SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Extracting Semantically Meaningful Context Windows around Class-Specific Keywords

Alexandra Seibicke

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Extracting Semantically Meaningful Context Windows around Class-Specific Keywords

Extraktion von semantisch sinnvollen Kontextfenstern um klassenspezifische Schlüsselwörter

Author:	Alexandra Seibicke
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Stephen Meisenbacher, M.Sc.
	Tim Schopf, M.Sc.
Submission Date:	15.12.2023

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.12.2023 Alexandra Seibicke

Acknowledgments

I would like to thank deeply my advisors, Stephen Meisenbacher and Tim Schopf, for guiding me through my thesis, for constant feedback and also brainstorming. Furthermore, I want to express gratitude to Prof. Dr. Florian Matthes for granting me the opportunity to write this thesis at his chair. Moreover, thanks also to my friends and family for their support and encouragement. Special thanks also to my boyfriend, who stood by me and supported my through challenging times.

Abstract

In our present time, automatic text generation, recognition and translation have become more and more important. With the rise of chatbots like ChatGPT, the use of artificial intelligence in natural language has increased drastically. But to use artificial intelligence for natural language processing, numerous texts have to be pre-processed. Doing so requires annotating texts and extracting keywords, which must be put in meaningful contexts. A word can contain various meanings depending on the context, for example "bank" could be a credit institute, or a shore, or even a verb. To figure out the meaning of a word, a context window must be set, which is the scope around a word used to identify its meaning. The arbitrariness of how long a context window may cause ambiguity, which makes extracting meaningful and useful context windows a significant challenge. This thesis addresses the task of finding such context windows.

For this purpose, we extract keywords from sentences and implement various approaches to determine meaningful context windows. We develop pre- and post-processing steps and evaluate our results. These results can then be used for further by natural language processing pipelines.

We investigate the current state-of-the-art approaches for Word Sense Disambiguation and address the question on how these approaches can be combined with clustering techniques for class-based context filtering. Further, we study what method evaluation approaches are most appropriate to assess the effectiveness of filtering context windows of sentences and the cohesiveness of the window extraction. In addition, we investigate which evaluation approaches are most appropriate to assess the effectiveness of the filtering step and the cohesiveness of the window extraction.

Contents

1. Introduction

1.1. Motivation

With the rise of chatbots like ChatGPT, the use of artificial intelligence in natural language has become more and more important in our present time. For automatic text processing, word disambiguation can be a problematic impediment for training models. Humans often derive the proper meaning of a word subconsciously by participating in many discussions over a lifetime. Associating the correct meaning with a word is, therefore, not a trivial task. Thus, finding, trimming, and evaluating meaningful scopes around a given word of a text plays a crucial role in numerous tasks in the natural language processing field.

This thesis is part of the pipeline of the CreateData4AI (CD4AI) project[1]. The project aims to develop a framework that helps domain experts annotate text using Natural Language Processing algorithms[1]. This is conducted in several sub-tasks:

1. **Keyword Extraction** Domain experts first define classes used to extract keywords with unsupervised techniques. Further, related words and phrases are suggested.
2. **Context Window Extraction** To associate a correct meaning of the keywords given a text, meaningful context windows are extracted.
3. **Context Rule Creation** Domain experts will further evaluate these windows and select those that best describe the meaning, creating context rules. This will serve as a basis for automated data set creation.
4. **Extrapolation** The last step uses the set of context rules and extrapolates a finite set of rules for a theoretically infinite number of unseen documents.

1.2. Research Questions

For this thesis, we will investigate the following research questions:

- **RQ1: What are the current state-of-the-art approaches for Word Sense Disambiguation?**

[1] *CreateData4AI*. URL: `https://wwwmatthes.in.tum.de/pages/nqpi6qljq0x9/CreateData4AI-CD4AI` (visited on 12/07/2023).

For the first research question, we will research exhaustively different approaches and conduct a brief overview.

- **RQ2: How can these approaches be combined with clustering techniques for class-based context filtering?**

  To answer the second research question, we will first explore different clustering techniques for class-based context filtering and evaluate them on our use case. Next, we will discuss how to combine clustering techniques with Word Sense Disambiguation.

- **RQ3: What methods can be leveraged to trim meaningful context windows from text chunks containing keywords?**

  The third research question will be answered by inspecting different methods we came up with and discussing challenges and alternatives.

- **RQ4: Which evaluation approaches are most appropriate to assess the effectiveness of the filtering step and the cohesiveness of the window extraction?**

  The last research question will be examined at the end of the thesis together with a comprehensive evaluation of our approaches.

1.3. Thesis Outline

This thesis is structured in seven chapters.

The first chapter introduces the motivation for this thesis, presents the research questions, and offers an overview of the structure.

The second chapter establishes background information by introducing the current state-of-the-art approaches for Word Sense Disambiguation and, therefore, answers the first research question and introduces context windows. Further, we conduct exhaustive research on clustering techniques and answer the second research question.

Subsequently, the third chapter introduces related works and evaluates these in our use case.

The third research question is answered in the fourth chapter, which introduces several approaches which are compared on a simple example as well.

In addition, the fifth chapter presents the results and answers the last research question by evaluating the presented approaches. In addition, an expert evaluation is conducted.

The sixth chapter provides a discussion about the results, challenges and future work, and the last chapter presents an overall conclusion of this thesis.

2. Background

In this chapter, we will establish some background information needed for the further course of this thesis. We will first talk about Word Sense Disambiguation and answer the first research question, we will introduce clustering and clustering techniques, and conclude with answering the second research question.

2.1. Word Sense Disambiguation

Many words in the human language have ambiguous meanings, for example, a "bank" can be a financial institution or an edge of a river. To get the correct meaning of the word, we humans rely on a "context" embedded in further words or sentences beneath the desired word. Often enough we also know the different meanings of a word and just assume one meaning since the other meanings do not make "sense".

We have to rely on alternative approaches for machines to get the proper meaning. Word Sense Disambiguation can be summarized as follows by Navigli et al. [2]:

> The computational identification of meaning for words in context is called word sense disambiguation (WSD).

Mallery [3] also describes WSD as an AI-complete problem which means it is equivalent to the Turing Test.

Navigli [2] further provides us with the following definition of a WSD:

When viewing a text as a sequence of words without punctuation, WSD is the task of finding a mapping from these words to the senses. The mapping should be a subset of the available senses of the word.

Ideally, the mapping should represent one sense for each word, but it can also be more senses per word.

2.1.1. Knowledge-Based Approaches

For finding available senses to a word, external knowledge sources like Thesaurus, Ontologies, and so on are created that use the word and its context to find suitable mappings. A famous lexicon for words is the Princeton database WordNet [4], which contains a wide range of synonyms of words and more. To find the context of a word, it is necessary to first preprocess the word, for example with tokenization, part-of-speech tagging, lemmatization, and more. The advantages of knowledge-based approaches include robustness to noisy data and interpretable outcomes. However, it heavily relies on expert knowledge and provides limited scalability.

There are various Knowledge-Based Approaches like Lesk's algorithm, Walker's algorithm, Selectional Preferences, to name a few. In the following sections, we will introduce some algorithms and approaches.

Lesk Algorithm

Lesk Algorithm was developed by Michael E. Lesk in 1986 [5]. The algorithm merely counts the number of overlaps between all dictionary definitions of a word and the words surrounding it excluding stop words. Stop words are words like "the", "a", "and" [6]. These words surrounding a word are also called a "context window".

For example, Lesk shows that a pine is a "kind of evergreen tree with needle-shaped leaves.. ." by definition of the Oxford Advanced Learner's Dictionary of Current English, and cone is a "fruit of certain evergreen trees..." [5]. In both definitions, evergreen and tree are used to define these words, indicating they are similar hence if we consider a text containing pine and cone in a context window, a program can conclude the word "pine" is likely a tree [5].

Although easy to implement and generalize, this algorithm has low accuracy (around 50-70%) and low recall since many words have similar meanings, but no overlap in definition [6]. The algorithm has since been developed by multiple researchers, like Viveros-Jiménez et al. [7] who proposed to adapt the context window by removing the target word and more. Also, WordNet [4] was inspired by the Lesk algorithm.

Semantic Similarity

In essence, semantic similarity uses a knowledge base of all available senses of a word. Now given a word in a text, we collect the senses of all words in a context window except the target word and sum the contribution of the most appropriate sense of each word. The sense with the highest sum is chosen [2].

Many adaptations of this method have been implemented since, for example, Rada et al. [8] calculated the shortest distance in WordNet [4] between pairs of word senses [2].

Mittal et al. [9] used an Ordered Weighted Averaging Operator (OWA) to calculate the senses and later used an expert evaluation to further optimize the algorithm.

Selectional Preferences

For each word, semantic relationships are collected and inappropriate senses of words are excluded leaving only senses that align with established rules. The occurrences of word pairs with syntactic relations are counted and the word sense with the most common frequency is selected [10].

Heuristic Approaches

Additionally, heuristic approaches can be used for Word Sense Disambiguation. According to Gujjar et al. [10], there are three types of heuristics:

1. Identifying all possible word senses with the assumption that one particular sense occurs more frequently than others.
2. A word occurring multiple times in a text is likely to maintain the same meaning.
3. Nearby words can indicate the correct meaning of a word.

Walker's Algorithm

Walker's Algorithm uses a thesaurus to assign each sense of a word a thesaurus class. The context in which a word appears is used to match it with a thesaurus category [11].

Graph-Based Methods

Graph-based methods explore the possibility of a relationship between words. For example, an "Is-A" relationship chain could be "Apple" "Is-A" "Fruit" "Is-A" "Plant". A graphic version is shown in figure 2.1.

Fruit
Is-A
Fruit
Is-A
Fruit

Figure 2.1.: An "Is-A" relationship of an apple

This is also explored by adding other relationships, such as "Kind-Of" or "Is-Like".

Based on this approach, Galley and McKeown [12] developed an algorithm that first builds a graph where all possible senses of a word are connected with the word. Next, if the senses of a current word are connected with the senses of a previous word, a connection is established between the appropriate words and senses. For deciding the meaning of a word, its senses, and their respective connections are summed by giving the sense the highest score that has the most connections chosen as the proper sense.

2.1.2. Machine Learning Approaches

One major separation of Machine Learning Techniques is supervised vs unsupervised learning. In supervised learning, a labeled data set is needed and we train the model to predict the corresponding label of a datum. The focus is on learning the relationship between input and output data. Classification and regression can be solved with supervised machine learning. Supervised learning often leads to high accuracy and is suitable for a variety of problems that lead to the development of many well-established algorithms. However, it can be sensitive to outliers and noise. In addition, it lacks generalization and is not scalable, since it requires labeled data [10].

In unsupervised learning, we train the model without labeled data and allow the model to discover new patterns and relationships. This learning type is typically used for clustering or association. Since no annotated data is needed, it can overcome the limitations of Supervised Machine Learning and Knowledge-Based Approaches. However, it performs often worse than Supervised Machine Learning approaches. Additionally, interpreting results can be difficult [10].

Semi-supervised learning contains both labeled and unlabeled data. It can be used for classification and regression tasks. Since this approach is a combination of supervised and unsupervised techniques, it can improve the performance of limited labeled data. However, obtaining and managing labeled data may be difficult and it may not outperform the former approaches. In addition, it may suffer from error propagation due to incorrect labels [10].

In the following sections, we will introduce the most important approaches of different machine learning techniques with a focus on the techniques we used for our approaches.

Decision List and Decision Tree

Rivest [13] introduced decision lists as an easy list of if-else statements. A simple example can be shown in 1.

Algorithm 1 A simple Decision List

```
if $f_1$ then
    output $b1$
else
    if $f_2$ then
        output $b2$
    end if
end if
```

A decision tree consists of decision lists ordered in a tree-like structure. We first begin with a simple decision function, and depending on the output of the function, a subsequent function is evaluated, as shown in figure 2.2.

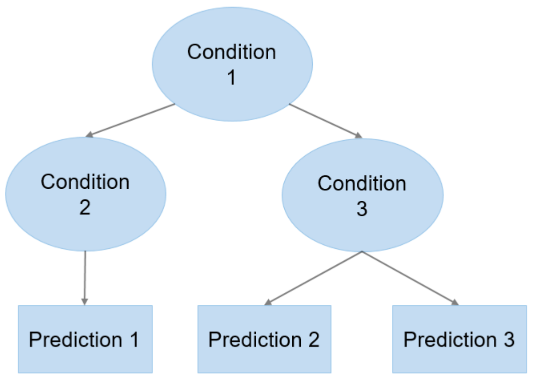


Figure 2.2.: A simple Decision Tree

In further chapters, we will use a RandomForest Classifier for our task. RandomForest is a strategy where multiple decision trees are combined to increase accuracy, making them sometimes the best performing types of classifiers [14]. Additionally, only a subset of features is considered for each condition evaluation making each tree marginally different.

Further, we use XGBoost as an alternative to RandomForest. It works similarly to RandomForest, but each decision tree learns from the tree before and is trained only on the data that the last tree did not perform well [14].

Naive Bayes

Machine learning models can utilize the Bayes Theorem for prediction using prior predicted data and the new information for the given data.

Using Bayes' theorem, we can calculate the posterior from prior knowledge given data, as you can see in equation 2.1.

$$P(\theta|\mathbf{D}) = P(\theta)\frac{P(\mathbf{D}|\theta)}{P(\mathbf{D})} \tag{2.1}$$

Martins [15] further explains that the Gaussian classification assumes that each class follows a normal distribution and that the parameters that predict the output are independent.

Support Vector Machines

Another approach that has gained much attention in recent years is support vector machines. A support vector machine (SVM) aims to identify a hyperplane that isolates classes with the highest possible margin, i.e., the maximum distance between data points of both classes. Support vectors represent data points lying closest to the hyperplane.

Neural Networks

Neural Networks consist of pairs of input features and the desired response that is processed by node layers containing an input layer, one or more hidden layers and an output layer. Each layer consists of nodes connected to one another and has an associated weight. An example illustration is provided by figure 2.3.

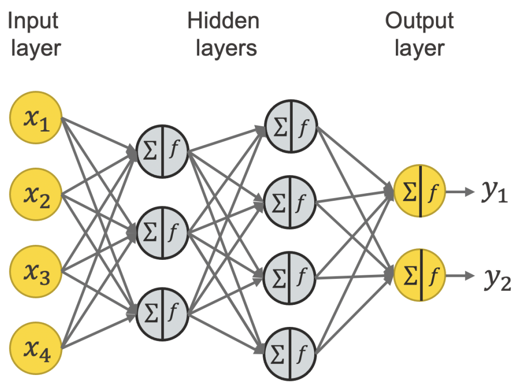


Figure 2.3.: A Neural Network [16]

Convolutional Neural Networks (CNNs) in NLP use filters and strides to process n tokens of a given sentence. Intuitively, the first layers capture and learn more primitive features and deeper layers detect more complex and more detailed features [17].

Recurrent neural networks (RNNs) use feedback loops to predict future tokens using the previous outputs of the network [18]
In Graph Neural Networks (GNN), text is represented as a graph, with nodes denoting words, and edges denoting relationships between words [18].

Context and Word Clustering

Context and Word Clustering are both techniques that are used in unsupervised machine learning.
Martin-Wanton et al. [19] implemented context clustering that represents words through context vectors that are organized into clusters with each cluster corresponding to a word sense. A word is transformed into a vector and captures the frequency of its co-occurrences with other words. By applying clustering techniques, a final sense can be assigned to a word [10].

Word Clustering denotes the grouping of words that are semantically similar together. There are various measurements for similarity available, but often features of words are analyzed, like part of speech tag and more. Clustering techniques can be used to classify meanings.

2.2. Overview of state-of-the-art approaches

To understand current trends and techniques, establishing a background is needed which we presented in the previous sections. In this section, we propose some of the most notable techniques and current research in recent years.

First, we want to include some notable databases and data sets. Although it is a rather old database, WordNet [4] is still a popular database, since it is well established with many entries.

Recently, BabelNet [20] gained interest with its semantic network where nodes are multilingual synsets [21].

SemCor [22] contains annotated data and is still used today, and for evaluation, different data sets called SensEval and SemEval are available that are still being further developed. For non-English languages, local and customized data is often required and for many languages, there is not a well-established database available [21].

As for approaches, pre-trained language models gain more and more attention and generate promising results. Many knowledge-based methods are completely outperformed by machine learning models and, therefore, not relevant for today's state-of-the-art research [21]. Nevertheless, many models use knowledge-based techniques as a basis or for improvements of their predictions [21]. Therefore, it is necessary to first gain an understanding of these techniques before we can dive into recent developments.

For these models, it is also crucial to develop more data as many models see significant improvements when training on more data as well as adding diverse information [21].

As for specific techniques, BERT [23] (Bidirectional Encoder Representations from Transformers) and GPT [24] (Generative Pre-trained Transformer) have gained a lot of attention in recent years. BERT is used as a pre-training step and consists of an encoder that processes a sequence of words at once and learns the context of a word. Whereas GPT takes n tokens and predicts one output and with a lot of data and training, can become incredibly powerful [24].

These approaches are further developed such as RoBERTa, XLNET, Chat-GPT, and more. According to Bevilacqua et al. [21], SREFKB and ESCHER belong to the best-performing models.

Another powerful approach is to ensemble methods that combine the predictions of multiple models to increase accuracy and robustness. However, the overhead and complexity can increase drastically and represent major challenges [25].

Further, techniques like attention [26] gain popularity which allows models to focus only on specific parts of the sentence. Attention concentrates on the parts with the most relevant information concentrated.

Despite recent improvements in Word Sense Disambiguation, this problem is not solved. Especially for multi-language tasks, more research, databases, and evaluation are needed [21].

2.3. Clustering Techniques

Clustering Techniques can be used to identify related points or objects and their relationships. The exact type of relation can be very diverse, it can be classified based on features as well as measuring the distance of points.

A first distinction of clustering techniques can be soft and hard clustering. Hard clustering denotes that a point is either in the class or not, whereas soft clustering categorizes a point in a class through a probability [27].

Further, Rai et al. [27] describe the following various kinds of clustering:

- Well-separated clusters: Any point in this cluster is closer to every other point in this cluster than any point not in the cluster.
- Center-based clusters: Any point in this cluster is closer to the center of this cluster than to the center of any other cluster. This center is also often called a "centroid".
- Contiguous clusters: Any point in this cluster is closer to one or more other points in this cluster than to any point not in the cluster.
- Density-based clusters: A cluster is described as a dense region of points that is separated by low-density regions.
- Shared Property or Conceptual Clusters: Points in a cluster share a common property of a concept.
- Described by an Objective Function: A cluster is described by an object function that is minimized or maximized.

Common challenges in creating clusters are handling outliers, scaling, handling noise, and more. Therefore, selecting appropriate clustering techniques for a given task is not trivial. In the following sections, we will introduce some of these techniques.

2.3.1. Hierarchical Clustering

Hierarchical clustering aims to build a hierarchy of clusters through a series of partitions.

The advantages of hierarchical clustering are embedded flexibility regarding the level of granularity, easy handling of various forms of similarity, and easy application.

Nevertheless, it may be challenging to establish a clear termination criterion, and most algorithms do not revisit the clusters for improvement [28].

Agglomerative Clustering

For Agglomerative Clustering, each data point is first assigned to its own cluster merging together to form final clusters [28].

2.3.2. Partition Clustering

Partitioning Clustering, on the other hand, decomposes data into clusters with each cluster being represented by a centroid.

K-Means

An established implementation of partition clustering is K-Means. Given the data, the goal is to find k clusters each with a centroid calculated by the mean of the cluster. This is repeatedly optimized until the centroids do not change.

Limitations of the algorithm occur when clusters are of different sizes, densities, or shapes and also when the data contains outliers.

2.3.3. Grid Clustering

Grid clustering focuses on building geometric structures of an object in the space, their relationships, and properties. The goal is to develop several hierarchical levels of groups of objects [27].

2.3.4. Density Based Clustering

Density Based Clustering analyzes the data on densities and creates clusters accordingly.

DBSCAN

DBSCAN is a density-based clustering algorithm that handles outliners very well since it groups 'densely grouped' data points into a single cluster [28].

2.3.5. Probabilistic Clustering

Probabilistic Clustering assumes the data is distributed according to one or multiple specific distributions. It classifies the data according to whether they belong to a specific distribution or not [28].

Gaussian Mixture Models

Gaussian Mixture Models (GMM) assumes the data is generated by Gaussian distributions [28].

2.3.6. Exemplar-based Clustering

In Exemplar-Based Clustering, the goal is to identify representative data points, and each data point is associated with an exemplar [29].

Affinity Propagation Clustering

In Affinity Propagation Clustering, data points communicate with each other and clusters are created by comparing similarities between points. The data points exchange messages and form a consensus on how to create clusters [28].

2.3.7. Combining Word Sense Disambiguation Approaches with clustering techniques for Context Filtering

Context filtering is essential for Word Sense Disambiguation. Take a look at the following examples:
"I am going to the bank to withdraw money".
"The fishermen set up their camp on the lush green bank of the river".
Finding the correct context window that embeds the context to define the meaning of the word "bank" is not trivial. It is necessary to set the context window size large enough to include keywords like money or river.
But it is also essential to set the context window not too large since false clues can be included, and given a context window, finding the corresponding class can be difficult.

As explained previously, Martín-Wanton et al. [19] showed how context clustering can be used for the problem of word sense disambiguation. The key idea is to use clustering techniques to assign a final word sense from a vector of word senses.

Further, Dolan [30] describes the problem that some words do not have an obvious mapping in some sense. Some words may be mapped to multiple or none senses, and with the increasing complexity of available senses to a word (for example by using multiple dictionaries), the probability for multiple senses increases, whereas for a too-small selection of senses, none senses may be assigned. Clustering is a suitable technique to approach this problem. The combination and clustering of the senses for words according to multiple dictionaries can give a rough indication of the semantic connections between two entries.

In our data set, we have more than 2 million descriptions and more since a description can contain multiple or no keywords. Therefore, it is necessary to use a clustering technique that can handle big data. Further, we know priory how many clusters we need, that is, 21 for all 21 keyword classes. Therefore, we conclude the k-means algorithm is suitable for our use case.

However, in our version, we have a well-defined small set of keywords that do not overlap. Sentences without a keyword are ignored. Nonetheless, the new version of seed keywords has a wide range of available keywords per class, so this will be a problem for future work. Still, we implemented a small method that can be used for future work using k-means, as explained in the methodology chapter.

Another problem where clustering can be utilized is assigning a context window to the corresponding keyword class and finding the keyword. That means to essentially classify the context windows back to classes and detect the keyword. Again, our method can be used for this task as explained in the methodology chapter.

3. Related Work

Using context windows to classify a text into categories is a difficult problem statement in the context of spam detection, fake news detection, and similar. Older research only focuses on given classifications like is-spam and non-spam and fails to incorporate the context the words are in.

Nevertheless, recent research acknowledges the need for context in classification problems, and in this chapter, we will highlight notable research and methods.

3.1. Span Detection

Detecting context windows can be redefined in detecting the start and end of a span. Joshi et al.[31] introduced SpanBERT which is a self-supervised pre-training method that uses BERT to predict spans of text. They repeatedly outperformed BERT on tasks like question answering and coreference resolution. The authors introduced the auxiliary objective SBO which tries to use only the representations of the tokens at the span's boundary to predict the span.

Sung et al.[32] created a multi-task stacked Bi-LSTM to detect antecedents and consequent conditional statements. Similarly, Pavlopoulos et al. [**Pavl**] work on detecting toxic spans.

3.2. Masking

An alternative approach to detecting a token window includes masking. Masking is the process of masking one or more tokens of a sentence and the model predicts the masked words. This can be used to mask a context window that needs to be predicted.

Sun et al. [33] created ERNIE which aims to learn entity-level masking and phrase-level masking.

MASS is a sequence-to-sequence pre-training method for encoder-decoder-based language generation proposed by Song et al. [34]. Given the remaining part of the sentence, MASS can generate the missing sentence fragment.

Chan et al. [35] presented KERMIT, an insertion-based framework that can generate text and sequences.

3.3. Classifier

Mugdha et al. [36] built a classifier for Fake News Detection in headlines in Bengali. They achieved using the Gaussian Naive Bayesian Classifier an accuracy of 87%. They used TF-IDF for feature extraction and an additional tree classifier.

Kundu et al. [37] developed a model for the identification of non-standard words (NSW) in the Bengali news corpus. For example, '1998' could be a year or a number. What is notable about their research is that the researchers have to create suitable context windows to classify the words correctly. However, the size of the context window depends on the semiotic classes and they struggle with the complexity of the Bengali language.

Instead of only considering words associated with abuse, Menini et al. [38] recognized that context is important to classify a word as abusive or not. The researchers compared annotations with context and without, and while context-aware training is more challenging than word classification based on simple matches, the researchers also concluded it is necessary for classifying certain text phrases as abusive or nonabusive.

Similarly, Masood et al. [39] aimed to classify tweets using past history. Their best-performing model (LSTM) uses all available features (like the past history of 24 hours), which emphasizes the need for context in classification tasks.

3.4. Evaluating Context Windows

Lison et al. [40] researched context windows dependent on the hyperparameters maximum windows size, weighting scheme, window position, and stop words removal. They trained continuous Skip-Gram models on two English-language corpora and evaluated the models on lexical similarity and analogy tasks. They concluded that the presence of cross-sentential context and right-side context windows leads to superior context windows.

4. Methodology

In this chapter, we will initially introduce basic terminology and the data set and then outline the pre-processing steps. We will introduce different manual approaches that we implemented and introduce machine learning approaches. We will talk about labeling the data set, feature extraction, and post-processing steps. Lastly, we will compare the context windows of all approaches on a straightforward example.

4.1. Terminology

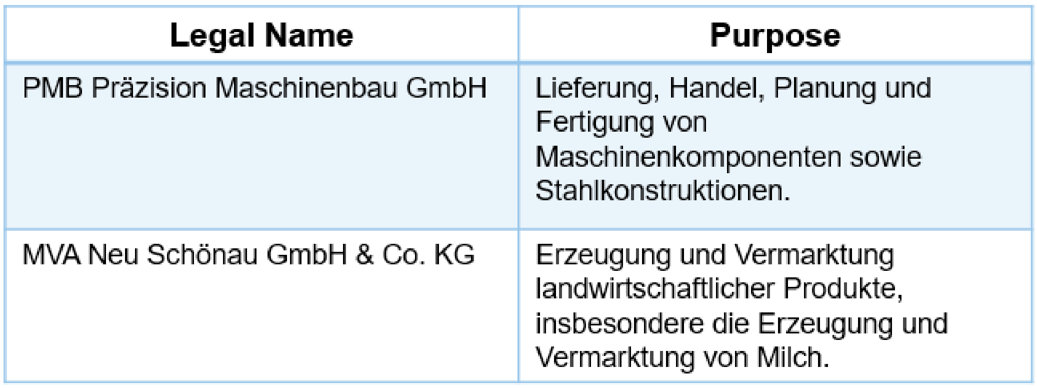
For this chapter, it is essential to establish certain terms:

- Token: A token is a word, punctuation, symbol, etc. in a sentence. We do not regard whitespaces as tokens.
- Stop words: Stop words are not meaningful tokens in a sentence. This can be punctuation, but also words like "aus", "gegen", "einer", "zum", "man", "ganz", … We will use the spaCy[1] collection of German stop words to identify stop words.
- Class: A class is a predefined set of items. In our thesis, we use the classes provided by the previous step in the pipeline, and each class contains keywords that are semantically related.
- Keyword: In our thesis, we refer to the specific words in the classes as keywords and matched keywords are keywords that are represented in the description but not in the classes. A matched keyword contains a keyword, for example, "Handelskammer" is the matched keyword of "Handel".

4.2. Data set

The data set German Business Registry consists of over 2,3 million entries, with varying descriptions of the purpose of the businesses. The descriptions are different lengths, so one challenge was to find context windows that cover enough context for lengthy sentences spanning over five lines as well as short sentences containing under 10 words. Figure 4.1 captures a small part of the German Business Registry data set.

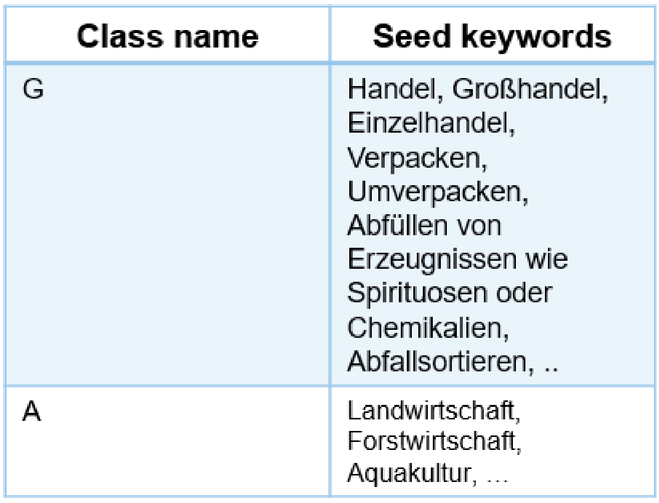
[1]*spaCy*. URL: `https://spacy.io/` (visited on 12/02/2023).



Legal Name	Purpose
PMB Präzision Maschinenbau GmbH	Lieferung, Handel, Planung und Fertigung von Maschinenkomponenten sowie Stahlkonstruktionen.
MVA Neu Schönau GmbH & Co. KG	Erzeugung und Vermarktung landwirtschaftlicher Produkte, insbesondere die Erzeugung und Vermarktung von Milch.

Figure 4.1.: Example of the German Business Registry data set

The seed keywords we use are created in the previous step of the pipeline of the Create-Data4AI project. They are classified into 21 classes, ranging from *A* to *U* with 15-20 keywords per class. We worked with a primary version of the keyword set, the final version contains a wider range of words than the version we used for training and evaluation. If the keyword consists of more than one word, we disregard the keyword. Figure 4.2 shows some examples of seed keywords.



Class name	Seed keywords
G	Handel, Großhandel, Einzelhandel, Verpacken, Umverpacken, Abfüllen von Erzeugnissen wie Spirituosen oder Chemikalien, Abfallsortieren, ..
A	Landwirtschaft, Forstwirtschaft, Aquakultur, ...

Figure 4.2.: Example of Seed Keywords

4.3. Pre-Processing

First, we have to examine the descriptions for keywords and match the corresponding description.

We accept not only exact matches, but also partial matches. For example, if the keyword is

Handel, then suitable matches would be *Handels*, *verhandeln* or *Einzelhandel* but not *Händler* or *Markt*. These matches with their corresponding sentences are collected in a data frame.

Further, we use the open-source library spaCy[2] to pre-process a given sentence. SpaCy removes white spaces, tokenizes the sentence, performs part of speech tagging on each token, parses the dependencies between tokens, and more.

As explained, our keywords and the matches are quite straightforward with no overlapping of keywords in classes. Therefore, we can simply use the whole sentence given the keyword for further processing. Nonetheless, in the next section, we explain how clustering can be utilized for overlapping classes.

4.3.1. Future Work

We implemented a small method that processes a sentence and creates features by counting how many times a keyword in this sentence belongs to a certain class.

For example, the sentence "Gegenstand des Unternehmens sind die Herstellung, die Verarbeitung, der Vertrieb und die Vermittlung von Formaten und Verpackungen aus Wellpappe und deren Zubehör sowie der Handel mit diesen Produkten und artverwandten Produkten sowie von Erzeugnissen zur Ergänzung oder Förderung dieser Tätigkeit." contains "Herstellung" which belongs to the class C and "Handel" which belongs to the class G. So this sentence contains one keyword of class B and one of class C.

For overlapping keywords, it may be necessary to create chunks of sentences instead of processing the whole sentence for the models to determine the correct context.

Manual approaches like the Dependency Approach or Naive Approach can be utilized to create these chunks.

Another possibility could be to create N-Grams of the sentences and filter these N-Grams whether they contain the keyword. To evaluate which approach is more suitable, more research and investigation is necessary.

In a jupyter notebook, we can process sentences or chunks with the help of our method, creating a vectorized dictionary of features and using k-means to cluster the sentences and assign a class to each sentence. We tested it on some examples of our data set and we got good assignments. This can be used to create a data set where sentences or parts of a sentence are assigned to a class. As mentioned, we did not use this for our approaches since it was not necessary, but it can be used for future work.

In the next sections, we describe our approaches after pre-processing.

[2]*spaCy*. URL: `https://spacy.io/` (visited on 12/02/2023).

4.4. Manual Approaches

4.4.1. Basic Approach

An intuitive approach to discover context windows would be to simply regard the x number of words beneath the keyword.

For example, the context window of the sentence "Lieferung, Handel, Planung und Fertigung von Maschinenkomponenten sowie Stahlkonstruktionen." with the keyword "Handel" would be "Lieferung, Handel, Planung und" with $x = 3$. Therefore, the context window contains two tokens to the left, the keyword, and three tokens to the right. As you can see in this example, the context window ending can be optimized by setting $x = 2$ and therefore removing the token "und".

We experimented with different context window sizes and first examined the keywords of the class *G* that contained keywords like *Handel*. By examining the context windows and the location of the keyword in the sentence we discovered that quite often the keyword is in the beginning of the sentence.

Hence it could make sense to not establish the context window x words left and right of the keyword, but to adopt a more flexible approach and shift the windows to one word on the left side of the keyword and three words on the right side.

However, this phenomenon does not consistently occur with the other keyword classes, for example for the class *K* which consists of keywords around the topic insurance, the keyword is often in the middle of the sentence or near the end.

Another problem with this approach is the descriptions are of very varying lengths. So by simply regarding the x words around the keyword, for lengthy sentences only a small part of the sentence is covered, and therefore, it is not guaranteed that enough context will be captured. On the other hand, for short sentences, this could mean that the whole sentence is included in the context window. Therefore, we conclude this approach is not very suitable for our use case.

4.4.2. Naive Approach

The open-source library *spaCy*[3] offers powerful tools to pre-process a sentence and tokenize the sentence, obtain the *Tag of Speech* of each token, obtain the noun phrases, and many more. For our use case, we will regard the dependency parser of *spaCy*.

First, the dependency parser of *spaCy* determines the root word of a sentence, then in a tree-like structure each word may have children connected with their respective head by dependency arcs. These arcs are labeled by the term *dep* which describes the syntactic relation between head and children. Each token may have children but it always has a head with the

[3]*spaCy*. URL: `https://spacy.io/` (visited on 12/02/2023).

head of the root being itself.

A simplification of the naive approach can be summarized in the following algorithm (2):

Algorithm 2 The Naive Approach Algorithm

```
$(matched\_keyword, description) \leftarrow find\_matches(seed\_keyword, dataframe)$
$context\_window \leftarrow$ 2 tokens beneath the keyword
for each token in context_window do
    $context\_window \leftarrow$ append children and head of token
end for
```

Note that in the proposed algorithm (2), we purely consider the immediate head and children of each token in the context window. We also experimented with considering including the children of the children and so on (i.e. doing more than one "jump") but this frequently lead to the same context window.

So the context window of the sentence "Lieferung, Handel, Planung und Fertigung von Maschinenkomponenten sowie Stahlkonstruktionen." with the keyword "Handel" would be "Lieferung, Handel, Planung und Fertigung" with three tokens beneath the keyword as maximum jump. This is a better context window than the baseline approach, since it does not end with a conjunction and contains the whole context of the sentence. An overview of the context windows of this sentence by all approaches is also shown at the end of this chapter. Additionally, figure 4.3 shows the dependencies of the proposed sentence. The token "von" is not included in the window since it would exceed the three tokens beneath the keyword limit.

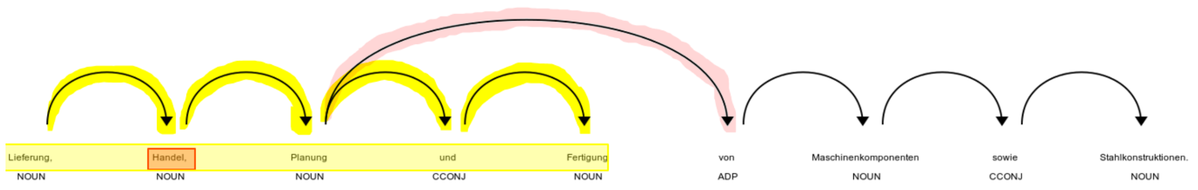


Figure 4.3.: Example of a context window with the Naive Approach

This mechanism turns out to be quite useful to overcome the problem of the difference in the sentence lengths we experienced in the data set. So especially for long sentences, even words that are further away from the keyword can be considered if there is some sort of syntactic relationship to the keyword. Simultaneously, short sentences can have a relatively small context window compared to the sentence size.

By further examining the context windows, we see that the tokens that are already included by the first jump have seldom further children that are not yet included.

This is also a result of the problem of how to find the optimal initial number of neighbors that should be directly included in the context window. There is nevertheless a problem that the context windows of short descriptions are too big and of long descriptions too small.

4.4.3. Dependency Approach

One solution to overcome the problem described in the Naive approach is to omit the number of neighbors considered in the context window altogether.

Therefore, instead of having a fixed window size, we can fix the number of jumps. Now we can establish a first context window dependent on the sentence length since large sentences tend to have their furthest children further away than short sentences.

This approach is described in the algorithm (3):

Algorithm 3 The Dependency Approach Algorithm

```
$(matched\_keyword, description) \leftarrow find\_matches(seed\_keyword, dataframe)$
$context\_window \leftarrow$ head and children of keyword
for each token in context_window do
    if token is a stop word then
        $context\_window \leftarrow$ append children and head of token
    end if
end for
$context\_window \leftarrow$ all tokens between the borders of the context window
```

So the context window of the sentence "Lieferung, Handel, Planung und Fertigung von Maschinenkomponenten sowie Stahlkonstruktionen." with the keyword "Handel" would be "Lieferung, Handel, Planung" with only one maximum jump. Increasing the jumps in this sentence to reach the token "Fertigung" would yield a significantly big context window since we use the jump to the token "von" which would be not optimal, as shown in figure 4.4.

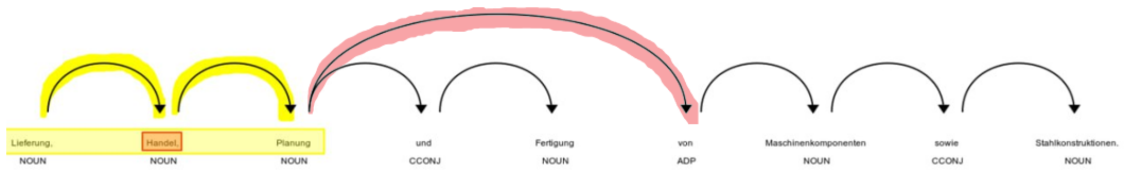


Figure 4.4.: Example of a context window with the Dependency Approach

We additionally inspect more context windows more closely and conclude that increasing the number of jumps does often not yield to different context windows, since a context window already consists of all tokens between the both farthest tokens. An increase of jumps naturally leads to small jumps in the middle of the context window and the farthest tokens often are reachable within one jump and do not have further children.

Nevertheless, with considering only one jump there may remain some cases where the jump "stops" at not meaningful tokens, such as commas and colons, i.e. stop words. Therefore, when the farthest token is a stop word, we jump another time. Note that *spaCy* already sorts the children of a token by their index.

4.5. Machine-Learning Approaches

As explained in the background chapter of this thesis, context windows can also be discovered by Machine Learning Algorithms. In our case, we will use supervised machine learning methods since we classify each token as "in context window" or "not in context window", i.e. as ′1′ or ′0′.

Labeling

As we use supervised Machine Learning Approaches, we require labeled data to train on. For this purpose, we use the keyword classes *A*, *B*, *C*, *D* and *G* and extract 150 descriptions per class. Additionally, we extract 50 descriptions of the class *K*.

We use *Prodigy*[4] to label each description with the keyword, the context window, the start of the context window, and the end of the context window. We will use these annotations to create an ideal label we can later use for training.

Approaches

There are different approaches possible on how exactly the models should predict a context window.

The first approach we implemented is to predict the start and the end of the context window. This approach yielded rather obscure context windows because of the variety of the data set and "optimal" context windows. Some sentences are extremely long and have therefore long context windows, and some are relatively short with short context windows, however, that may cover almost the whole sentence. In addition, the keyword may be at the beginning or at the end. Overall, this approach turned out to be not optimal given our data. By only predicting the start and the end, we have just 2 tokens that get classified in one class, and all other tokens are classified in another class. Machine Learning models struggle to learn the start and end because of the lack of different features of many tokens and the varying data.

The second approach is rather a classification problem where we classify each token as whether it is a part of the context window or not. Now, many tokens can be classified in one class which leads to more features for the machine learning model to learn. By experimenting and observing some example data, we conclude that this approach is more suitable for our data set.

[4]*Prodigy*. URL: `https://prodi.gy/` (visited on 12/02/2023).

4.5.1. Feature Extraction

For creating the data set we later train on, it makes sense to not consider the text of a word, but rather its features like the dependency label of the token, the index of the token in the text, how far the token is comparing to the keyword and so on. This also lead to a better generalization.

Therefore, the first step is to find meaningful features and create a data set. In the following, we list the features we utilized:

- dep_: Syntactic dependency
- pos: Part of Speech
- tag: Detailed Part of Speech Tag
- n_lefts: Number of tokens that are to the left of the token
- n_rights: Number of tokens that are to the right of the token
- is_stop: Is the token a stop word?
- is_start: Does the token start a sentence?
- is_end: Does the token end a sentence?
- is_ascii: Does the token consist of ascii characters?
- is_alpha: Does the token consist of alphabetic characters?
- is_lower: Is the token text in lowercase?
- is_child: Is the token a child of the keyword?
- is_head: Is the token the head of the keyword?
- sentence_length: Length of the sentence
- index_of_keyword: Index of the keyword
- distance_to_keyword: Euclidean distance between keyword and token
- child: Index of the token's first child
- head: Index of the token's head
- shape: Shape of the token (like xxxx, XXX)
- in_sentence: Is the token in the same sentence as the token?

To feed the features in a Machine Learning Model, we have to transform the features into a numerical representation using Dictvectorizer of the scikit-learn library.

For each approach, we mix the data set and use a split of 90% training data and 10% validation data.

4.5.2. Random Forest Approach

For the first approach, we use the Random Forest Classifier of the scikit-learn library. The algorithm uses decision tree classifiers on various sub-samples of the data set and averages the results.

We train the Random Forest Classifier on the training set and inspect some context windows. Additionally, we inspect the features of the Random Forest Classifier. As expected, some features are simply not as important for context windows classification as other features.

In a decision tree, a feature is more important than another feature if it decreases the impurity strongly. This can be averaged and a final importance can be computed. We can set a threshold, for example 0.0115, and merely consider the features above the threshold. For better comparisons in later steps, we can print kept features and include an additional method that eliminates withhold features in future data set pre-processing. A visualization of the feature's importance is shown in figure 4.5.

The most important features are distance to keyword, sentence length, index of keyword, is child, child, head, is head and in sentence. The least important features contain different shapes and tags.

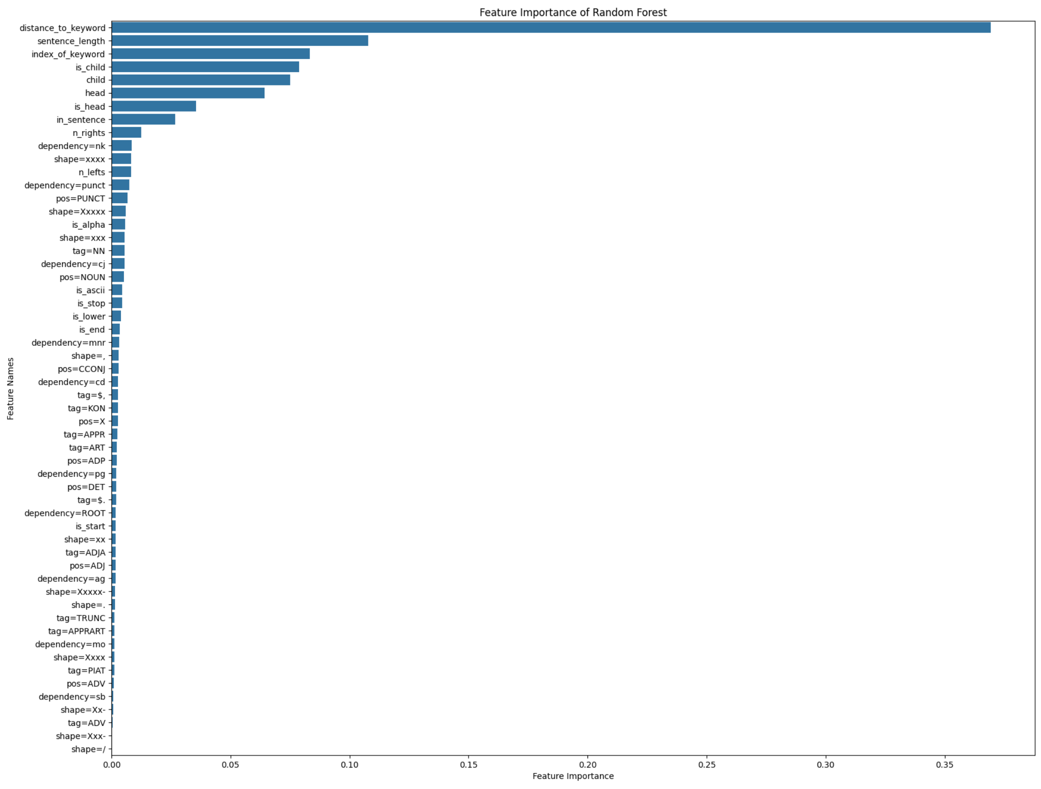


Figure 4.5.: Feature Importance of the Random Forest Classifier

Further, we optimize the validation set by using randomized search on hyper parameters by scikit-learn. We use varying numbers of estimators from 50 to 1000 and additionally vary the maximal depth from 1 to 150.

The randomized search includes a 5-fold cross-validation and yields an optimal number of estimators of 700 and a max depth of each tree of 10. By using these hyperparameters, we achieve a mean cross-validated score of 0.932 on the validation set.

4.5.3. XGBoost Approach

Similar to the Random Forest Classifier, XGBoost uses decision trees as a learning algorithm combined with gradient boosting.

Once more, we train our model on the training set and optimize on the validation set using a randomized search on hyperparameters by scikit-learn. The hyperparameter distribution includes varying maximal depth from 1 to 150, learning rates from 0.1 to 0.0001, sub-samples from 0.5 to 1, and a number of estimators from 50 to 1000.

By inspection of the feature importance, we discover they are similar to the Random Forest Approach with several unimportant features that we can remove, the remaining features are shown in figure 4.6.

The most important features are distance to keyword, part of speech tag punct, dependency punct, index of keyword, is child, in sentence and several tags and pos.

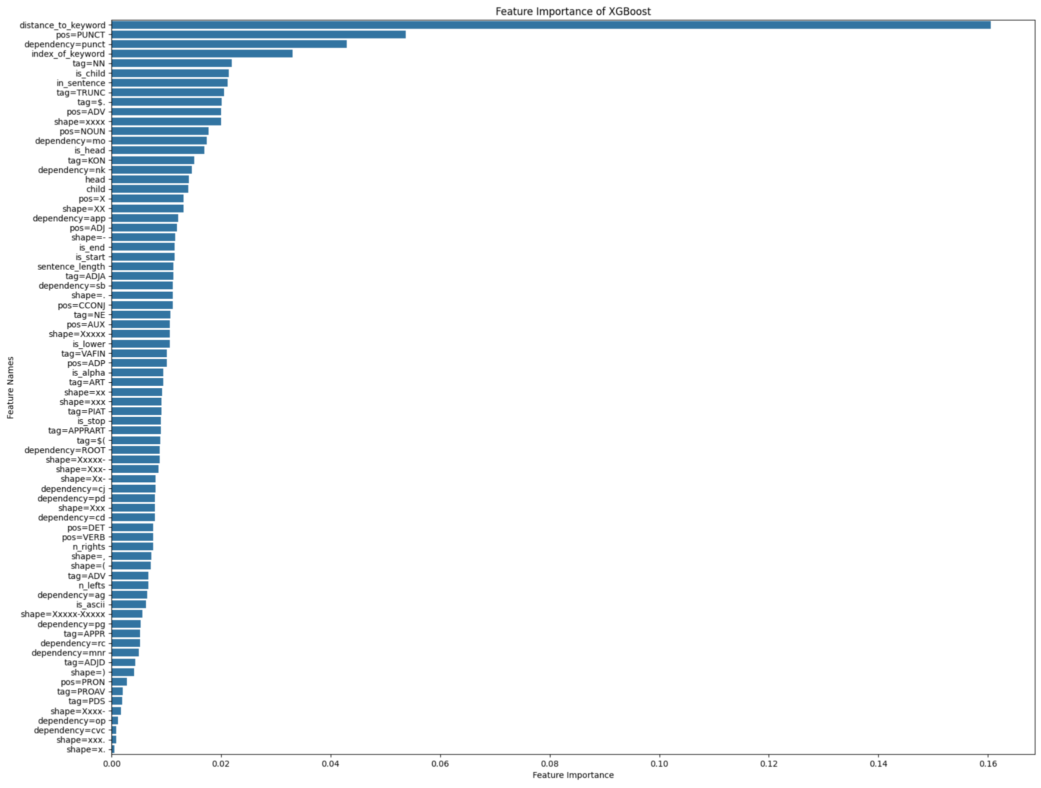


Figure 4.6.: Feature Importance of the XGBoost Classifier

The best set of hyperparameters includes a subsample of 0.7, a number of estimators of 100, a maximal depth of 80, and a learning rate of 0.01. Using these hyperparameters, we achieved a mean cross-validated score of 0.934 on the validation set.

4.5.4. Gaussian Approach

The second approach uses the Gaussian Classifier of the scikit-learn library based on Laplace approximation. We use the default kernel "1.0 * RBF(1.0)" with RBF being the Radial basis function kernel as described in (4.1) by Sreenivasa [43].

$$K(x, x') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{4.1}$$

Additionally, we adapt the features by using the permutation importance of the scikit-learn library and calculate the mean of each feature. As a result, many features are not relevant and some features decrease the accuracy of the predictions. The remaining features and their mean are shown in figure 4.7.

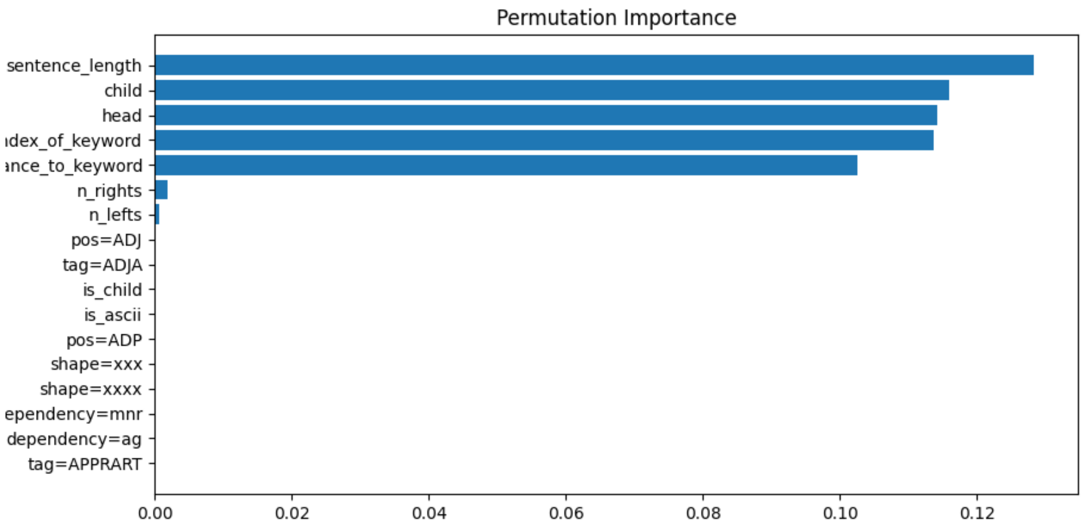


Figure 4.7.: Feature Importance of the Gaussian Classifier

4.6. Post-Processing

Although the scores on the validation sets are already quite high, by further inspection of the data we see there are, nonetheless, some problems occurring:
For example, the Gaussian Approach does sometimes not include the keyword in the context window when tested on a very different data set. In addition, some context windows contain "holes", so a single token in the middle may not be categorized as part of the context window, for example, if the token contains a very specific character that not many descriptions have, like "§". For these purposes, we decided to implement the following post-processing steps:

1. If the model did not predict anything, the context window includes just the keyword.
2. If the model did not include the keyword in the context window, extend the context window to include the keyword.
3. If there are holes in the predicted context window, find the largest chains and merge them.
4. If the context window does not start and end with a *NOUN*, then search the 3 tokens beneath to find the nearest *NOUN* and include it in the context window.

These steps ensure we will always include the keyword in the predicted context window and yield overall better context windows.

4.7. Classifying context windows back to keywords

Another problem where clustering can be utilized is assigning a context window to the corresponding keyword class and finding the keyword. So essentially to classify the context windows back to classes and detect the keyword.

Our created method from future tasks can be used and finding the keyword can be done by a simple matching similar to what we used for finding the keyword in a sentence once the context window is assigned to a certain class. We tested this for a few examples and the results are promising, but as mentioned for the newest version of the keywords the method may need improvements.

4.8. Overview of the context windows of all approaches on a straightforward example

To conclude this chapter, we will propose a small overview over the context window of the sentence "Lieferung, Handel, Planung und Fertigung von Maschinenkomponenten sowie Stahlkonstruktionen." with the keyword "Handel" in table 4.1.

Basic Approach with x=3	Lieferung, Handel, Planung und
Naive Approach with x=3	Lieferung, Handel, Planung und Fertigung
Dependency Approach with 1 jump	Lieferung, Handel, Planung
Random Forest Approach	Lieferung, Handel, Planung und Fertigung
XGBoost Approach	Handel, Planung
Gaussian Approach	Lieferung, Handel, Planung und Fertigung von Maschinenkomponenten sowie Stahlkonstruktionen

Table 4.1.: Overview of all approaches on a straightforward example

As you can see in 4.1, the context window of the Gaussian Approach is very large and contains unnecessary tokens. Whereas the context window of XGBoost is very small containing only 3 tokens. The manual approaches yield decent context windows as well as the Random Forest Approach.

5. Results

In this chapter, we will discuss the results of our different approaches using our hand-labeled data set of the keyword classes *A*, *B*, *C*, *D*, and *G*. We used 150 descriptions for each class. In the following, we will call a manual labeled data set that matches keywords of the class *A* to *D* but not *G*, *G data set*. Similar for the respective classes. We will use these datasets to evaluate the approaches.

Additionally, for each approach, we will evaluate the context windows on the 50 hand-labeled data of the keyword set *K*, in the following called *full_dataset*.

For evaluation, we will use the metrics Precision (P), Recall (R), and F1. The formulas are given in equation 5.1 with TP being True Positive, FP being False Positive and FN being False Negative.

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \\ F1 = \frac{2*Precision*Recall}{Precision+Recall} &= \frac{2*TP}{2*TP+FP+FN} \end{aligned} \tag{5.1}$$

However, having a high F1, recall, and the precision score does not necessarily mean the predicted context windows are optimal. For this purpose, we will additionally conduct a survey where human experts will select the most appropriate context window for some examples.

5.1. Test Data set

Regarding the manual approaches, we predict the context windows of the respective data set and compare these with the labeled context windows.

For each data set, we will evaluate the machine learning models on a test data set. The results are shown in table 5.1.

The XGBoost Approach yields the second-highest precision and a similar F1 score as RandomForest. Regarding the manual approaches, the Basic and Naive Approach yield very similar results whereas the Dependency approach yields the worst precision but a higher F1 score than the Basic and Naive Approach.

Dataset	Validation			Test		
Model	P	R	F1	P	R	F1
Basic	72.45	48.92	58.40	72.49	52.24	60.72
Naive	72.45	48.92	58.40	72.49	52.23	60.72
Dependency	55.15	73.82	63.13	52.99	69.72	60.22
RandomForest	75.36	56.59	64.64	73.39	60.55	66.36
XGBoost	79.17	54.23	64.37	79.05	60.34	68.44
Gaussian	52.16	46.78	49.32	51.24	57.14	54.03

Table 5.1.: Results on the test and validation data set

Additionally, we evaluate each approach on the training set, as shown in table 5.2. The Gaussian Approach yields very high scores overall despite not achieving high results in the validation and test set.

Model	P	R	F1
Basic	73.10	49.22	58.83
Naiv	73.10	49.22	58.83
Dep	59.05	74.34	65.82
RandomForest	82.19	63.90	71.90
XGBoost	87.72	96.63	73.76
Gaussian	91.32	95.81	93.41

Table 5.2.: Results on the training data set

Further, we evaluate whether the post-processing does improve the results for the machine learning models, the results on the validation set are shown in table 5.3.

PP denotes using post-processing and we shorten RandomForest to RF to improve readability.

Table 5.3 shows that the the RandomForest Approach and the XGBoost Approach perform regarding the F1 score slightly better with the post-processing step included whereas the Gaussian Approach has the same result.

The RandomForest Approach and the XGBoost Approach both have a significantly lower precision and a considerably higher recall with post-processing.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	75.36	84.61	79.17	88.63	52.16	52.15
R	56.59	50.64	54.23	43.09	46.78	46.78
F1	64.64	63.36	64.37	57.99	49.32	49.32

Table 5.3.: Results on the validation data set with and without post-processing

The results of the test data set are shown in table 5.4.

Table 5.4 shows that the RandomForest Approach performs slightly better without post-processing regarding the F1 score, whereas the Gaussian and the XGBoost Approach both perform better with post-processing.

The RandomForest Approach and the XGBoost Approach both have a significantly lower precision and a considerably higher recall with post-processing whereas the Gaussian Approach has a higher precision and recall with the post-processing step included.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	73.39	82.67	79.05	87.02	51.24	49.79
R	60.55	58.00	60.34	52.88	57.14	50.53
F1	66.36	68.17	68.44	65.78	54.03	50.16

Table 5.4.: Results on the test data set with and without post-processing

5.2. Cross Validation Data set

For the manual approaches, we will proceed as before.

Since we use the data sets for training for the machine learning approaches, we will adapt the models to get a correct evaluation. We will train the models on the data sets with a train/val split of 90% / 10% holding out the respective data set.

For example for data set A, the models are trained on the data sets B, C, D, and G, then we will evaluate the data set A.

For the machine-learning models, we use post-processing to further enhance the prediction.

5.2.1. Evaluation on data set A

Table 5.5 shows the results of evaluating the data set A. The Gaussian Classifier yields very bad results with an F1 score of only 45.88% whereas the XGBoost achieves an F1 score of 71.50%.

Model	Basic	Naive	Dependency	RandomForest	XGBoost	Gaussian
P	65.50	65.50	66.46	64.78	73.89	39.53
R	63.46	63.46	74.29	69.42	69.28	54.67
F1	64.47	64.47	70.16	67.02	71.50	45.88

Table 5.5.: Results on data set A

To compare if the post-processing did improve the predictions of the context windows, we additionally run the models with and without post-processing, as shown in table 5.6.

The RandomForest Approach performs slightly better (F1 score 69.92%) without the post-processing step, whereas the Gaussian Approach and the XGBoost Approach perform worse without the post-processing regarding the F1 score.

The RandomForest Approach and the XGBoost Approach both have a significantly lower precision and a considerably higher recall with post-processing. Whereas for the Gaussian Approach, the precision and the recall are higher, with the recall being significantly higher with post-processing.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	64.78	75.71	73.89	97.26	39.53	39.01
R	69.42	64.95	69.28	52.78	54.67	47.09
F1	67.02	69.92	71.50	68.42	45.88	42.67

Table 5.6.: Results on data set A with and without post-processing

5.2.2. Evaluation on data set B

The next table 5.7 shows the results of evaluating the data set B. Again, the Gaussian Classifier yields very bad results with an F1 score of only 48.73% whereas the XGBoost achieves an F1 score of 68.97%.

Model	Basic	Naive	Dependency	RandomForest	XGBoost	Gaussian
P	69.52	69.52	53.11	65.06	78.73	41.40
R	66.18	66.18	73.13	69.28	61.36	57.89
F1	67.81	67.81	61.53	67.10	68.97	48.73

Table 5.7.: Results on data set B

The difference of the machine learning models with and without post-processing is shown in table 5.8.

Again, the Random Forest Approach performs better without the post-processing step, whereas the XGBoost and Gaussian Approaches perform better with the post-processing step, regarding the F1 score.

Similar to data set A, the RandomForest Approach and the XGBoost Approach both have a significantly lower precision and a considerably higher recall with post-processing for data set B. Whereas for the Gaussian Approach, the precision and the recall are higher, with the recall being significantly higher with post-processing.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	65.06	79.35	78.73	86.13	41.40	39.96
R	69.28	61.63	61.368	51.47	57.89	50.80
F1	67.10	69.38	68.97	64.44	48.73	44.73

Table 5.8.: Results on data set B with and without post-processing

5.2.3. Evaluation on data set C

The evaluation of the data set C is shown in table 5.9. The Gaussian Classifier now performs better with an F1 score of 65.59%, outperforming the XGBoost Classifier which scores an F1 score of 62.87%. Regardless, the RandomForest Classifier achieves an F1 score of 70%. The Basic and Naive Approach on the other hand only achieve a F1 score of 55.85%.

Model	Basic	Naive	Dependency	RandomForest	XGBoost	Gaussian
P	85.00	85.00	50.56	81.95	82.78	69.70
R	41.60	41.60	82.42	61.10	50.69	61.93
F1	55.85	55.85	62.67	70.00	62.87	65.59

Table 5.9.: Results on data set C

In table 5.10 the difference of the machine learning models with and without post-processing is shown.

For data set C, the Random Forest and the Gaussian Approach perform better on the F1 score with the post-processing step whereas the XGBoost has a lower F1 score with the post-processing step.

Adding the post-processing step leads to a higher precision for the RandomForest and the Gaussian Approach and a lower precision for the Gaussian Approach.

The recall on the other hand is higher without the post-processing step for all approaches.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	81.95	72.90	82.78	86.13	69.70	67.75
R	61.10	64.60	50.69	51.47	61.93	62.46
F1	70.00	68.50	62.87	64.44	65.59	64.53

Table 5.10.: Results on data set C with and without post-processing

5.2.4. Evaluation on data set D

The Gaussian Classifier performs again only 45.15% on the F1 score for the data set D, as shown in table 5.11. The XGBoost Classifier slightly outperforms the RandomForest Classifier with an F1 score of 67.78%.

Model	Basic	Naive	Dependency	RandomForest	XGBoost	Gaussian
P	68.45	68.45	63.29	75.04	80.99	43.17
R	58.79	57.79	75.03	60.40	58.26	47.32
F1	62.67	62.67	68.66	66.93	67.78	45.15

Table 5.11.: Results on data set D

Table 5.12 shows the difference of the machine learning models with and without post-processing.

Regarding the F1 score on data set D, all approaches perform better with the post-processing step.

The XGBoost Approach scores significantly higher in precision with the post-processing step, the Gaussian Approach slightly higher, and the XGBoost Approach lower.

The XGBoost and the Gaussian approaches both have a significantly higher recall with the post-processing step whereas the RandomForest Approach has a slightly higher recall.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	75.04	70.47	80.99	85.15	43.17	41.89
R	60.40	60.99	58.26	46.37	47.32	41.14
F1	66.93	65.39	67.78	60.05	45.15	41.51

Table 5.12.: Results on data set D with and without post-processing

5.2.5. Evaluation on data set G

The results of the last data set G are shown in table 5.13. The Gaussian Classifier performs a bit better than the other classes with an F1 score of 53.43%, outperforming the XGBoost and the RandomForest Classifier achieves only 50.64% and 55.68% each.

Model	Basic	Naive	Dependency	RandomForest	XGBoost	Gaussian
P	78.02	78.02	64.31	78.33	79.66	64.80
R	35.31	35.31	67.36	43.19	37.12	45.46
F1	48.62	48.62	65.80	55.68	50.64	53.43

Table 5.13.: Results on data set G

For data set G, the table 5.12 shows the difference of the machine learning models with and without post-processing.

Similar to data sets A and B, the Random Forest Approach performs better without the post-processing step, whereas the XGBoost and Gaussian Approaches perform better with the post-processing step, regarding the F1 score.

Again mirroring the behavior of data sets A and B, the RandomForest Approach and the XGBoost Approach both have a lower precision and a higher recall with post-processing for the data set B. Similar trends are visible in the Gaussian Approach.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	78.33	83.93	79.66	88.69	64.80	66.84
R	43.19	42.19	37.12	32.44	45.46	43.86
F1	55.68	56.15	50.64	47.57	53.43	52.96

Table 5.14.: Results on data set G with and without post-processing

5.3. Evaluation on data set K

To have a final evaluation of these approaches, it could make sense to compute the mean and standard deviation of each data set.

Or, we could annotate a small sample of another data set and evaluate this data set. This resembles the actual use case the most and we get the additional advantage of increasing the training data.

So for evaluating the K data set, we will train the models on all classes A, B, C, D, and G with a train/val split of 90% / 10% and evaluate on 50 data samples of class K.

The results are shown in table 5.15. The Gaussian Classifier performs the worst with an F1 score of 50.74% and the XGBoost performs the best with an F1 score of 77.61%. The Dependency Approach has a slightly worse F1 score of 68.42% than the Basic and Naive Approach which scores 70.67%.

Model	Basic	Naive	Dependency	RandomForest	XGBoost	Gaussian
P	64.90	64.90	62.15	70.43	79.19	40.71
R	77.56	77.56	76.10	77.56	76.10	67.32
F1	70.67	70.67	68.42	74.48	77.61	50.74

Table 5.15.: Results on data set K

The last table 5.16 shows the difference of the machine learning models with and without post-processing.

For the data set K, we trained the models with data sets A, B, C, D, and G and tested on the data set K.

Despite leading the post-processing step for some data sets and approaches (mainly the RandomForest Approach) to worse results than without post-processing, this does not seem the case with the data set K.

Here, the post-processing leads to overall better precision, recall, and F1 values with the highest difference of the F1 score with nearly 7% for the XGBoost Approach, a better precision of 9% for the XGBoost Approach, and a better recall of approximately 9% for the RandomForest Approach.

Model	RF PP	RF	XGBoost PP	XGBoost	Gaussian PP	Gaussian
P	70.43	67.79	79.19	70.34	40.71	39.41
R	77.56	68.78	76.10	70.73	67.32	59.02
F1	74.48	68.28	77.61	70.56	50.74	47.27

Table 5.16.: Results on data set K with and without post-processing

5.4. Expert Evaluation

Scoring metrics like precision, recall and the F1 score have only limited informative value on how effective and coherent the context windows are. Consulting human experts is crucial to have a final evaluation of the quality of context windows.

For the expert evaluation, we extracted five random samples of each data set. We calculate for each approach the context windows of each sample and summarize the results in an Excel sheet. Additionally, we calculate for the approaches Basic Approach and Naive Approach x=2 as well as x=3 with one jump each. A short excerpt of the Excel sheet is shown in figure B.1 in the appendix B.

From this overview, we extract 18 sentences and present two possible context windows. For each context window, we ask if it represents a suitable context window for the presented sentence and also, which context window is preferred. In the following sections, we will present the results.

Please note that these results are primarily answered by non-experts as we have not yet collected enough answers from experts.

First, the questionnaire explains the task and what the experts should evaluate. The experts should especially pay attention to these three requirements:

- The context window should contain the keyword which is highlighted by bold writing.
- The context window should make sense grammatically.
- The context window should be representative of classes of WZ2008.

We collected three answers in total as of today. The first context window is generated by the RandomForest Approach, and the second context window by the Gaussian Approach, both with post-processing. For some sentences, the context windows are the same.

The experts were asked if the first context window (generated by RandomForest) was suitable as a first question and then if the second context window (generated by Gaussian) was suitable as a second question. For the third question, the experts should specify if they think the first context window is more suitable, the second, both, or neither. This is repeated for all 18 sentences.

An example sentence is: "Handel mit sowie die Entwicklung, Verarbeitung und Produktion von Papier, papierähnlichen Produkten, Kunststoffen, Verpackung und EDV-Zubehör sowie alle damit zusammenhängenden Tätigkeiten im weitesten Umfang."

Possible context windows are:
"Verarbeitung und Produktion" and "Handel mit sowie die Entwicklung, Verarbeitung und Produktion von Papier".

Figure 5.1 shows the answers to whether context windows generated by the RandomForest Approach are suitable or not. Out of 54 possible answers, the experts answered 17 times they were suitable (31.5%) and 37 were not suitable (68.5%).

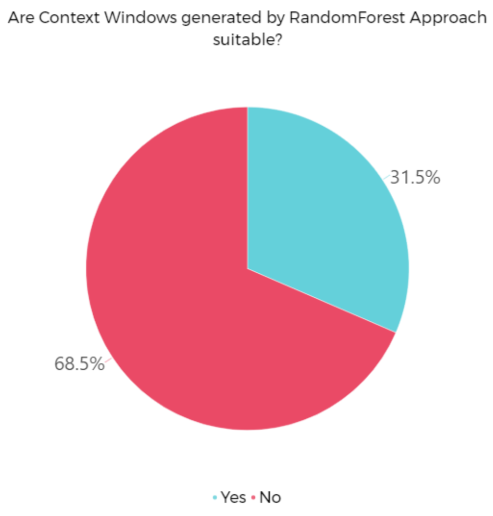


Figure 5.1.: Results of Expert Evaluation on RandomForest Appraoch

Similarly, figure 5.2 shows the answers to whether context windows generated by the Gaussian Approach are suitable or not. Out of 54 possible answers, the experts answered 32 times they are suitable (59.3%) and 22 they are not suitable (40.7%).

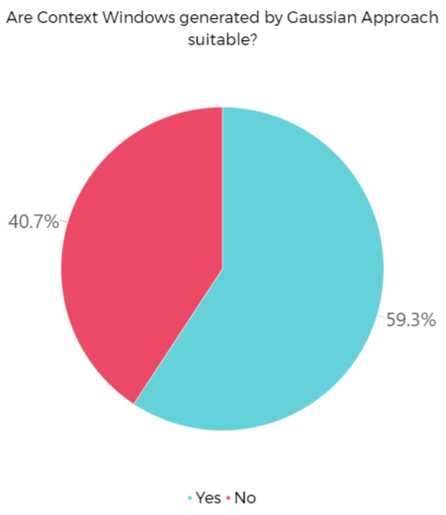


Figure 5.2.: Results of Expert Evaluation on Gaussian Appraoch

The last figure 5.3 shows the result of which approaches generate suitable context windows. Out of 54 answers, the experts answered 5 times the RandomForest Approach generates suitable context windows (9.3%), 15 times the Gaussian Approach (27.8%), 18 times neither (29.6%) and 16 times both (33.3%).

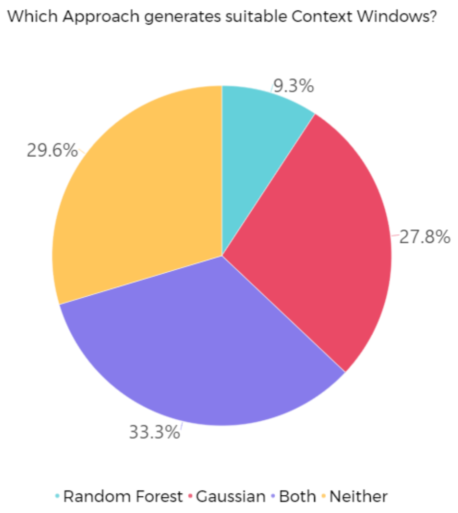


Figure 5.3.: Overall Results of Expert Evaluation

6. Discussion

This chapter discusses the results of our thesis, mentions the challenges encountered and proposes some approaches for future work.

6.1. Results

The Basic and the Naive Approach tend to have the same context window and therefore the same precision, recall and F1 score. Since the Naive Approach is an extension of the Basic Approach, this makes sense. The Naive Approach simply adds a dependency jump to expand the context window. But since we chose x=3 as a size, the context window is with 7 tokens already big enough to capture a possible jump.

One could add more jumps, but spaCy already sorts the children of a token from furthest away to nearest, so one dependency jump already jumps to the furthest token. Tokens within the already defined context window by the Basic Approach tend to be highly interconnected so one jump tends to be enough to jump to the furthest point and additional jumps do not alter the context window.

We could choose a smaller x and more jumps to create a more significant difference between these two approaches but this leads to the same context windows as before for the Naive Approach and worse context windows for the Basic Approach. Setting a higher x tends to create too big context windows, especially for the Naive Approach since the jumps tend to jump to almost the end of the sentence.

So we conclude that the Naive Approach does not improve the Basic Approach significantly.

The Dependency Approach on the other hand leads to a better recall, and worse precision but overall to a better F1 score than the Basic and Naive Approaches. That is, the Dependency Approach first jumps one time and when the reached token is a stop word, we jump again. Even with configuring only one jump, this leads to very big context windows in some cases, as explained before. This creates bad precision while maintaining a good recall.

The Machine Learning Approaches tend to score better in precision, recall and F1. Noticeable is the bad performance of the Gaussian Approach when the test set is not in the same distribution with a very good performance on the training set. This is explainable with the function of the Gaussian Classifier. Since it assumes a normal distribution, outliers have a low probability. Our data set can have, depending on the keyword class, very different features, like the exact position of the keyword in a sentence.

By examining the resulting context windows of the Gaussian Approach further, we discover that the context windows are either suitable for including all relevant information, or no context window was detected at all. Of course, this behavior drags down the metrics. The Gaussian Approach does have higher scores on the data set C and when examining the sentences of data set C further, we discover that many keywords are at the beginning of the sentence. Since the Gaussian Approach tends to create longer context windows, for data sets where the keyword is in the middle of the sentence this could lead to too big context windows. Whereas for sentences where the keyword is at the beginning of a sentence, the Gaussian Approach does not include too many tokens which leads to more accurate windows.

The XGBoost Approach tends to have a higher precision and a lower recall than the RandomForest Approach. XGBoost optimizes misclassified predictions by retraining the last tree that performed not well. Overall, the F1 score of the XGBoost Classifier is sometimes worse and sometimes better than the RandomForest Classifier. Evaluating the data set K, XGBoost has a higher F1 score.

The post-processing step tends to lower the precision and increase recall, meaning it increases the chance to include tokens in context windows even if the tokens may not belong to context windows. This could be further improved to include only meaningful tokens.

We experimented with leaving the last step of the post-processing out, i.e. do not necessarily end the context window with a *NOUN* by searching the three next tokens for a *NOUN*. This was not successful as the precision is still lower than without post-processing but now the F1 score is worse, i.e. the recall is less increased.

Still, on the data set K, including the post-processing increases recall, precision and F1 score, meaning that with more training and more data, the post-processing is more valuable. One explanation is that with increasingly diverse data, the chance for "holes" in the context window is increased since some classes tend to have their keyword in the beginning and others at the end. Post-processing eliminates these holes. Additionally, with more training the context windows will likely already end with a *NOUN*, meaning the post-processing step will not include unnecessary tokens.

We do not have a significant number of participants for the expert evaluation. Additionally, the answers were given by non-experts. Therefore, these conclusions are only preliminary. Nonetheless, we can see a tendency for the Gaussian Approach instead of the RandomForest Approach. The context windows generated by the Gaussian Approach tend to be longer and in more detail than the ones generated by the RandomForest Approach. Therefore, we can conclude that experts tend to prefer longer context windows.

For most of our labeled data, we preferred shorter context windows over longer ones. The data set consists of many enumerations of the tasks of a company which means that keywords that belong to different classes are quite near each other. For example, a significant number of descriptions describe the company that manufactures a product and trades with it. Manufacture ("Herstellung") and trade ("Handel") belong to the classes C and G respectively.

Therefore, we preferred shorter context windows when labeling the data, whereas the experts do not know which keywords may overlap and prefer longer context windows.

Further, the percentage of whether both context windows are preferred (33.3%) or neither (29.6%) is quite similar, with a slightly higher percentage of both context windows. We presume that we simply did not yet get enough feedback and with more reports, a more solid trend will be visible.

All in all, we conclude that our approaches especially the machine learning approaches are suitable for finding context windows around class-specific keywords. Through the post-processing step, we ensure that the context windows are cohesive and always include the keyword. With enough training, the Gaussian Approach may lead to better results and generate suitable context windows. The RandomForest and XGBoost Approaches on the other hand do not always predict optimal results, but more solid results overall.
Further, we can successfully detect the keywords given a context window using k-means, as investigated on a small part of the data. Still, this may change when using the newest version of the keywords.

6.2. Challenges

One of the major challenges we encountered was the fact that the data set consists of descriptions of very varying lengths and features. For manual approaches as well as machine learning models it was not trivial to find a suitable size of the context window. Even when manually labeling the data, for some sentences the context window could span over almost the whole sentence whereas some other descriptions are 5 sentences long.

Further regarding the data set, for very long sentences multiple keywords of diverse classes could match. We paid attention to manually label different sentences, but when scaling for the whole data set this could represent a problem for some machine learning models. A solution could be to assume that a context window should not contain multiple sentences and therefore, trim the description to contain only the same sentence the keyword is in. The problem is, that the German language tends to use very long-running sentences and some sentences are still too long. The data set contains many enumerations of different tasks of a company that can span several lines.

Another challenge was the keyword was located more at the beginning, at the middle, or at the end of the sentence, depending on the class. The Gaussian Classifier had difficulties even finding the keyword, resulting in often empty context windows that we post-process with to at least include the keyword. That, of course, dragged down the accuracy. So for some sentences, the Gaussian Classifier could predict very good context windows whereas for other sentences, the approach predicted an empty window.

Regarding the expert evaluation, we simply do not have enough responses. Additionally, larger context windows tend to be prioritized over smaller context windows. In the survey, we did not include the possibility of multiple keywords in a sentence although our data set often contains keywords belonging to different classes that are near each other like in an enumeration. This could be an improvement for future work, as explained in the next section.

6.3. Future Work

For this thesis, we developed several approaches for classifying whether a token is in the context or not. Another approach could be to find the start and the end of the context window by classifying whether a token is the start or end or not. Since our descriptions have very varying lengths, we concluded that this approach may be not suitable for our data but future work could include further comparing and evaluating the approaches and looking for alternatives.

We mainly used centered context windows for the manual approaches. By examining the data more closely, we discover that significant context can be more to the left or more to the right of the keyword depending on the class. We concluded that therefore, a center-based approach would cover most cases, and we often jump to the right to cover more relevant context. One could calculate the mean number of tokens on the right and left side of the keyword that capture the context window and experiment with different numbers as the initial window size. On the other hand, machine learning models do not require a predefined number of tokens for each side and have already good results. Nonetheless, one could compare an optimized version of the manual approaches with the machine learning models.

We merely implemented some basic machine learning models, so developing actual deep learning models like Long Short-Term Memory (LSTM) and others could lead to much higher results. Another notable approach could be to use semi-supervised learning, using the labeled data as input and an unsupervised approach to scale for unseen data.

Additionally, labeling a subset of all 21 classes could lead to different results since some of the descriptions of the classes are quite different in length, the location of the keyword, and more. For example, comparing the results of data set C in table 5.9, we observe the Gaussian Approach has a much higher F1 score than in other classes, as explained in the results section.

We used a premature release of the keywords for training, utilizing the newest version that also contains a very wide range of keywords could lead to different results, especially regarding the feature selection.

As for the pre- and post-processing, future work could include employing alternative methods of clustering and also improving the post-processing depending on the models.

The expert evaluation survey can be further improved to contain multiple keywords in a sentence and include options that contain overlapping context windows as well as short context windows.

7. Conclusion

This thesis develops several approaches for finding context windows given sentences and keywords and constitutes the second part of the pipeline of the CreateData4AI project [1].

We researched approaches extensively for Word Sense Disambiguation and provided an overview of well-established techniques as well as state-of-the-art approaches and answered the first research question.

By first extensively researching suitable clustering techniques, we develop pre-processing steps including classifying each description of the German Business Registry data set using k-means algorithm according to the keyword classes. This concludes the second research question.

As a subsequent step, we developed several approaches regarding extracting context windows.

This includes manual approaches like the Basic Approach, Naive Approach and Dependency Approach as well as several Machine Learning Approaches namely RandomForest Approach, XGBoost Approach and Gaussian Approach. While the Gaussian Approach scores very well on seen data, i.e. the training set, it is not suitable for the variety we deal with in the test data set. The XGBoost Approach and the RandomForest Approach score better overall given the data, whereas in expert evaluation the Gaussian Approach is preferred as it leads to optimal windows but does not always predict windows.

These developed approaches are suitable for trimming context windows for text chunks and whole sentences, which answers the third research question.

We manually labeled 800 descriptions extracted by the German Business Registry date set and compared the classification approach of each token with the detection of the start and end of the context window. While we did not implement the approach of detecting the start and the end of the context window fitting for our data because of the variety, it could be interesting to develop this approach further. Additionally, labeling more data to get a subset of all classes could lead to different results, since the descriptions of some classes differ strongly from other classes.

Further, we extracted suitable features for each Machine Learning Approach and developed post-processing steps to further improve the context windows. We kept in mind that for future models, these can be adapted to ensure optimal feature selection and post-processing.

Using the manually labeled descriptions for creating data sets, we use cross-validation to evaluate each approach and additionally, label 50 descriptions of another class which we use for a final evaluation. Additionally, we used an expert survey to analyze the context windows created by the RandomForest Approach and the Gaussian Approach. Although we did not collect enough responses, the Gaussian Approach seems to be preferred. This concludes the last research question.

Finally, we use k-means to classify the context windows according to the keyword classes and match the corresponding keyword of a context window. While generating satisfying results, this may need adaptation for the newest version of the keywords.

A. Repository Link

The repository can be accessed using the following link:
https://gitlab.lrz.de/CreateData4AI/masterthesis-alexandra-seibicke

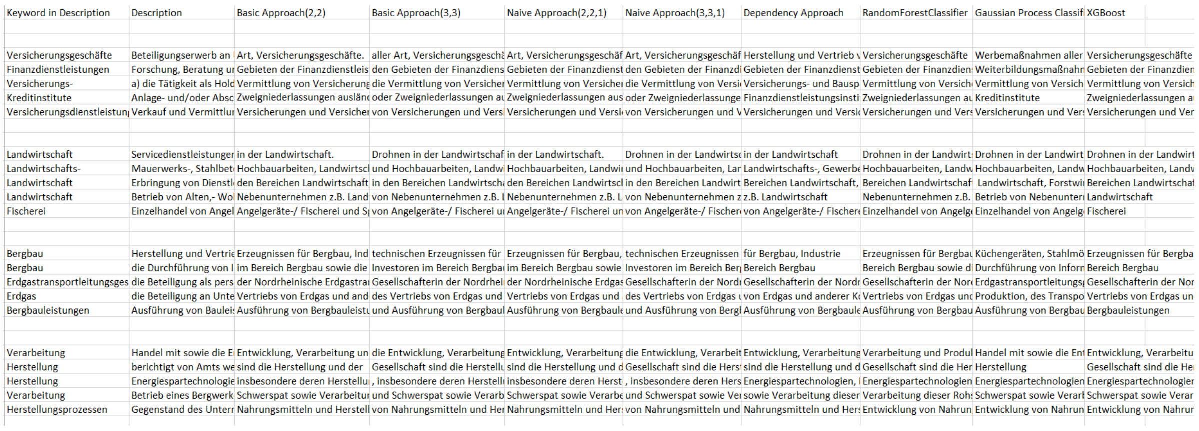
B. Excerpt of the Overview of all Approaches on five samples

For each data set A, B, C, D, G and K, we extract five random samples and calculate the context windows of each approach.

The data sets are used the same as in the previous chapters explained, for example for data set A this means the models are trained on B, C, D and G and evaluated on A.

Data set K is never used for training, only for evaluating.

The figure B.1 shows an excerpt of the context windows.



Keyword in Description	Description	Basic Approach(2,2)	Basic Approach(3,3)	Naive Approach(2,2,1)	Naive Approach(3,3,1)	Dependency Approach	RandomForestClassifier	Gaussian Process Classifi	XGBoost
Versicherungsgeschäfte	Beteiligungserwerb an	Art, Versicherungsgeschäfte.	aller Art, Versicherungsgeschä	Art, Versicherungsgeschäf	Art, Versicherungsgeschäf	Herstellung und Vertrieb v	Versicherungsgeschäfte	Werbemaßnahmen aller	Versicherungsgeschäfte
Finanzdienstleistungen	Forschung, Beratung ur	Gebieten der Finanzdienstleis	den Gebieten der Finanzdiens	Gebieten der Finanzdienst	den Gebieten der Finanzdi	Gebieten der Finanzdienst	Gebieten der Finanzdien:	Weiterbildungsmaßnahn	Gebieten der Finanzdien
Versicherungs-	a) die Tätigkeit als Hold	Vermittlung von Versicherung	die Vermittlung von Versicher	Vermittlung von Versicher	die Vermittlung von Versi	Versicherungs- und Bausp	Vermittlung von Versich	Vermittlung von Versich	Vermittlung von Versich
Kreditinstitute	Anlage- und/oder Absc	Zweigniederlassungen auslän	oder Zweigniederlassungen au	Zweigniederlassungen aus	oder Zweigniederlassunge	Finanzdienstleistungsinsti	Zweigniederlassungen au	Kreditinstitute	Zweigniederlassungen a
Versicherungsdienstleistun	Verkauf und Vermittlur	Versicherungen und Versicher	von Versicherungen und Versi	Versicherungen und Versi	von Versicherungen und V	Versicherungen und Versi	Versicherungen und Vers	Versicherungen und Vers	Versicherungen und Ver
Landwirtschaft	Servicedienstleistunger	in der Landwirtschaft.	Drohnen in der Landwirtschaf	in der Landwirtschaft.	Drohnen in der Landwirts	in der Landwirtschaft	Drohnen in der Landwirt:	Drohnen in der Landwirt	Drohnen in der Landwirt
Landwirtschafts-	Mauerwerks-, Stahlbet	Hochbauarbeiten, Landwirtsc	und Hochbauarbeiten, Landwi	Hochbauarbeiten, Landwi	und Hochbauarbeiten, Lar	Landwirtschafts-, Gewerbe	Hochbauarbeiten, Landw	Hochbauarbeiten, Landw	Hochbauarbeiten, Landv
Landwirtschaft	Erbringung von Dienstl	den Bereichen Landwirtschaft	in den Bereichen Landwirtscha	den Bereichen Landwirtsc	in den Bereichen Landwirt	Bereichen Landwirtschaft,	Bereichen Landwirtschaf	Landwirtschaft, Forstwi	Bereichen Landwirtschaf
Landwirtschaft	Betrieb von Alten,- Wol	Nebenunternehmen z.B. Land	von Nebenunternehmen z.B. L	Nebenunternehmen z.B. L	von Nebenunternehmen z	z.B. Landwirtschaft	Nebenunternehmen z.B.	Betrieb von Nebenunter	Landwirtschaft
Fischerei	Einzelhandel von Angel	Angelgeräte-/ Fischerei und Sp	von Angelgeräte-/ Fischerei u	Angelgeräte-/ Fischerei un	von Angelgeräte-/ Fischer	von Angelgeräte-/ Fischere	Einzelhandel von Angelge	Einzelhandel von Angelg	Fischerei
Bergbau	Herstellung und Vertrie	Erzeugnissen für Bergbau, Ind	technischen Erzeugnissen für	Erzeugnissen für Bergbau,	technischen Erzeugnissen	für Bergbau, Industrie	Erzeugnissen für Bergbau	Küchengeräten, Stahlmö	Erzeugnissen für Bergba
Bergbau	die Durchführung von I	im Bereich Bergbau sowie die	Investoren im Bereich Bergbau	im Bereich Bergbau sowie	Investoren im Bereich Ber	Bereich Bergbau	Bereich Bergbau sowie di	Durchführung von Inforn	Bereich Bergbau
Erdgastransportleitungsges	die Beteiligung als pers	der Nordrheinische Erdgastra	Gesellschafterin der Nordrhei	der Nordrheinische Erdgas	Gesellschafterin der Nordr	Gesellschafterin der Nordr	Gesellschafterin der Nor	Erdgastransportleitungsg	Gesellschafterin der Nor
Erdgas	die Beteiligung an Unte	Vertriebs von Erdgas und and	des Vertriebs von Erdgas und	Vertriebs von Erdgas und	des Vertriebs von Erdgas u	von Erdgas und anderer K	Vertriebs von Erdgas unc	Produktion, des Transpo	Vertriebs von Erdgas un
Bergbauleistungen	Ausführung von Baulei:	Ausführung von Bergbauleistu	und Ausführung von Bergbaul	Ausführung von Bergbaule	und Ausführung von Bergl	Ausführung von Bergbaule	Ausführung von Bergbau	Ausführung von Bergbau	Bergbauleistungen
Verarbeitung	Handel mit sowie die E	Entwicklung, Verarbeitung un	die Entwicklung, Verarbeitung	Entwicklung, Verarbeitung	die Entwicklung, Verarbeit	Entwicklung, Verarbeitung	Verarbeitung und Produl	Handel mit sowie die Ent	Entwicklung, Verarbeitu
Herstellung	berichtigt von Amts we	sind die Herstellung und der	Gesellschaft sind die Herstellu	sind die Herstellung und d	Gesellschaft sind die Herst	sind die Herstellung und d	Gesellschaft sind die Her	Herstellung	Gesellschaft sind die Her
Herstellung	Energiespartechnologie	insbesondere deren Herstellu	, insbesondere deren Herstell	insbesondere deren Herst	, insbesondere deren Hers	Energiespartechnologien, i	Energiespartechnologien	Energiespartechnologien	Energiespartechnologier
Verarbeitung	Betrieb eines Bergwerk	Schwerspat sowie Verarbeitur	und Schwerspat sowie Verarb	Schwerspat sowie Verarbe	und Schwerspat sowie Ve	sowie Verarbeitung dieser	Verarbeitung dieser Roh:	Schwerspat sowie Verarl	Schwerspat sowie Verar
Herstellungsprozessen	Gegenstand des Unterr	Nahrungsmitteln und Herstell	von Nahrungsmitteln und Her	Nahrungsmitteln und Her:	von Nahrungsmitteln und	Nahrungsmitteln und Her:	Entwicklung von Nahrun	Entwicklung von Nahrun	Entwicklung von Nahrur

Figure B.1.: Excerpt of the Overview of all Approaches on five samples

List of Figures

List of Tables

Bibliography

[1] *CreateData4AI*. URL: https://wwwmatthes.in.tum.de/pages/nqpi6qljq0x9/CreateData4AI-CD4AI (visited on 12/07/2023).

[2] R. Navigli. "Word Sense Disambiguation: A Survey". In: *ACM Comput. Surv.* 41.2 (Feb. 2009). ISSN: 0360-0300. DOI: 10.1145/1459352.1459355. URL: https://doi.org/10.1145/1459352.1459355.

[3] J. C. Mallery. "Thinking About Foreign Policy: Finding an Appropriate Role for Artificially Intelligent Computers". In: 1988. URL: https://api.semanticscholar.org/CorpusID:107849961.

[4] G. A. Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[5] M. Lesk. "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone". In: *Proceedings of the 5th annual international conference on Systems documentation*. 1986, pp. 24–26.

[6] *Lesk's Algorithm: A Method for Word Sense Disambiguation in Text Analytics*. URL: https://towardsdatascience.com/lesks-algorithm-a-method-for-word-sense-disambiguation-in-text-analytics-52c157a2fdff (visited on 12/07/2023).

[7] F. Viveros-Jiménez, A. Gelbukh, and G. Sidorov. "Simple window selection strategies for the simplified lesk algorithm for word sense disambiguation". In: *Mexican International Conference on Artificial Intelligence*. Springer. 2013, pp. 217–227.

[8] R. Rada, H. Mili, E. Bicknell, and M. Blettner. "Development and application of a metric on semantic nets". In: *IEEE transactions on systems, man, and cybernetics* 19.1 (1989), pp. 17–30.

[9] K. Mittal and A. Jain. "WORD SENSE DISAMBIGUATION METHOD USING SEMANTIC SIMILARITY MEASURES AND OWA OPERATOR." In: *ICTACT Journal on Soft Computing* 5.2 (2015).

[10] V. Gujjar, N. Mago, R. Kumari, S. Patel, N. Chintalapudi, and G. Battineni. "A Literature Survey on Word Sense Disambiguation for the Hindi Language". In: *Information* 14.9 (2023). ISSN: 2078-2489. DOI: 10.3390/info14090495. URL: https://www.mdpi.com/2078-2489/14/9/495.

[11] J. Q. Walker. "A node-positioning algorithm for general trees". In: *Software: Practice and Experience* 20.7 (1990), pp. 685–705.

[12] M. Galley, K. R. McKeown, E. Fosler-Lussier, and H. Jing. “Discourse Segmentation of Multi-Party Conversation”. In: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan: Association for Computational Linguistics, July 2003, pp. 562–569. DOI: 10.3115/1075096.1075167. URL: https://aclanthology.org/P03-1071.

[13] R. L. Rivest. “Learning Decision Lists”. In: *Sponsored Paper, NSF Grant DCR-8607494, ARO Grant DAAL03-86-K-0171 ve Siemens Corporation* (2001).

[14] C. Kingsford and S. L. Salzberg. “What are decision trees?” In: *Nature biotechnology* 26.9 (2008), pp. 1011–1013.

[15] *Gaussian Naive Bayes Explained With Scikit-Learn*. URL: https://builtin.com/artificial-intelligence/gaussian-naive-bayes (visited on 12/02/2023).

[16] *A Friendly Introduction to [Deep] Neural Networks*. URL: https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks (visited on 12/02/2023).

[17] *NLP with CNNs*. URL: https://towardsdatascience.com/nlp-with-cnns-a6aa743bdc1e (visited on 12/02/2023).

[18] *Recurrent Neural Networks and Natural Language Processing*. URL: https://towardsdatascience.com/recurrent-neural-networks-and-natural-language-processing-73af640c2aa1 (visited on 12/02/2023).

[19] T. Martín Wanton and R. Berlanga Llavori. “A clustering-based approach for unsupervised word sense disambiguation”. In: (2012).

[20] R. Navigli and S. P. Ponzetto. “BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network”. In: *Artificial intelligence* 193 (2012), pp. 217–250.

[21] M. Bevilacqua, T. Pasini, A. Raganato, and R. Navigli. “Recent Trends in Word Sense Disambiguation: A Survey”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Z.-H. Zhou. Survey Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4330–4338. DOI: 10.24963/ijcai.2021/593. URL: https://doi.org/10.24963/ijcai.2021/593.

[22] G. A. Miller, C. Leacock, R. Tengi, and R. T. Bunker. “A semantic concordance”. In: *Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*. 1993.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[24] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. “Improving language understanding by generative pre-training”. In: (2018).

[25] J. Jia, W. Liang, and Y. Liang. *A Review of Hybrid and Ensemble in Deep Learning for Natural Language Processing*. 2023. arXiv: 2312.05589 [cs.AI].

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[27] P. Rai and S. Singh. "A survey of clustering techniques". In: *International Journal of Computer Applications* 7.12 (2010), pp. 1–5.

[28] *8 Clustering Algorithms in Machine Learning that All Data Scientists Should Know*. URL: `https://www.freecodecamp.org/news/8-clustering-algorithms-in-machine-learning-that-all-data-scientists-should-know/` (visited on 12/02/2023).

[29] K. Erk and S. Padó. "Exemplar-based models for word meaning in context". In: *Proceedings of the acl 2010 conference short papers*. 2010, pp. 92–97.

[30] W. B. Dolan. "Word sense ambiguation: clustering related senses". In: *COLING 1994 Volume 2: The 15th International Conference on Computational Linguistics*. 1994.

[31] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. *SpanBERT: Improving Pre-training by Representing and Predicting Spans*. 2020. arXiv: `1907.10529 [cs.CL]`.

[32] M. Sung, P. Bagherzadeh, and S. Bergler. "CLaC at SemEval-2020 Task 5: Muli-task Stacked Bi-LSTMs". In: *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Ed. by A. Herbelot, X. Zhu, A. Palmer, N. Schneider, J. May, and E. Shutova. Barcelona (online): International Committee for Computational Linguistics, Dec. 2020, pp. 445–450. DOI: `10.18653/v1/2020.semeval-1.54`. URL: `https://aclanthology.org/2020.semeval-1.54`.

[33] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu. *ERNIE: Enhanced Representation through Knowledge Integration*. 2019. arXiv: `1904.09223 [cs.CL]`.

[34] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. *MASS: Masked Sequence to Sequence Pre-training for Language Generation*. 2019. arXiv: `1905.02450 [cs.CL]`.

[35] W. Chan, N. Kitaev, K. Guu, M. Stern, and J. Uszkoreit. *KERMIT: Generative Insertion-Based Modeling for Sequences*. 2019. arXiv: `1906.01604 [cs.CL]`.

[36] S. B. S. Mugdha, S. M. Ferdous, and A. Fahmin. "Evaluating Machine Learning Algorithms For Bengali Fake News Detection". In: *2020 23rd International Conference on Computer and Information Technology (ICCIT)*. 2020, pp. 1–6. DOI: `10.1109/ICCIT51783.2020.9392662`.

[37] C. Kundu, R. K. Das, and K. Sengupta. "Implementation of Context Window and Context Identification Array for Identification and Interpretation of Non Standard Word in Bengali News Corpus". In: *International Journal of Computational Linguistics Research* 4.4 (2013), pp. 159–171.

[38] S. Menini, A. P. Aprosio, and S. Tonelli. "Abuse is contextual, what about nlp? the role of context in abusive language annotation and detection". In: *arXiv preprint arXiv:2103.14916* (2021).

[39] M. A. Masood, R. A. Abbasi, and N. Wee Keong. "Context-Aware Sliding Window for Sentiment Classification". In: *IEEE Access* 8 (2020), pp. 4870–4884. DOI: 10.1109/ACCESS.2019.2963586.

[40] P. Lison and A. Kutuzov. "Redefining context windows for word embedding models: An experimental study". In: *arXiv preprint arXiv:1704.05781* (2017).

[41] *spaCy*. URL: https://spacy.io/ (visited on 12/02/2023).

[42] *Prodigy*. URL: https://prodi.gy/ (visited on 12/02/2023).

[43] *Radial Basis Function (RBF) Kernel: The Go-To Kernel*. URL: https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a (visited on 12/02/2023).