of the embedding of domain-specific terms that were not, or not significantly, present during the pretraining of the model.

Corpus Pre-Processing and Hyponyms Extraction: The corpus undergoes pre-processing to remove as many irrelevant terms as possible. Given that we are working with unprocessed texts, there are numerous words that only hold semantic interest, such as pronouns and verbs, which are not only irrelevant to our domain but also to the ontology learning. The pre-processing step also includes the removal of the non-lemma part of a word. Lastly, we extract hyponym relations to help our model in representing the terms in the embedding space.

Term Embedding: Following this, we compute the embedding of the terms, which enables us to perform mathematical operations on the terms.

Dimension Reduction: The output of Bert is 768 dimensions. We employ a dimension reduction algorithm to reduce the number of dimensions to 10. This effectively enhances the computational efficiency of our model. The model used also retains more global structure of the original data than other models. It also uses the synonyms extracted in the previous steps to increase or decrease the distance between certain terms to guide the subsequent clustering.

Term Filtering: After dimension reduction, some terms are clearly separated from others, without any Core Concept in close proximity. We can directly label these terms as irrelevant to our domain, without having to rely on clustering.

Semi-Supervised Clustering: Finally, we create clusters around the Core Concepts, using rules to constrain the clusters. This ensures that a cluster does not contain more than one Core Concept. During iteration, only a fraction of terms remain pseudo-labelled for the next iteration. The number of terms selected must be optimized to avoid overfitting the model with uncertain data.

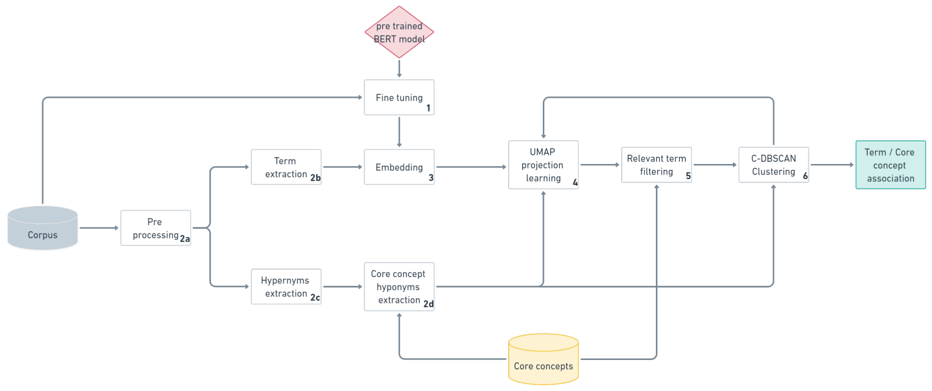

Fig. 1. Steps for term clustering around Core Concept

4 Experiments

4.1 Corpus

To conduct the experiment, we utilized a corpus on the regulation of AI. The corpus consists of the 208-page long proposal titled "Regulation Of The European Parliament And Of The Council Laying Down Harmonised Rules On Artificial Intelligence (Artificial Intelligence Act) And Amending Certain Union Legislative Acts". It comprises 2234 sentences and 56588 words. However, the vast majority of those words are not relevant to our domain, being mostly repeating words. Since we are only interested in noun phrases, we can eliminate articles, verbs, pronouns, etc. After the part-of-speech tagging and filtering with stopwords, we retained 920 terms to classify under the Core Concepts of AI regulation.

4.2 Benchmark: AI Regulation Ontology

To build an AI Regulation ontology, we initially defined a core ontology by identifying Core Concepts of the domain and relations between them. The selection was made according to existing ontology in the law domain (e.g.) and the terms to classify. Six general Core Concepts were created: Anything AI, Data, Organisation or Person, Risk, Law, and Market. Some of these are further specialized into subconcepts (see figure 2). This step was completed by eight computer science students, two computer science associate professors, and a researcher in the law field. This benchmark will then be used to evaluate our model.

For comparison, we also added another concept: Other. This concept will represent all the irrelevant terms that we found. As we will see later, handling the irrelevant terms proved to be quite challenging. As evident in figure 3, which shows the composition and size of each class related to a Core Concept,

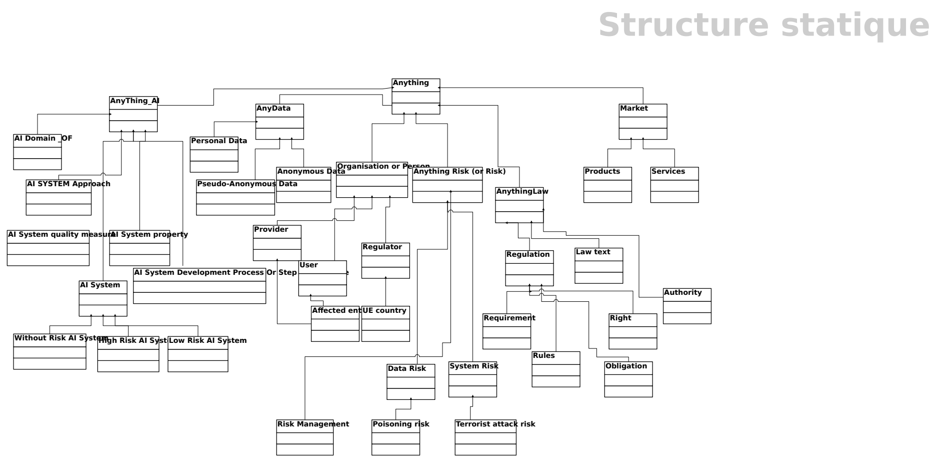

Fig. 2. The Core Concept centered ontology built by two computer science associate professors and a researcher in the law field.

the classes are not balanced. To minimize the impact on the results, we decided to split some Core Concepts according to the ontology. The revised term distribution can be seen in figure 4

4.3 Metrics

To measure our model's performance, we use four metrics: accuracy, precision, recall, and F1-score.

Accuracy quantifies the total number of correct predictions made by the model out of all predictions. It is calculated by taking the ratio of the number of correct predictions (both positive and negative) to the total number of predictions. The formula used is: (TP + TN) / (TP + FP + FN + TN).

Precision, also known as "positive predictive value", quantifies the number of correct positive predictions made. It is calculated by dividing the number of true positives by the sum of true positive and false positive predictions. The formula used is: TP / (TP + FP).

Recall, also known as "sensitivity", "hit rate", or "true positive rate" (TPR), quantifies the number of correct positive predictions made out of all actual positive instances. It is calculated by dividing the number of true positive predictions by the sum of true positive and false negative predictions. The formula used is: TP / (TP + FN)

Finally, the F1-score combines precision and recall into a single number. It is the harmonic mean of precision and recall, giving equal weight to both components. As our dataset is imbalanced, this is an appropriate metric to evaluate our model's performance. The formula used is: 2 * (Precision * Recall) / (Precision + Recall)

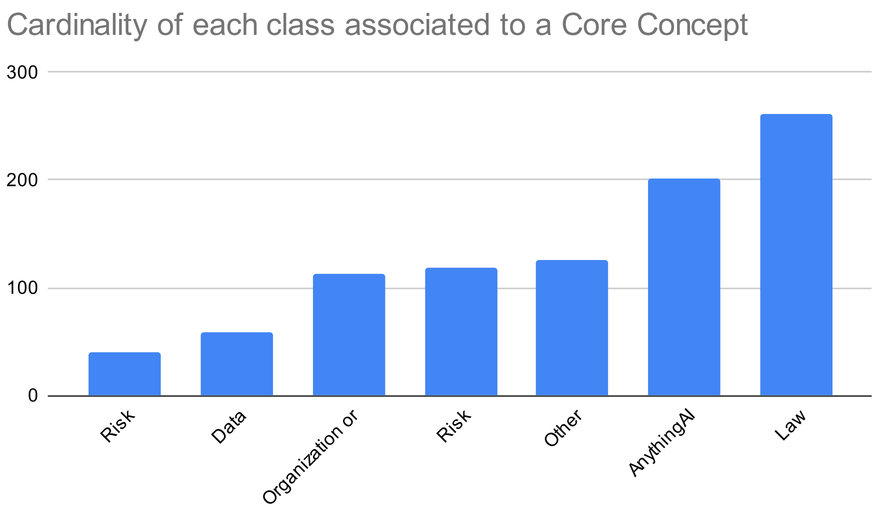

Fig. 3. Cardinality of each class associated to a Core Concept.

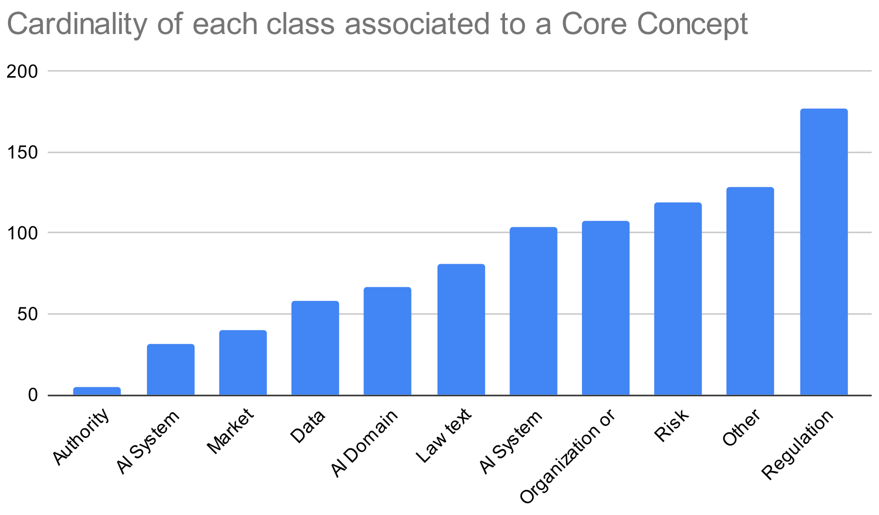

Fig. 4. Cardinality of each class associated to a Core Concept.

Given our task involves multilabel classification, we need to average these metrics over the different labels. To do this, we can use three methods referred to as "micro", "macro", and "weighted" in the scikit-learn documentation [4]. We disregarded the "micro" average as it yielded the same result for every metric. Moreover, the "macro" average does not take into account label imbalance. Given our dataset is heavily imbalanced, we decided to conduct our experiments using the "weighted" average. This method calculates metrics for each label, and finds their average weighted by support (the number of true instances for each label). This adjustment to 'macro' accounts for label imbalance, but can result in an F-score that is not between precision and recall.

4.4 Evaluation

Interest of dimension reduction fine-tuned with Core Concepts As we can visualize in the figure 5,, UMAP is able to dissociate each Core Concept. On the first iteration (Fig. 5 a)), some Core Concepts are still very close and the clusters can overlap. On the second iteration, UMAP can use the data extracted from the previous iteration to reposition the points in the embedding space. We can see in the figure 5 b) that the Core Concepts start to be more evenly spaced out. Finally, on the third iteration, the Core Concepts are almost evenly distributed. We can also see that the Core Concept in the top right corner is place further out in each iteration.

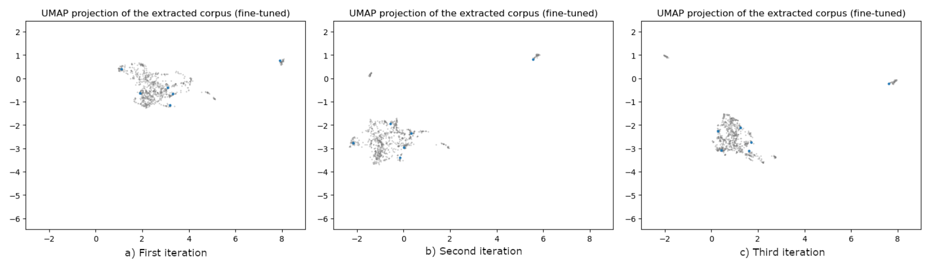

Fig. 5. First three iterations. (Blue points are Core Concepts). a) First iteration. The Core Concepts are still very close as UMAP could not efficiently move them apart yet. b) Second itération. UMAP starts to make the distinction between the Core Concepts. c) Third iteration. The Core Concepts are well separated.

We can then compare the performance of the dimension reduction model. We tested UMAP against a more basic PCA and without reducing the embedding dimension. We also wanted to focus on the number of iterations and finding a possible link between the classification and the clustering metrics to be able to choose the best number of iterations based on the corpus studied.

We observe on table 1 that not reducing the dimensions yields terrible results. Furthermore, PCA offers better results but still can't grasp the overall structure

Table 1. Comparison of different dimension reduction models. As we iterate over UMAP in our model, we tested it for multiple iteration values. Using "weighted" average.

Dimension reduction model	Accuracy	Precision	Recall	F1-score	Silhouette	Davies-Bouldin	Calinski-Harabasz
UMAP (1 iteration)	0.334	0.376	0.334	0.329	0.001	5.770	8.799
UMAP (2 iterations)	0.374	0.422	0.374	0.372	**0.013**	5.8	9.680
UMAP (5 iterations)	**0.398**	**0.482**	**0.398**	**0.405**	**0.013**	6.862	9.419
UMAP (10 iterations)	0.395	0.471	0.395	0.383	0.006	**5.271**	**10.681**
UMAP (15 iterations)	0.360	0.442	0.360	0.364	0.012	5.675	9.373
PCA	0.306	0.380	0.306	0.314	0.003	6.244	8.202
None	0.161	0.440	0.161	0.119	-0.028	7.436	2.931

of the embedding. Whereas UMAP, even without iterating, seems to preserve some kind of global structure, enabling the clustering algorithm to better cluster the terms.

Interest of the semi-supervised clustering Next, we tested different clustering and classification methods for comparison. For the classification task, the classifier is trained to predict if a word is an hyponym of a Core Concept. We used a support vector classifier and an artificial neural network to see if the model complexity would have a positive impact on the classification. As we have only a small corpus, we used boostrap aggregating (bagging) for the support vector classifier. To conduct this experiment, we replaced C-DBSCAN with other model, but kept all the other steps.

Table 2. Comparison between our model and other classifiers / clusterers.

Clustering model / Classifier	Accuracy	Precision	Recall	F1-score
C-DBSCAN	**0.398**	**0.482**	**0.398**	**0.405**
DBSCAN	0.224	0.083	0.224	0.087
SVM (Bagging)	0.384	0.227	0.384	0.264
SVM (no Bagging)	0.385	0.227	0.385	0.264
ANN	0.364	0.444	0.364	0.361

The results in table 2 show the usefulness of creating the clusters around the Core Concepts. We can see that DBSCAN have poor results in comparison to C-DBSCAN, mainly because it wasn't able to create as many clusters as there are Core Concepts. We also see that even though we had few training data, the performance of the neural network is close to C-DBSCAN, while remaining worse. Finally, we can see that the results between SVM with and without bagging are quite similar.

Interest of the combination of UMAP and C-DBSCAN and the iteration loop We wanted to compare the overall performance of our model compared to some baselines. We choose to compare our best model (UMAP, C-DBSCAN, 5 iterations) against our model with only 1 iteration, a simple support vecteur classifier and an artificial neural network. This allows us to compare our model to well known, and more generic, techniques.

Table 3. Performance comparison between our model and two baselines

Core Concepts choice	Accuracy	Precision	Recall	F1-score
UMAP + C-DBSCAN (1 iteration)	0.334	0.376	0.334	0.329
UMAP + C-DBSCAN (5 iterations)	**0.398**	**0.482**	**0.398**	**0.405**
SVM	0.353	0.406	0.352	0.245
ANN	0.388	0.403	0.388	0.362

We already demonstrated that the iteration loop was important within the model. But the results in table 3 show that our model needs the iteration process to compete with other more generic models. We can also note that simply using an artificial neural network without UMAP leads to better performance.

Importance of the Core Concepts choice As we discussed earlier, our classes are imbalanced and this can lead to biased results. Numerous techniques exist to re-balance a dataset We also tried to go deeper in our Core Concepts ontology. For our experiments, we used the top level Core Concepts, but by splitting those associated with the most terms into "sub" Core Concepts, we can try to re balance our dataset. Figures 3 and 4 show such separation. We chose to split "Anything AI" into "AI Domain", "AI System" and "AI System Development Process" and "Law" into "Regulation", "Law Text" and "Authority". As we can see in figure 4, the classes are still not balanced but the absolute cardinality difference is smaller.

Table 4. Table captions should be placed above the tables.

Core Concepts choice	Accuracy	Precision	Recall	F1-score
Top level	**0.398**	**0.482**	**0.398**	**0.405**
Re-balanced	0.246	0.276	0.246	0.232

As shown in the table 4, using more generic Core Concepts results in better performance. This might be due to the fact that more generic terms have a higher chance to be present in the corpus and so more hyponyms can be found. The embedding may play its role in the gain of performance as a generic term might have a more accurate embedding than a more specific one.

Impact of the Core Concept labels e also investigated the impact of adding a keyword in front of the Core Concept. This is due to how the ontology was firstly written, where some Core Concept had the tag "any" or "anything" to denote that they are generic terms. However, this addition can have a real impact on the model performance as it changes the embedding.

Table 5. Comparison between different labels for the Core Concepts.{CC} is a placeholder for the true Core Concept label.

Core Concepts labels	Accuracy	Precision	Recall	F1-score
Core Concept (CC)	**0.398**	**0.482**	**0.398**	**0.405**
Anything{CC}	0.356	0.411	0.256	0.351
Anything {CC}	0.372	0.419	0.371	0.374
Any{CC}	0.381	0.451	0.381	0.382
Any {CC}	0.333	0.422	0.333	0.313

Table 5 shows that adding a keyword in front of the Core Concept label worsen the model's performance. This is likely due to the embedding model, positioning the Core Concept closer as they share a common keyword. As the Core Concept are placed closer to each other, the performance of the clustering algorithm will decrease as more terms will overlap between multiple possible clusters. Adding a space between the keyword and the Core Concept doesn't seem to have a definitive impact on the performance. For the keyword *Anything*, the space resulted in an increase of performance, but for *Any*, it was the opposite.

Table 6. Performance Comparison Across Different Embedding Models.

Embedding model	Accuracy	Precision	Recall	F1-score
Bert	**0.398**	**0.482**	**0.398**	**0.405**
Bert (fine tuned)	0.356	0.445	0.356	0.355
Word2Vec	0.296	0.235	0.296	0.251

Other evaluations As shown in table 6, BERT without fine tuning was the most efficient model. Although we would expect a fine tune model to be more efficient on our data, this can be caused by two things. Firstly, our corpus is really small, the model doesn't have enough data to be trained on. Secondly, The training method wasn't optimal. We fine tuned the model on a subject recognition task, training the model to recognize if two sentences have the same subject. To save labeling time, this was done in a semi supervised manner, resulting in a semi accurate training.

4.5 Repeatability

We compared our results to Hao Huang et al. [7] on their corpus to see how corpus size would affect our model's performance. However, as stated in their

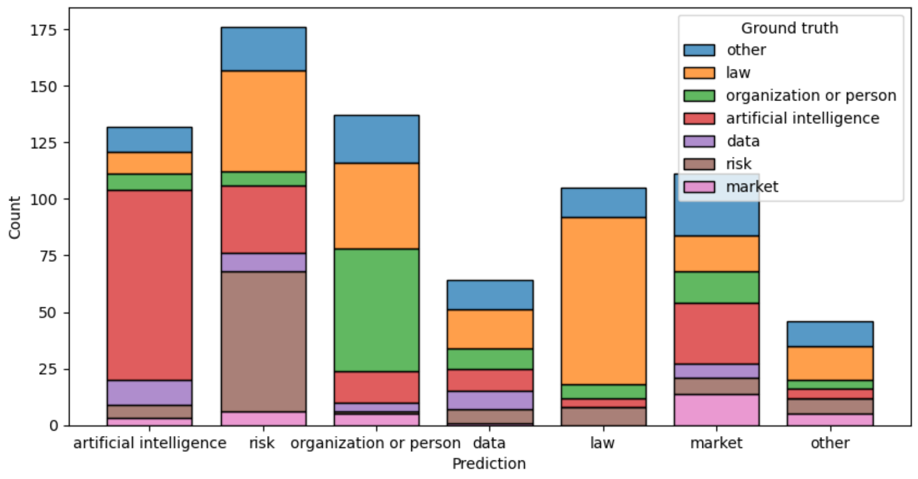

Fig. 6. Model's prediction for each Core Concept (5 iterations).

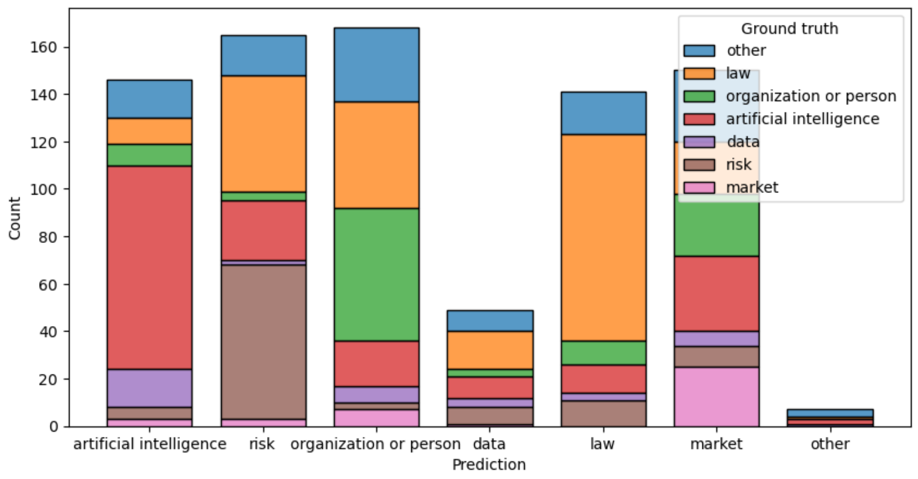

Fig. 7. Model's prediction for each Core Concept (10 iterations).

article, their corpus is heavily imbalanced toward the *other* class. As we consider this class as noise, or irrelevant data, this could have a negative impact on our performance. The first thing that we noticed was the difference of time it took to perform the computation. On our corpus, code optimization was not required and the not optimized part of the code remained unnoticed. However, when executing the model on a bigger corpus, we could measure an increase in computing time. With this data, we measured a time complexity between O(n) and O(n^2).

Table 7. Comparison between Core seeded LDA and our model.

Model	Performance
CC-seed	0.5804
UMAP + C-DBSCAN (1 iteration)	0.5194
UMAP + C-DBSCAN (5 iteration)	0.5210
UMAP + C-DBSCAN (10 iteration)	0.5107

As we can see in table 8, the performance of the model follows the same pattern as on our dataset. The classification yields better results with 5 iteration although two clustering metrics are better with 10 iteration. As the results are quite close between 5 and 10 iterations, we can consider that doing 5 iteration regarless of our data is a good baseline.

Table 8. Comparison of our model model's iteration process on another dataset.

Core Concepts choice	Accuracy	Precision	Recall	F1-score	Silhouette	Davies-Bouldin	Calinski-Harabasz
UMAP + C-DBSCAN (1 iteration)	0.440	0.448	0.440	0.435	0.013	6.482	20.680
UMAP + C-DBSCAN (5 iterations)	**0.521**	**0.570**	**0.521**	**0.526**	0.017	**5.065**	24.324
UMAP + C-DBSCAN (10 iterations)	0.511	0.560	0.511	0.517	**0.018**	5.456	**25.174**

5 Conclusion

In this research, we introduced a novel methodology for term clustering around Core Concepts in the ontology learning domain. Our approach, heavily influenced by semi-supervised learning principles, leverages the strength of the embedding model BERT and the dimension reduction algorithm UMAP, coupled with C-DBSCAN clustering.

The experimental results have shown promising outcomes, especially in the context of imbalanced classes. Our method outperforms other models, including the standard SVM or ANN. We also demonstrated that the iteration loop significantly increased the model's performance.

A noteworthy point is the importance of the choice of Core Concepts. More generic Core Concepts resulted in better performance, likely due to their higher chance of being present in the corpus and more accurate embeddings. However, caution must be taken when adding keywords to Core Concept labels as it may affect the embedding and thus, the model performance.

An intriguing avenue for future work would be to optimize the handling of irrelevant terms. As our experiments showed, irrelevant terms can have a significant impact on model performance. Furthermore, implementing our method on a larger corpus, perhaps with more balanced classes, could present an interesting challenge. As shown in our repeatability tests, while our model is scalable to a certain extent, further optimization may be necessary for larger datasets.

Overall, this work contributes a new perspective to the field of ontology learning, providing a robust and adaptable approach for term clustering around Core Concepts.

References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) Database Theory — ICDT 2001. pp. 420–434. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
2. Breiman, L.: Bagging Predictors. Machine Learning **24**, 123–140 (1996). https://doi.org/10.1007/BF00058655
3. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020)
4. Cournapeau, D.: https://scikit-learn.org/stable/modules/model$_{e}valuation.html$
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018), http://arxiv.org/abs/1810.04805, cite arxiv:1810.04805Comment: 13 pages
6. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics (1992), https://aclanthology.org/C92-2082
7. Huang, H., Harzallah, M., Guillet, F., Xu, Z.: Core-concept-seeded lda for ontology learning. Procedia Computer Science **192**, 222–231 (2021). https://doi.org/https://doi.org/10.1016/j.procs.2021.08.023, https://www.sciencedirect.com/science/article/pii/S1877050921015106, knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021
8. van der Maaten, L., Hinton, G.: Viualizing data using t-sne. Journal of Machine Learning Research **9**, 2579–2605 (11 2008)
9. Martel, F., Zouaq, A.: Taxonomy extraction using knowledge graph embeddings and hierarchical clustering. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. p. 836–844. SAC '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3412841.3441959, https://doi.org/10.1145/3412841.3441959

10. Maćkiewicz, A., Ratajczak, W.: Principal components analysis (pca). Computers & Geosciences **19**(3), 303–342 (1993). https://doi.org/https://doi.org/10.1016/0098-3004(93)90090-R, https://www.sciencedirect.com/science/article/pii/009830049390090R
11. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (2020)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023)
14. Wolpert, D.H.: Stacked generalization. Neural Networks **5**(2), 241–259 (1992). https://doi.org/https://doi.org/10.1016/S0893-6080(05)80023-1, https://www.sciencedirect.com/science/article/pii/S0893608005800231