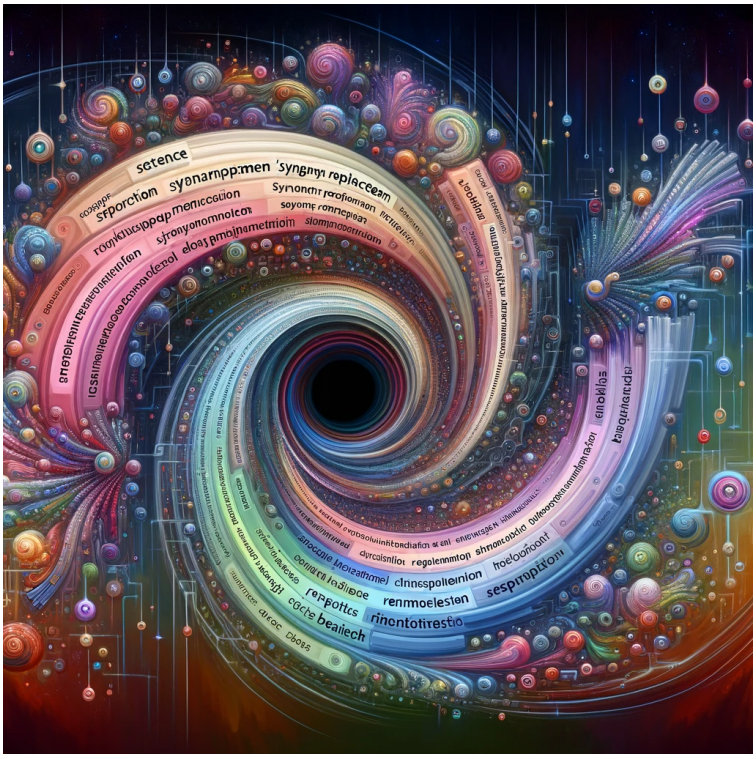
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Leveraging Data Augmentation for Better Named Entity Recognition in Low-Resource Settings

Master's thesis in Computer science and engineering

Philip Björnerud

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Leveraging Data Augmentation for Better Named Entity Recognition in Low-Resource Settings

Philip Björnerud

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Leveraging Data Augmentation for Better Named Entity Recognition in Low-Resource Settings
Philip Björnerud

Supervisor: Dana Dannélls, Department of Swedish, Multilingualism, Language Technology
Supervisor: Dimitrios Kokkinakis, Department of Swedish, Multilingualism, Language Technology
Examiner: Jean-Philippe Bernardy, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The illustration provides an abstract and artistic interpretation of Data augmentation in natural language processing. Created by DALL · E.

Typeset in LaTeX
Gothenburg, Sweden 2024

Leveraging Data Augmentation for Better Named Entity Recognition in Low-Resource Settings
Philip Björnerud
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis investigates the challenges in the field of Natural Language Processing (NLP), with a focus on Named Entity Recognition (NER), a subtask within NLP that involves classifying entities. Addressing the issue of data scarcity, which is particularly critical in non-English languages like Swedish, this study investigates various data augmentation methods by fine-tuning the transformer-based model, KB-BERT. The datasets are simulated as low-resource settings, drawing inspiration from the study X Dai and H Adel (2020) [1] work, using three sets of training data containing 50, 150, and 500 instances respectively. The thesis also explores whether a newly developed state-of-the-art data augmentation method can outperform other data augmentation methods in enhancing an NLP model, centering on three data augmentation methods: Synonym replacement, Mention replacement, and AugGPT, the last being a state-of-the-art method. The findings of this study highlight that synonym replacement emerged as the most effective data augmentation method across various low-resource settings, achieving the highest F1-score increase in all scenarios. AugGPT achieved the second highest average F1-score, while mention replacement achieved the lowest across the tested settings.

Keywords: Named Entity Recognition, Data Augmentation, Low-Resource Settings, Synonym Replacement, Mention Replacement, AugGPT.

Acknowledgements

I want to thank Dana Dannélls and Dimitrios Kokkinakis for all their help and guidance during this project. Their support and advice have been really important to me. Thank you both so much!

Philip Björnerud, Gothenburg, 2024-01-25

Contents

List of Figures

List of Tables

1

Introduction

The rapid growth of data, combined with the advancements in technology, has positioned Natural Language Processing (NLP) at the forefront of machine learning (ML) research. However, the quality of NLP models depends highly on the quality and quantity of available training data. This is because these models learn based on the patterns from the training data. This dependency poses a unique set of challenges for languages like Swedish, where the availability of annotated data is limited. With limited training data available, it can be challenging to train reliable NLP models that generalize well. One of the important NLP subtasks is Named Entity Recognition (NER)[2], which is a task that involves recognizing and classifying entities (e.g., persons, organizations, locations, medical terms, time expressions, etc) within a given sentence. NER is an important task for understanding the meaning of human language since the method not only helps in extracting data from large volumes of text but also aids in analyzing the text efficiently. The availability of training datasets significantly impacts the NER model's performance, particularly in low-resource environments, like newly developed domains. This is particularly challenging as many non-English languages and specific domains lack annotated data. Thus, finding ways which improve NER models in low-resource environments can make advanced language tools more available, and enable the development of new NLP applications across various domains.

There are several ways to mitigate the dependency on labeled data, one way is to use fine-tuned models, which are pre-trained models that are adjusted for specific tasks. Another approach has been to use methods such as data augmentation to enhance the ML model's effectiveness in NER tasks in low-resource environments. Interestingly, these two methods can be effectively combined, by using data augmentation to enrich the dataset before fine-tuning the model. However, utilizing data augmentation is an area of ongoing research in numerous NLP tasks [3]. Since, the area is constantly developing, where new data augmentation methods have emerged with the help of large language models.

To demonstrate the significance of data augmentation methods within low-resource settings, this study aims to evaluate the performance of both existing and newly developed state-of-the-art data augmentation methods. By simulating different sets of environments where annotated data is scarce, the study seeks to understand how effectively these augmentation methods can mitigate the challenges posed by limited data availability.

1.1 Problem

NLP tasks depend heavily on the quality and quantity of available training data. However, as mentioned in the introduction, this poses a set of challenges for low-resource languages like Swedish, where annotated data is very limited. Annotation can be very expensive and time-consuming and it is also required to have specialized knowledge, making it a costly and often tedious process [3]. The lack of annotated data can limit the potential of the NLP models. It has also been shown to be a problem for particular tasks like NER. Therefore, it is a noteworthy problem to address.

A potential solution to this issue is data augmentation, a method designed to expand and improve the datasets by creating variants of the existing data. While data augmentation is a promising approach, it is not without limitations. Some studies have pointed out that certain data augmentation methods can be ineffective, thereby it is important to also explore other state-of-the-art methods [4].

Thus, this thesis aims to address three problems:

1. Can various data augmentation methods, including modern, cutting-edge methods, effectively improve a NER model's performance, particularly in a low-resource setting?

2. Can a newly developed state-of-the-art data augmentation method demonstrate superior performance in enhancing an NER model over existing methods?

3. How do different data augmentation methods impact the NER model in different low-resource settings?

1.2 Purpose

This research explores whether data augmentation can enhance performance in low-resource settings with Swedish language datasets. Moreover, it has been inspired by the X Dai and H Adel (2020) [1] work, which demonstrated that simple augmentation methods can enhance the performance of transformer-based models, particularly for very small training sets, which simulated a low-resource setting environment.

To determine if data augmentation can improve performance, we will be analyzing subsets of the Swe-NERC Version 1 dataset [5]. The subsets will be selected following the approach proposed by X Dai and H Adel (2020) [1] but with a slightly modified approach. The three subsets will consist of 50, 150, and 500 sentences from the training set, which will be used to create small, medium, and large training sets.

For the three subgroups, we will use both traditional data augmentation methods, including synonym replacement and mention replacement, as well as modern method like AugGPT, which utilizes the capabilities of ChatGPT [6]. These methods will be used to determine whether augmentation can improve scenarios where the data

resource is very scarce, by applying these methods to the scarce dataset and then fine-tuning them separately with the same initial transformer model. To our knowledge, there exists no study where ChatGPT has been used as a data augmentation method for Swedish NER tasks. Thus, this is an interesting and unexplored area for our study. Should our study yield successful results, AugGPT could potentially emerge as a data augmentation method for low-resource languages such as Swedish.

The contributions of this thesis include:

1. An investigation on the effects of data augmentation methods in various low-resource scenarios, categorized as small (50 instances), medium(150 instances), and large(500 instances) to determine their effectiveness in enhancing a NER model.
2. A introduction of cutting-edge data augmentation method, such as AugGPT, into the context of Swedish NER, a research field that has not been applied before.
3. A study of how F1-score relates to cosine-similarity when augmenting data.
4. A study of various data augmentation methods on the Swe-NERC dataset, a 21st-century Swedish text.

1.3 Limitations

One of the limitations of this thesis is the scope of ML models and datasets explored. Due to the time constraints for this research, we have limited our study to use a single ML model, Swedish BERT (KB-BERT) [7], and only the Swe-NERC dataset. While this focused approach allows for a more in-depth analysis of the selected model and dataset, it also leaves room for questioning the generalizability of the results. Moreover, there exist endless data augmentation methods and model combinations that could potentially improve the robustness of the solutions we propose. Given the time limit of this work, we have to restrict the scope of what data augmentation methods and models to use.

In this study, we use ChatGPT version 3.5, a large language model developed by OpenAI, to augment data. However, it's important to note that while the results we have generated are unique and not fully reproducible in their exact form, similar outcomes can be achieved through repeated experimentation.

1.4 Ethical considerations and risks

One ethical element to consider is the possibility of biases in the dataset. Social media data often reflect the biases in society, which can propagated into the ML models. The data is open-sourced, which largely mitigates problems over data privacy and anonymization.

2

Theory

In this chapter, we will introduce the theory for our study, focusing on Transformer architecture, BERT, Named Entity Recognition, and Data augmentation.

2.1 Neural networks

An artificial neural network model is a simplified and generalized version of the brain's actual neurons, which in recent years have achieved a significant accomplishment in a broad range of domains, such as healthcare, business, education, and economics [8].

A neural network consists of three types of node layers, which include an input layer, at least one hidden layer, and an output layer. The input layer receives numerical data in the shape of feature vectors and passes it on to the hidden layers for processing. The functionality of the input layer is to work as an interface between the neural network and the inputs. Once the numerical data has left the input layer, it will pass through the hidden layers, where most of the computation occurs. The concept of the hidden layers is to detect the underlying features of the input data. When the data reaches the output layer, it will perform its last computation to produce an output, which is used for decision-making. [9]

Each node layer will apply a linear transformation and add a bias vector to the data that passes through from the previous layer. From this result, a non-linear transformation will be applied through a so-called activation function, which is an important feature of the architecture since it introduces non-linearity, and will help the network to capture complex patterns. There are different types of activation functions and the selection can influence the network performance since each is suited for different tasks and network architectures. One of the more common activation functions is Rectified Linear Units (ReLU) [10]. The equation is defined as $f(x) = \max(0, x)$, where if x (the input to the function) is positive or zero, the output is x itself, and if x is negative, the output is 0.

After feeding the input data, the network computes a prediction, and the difference between the output and the true output value is calculated using a loss function. The loss function computes the error, which denotes how distant the neural networks predictions are from the true values. The objective of the network is to minimize errors, which is often accomplished by something called backpropagation. During

the process of backpropagation, the gradients of the loss function for the model parameters are computed, which is later used to update the weights of the nodes. The entire process is called one training step [11].

During training, hyperparameters influence the structure of the network and how it learns, which affects the model's performance. Some common hyperparameters are learning rates, warm-up steps, early stopping rates, and batch sizes [12]. The learning rate affects how much the network updates its weights based on the loss gradient [13]. A learning rate that is too high may cause the model to overshoot the minimum loss value, preventing it from converging to the optimal solution. However, a too-low learning rate will cause the model to learn slowly, and possibly make the model not converge at all. Warm-up steps are an extension of the learning rate, which is designed to provide a model with an acceptable learning rate. It starts initially with a reduced learning rate, after these steps, the network then adopts the specified learning rate. Early stopping is a technique that is used as a security mechanism to ensure the model does not overfit. During training, early stopping monitors network performance and stops the learning when the model's performance stops improving [14]. Batch size is a hyperparameter that specifies the number of training examples used in one iteration. A smaller batch size often provides the network with lower generalization error, and a larger batch size helps the network to converge faster [15].

2.1.1 Attention

Traditional models within the NLP domain have often struggled with long sequences, while also maintaining context [16]. Attention mechanisms are a crucial tool within neural network-based NLP as the architecture allows the network to focus on relevant parts of input data for generating outputs [17]. Attention archives this by assigning varying degrees of importance to different parts of the input, allowing the network to focus on relevant parts and provide contextual attention. Figure 2.1 illustrates how a long short-term memory network (LSTMN) [18] processes the sentence "The FBI is chasing a criminal on the run" with the help of the attention mechanism. The red color indicates the word currently being processed, while the blue color marks previously processed words that the network considers relevant to the present word. The intensity of the blue color illustrates the degree of memory activation, illustrating how the model links the current word with its previously examined words [18].

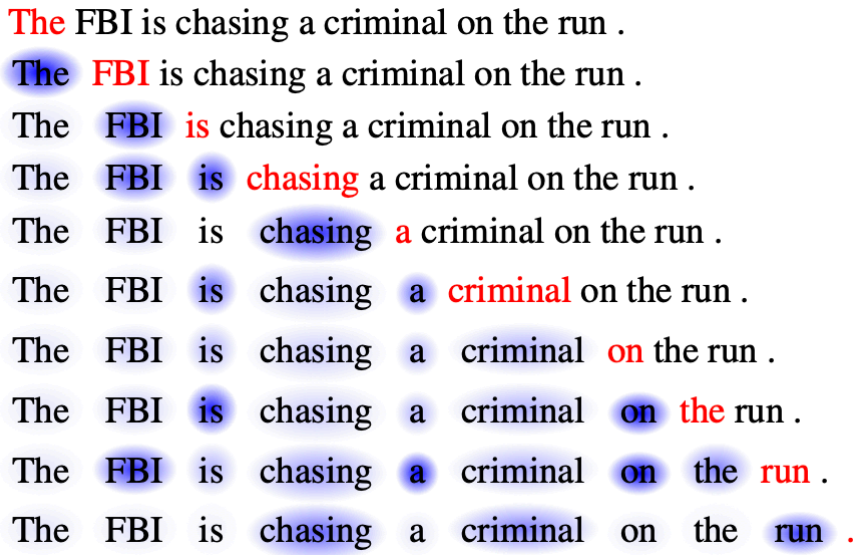


Figure 2.1: Example of an LSTM Network employing an attention mechanism to highlight relational memory activation. Credit to the paper [18]

The mechanism of attention involves the transformation of a query and a set of key-value pairs into an output. This output is then calculated as a weighted sum of the values, which serves to estimate the relevance of each value with respect to the given query context.

Self-attention [19] is an extension of the attention mechanism. The difference is that it further allows the model to consider the importance of some parts within a sequence. Thus, allowing the model to examine the relationship between the input sequence and the output sequence. In the context of self-attention mechanisms, the components are the query (Q), keys (K), and values (V). The attention mechanism is calculated as the dot product of the Q value and the K value, where the dot product can also be scaled, resulting to a particular attention mechanism called Scaled Dot-Product Attention [20]. The equation for the Scaled Dot-Product Attention can be seen at equation 2.1

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

In this equation, d_k is the dimensionality of the K vectors, and the dot product is scaled by $\sqrt{d_k}$, which is a feature of the Scaled Dot-Product Attention mechanism.

2.1.2 Transformers

The transformer architecture was introduced by Ashish Vaswani et al. [20] and since its introduction, transformer models have significantly improved the state-of-the-art applications for numerous NLP tasks. Transformers are a popular choice in deep learning due to their ability to leverage parallel processing, which lets the mecha-

nism handle entire sequences in parallel. This handling improves the computational efficiency.

The architecture of transformers is illustrated by Figure 2.2. In the processing of data, two primary components are used in the model, the encoder component, which is on the left of the architecture, and the decoder, which is on the right of the architecture. Each of these main components is made up of sub-components, which operate sequentially. The objective of the encoder component is to analyze the input data and store the relevant information in a memory for the decoder to use later. The encoder component contains multiple layers that work together to capture the relationships between the input data. In contrast, the decoder's primary role is to create an output sequence, using the context provided by the encoder. It does this by obtaining the output from the encoder and with its previous output [20].

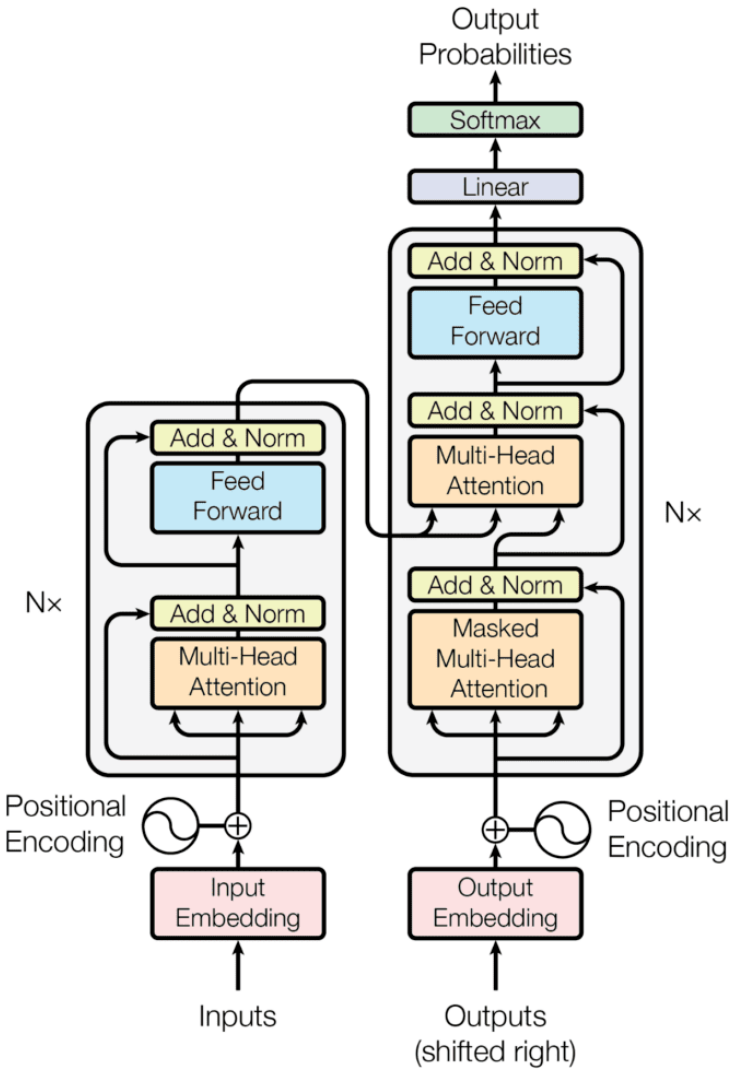


Figure 2.2: Transformers architecture

The encoder component consists of six layers. Each layer consists of two sub-layers,

a multi-head self-attention mechanism, and a position-wise fully connected feed-forward network. Each sub-layer is surrounded by a residual connection and a normalization layer, which results in outputs of dimension 512 [20]. The decoder component consists of six identical layers. However, it also includes an additional sub-layer for multi-head attention over the encoder's output. Additionally, the self-attention sub-layer in the decoder is modified to prevent forward-looking attention, which ensures the predictions only depend on previously known outputs [20]. The primary objective for creating this architecture was to solve the issue of past architectures being unable to maintain the strength of the pre-trained representations, particularly when using the fine-tuning method.

2.1.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model that was introduced by Google in 2018 [21]. The BERT model is developed using the transformer architecture, with the primary objective of managing the problem of previous architectures being unable to maintain the effectiveness of pre-trained representations when using the fine-tuning method. The main drawback with the previous models was due to the models being unidirectional, which limited the available options for pre-training architectures. Thus, those models were not optimal for two different tasks, tasks that were in sentence-level, and tasks that required context from both directions [21]. However, BERT tackles this problem by using something called a Masked language model (MLM). The technique functions by hiding selected tokens in the input at random, forcing the model to identify the original vocabulary of these masked tokens, which enhances the model's understanding of the sounding context [21].

Figure 2.3 illustrates the two main stages of BERT's training, the pre-training and the fine-tuning phase.

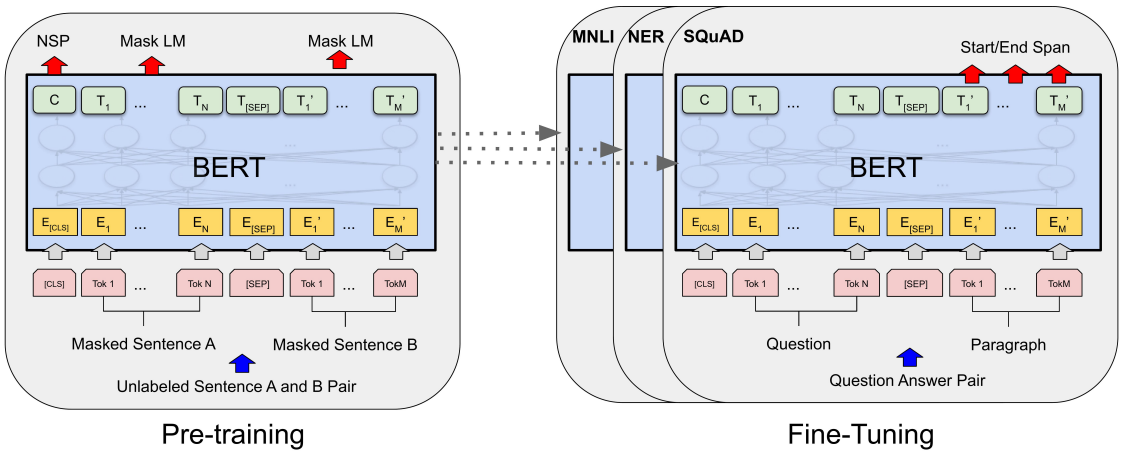


Figure 2.3: BERT's pretraining and the fine-tuning phase

During the pre-training phase, the BERT model will process unlabeled sentences with two methods. The first method is called the Masked Language Model. The MLM method will label certain words, which will later be masked, which will enable the model to predict words with the use of the surrounding context. This will strengthen BERT's ability to understand the surrounding context. The second method is called Next-sentence prediction (NSP). NSP is used to decide what relationship two sentences have to each other. In this approach, the BERT model develops an understanding of the context at the paragraph level, by analyzing the relationships between sentences. This will help the model to achieve a better understanding of the content. The left part of Figure 2.3 illustrates BERT's analysis of how two sentences are contextually related.

During the fine-tuning phase, which is illustrated in the right part of Figure 2.3, BERT is customized for particular NLP tasks and will optimize all parameters with the task-specific inputs and outputs. The pretraining phase is an unsupervised method, whereas the fine-tuning phase is supervised. Thus, throughout this phase, BERT requires labeled data since labeled data will provide important direction for the model's learning process during fine-tuning.

2.1.3.1 KB-BERT model

KB-BERT was developed by the National Library of Sweden's KBLab [7]. The model aimed to tackle the lack of dedicated resources specifically for the Swedish language. Since the dependency on the previous models like Googles multilingual M-BERT [1] and Arbetsförmedlingens model [2] was not fully optimized for the Swedish language.

The effectiveness of KB-BERT is based on the quality of its training data, which contains newspapers, reports, Swedish Wikipedia texts, and emails. As a result, the model encapsulates a wide range of elements of the Swedish language and the dataset ensures the model's understanding of various linguistic semantics.

KB-BERT uses the foundation of Google's BERT architecture, the training of the model was conducted on the TensorFlow Research Cloud, which allowed for extensive computational processing. The model underwent strict testing and benchmarking, particularly using NER and Part-Of-Speech tagging (POS), where it illustrated higher accuracy than the existing language models.

2.1.4 GPT

The Generative Pre-trained Transformer (GPT) architecture, was developed by OpenAI [6], and is fundamentally based on the transformer model. The model is pretrained on a wide range of texts scraped from the internet, which enables it to understand and produce human-like text. OpenAI has progressed through the GPT series, including GPT-1, GPT-2, GPT-3, GPT-4, and recently GPT-4 Turbo, which illustrates a continuous increase in the model's size and complexity of the algorithms.

[1] `https://github.com/google-research/bert/blob/master/multilingual.md`
[2] `https://github.com/af-ai-center/SweBERT`

ChatGPT, based on the GPT-3.5 and GPT-4 architecture, is a culmination of these developments. Unlike its predecessors, ChatGPT is fine-tuned specifically for conversational tasks. It uses both supervised learning and reinforcement learning from human feedback (RLHF)[6], a method in which its responses are more aligned with human-like responses. This fine-tuning process allows ChatGPT to demonstrate more contextual understanding.

2.2 Named Entity Recognition

Named Entity Recognition (NER) is a subfield of Natural Language Processing (NLP) that focuses on determining and classifying named entities in textual data. Some commonly used entities are names of people, organizations, locations, and words that describe time. However, the scope of entities can be expanded depending on the specific needs of a task. For example, in medical applications, entities like drugs, symptoms, and diseases could be used for the task [22]. The primary goal of the NER task is to extract structural information from textual data. One of the issues with NER applications is the understanding of the context in which an entity appears. The same entity can appear with a different meaning in different contexts. Thus, making it hard to determine which class an entity should belong to. For instance, the word "Volvo" can be defined as an artifact, while simultaneously referring to Volvo as an organization. Some common methods for performing NER tasks are rule-based, and Neural network approaches. However, some hybrid approaches combine rule-based and neural networks [23].

2.2.1 Ruled-based and lexicon-based approaches

The rule-based method is a technique where linguistic specialists set linguistic rules to identify named entities [24]. These rules are based on the structure of the language and common patterns that indicate the characteristics of a specific named entity.

There are various strategies to construct rules for the ruled-based method. Lexical rules are often derived by analyzing the words that are often based on specific word structures. For example, lexical rules might identify capitalized words used in certain contexts as nouns, hence the word is a potential entity class. In contrast, contextual rules focus on individual words to consider the whole sentence. These rules analyze the whole sentence to understand an entitys classification. For example, a word that follows prepositions like "at" could be classified as an entity location. The effectiveness of the rule-based method largely depends on the construction of the rules because certain rules might not cover all entity variations, especially those characterized by misspellings. Since these examples can differ from the rule criteria [25].

2.2.2 Neural network approaches

Deep learning methods can perform tasks without the reliance on predefined rules and self-made features. Recurrent Neural Networks (RNNs), Bi-directional Long

Short-Term Memory (BiLSTM) networks, and Convolutional Neural Networks (CNNs) have become common models in the domain of NER tasks [26]. These deep learning architectures are widely used due to their features to effectively process sequential data, which is crucial when classifying named entities.

Transfer learning has become a crucial strategy in deep learning NER tasks, particularly useful for low-resource settings [26] since it involves training models on large text corpora followed by fine-tuning for independent NER tasks. This method takes advantage of the first model's understanding of language structure, resulting in more efficient recognition of named entity classes.

2.3 Data augmentation

The general goal of data augmentation is to construct diverse new data samples, which also maintain the diversity of the data. The process involves generating new data from existing data by applying various transformative methods. This exposes the model to a wider range of variations. Thus, improves the model's generalization for unseen data. Data augmentation is applied in various fields, notably in domains such as computer vision and NLP [27]. Within the domain of computer vision, a variety of methods are used to augment data, including methods such as rotation, scaling, and color changing, which are applied directly to the image pixels [28]. These methods generate diverse visual representations of the model, by varying images in ways that simulate variations that represent real-world scenarios. Although data augmentation has proven to be a practical tool in computer vision, it introduces a different set of challenges in NLP. Instead of dealing with pixels, augmentation methods in NLP focus on augmenting textual data, where each word might have a specific semantic meaning. Thus, such methods require a different approach to augment the data. Due to the complex structure of language, a specific methodology is required, where it is essential to augment the data in a way that ensures the method introduces variations into the data while preserving some contextual meaning [29]. This is because the structure of the data influences the model's performance and if the data differs too much from the original meaning, the model could learn inaccurate patterns.

In the domain of NLP, data augmentation operates on four distinct levels as illustrated in Figure 2.4. These levels are:

- **Character level**: This involves editing text at the character level, Some common methods are random insertion, deletion, or replacement of characters. These methods enhance the model's capability to process texts with spelling errors by incorporating noise-like mistakes during training. Thus, this approach trains the model to identify texts with similar errors more effectively, improving its overall performance in handling such errors. [30].
- **Word level**: This method changes text at the word level, some common methods are random deletion, word swap, and synonym replacement to increase the diversity of the dataset. Thus improving the robustness of NLP models against variations in language usage [31].

- **Phrase level**: Phrase level augmentation changes chunks of text. A common method is back-translation, which is a process of translating a sentence into another language and then translating it back to the original language, aiming to maintain the same semantic content while potentially altering the structure of the sentence [32].
- **Document level**: This approach alters entire documents. Some methods involve reordering paragraphs or adding context to the model's ability to understand and generate contextually relevant text.

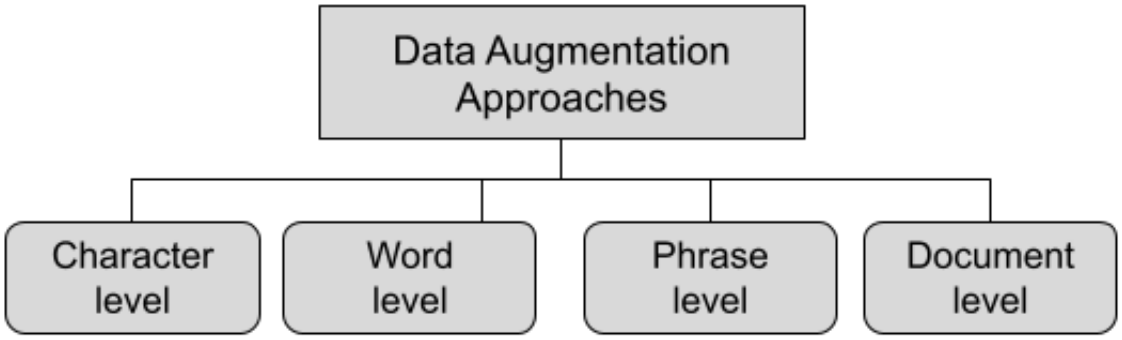


Figure 2.4: Data augmentation levels

Each level of augmentation is designed to simulate different types of variations and challenges that an NLP model might encounter in real-world applications.

3

Related work

In the related work chapter, we will cover other work, which is related to our thesis study. We will explore three areas: Swedish NER Resources, Low-Resource Settings in NLP, and data augmentation in NLP.

3.1 Swedish NER resources

For the Swedish language, there have been several studies focusing on the field of NER tasks. The paper by Ek et al. (2011) [33] addresses a NER system designed for short text messages on mobile platforms. This research tackled problems such as the lack of structured data, and shorthand expressions, which are diverse language structures with slang and emojis. The study's focus was on identifying entities such as locations, names, dates, times, and phone numbers. The study employed a rules-based method, which used a combination of regular expressions and logistic regression classifiers to process the text efficiently, achieving an F-score of 0.86.

The paper by KBLab [7] pioneers a new direction for NER tasks in NLP with the development of the Swedish BERT model, "KB-BERT". This study highlights the creation and training of a language-specific BERT model for Swedish. The KB-BERT model outperforms existing established NER models in various NLP tasks, models such as Swedish Public Employment Service and Googles M-BERT. The training of the model involved one million steps, utilizing a maximum sequence length of 128 tokens and employing a batch size of 512. The results from this training were as follows: at 10k steps, the NER F1-score was 0.8687, at 50k it was 0.912, at 150k it reached 0.918, at 350k it increased to 0.926, it was 0.925 at 700k, slightly decreased to 0.923 at 1M, and peaked at 0.927 for 2M steps. However, the research also acknowledges the ongoing challenges in NLP for lesser-used languages, particularly highlighted by the limited availability and quality of training data for languages like Swedish.

Stockholm-Umeå Corpus (SUC) has traditionally been used as a golden standard benchmark dataset for NER. However, SUC does not grasp the linguistic meaning of the 21st-century Swedish language. The study conducted by Ahrenberg (2020) introduced a new gold standard resource for Swedish Named-Entity Recognition [34], the Swe-NERC Version 1 dataset. The new resource uses several categories for the NER classification task, including Event (EVN), Organisation (GRO), Location (LOC), Treatment (MNT), Person (PRS), Symptom (SMP), Time entity (TME),

and WorkOfArt/Product (WRK). The categories of symptoms and treatment are new additions that are not used in the SUC dataset. The texts included in this resource are mainly from around 2010, which provides a more modern language. Thus, some of the texts have been extracted from various Swedish social media websites, such as Flashback and Familjeliv, as well as from the Smittskydd website and Wikipedia. The paper from Spårkbanken Text used the pre-trained BERT language model for Swedish developed at KB-Lab, KB-Bert [35], where they integrated this with a NER system devised by Kamal Raj [1]. The paper observed the average result from precision, recall, and F1-score are nearly 10 % points below the results obtained when using the SUC dataset with the same ML model. The paper by Ahrenberg (2020) mentions that it is not a fair comparison to evaluate the newly created resource against SUC 3.0 dataset since the SUC lacks name entity classes for Treatments and Symptoms.

3.2 Low-Resource settings in NLP

The challenge of reaching high performance in NLP tasks under low-resource settings has led to different approaches in data augmentation. Several studies highlight methods and their effectiveness in such simulation settings.

The paper by X Dai and H Adel (2020) [1] study focuses on data augmentation for NER tasks in the biomedical domains. The study demonstrated that even simple augmentation techniques could boost the performance of both recurrent and transformer-based models. The low-resource environment was simulated by creating small, medium, and large training sets, selecting the first 50, 150, and 500 sentences containing at least one entity mentioned from the training set. This approach showed notable improvements over strong baselines, especially in smaller training sets.

The work of Ahin, G. G. (2022) [36] delves into another direction of text augmentation techniques in low-resource settings. The study found that text augmentation greatly improved NLP models in three domains, dependency parsing, followed by part-of-speech tagging, and semantic role labeling. The study also highlighted that character-level methods were the most consistent performers in enhancing a model in a low-resource setting, the paper sampled 250, 500, and 1,000 sentences from the training data.

The paper by Wei, J., & Zou, K. (2019) [37] presents an EDA (easy data augmentation) approach, which uses four different data augmentation methods: synonym replacement, random insertion, random swap, and random deletion. The paper illustrated that EDA methods improved performance in text classification tasks, particularly for smaller datasets. The study conducted experiments using varying fractions of training data, showing that training with data augmentation methods on just 50% of the data could achieve the same accuracy as training with the full dataset.

[1] `https://github.com/kamalkraj/BERT-NER`

3.3 Data augmentation in NLP

In the context of NER tasks, several studies have illustrated the successful utilization of different data augmentation techniques to solve the challenge of limited data availability. However, when it comes to research focusing specifically on the Swedish language, there is a noticeable gap of lack of studies.

The paper by Erd et al.(2022) [38] applied three data augmentation techniques to a German Legal Entity Recognition dataset, addressing the challenge of limited labeled data within German legal text. The study used synonym replacement, mention replacement, and back translation methods as their data augmentation methods. These methods were trained on models that used BiLSTM-CRF [39] and a transformer-based architecture. The key findings were that synonym and mention replacements yielded similar improvements, with mention replacement being more time-efficient. However, back translation faced challenges due to the complex nature of legal texts. In terms of performance gains, mention replacement using the XLM-RoBERTa [40] model showed the most influential improvement, especially with smaller datasets. The paper highlighted potential future improvements for each data augmentation method. For back translation, the paper suggested improving the structural framework of the technique to more effectively manage the complexities of legal language, for mention replacement, the paper suggested expanding the method by integrating a knowledge base to introduce new entities that were more relevant, and for synonym replacement, it highlighted the need to adjusting the method to make sure the new words aren't too identical to the original ones.

Motivated by the success of the current large language models, specifically ChatGPT, the paper by Dai et al, 2023 [4] proposes a new data augmentation approach, AugGPT. AugGPT rephrases sentences in training samples into multiple semantically varied instances that enhance the dataset for model training. The study utilized three distinct datasets. The Amazon dataset, which is composed of customer reviews, the symptoms dataset, sourced from Kaggle, which is composed of descriptions of medical symptoms, and the PubMed20K dataset, which contains annotated data from the biomedical field. These datasets were chosen for their relevance to the study's focus on clinical NLP, as annotated data is often limited. The AugGPT method consists of two main steps. First, it uses ChatGPT to create new sentences, which are semantically similar data samples to the original data. Second step, it trains a BERT-based sentence classifier using these newly generated samples. The results reveal that AugGPT performs better than the current data augmentation method within text classification tasks with limited data.

4

Data

In the Data chapter of this report, we provide an overview of the datasets used for fine-tuning our models. In this chapter, we will explore the formatting and content of the dataset.

4.1 Data

For this study, we used the Swe-NERC dataset [5], a corpus focused on NER in the 21-century Swedish language. The dataset has been sourced by scraping various social media platforms and websites, providing a wide range of data on different genres of text and also unfiltered language usage. Unlike traditional publications or edited works, social media texts often provide language that is unfiltered and raw. During the inspection of the Swe-NERC dataset, I found a range of spelling errors and grammatical inconsistencies. For instance, the Swedish word "original" was often misspelled as "orginal," and common expressions like "tyvärr," and "i så fall" also appeared in incorrect spelling such as "tyvär" and "isåfall". The Swe-NERC dataset contains the following genres:

1. **bloggmix:** Texts from blogs that focus on the life of young people.
2. **Familjeliv-barnhälsa:** Posts from a social forum that discusses children's health.
3. **Flashback-fordon:** Posts that are centered around cars and other vehicles.
4. **SIC:** Blog that is describing everyday activities.
5. **Göteborgsposten:** Articles from daily newspaper .
6. **Smittskydd:** Texts from a medical periodical.
7. **Wikipedia-krig:** Wikipedia articles on wars history.

In Table 4.1, the distribution of named entities within the Swe-NERC dataset is outlined by genre. This table provides an enumeration of the number of tokens and instances of named entities for each genre. Highlighting the dataset's diversity.

Table 4.1: Distribution of Named Entities in Swe-NERC Dataset

Genre	**Tokens**	**Instances**
Bloggmix	29,052	1,079
Familjeliv-barnhälsa	25,867	768
Flashback-fordon	23,824	940
SIC	9,938	397
Göteborgsposten	20,971	1,549
Smittskydd	16,927	928
Wikipedia-krig	15,855	1,815

Table 4.2: Named Entity Distribution by genre in the Swe-NERC Dataset. This table presents the total number of tokens and instances of named entities across different genres, providing insights into the dataset's composition and the prevalence of named entities in each genre.

4.1.1 Entities

The Swe-NERC dataset includes a wide range set of named entities across different categories. The different categories and their associated metrics are enumerated as follows:

1. **Event (EVN)**: This entity contains 564 tokens and 275 instances. Entities in this category refer to occurrences of events like gatherings, natural disasters, or historical events.
2. **Organisation (GRO)**: This entity contains the names of corporations, institutions, and other types of organizations. It holds 1,910 tokens and 1,439 instances.
3. **Location (LOC)**: This entity contains geographical locations such as countries, cities, and landmarks. It holds 1,205 tokens and 1,031 instances.
4. **Treatment (MNT)**: This entity contains medical treatments, drugs, and medical procedures, with 267 tokens and 197 instances.
5. **Person (PRS)**: This entity contains fictional or real names. It holds 2,186 tokens and 1,323 instances.
6. **Symptom (SMP)**: This entity contains 1,248 tokens and 792 instances. Symptoms and medical conditions are classified under this entity.
7. **Time entity (TME)**: This entity contains 2,806 tokens and 1,377 instances. Time-related words such as dates, and periods related to time are used here.
8. **Work of Art / Product (WRK)**: This entity contains 2,081 tokens and 1,042 instances. Names of artistic works and movies are included here.
9. **Other (O)**: This category is for words (or entities) that do not fit into any of the above. It includes 130,239 tokens.

5

Methods

In this chapter, we will present the methods used in this thesis. Iterative processes have been used to shape the approaches. Here, we will explain how we preprocessed the data before training, what data augmentation methods we used for our thesis, and our model implementation. The pipeline described in Figure 5.1 illustrates a view of the study's implementation.

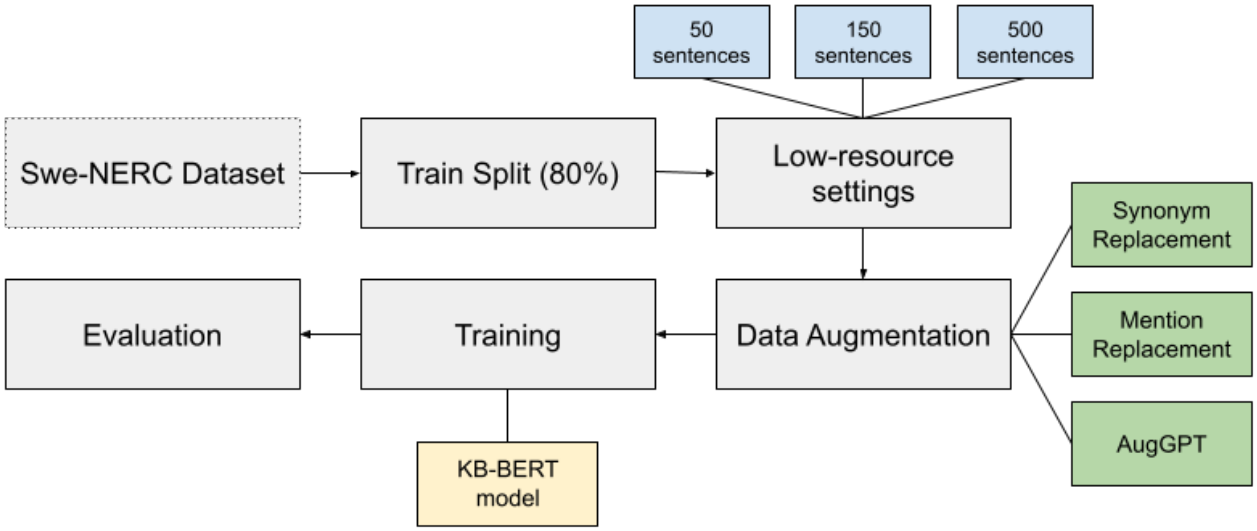


Figure 5.1: Pipeline

The overall methodology begins with the Swe-NERC Dataset. This dataset was split and 80% of the data was allocated for training. However, further modifications were made to the training data to simulate low-resource settings. Therefore, three different subsets of the training data were created, corresponding to 50, 150, and 500 sentences, where we ensured to have a variety of genres. These three subsets were then used for fine-tuning the model.

In the data augmentation phase, different techniques were implemented, such as syn-

onym replacement, mention replacement, and AugGPT, which generated multiple new datasets.

The chosen model for this study was KB-BERT, which was independently fine-tuned with our three low-resource training sets, from which we derived the baseline F1-scores from. Additionally, we independently trained the model with new training sets, generated with our data augmentation methods applied to the low-resource training sets. Further details on the training procedure are provided in Sections 5.2.2 and 5.3.2.

5.1 Preprocessing

Preprocessing is a crucial step for NLP tasks. The method serves as an approach for structuring raw data into an appropriate structure before feeding it to the ML model. For this section, we will detail the steps we took to preprocess the data.

5.1.1 Data Preparation and Tokenization

During this step, we were involved in several stages of data preparation before training the machine-learning model.

The first step was to prepare data for NER by filtering representative sentences with two filter criteria. The first filter criteria saved sentences that contained at least one label other than "O", and removed all sentences only containing the class "O". In NER tasks, "O" stands for non-entity class and usually indicates that the word or token doesn't belong to any named entity category. By choosing sentences with at least one named entity, we ensured the model got entities to learn from during the training. The second part of the filtering criteria involved the sentence length, which was done by limiting the sentences by 25 tokens. This length criterion made sure that the model was trained on sentences that were reasonably sized, which was essential for computational efficiency, and also enabled the model to achieve better generalization.

Our second step was to tokenize the data. Tokenization is a method used in NLP to split a text into smaller parts. This method is important for preparing text before using ML models since it breaks down language into smaller parts, which is easier for the model to process. For this part, we employed the AutoTokenizer from the Hugging Face Transformers library [41]. The AutoTokenizer we used was the "KBLab/bert-base-swedish-cased-ner" model [1]. Despite the sentences already being split into tokens, the KBLab's tokenizer was necessary to use, since it provides a specific input format, which the BERT model needs. Another reason for using that tokenizer was due to its incorporation of special tokens, such as [UNK] for unknown words, [SEP] and [CLS] for separating and classifying sentences, respectively, and [PAD] for padding shorter sequences to a uniform length, which is important enabling the model's understanding of sentence context. The tokenizer from KBLab is a standard BERT base tokenizer for the Swedish language, which has been trained

[1] `https://huggingface.co/KB/bert-base-swedish-cased`

on a wide collection of text sources, and has a vocabulary size of around 50,000 words. The tokenizers converted words into token IDs and generated metadata for the BERT model, including attention masks and segment IDs. These components are important as they allow the BERT model to process the input data effectively.

5.1.2 Simulating Low-resource settings and data split

In the field of NLP, situations, where data are scarce, are often referred to as "low-resource settings".

Within the context of this thesis, an attempt was made to replicate and evaluate performance under these low-resource scenarios. The initial dataset was split with a ratio of 80% for training, 10% for validation, and 10% for testing. The training data was further modified to simulate a low-resource setting by creating three unique training subsets to simulate various low-resource scenarios, these subsets were then used as our training datasets when fine-tuning our model (the remainder of the training data was not used). These subsets were also created to contain a diverse range of genres from the SWE-NERC dataset, which was achieved by randomizing the data. Additionally, we made sure that the distribution of entities in these subsets closely resembled the entity distribution found in the original dataset.

The three distinct training subsets are outlined below:

- **Small**: This subset included 50 sentences.
- **Medium**: This subset included 150 sentences.
- **Large**: This set included 500 sentences.

The goal of creating these subsets was to determine the model's performance when trained on a very constrained amount of data.

Figure 5.2 illustrates the distribution of each entity class between the training, validation, and test datasets split, excluding the "O" class. Each bar represents the percentage of a specific entity relative to the total number of non "O" entities within its respective dataset. The training dataset illustrates the initial distribution used for training the model. The test dataset was created with a unique set of examples that the model has not encountered before. Using this method guarantees that the test set performance of the model is a reliable measurement.

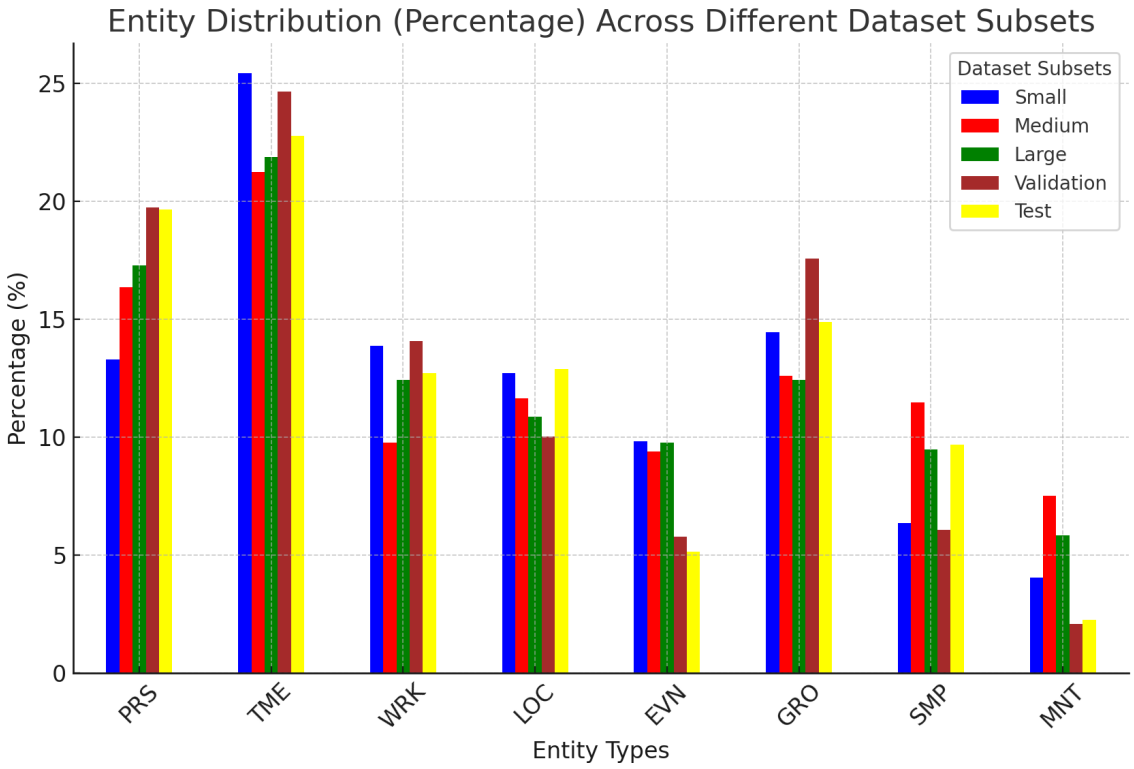


Figure 5.2: Entity class percentages

5.2 Data augmentation

In this section, we will present the chosen data augmentation methods utilized in our study, along with the specific parameters applied for each method.

5.2.1 Augmentation methods

In this section, we will detail the data augmentation methods used to enhance the model. Some of the real augmented examples and their corresponding results can be viewed in Appendix A.

5.2.1.1 Synonym replacement

Our method, synonym replacement is inspired by the paper Dai, X., & Adel, H. (2020) [1]. The idea behind this method was to substitute some selected words which were labeled with the entity "O". The selected word was substituted with its related synonym, allowing us to generate sentences that maintain semantic similarity while introducing some variation to the model. This method ensured that named entity classes, which held contextual meanings, remained unchanged. Thus, the method could deliver a balanced combination of various sentences, while also keeping the contextual meaning of the original sentences. For this method, we primarily used Språkbanken's lexicon Swesaurus [42] as a resource. The resource was mainly used

because it is a reliable source specifically designed for Swedish synonyms. The resource contains around 12,764 synonyms. However, when the method could not identify a synonym, we supplemented it with an external API to ensure completeness. For this objective, we utilized the API provided by Synonymord [43]. Sometimes we obtained multiple synonyms for a specific word. In such cases, we randomly selected one synonym to use in our dataset.

Table 5.1 shows how one sentence can be augmented three times with the synonym replacement method.

Original Sentence				
Token	Anders	went	to	market
Label	PRS	O	O	O
Augmented Example 1				
Token	Anders	**traveled**	**toward**	**marketplace**
Label	PRS	O	O	O
Augmented Example 2				
Token	Anders	**journeyed**	**into**	**bazaar**
Label	PRS	O	O	O
Augmented Example 3				
Token	Anders	**moved**	**towards**	**emporium**
Label	PRS	O	O	O

Table 5.1: Augmentation of the sentence "Anders went to the market" using synonym replacement for non-entity tokens. In this case, "Anders" is an entity that is not within the "O" class, labeled as a person. Thus, it was not changed. The words "went," "to," and "market" are "O" entity classes, and are the targets for synonym replacement. The process involves finding synonyms for these words and replacing them to create new sentences with similar meanings but different phrasing.

5.2.1.2 Mention replacement

Our second data augmentation is inspired by Erd, R., Feddoul, L., Lachenmaier, C., Mauch, M. J. (2022) [38]. Mention replacement is a method for enriching training data for NER models. The method targets entities in sentences labeled with classes other than "O". The mentioned replacement method involves replacing labeled entities in the dataset with different entities of the same entity class. Entities labeled as "O" were left out. This method ensured the introduction of variations in the dataset while maintaining the grammatical structure of the original sentences.

In implementing our approach, we utilized a collection of entity words from the Språkbanken language resources [44] [45]. The SMP class included 3,206 words related to medical symptoms. Entities associated with WRK and GRO classes included 3,163 words. The TME class provided a collection of 105 words, the EVN class included 797 words, the MNT class provided us with 2,773 words, the PRS class contained 1,449 words, and the LOC class included 107,190 words that correspond to locations.

For our implementation, we started by creating a function to load entities from a text file, and we categorized them by their respective class labels. Then, we implemented a function to shuffle these entities and generate a unique set of replacements for each entity class, intentionally excluding the non-entity "O" class. Our augmentation procedure involved iterating over each sentence in the dataset and replacing named entities with their randomly generated alternatives. This was repeated to produce multiple augmented variants of each original sentence.

The table 5.2 illustrates augmentation example with the method mentioned replacement,

Original Sentence						
Token	Yesterday	,	Julia	visited	a	town
Label	O	O	PRS	O	O	O
Augmented Example 1						
Token	Yesterday	,	**Gustav**	visited	a	town
Label	O	O	PRS	O	O	O
Augmented Example 2						
Token	Yesterday	,	**Alex**	visited	a	town
Label	O	O	PRS	O	O	O
Augmented Example 3						
Token	Yesterday	,	**Emma**	visited	a	town
Label	O	O	PRS	O	O	O

Table 5.2: Our mentioned replacement method was used for the sentence "Yesterday, Julia visited a town" to generate augmented versions. We used the resource from Språkbanken, which provided us with a list of alternative names corresponding to specific entity classes. 'Julia', classified under the PRS class, was replaced with other names like 'Gustav', 'Alex', or 'Emma'.

5.2.1.3 AugGPT

For our third method, we got inspired by the approach by Dai et al, 2023 [4]. However, some modification was made, since the Dai et al approach was made for text classification problems. Our modified method was designed to preserve non-'O' classes while changing the surrounding context. Thus, extending the dataset without altering the semantic structure, and creating diverse linguistic variations around the entities.

During the augmentation process, a prompt is constructed to guide the GPT model in rewriting the sentence. The prompt, written in English for clarity here, instructs the model to "Rewrite the sentence x times, retaining the following words [list of entities]". The prompt used for this thesis can be seen in Appendix 2. The construction of this prompt is crucial since it tells the model which words are essential to preserve while altering the rest of the sentence. If a word containing an entity (which should be retained) is altered, it becomes impossible to map the entity back to its original class, as we no longer know to which class it belonged.

Figure 5.3 illustrates an overview of the AugGPT method pipeline, showcasing how the method maintains specific entities within a sentence during augmentation. These entities, referred to as "word1" and "word2" in the figure are the entities not belonging to the "O" class in that sentence. The AugGPT model takes the original sentence and, through a predefined prompt is instructed to construct new sentences that maintain the original meaning but with some variation.

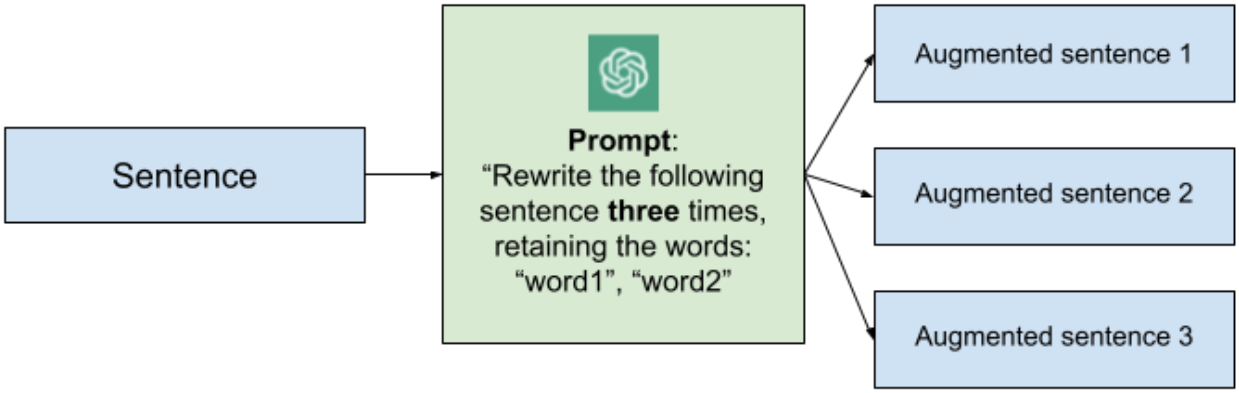


Figure 5.3: AugGPT method pipeline

The table 5.3 illustrates augmentation example with the method AugGPT,

Original Sentence						
Token	Yesterday	,	Anders	went	to	school
Label	O	O	PRS	O	O	O
Augmented Example 1						
Token	Anders	attended	school	yesterday	.	
Label	PRS	O	O	O	O	
Augmented Example 2						
Token	School	was	where	Anders	spent	yesterday
Label	O	O	O	PRS	O	O
Augmented Example 3						
Token	Yesterday	saw	Anders	heading	to	classes
Label	O	O	PRS	O	O	O

Table 5.3: In our example, "Anders" is the entity to be maintained. Our method is using the prompt: "Rewrite the following sentence Yesterday, Anders went to school three times". This ensures that the name 'Anders' is included in each new sentence and that the output is three distinct sentences that will keep the entity's context.

5.2.2 Parameter setup and augmentation process

In this section, we will detail the specific settings used for data augmentation. The first key tuning parameter relates to the number of generated instances of the raw dataset, for every raw data sample in the training set(50,150,500). We selected an

augmentation level parameter from a predefined list of ratios: {1, 3, 5}, which is how many augmented instances are created from each original sentence from that raw dataset. For example, if we are working with the small training set with 50 raw sentences and select a ratio of 3, this would result in the outcome of 150 new augmented sentences. Likewise, if 500 raw sentences are selected with a ratio of 5, it would result in the outcome of 2500 new augmented sentences. Our second setting parameter was a "p-value", which sets the probability of whether a specific word in a sentence will be augmented or not. The following p-values [0.3, 0.5, 0.7] were used. Synonym replacement relies on a probabilistic approach where each word in a sentence that is within the "O" class entity has a certain probability of being replaced with a synonym, making the method dependent on the p-value parameter. In contrast, AugGPT doesnt focus on modifying individual words, instead, it generates entirely new text based on the entity classes that are not the "O" class. Similar to the mentioned replacement method which works on a different approach, where the method replaces all occurrences of entities that are not within the class "O" in each sentence, rather than making probabilistic changes to individual words.

In our methodology, each data augmentation method was applied once to the three training datasets, where 15 new training sets were created from each raw training set. In total, we created 45 new augmented training datasets by combining all three methods and their respective parameters. Synonym replacement contributed to 27 new training datasets in total (because of its two parameters used), while the mentioned replacement and AugGPT contributed to 9 each. The procedure can be viewed in Figure 5.4. The KB-BERT model was then independently fine-tuned using each of the new training datasets.

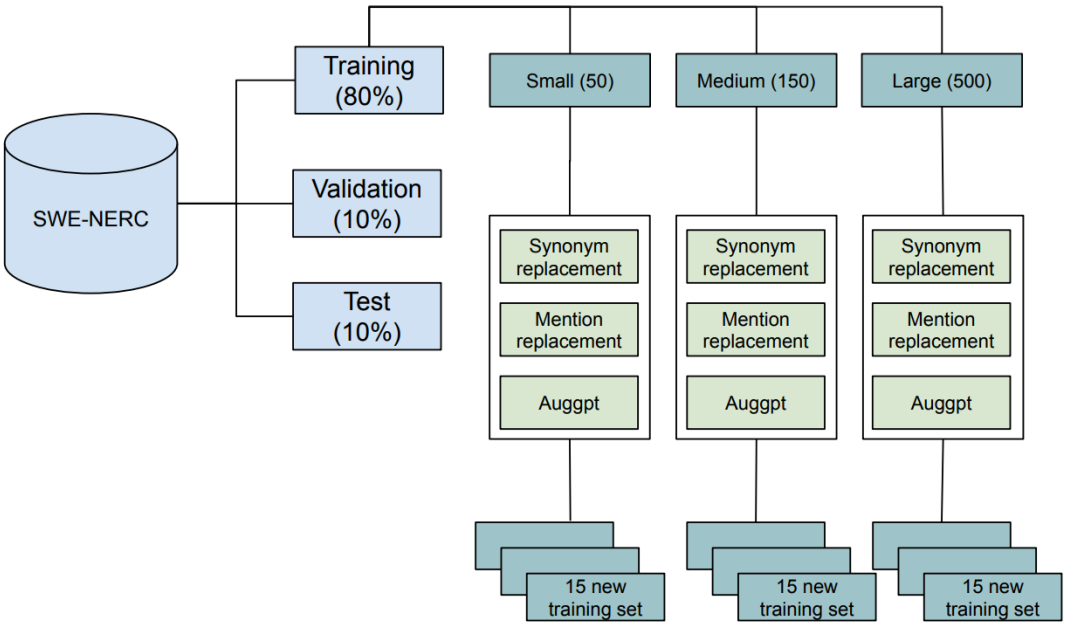


Figure 5.4: New training datasets

5.3 Model implementation and training

In this part, we will explain the implementation of the model used in the thesis, along with the choice of our hyperparameters, and how we fine-tuned the model.

5.3.1 Model

In this study, we used the fine-tuned model checkpoint "KBLab/bert-base-swedish-cased-ner" [2], which is fine-tuned NER model, with similar NER task.

Choosing this specific model was motivated by several factors, including:

- **Language**: The KB-BERT model is designed for the Swedish language. Thus, making the model a suitable choice for our study.

- **Task Optimization**: The model offers a useful basis to evaluate the level of training performance in low-resource settings since it is already fine-tuned for a similar NER task.

The model was built using PyTorch [46] and Hugging Face's transformers library [47]. PyTorch handled the deep learning tasks, and the transformers library made it easy to use the pre-trained model and tools. Thus, making the implementation straightforward.

5.3.2 Creating baseline and fine-tuning the model

Our study focuses on the influence of data augmentation on model performance within low-resource settings. Thus, it was essential to create a reliable baseline. The baseline serves as our reference for our data augmentation methods. Figure 5.5 illustrates the creation of our baselines.

[2] `https://huggingface.co/KB/bert-base-swedish-cased-ner`

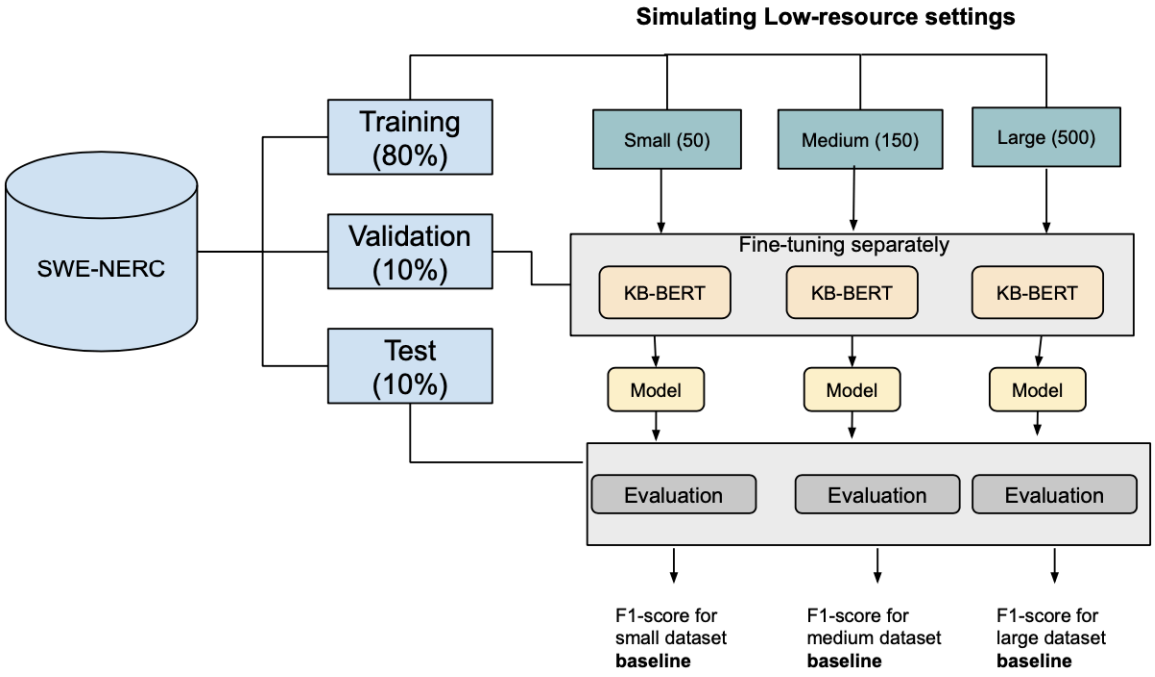


Figure 5.5: Baselines creation

The baseline refers to the F1-score acquired from the KB-BERT model when fine-tuned on the small, medium, and large datasets, where no data augmentation was applied. We created our baseline by creating three raw datasets reflecting the common limitations found in low-resource settings, one with 50 sentences, a second with 150 sentences, and a third, with 500 sentences. Our approach to establishing the baseline involved fine-tuning the KB-BERT model on each of these new training sets independently.

The fine-tuning was conducted by first downloading the pre-trained model, KB-BERT from the Hugging Face Transformers library. We instantiate the model by using the Trainer class from the library, where we configured the pre-trained KB-BERT model with hyperparameter optimization to optimize the model's performance during the fine-tuning phase.

Following the establishment of the baseline with the raw datasets, we applied our data augmentation methods to each dataset, creating new datasets, which were illustrated in Figure 5.4. The augmented datasets were then fine-tuned on the same initial KB-BERT model, with the same configurations as the fine-tuning for our baseline. Figure 5.6 illustrates an example of the pipeline for the large training set, the same procedure was done on low and medium training sets.

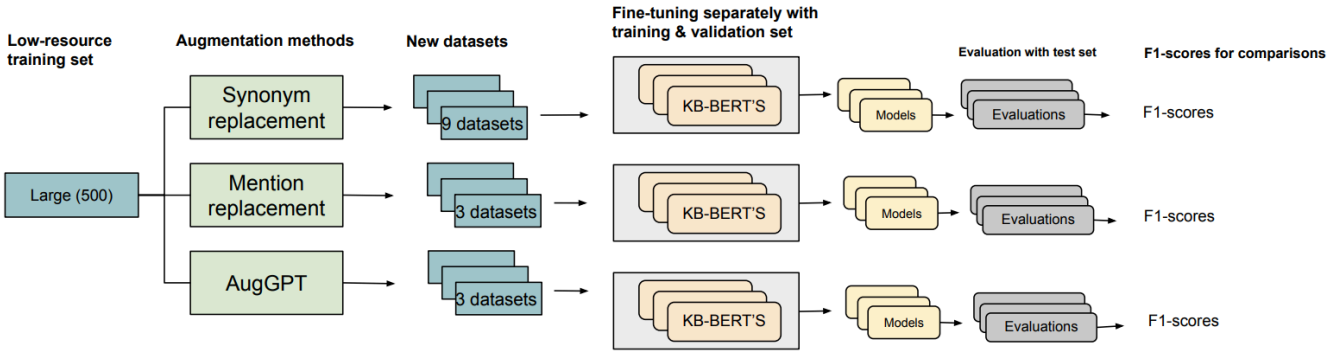


Figure 5.6: Augmented datasets fine-tuning

5.3.3 Hyperparameter optimization

Hyperparameters were optimized during the training process of our model to increase the model's ability for learning and data generalization. We implemented a random search method [48], which allowed us to have a wide exploration of the best parameter values, increasing the possibility of selecting the best combination of hyperparameters. Random search operates by randomly choosing combinations of hyperparameters from a specified range for each hyperparameter. This randomized strategy is effective in quickly deciding the most optimal parameters, often requiring fewer iterations compared to other methods, such as grid search [48], which explores all possible combinations of hyperparameters.

Below, we describe the individual hyperparameters we used for our study:

- **Max Epochs:** Our training was conducted with a maximum of 20 epochs to allow for wide performance evaluation.
- **Learning Rate:** We began with an initial learning rate of 2×10^{-5}. Through our random search, we tested learning rates ranging from 5×10^{-6} to 5×10^{-5}, with 2×10^{-5} being the starting point.
- **Weight Decay:** We utilized a weight decay of 0.01 to negate potential overfitting. The parameter was varied between 0.001 and 0.1, during the fine-tuning phase.
- **Warm-Up Steps:** A warm-up phase using between 10 and 500 steps was used to scale the learning rate up.
- **Batch Size:** We chose a batch size in the range of 8 to 32.
- **Early Stopping:** An early stopping criterion was implemented to halt training if the validation performance did not improve for a certain number of epochs. We set a maximum of 5 epochs.

5.4 Evaluation

In this section, we evaluated our NLP model and methods, focusing on their impact in different low-resource settings and the impact augmentation had on the score. Our evaluation was based on metrics such as the F1-score to measure entity recognition accuracy and the cosine similarity to understand how augmentation affected the score.

5.4.1 Evaluation of Data augmentation effectiveness in various low-resource settings

In this subsection, we evaluated the performance of our model on datasets of 50, 150, and 500 sentences, to simulate different low-resource environments. To measure the models' entity recognition abilities, we used the F1-score metric. For calculating the F1-score, we used the seqeval metric library [49], with the micro-averaging. This metric allowed us to evaluate the model's performance by considering the correct and incorrect predictions of entity classes.

For this evaluation process, we began by setting a baseline, where we fine-tuned the KB-BERT model on each of the raw datasets independently, recording the F1-scores. Following the baseline result, we applied our three data augmentation methods: Synonym Replacement, Mentioned replacement, and AugGPT, each with multiple parameters, which created multiple new datasets. We fine-tuned the KB-BERT model independently on each of the augmented datasets to evaluate the effect of each data augmentation method and its parameters. We then recalculated the F1-scores for the models fine-tuned on the augmented datasets and compared them to their respective baseline F1-scores, and to each other. This evaluation provided us with insights into how data augmentation affected performance in various low-resource settings.

5.4.2 Augmentation Impact

To evaluate the impact our augmentation method had on our model's performance, we used the Swedish sentence-BERT transformer [3]. The method involved calculating the cosine similarity [50] between the original sentences and their augmented version, which provided us with an understanding of the effect the augmentation had on the performance of the model. The metric calculates the degree of similarity of two sentences. Thus creating a basis for data augmentation evaluation. A score of 1 implies that the two sentences are semantically identical, meaning the augmentation method has made no changes to the original meaning of the sentence. A score lower than 1 indicates deviation during the augmentation method. For each method and dataset, we calculated the average value of all these cosine similarity scores with the Swedish sentence-BERT transformer. The mean value provided us with a general view of how the augmentation preserved the original sentence's meaning with the method. We then extended our approach by creating a plot that displayed the

[3] `https://huggingface.co/KBLab/sentence-bert-swedish-cased`

cosine similarity values against the F1-score, which illustrated the influence of cosine similarity on the F1-score for each method and dataset.

6

Results

The Results section of this thesis is structured to analyze the impact of various augmentation methods on dataset performance. We begin each section by presenting an overview of a specific augmentation method, highlighting its average results across all used parameters. This is followed by detailed subsections where we illustrate how each parameter influences performance across different datasets: small (50 instances), medium (150 instances), and large (500 instances). Subsequently, we will present a comparison analysis of the different methods used in this thesis. The chapter will also present an analysis of the relationship between cosine similarity and F1-score across each dataset.

6.1 Baseline

The baseline F1-scores for small, medium, and large training sets were recorded at 0.384, 0.445, and 0.639, respectively, which served as a reference for evaluating the impact of the augmentation.

6.2 Synonym replacement

In evaluating the effectiveness of synonym replacement as a data augmentation method, we observed significant improvements in the F1-scores across various training set sizes. The results displayed in Table 6.1 illustrate the impact of the synonym replacement method, where both the percentage increase and the Delta are calculated based on the average result from all different parameter settings used for the specific dataset size. For the synonym replacement method, the F1-scores showed significant increases for all low-resource settings. For the small training set, the increase went from 0.384 to 0.409. The medium and large training sets, however, showed more noticeable improvements. The F1-score for the medium training set went from 0.445 to 0.538, and for the large set, it went from 0.639 to 0.728. For the average improvement on the F1-score with the synonym replacement method, we observed increases of 0.025, 0.093, and 0.089 for the small, medium, and large sets, respectively. The average delta increase across all sizes was calculated at 0.069. These enhancements result in percentage increases of 6.48%, 21.00%, and 13.93%.

Synonym replacement	Small	Medium	Large	Overall Avg. Increase
Baseline	0.384	0.445	0.639	-
F1-Score	0.409	0.538	0.728	-
Percentage Increase	6.48%	21.00%	13.93%	13.80%
Δ	0.025	0.093	0.089	0.069

Table 6.1: Average F1-scores and their percentage increases for different sizes of training sets in synonym replacement augmentation, including delta increases.

6.2.1 Small training set

In the analysis of the impact of synonym replacement on the F1-score, Figure 6.1 and Table 6.2 provide an overview of the outcomes at various augmentation levels and p-values. The data shows a fluctuation in F1-scores depending on the augmentation parameters. For small-sized datasets, the best parameter setting was an augmentation level of 5 with a p-value of 0.7, resulting in the highest F1-score of 0.471. The least effective setting for small-sized datasets occurred at an augmentation level of 1 with a p-value of 0.5, where the F1-score increased to only 0.386.

At augmentation level 1 and a p-value of 0.3, the F1-score increased to 0.390. When the p-value was raised to 0.5, the F1-score increased to 0.386, and at a p-value of 0.7, it further increased to 0.394. At augmentation level 3 with a p-value of 0.3, the F1-score significantly increased to 0.431. With a p-value of 0.5 at this level, the F1-score was 0.398, and at a p-value of 0.7, it further increased to a notable 12.50% improvement from the baseline. At the highest augmentation level of 5, a p-value of 0.3 led to an F1-score of 0.391. A p-value of 0.5 resulted in a score of 0.387, while a p-value of 0.7 significantly improved the score to 0.471.

Synonym replacement methods	F1-score	Percentage increase
Baseline	0.384	-
Augmentation level=1, p-value=0.3	0.390	1.56%
Augmentation level=1, p-value=0.5	0.386	0.52%
Augmentation level=1, p-value=0.7	0.394	2.60%
Augmentation level=3, p-value=0.3	0.431	12.24%
Augmentation level=3, p-value=0.5	0.398	3.65%
Augmentation level=3, p-value=0.7	0.432	12.50%
Augmentation level=5, p-value=0.3	0.391	1.82%
Augmentation level=5, p-value=0.5	0.387	0.78%
Augmentation level=5, p-value=0.7	0.471	22.66%
Average	**0.409**	**6.48%**

Table 6.2: This table displays F1-scores and their percentage increases across different levels of augmentation and p-values in synonym replacement methods. The table illustrates the impact each parameter setting had on a small training set compared to the baseline.

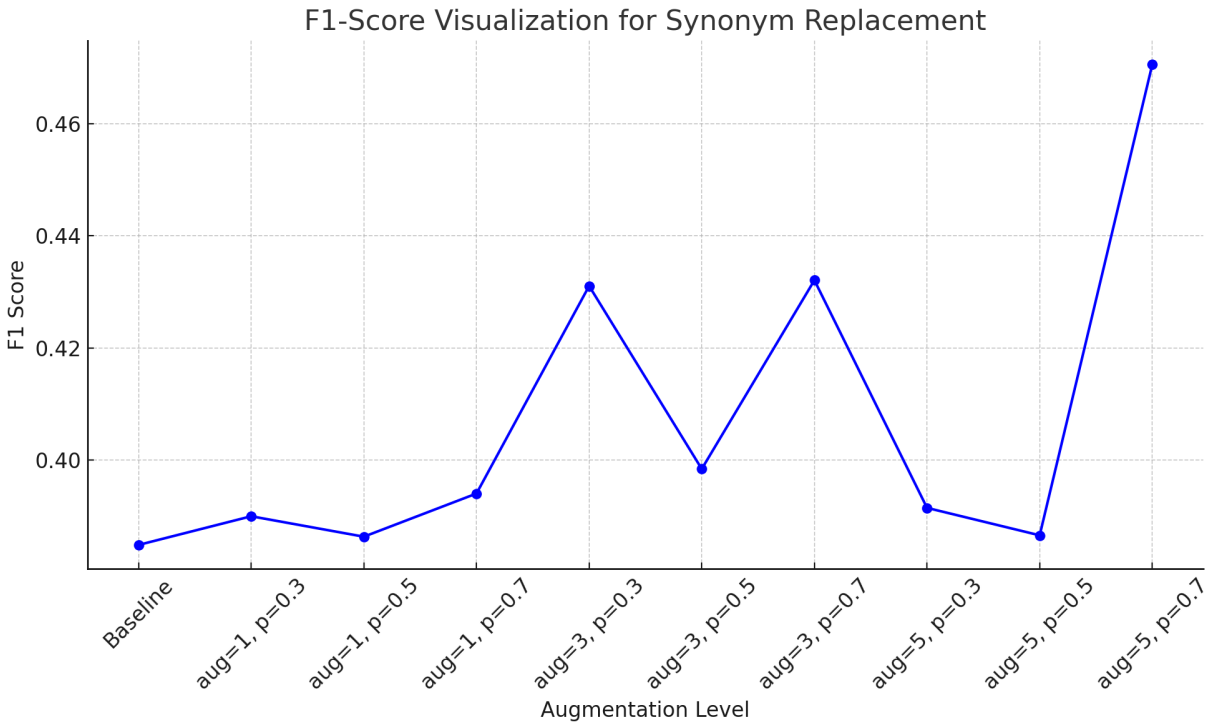


Figure 6.1: F1-score comparison with a line chart for synonym replacement with a large-sized dataset

6.2.2 Medium training set

In the study of the medium-sized dataset, different levels of synonym replacement had varied effects on F1-scores. Fluctuations in F1-scores were also observed here across various parameter settings, which can be seen in Figure 6.2 and Table 6.3. For medium-sized datasets, the best parameter setting was an augmentation level of 5 with a p-value of 0.7, resulting in the highest F1-score of 0.622. The least effective setting for medium-sized datasets occurred at an augmentation level of 5 with a p-value of 0.5, where the F1-score decreased to 0.460.

At an augmentation level of 1 with a p-value of 0.3, the F1-score improved to 0.494. When the p-value increased to 0.5, the score slightly decreased to 0.488. With a further increase in p-value to 0.7, the F1-score reached 0.496. At an augmentation level of 3 and a p-value of 0.3, the F1-score significantly jumped to 0.549. However, increasing the p-value to 0.5 at this level resulted in a reduced F1-score of 0.525. The highest score in the study, 0.608, was observed at this augmentation level with a p-value of 0.7. With the augmentation level of 5, the outcomes varied with different p-values. At a p-value of 0.3, the F1-score was 0.605. A decrease to 0.460 was noted at a p-value of 0.5. The largest improvement, with an F1-score of 0.622, occurred at a p-value of 0.7.

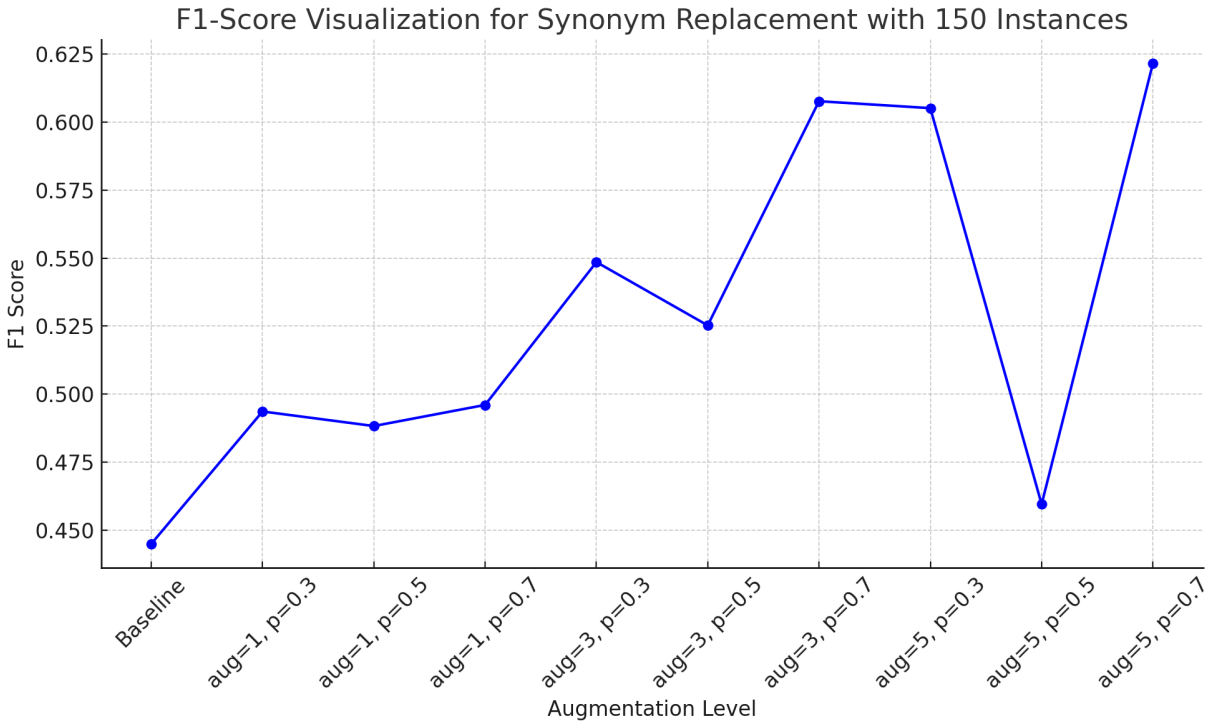


Figure 6.2: F1-score comparison with a line chart for synonym replacement with a medium-sized dataset

Synonym replacement methods	**F1-score**	**Percentage increase**
Baseline	0.445	-
Augmentation level=1, p-value=0.3	0.494	10.98%
Augmentation level=1, p-value=0.5	0.488	9.69%
Augmentation level=1, p-value=0.7	0.496	11.47%
Augmentation level=3, p-value=0.3	0.549	23.37%
Augmentation level=3, p-value=0.5	0.525	17.99%
Augmentation level=3, p-value=0.7	0.608	36.63%
Augmentation level=5, p-value=0.3	0.605	35.95%
Augmentation level=5, p-value=0.5	0.460	3.38%
Augmentation level=5, p-value=0.7	0.622	39.78%
Average	**0.538**	**21.00%**

Table 6.3: This table displays F1-scores and their percentage increases across different levels of augmentation and p-values in synonym replacement methods. The table illustrates the impact each parameter setting had on a medium training set compared to the baseline.

6.2.3 Large training set

In the study involving a large-sized dataset, the effects of various levels of synonym replacement on F1-scores were analyzed, and the result is illustrated in Figure 6.3 and Table 6.4. For large-sized datasets, the best parameter setting was an augmentation level of 3 with a p-value of 0.3, resulting in the highest F1-score of 0.753. The least effective setting for medium-sized datasets occurred at an augmentation level of 1 with a p-value of 0.5, where the F1-score decreased to 0.696.

At an augmentation level of 1 and a p-value of 0.3, the F1-score increased to 0.709, showing a 10.92% improvement. A rise in the p-value to 0.5 led to a slightly reduced F1-score of 0.696, but it was still 8.85% above the baseline. Further increasing the p-value to 0.7 resulted in an F1-score of 0.712, an 11.42% improvement. At an augmentation level of 3, the F1-score went up to 0.753 with a p-value of 0.3, which resulted in a 17.80% increase. However, at p-values of 0.5 and 0.7, the F1-scores were 0.744 and 0.728 respectively, indicating decreasing returns at higher synonym replacement probabilities. At the highest tested augmentation level of 5, the F1-score was 0.736 with a p-value of 0.3. With p-values of 0.5 and 0.7, the F1-scores were 0.743 and 0.730 respectively, indicating improvements of 16.21% and 14.22%.

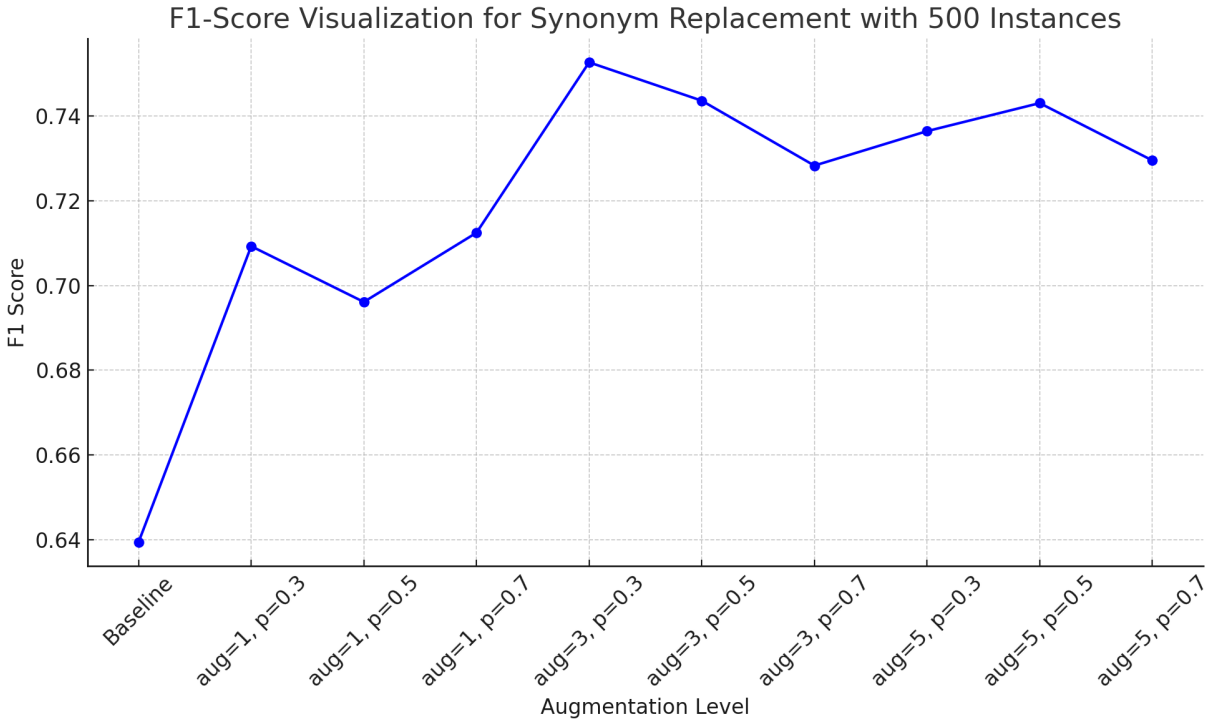


Figure 6.3: F1-score comparison with a line chart for synonym replacement with a large-sized dataset

Synonym replacement methods	**F1-score**	**Percentage increase**
Baseline	0.639	-
Augmentation level=1, p-value=0.3	0.709	10.92%
Augmentation level=1, p-value=0.5	0.696	8.85%
Augmentation level=1, p-value=0.7	0.712	11.42%
Augmentation level=3, p-value=0.3	0.753	17.80%
Augmentation level=3, p-value=0.5	0.744	16.37%
Augmentation level=3, p-value=0.7	0.728	13.90%
Augmentation level=5, p-value=0.3	0.736	15.16%
Augmentation level=5, p-value=0.5	0.743	16.21%
Augmentation level=5, p-value=0.7	0.730	14.22%
Average	**0.728**	**13.85%**

Table 6.4: This table displays F1-scores and their percentage increases across different levels of augmentation and p-values in synonym replacement methods. The table illustrates the impact each parameter setting had on a large training set compared to the baseline.

6.3 Mention replacement

In our study, we investigated the effect of mention replacement on model performance across training sets of varying sizes, as detailed in Table 6.5. The results displayed in the table illustrate the impact of the mentioned replacement method, where both the percentage increase and the Delta are calculated based on the average result from all different parameter settings used for the specific dataset size.

Upon implementing the mentioned replacement method, a diverse impact on the F1-scores was observed. For the small training set, the F1-score decreased from 0.384 to 0.343, indicating a reduction in model performance by 10.68%. Contrariwise, in the medium and large training sets, mentioned replacement led to an increase in F1-scores. The medium set experienced a slight improvement from 0.445 to 0.452, a modest increase of 1.49%, while the large set showed a more significant improvement from 0.639 to 0.675, a 5.61% increase. The overall average delta change in F1-score, calculated across all sizes, was approximately 0.00067, a minimal average change. This indicates that while mentioned replacement can have varying effects depending on the dataset size.

	Small	**Medium**	**Large**	**Avg. Increase across datasets**
Baseline	0.384	0.445	0.639	-
Mention Replacement	0.343	0.452	0.675	
Precentage Increase	-10.68%	1.49%	5.61%	-1.19
Δ	-0.041	0.007	0.036	0.00067

Table 6.5: F1-score Comparison for mention replacement across different sizes of training sets, showing percentage and delta changes.

6.3.1 Small training set

In the study involving a small-sized dataset, the effects of various levels of mention replacement on F1-scores were analyzed, the result is illustrated in Figure 6.4 and Table 6.6. For small-sized datasets, the results demonstrate that the data augmentation method mentioned replacement led to a reduction in model performance across all tested levels. When augmentation was applied at level 1, the F1-score reduced to 0.343, which resulted in a decrease of 10.68%. Increasing the augmentation to level 3 resulted in a slightly higher F1-score of 0.351, yet still 8.61% lower than the baseline. At the highest augmentation level of 5, the F1-score further declined to 0.336, a 12.51% decrease from the baseline.

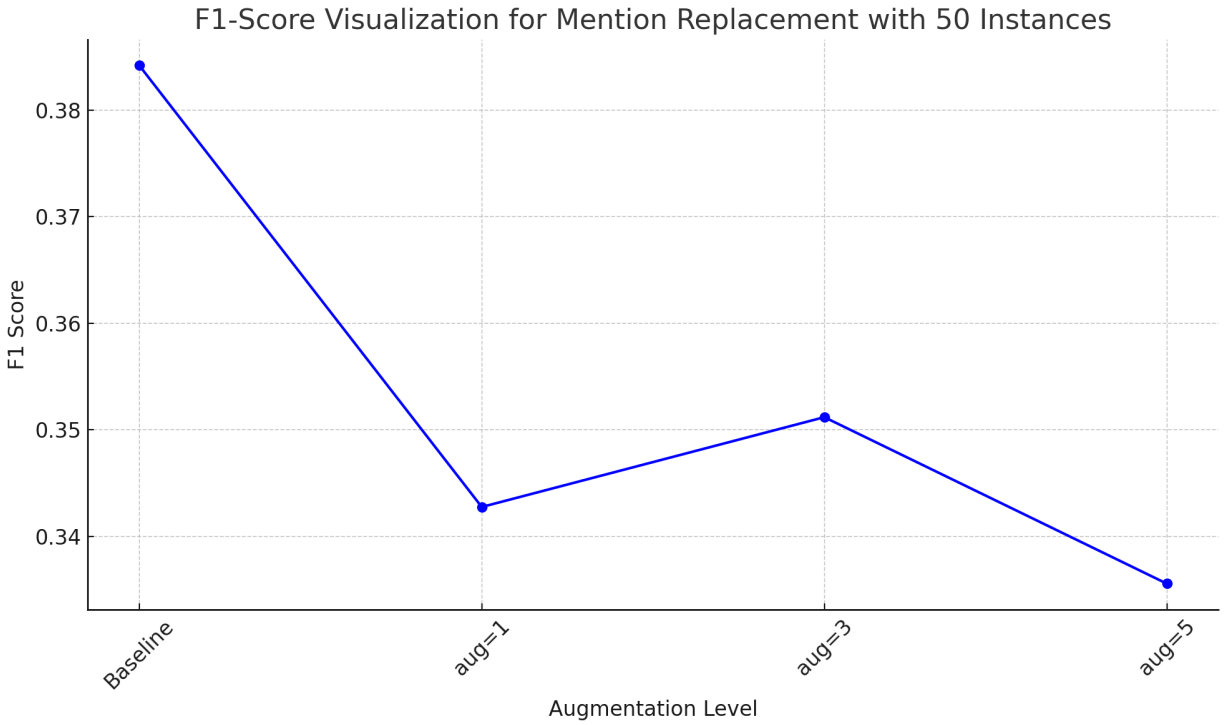


Figure 6.4: F1-score comparison with a line chart for mention replacement with a small-sized dataset

Mention Replacement Methods	**F1-score**	**Percentage Change**
Baseline	0.384	-
Augmentation Level=1	0.343	-10.68%
Augmentation Level=3	0.351	-8.61%
Augmentation Level=5	0.336	-12.51%
Average	**0.343**	**-10.68%**

Table 6.6: This table displays F1-scores and their percentage increases across different levels of augmentation in the mentioned replacement method. The table illustrates the impact each parameter setting had on a small training set compared to the baseline.

6.3.2 Medium training set

In the investigation that included a medium-sized dataset, we examined how replacing mentions at different levels affects F1-scores. The outcomes are illustrated in Figure 6.5 and Table 6.7. The data shows a fluctuation in F1-scores depending on the augmentation parameters. For medium-sized datasets, the best parameter setting was an augmentation level of 5, resulting in the highest F1-score of 0.452, a 4.95% increase compared to the baseline. The least effective setting for medium-sized datasets occurred at an augmentation level of 3, where the F1-score decreased to 0.427, a -4.17% decrease compared to the baseline. When augmentation was applied at level 1, there was a slight increase in the F1-score to 0.461, an improvement of 3.64% over the baseline.

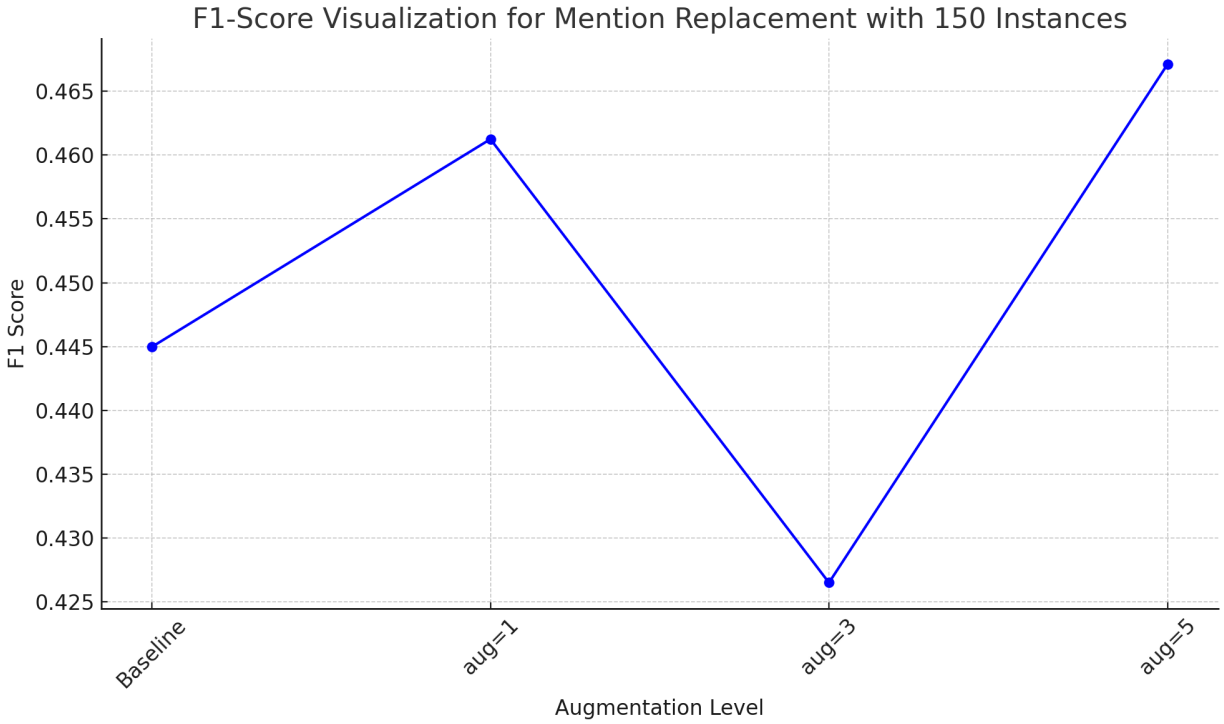


Figure 6.5: F1-score comparison with a line chart for mention replacement with a medium-sized dataset

6.3.3 Large training set

In the mention replacement study with a large-sized dataset, the F1-scores displayed a positive trend with increasing levels of augmentation. The results are visualized in Figure 6.6 and presented in Table 6.8. With an augmentation level of 1, the F1-score increased to 0.655, indicating a 2.49% improvement over the baseline. When the augmentation level was increased to 3, a more significant improvement was observed, with the F1-score reaching 0.685, a 7.19% increase compared to the baseline. Further, augmentation at level 5 resulted in an F1-score of 0.686, which resulted in a 7.34% improvement from the baseline.

Mention Replacement Methods	F1-score	Percentage Change
Baseline	0.445	-
Augmentation Level=1	0.461	3.64%
Augmentation Level=3	0.427	-4.17%
Augmentation Level=5	0.467	4.95%
Average	**0.452**	**1.49%**

Table 6.7: This table displays F1-scores and their percentage increases across different levels of augmentation in the mentioned replacement method. The table illustrates the impact each parameter setting had on a medium training set compared to the baseline.

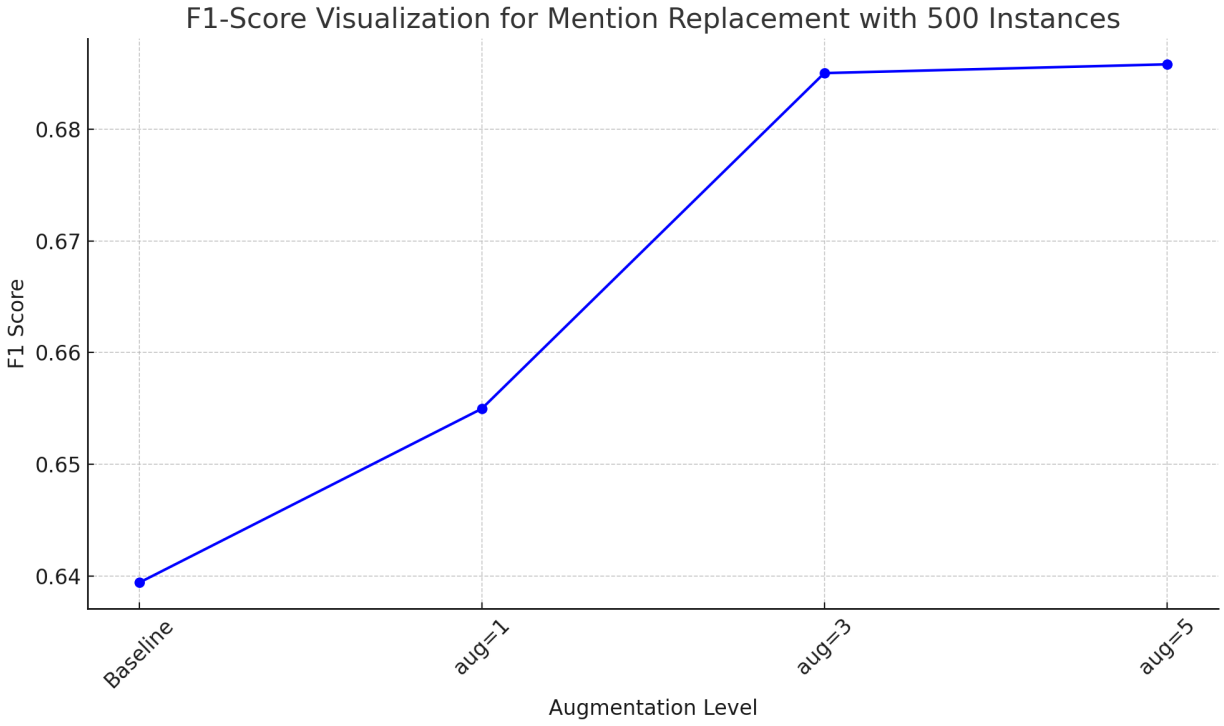


Figure 6.6: F1-score comparison with a line chart for mention replacement with a large-sized dataset

Mention Replacement Methods	F1-score	Percentage Change
Baseline	0.639	-
Augmentation Level=1	0.655	2.49%
Augmentation Level=3	0.685	7.19%
Augmentation Level=5	0.686	7.34%
Average	**0.675**	**5.61%**

Table 6.8: This table displays F1-scores and their percentage increases across different levels of augmentation in the mentioned replacement method. The table illustrates the impact each parameter setting had on a large training set compared to the baseline.

6.4 AugGPT

The results displayed in Table 6.9 illustrate the impact of the AugGPT method, where both the percentage increase and the Delta are calculated based on the average result from all different parameter settings used for the specific dataset size.

AugGPT method improved F1-scores across all training set sizes. The small training set had an F1-score increase from 0.384 to 0.397, reflecting a 3.39% change. The F1-score for the medium-sized training set increased from 0.445 to 0.524, which is a 17.74% improvement. For the large training set, the F1-score improved from 0.639 to 0.663, a change of 3.68%. While the relative improvement is less pronounced compared to the medium set, it still signifies the positive impact of AugGPT in larger datasets. The overall average delta change in F1-score, calculated across all sizes, was approximately 0.0387, a change of 927%.

	Small	**Medium**	**Large**	**Avg. Increase across datasets**
Baseline	0.384	0.445	0.639	-
AugGPT	0.397	0.524	0.663	-
Precentage Increase	3.39%	17.74%	3.68%	8.27 %
Δ	0.013	0.079	0.024	0.0387

Table 6.9: F1-score Comparison for AugGPT across different sizes of training sets, showing percentage and delta changes.

6.4.1 Small training set

In the experiment involving AugGPT with a small-sized dataset, the F1-scores displayed varying results at different augmentation levels. The outcomes are shown in Figure 6.7 and Table 6.10. At an augmentation level of 1, the F1-score slightly decreased to 0.374, showing a reduction of 2.67%, which also was the lowest F1-score. When the augmentation level was increased to 3, the F1-score improved to 0.400, which resulted in an increase of 4.09% from the baseline. Further increasing the augmentation level to 5 led to the highest F1-score of 0.418, an 8.80% improvement over the baseline.

AugGPT Augmentation Methods	**F1-score**	**Percentage Change**
Baseline	0.384	-
Augmentation Level=1	0.374	-2.67%
Augmentation Level=3	0.400	4.09%
Augmentation Level=5	0.418	8.80%
Average	**0.397**	**3.39%**

Table 6.10: This table displays F1-scores and their percentage increases across different levels of augmentation in the AugGPT method. The table illustrates the impact each parameter setting had on a small training set compared to the baseline.

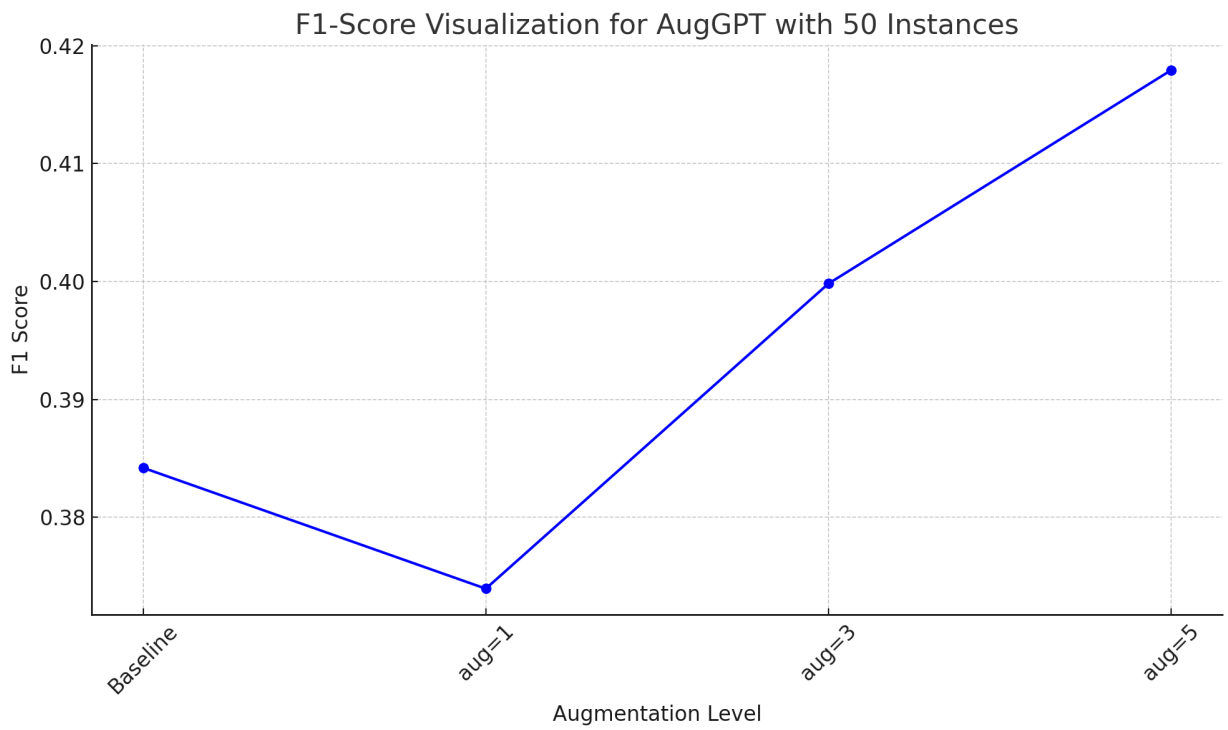


Figure 6.7: F1-score comparison with a line chart for AugGPT with a small-sized dataset

6.4.2 Medium training set

In the research that included a medium-sized dataset, we examined how the method AugGPT affects F1-scores at different augmentation levels. The outcomes are shown in Figure 6.8 and Table 6.11. The data shows a fluctuation in F1-scores depending on the augmentation level.

With an augmentation level of 1, the F1-score increased to 0.482, which resulted in an improvement of 8.27% over the baseline. Increasing the augmentation level to 3 led to a more substantial improvement, with the F1-score rising to 0.550, a 23.61% increase compared to the baseline. At the highest tested augmentation level of 5, the F1-score was 0.540, which is a 21.34 improvement over the baseline. Although slightly lower than the F1-score at augmentation level 3.

AugGPT Augmentation Methods	**F1-score**	**Percentage Change**
Baseline	0.445	-
Augmentation Level=1	0.482	8.27%
Augmentation Level=3	0.550	23.61%
Augmentation Level=5	0.540	21.34%
Average	**0.524**	**17.74%**

Table 6.11: This table displays F1-scores and their percentage increases across different levels of augmentation in the AugGPT method. The table illustrates the impact each parameter setting had on a medium training set compared to the baseline.

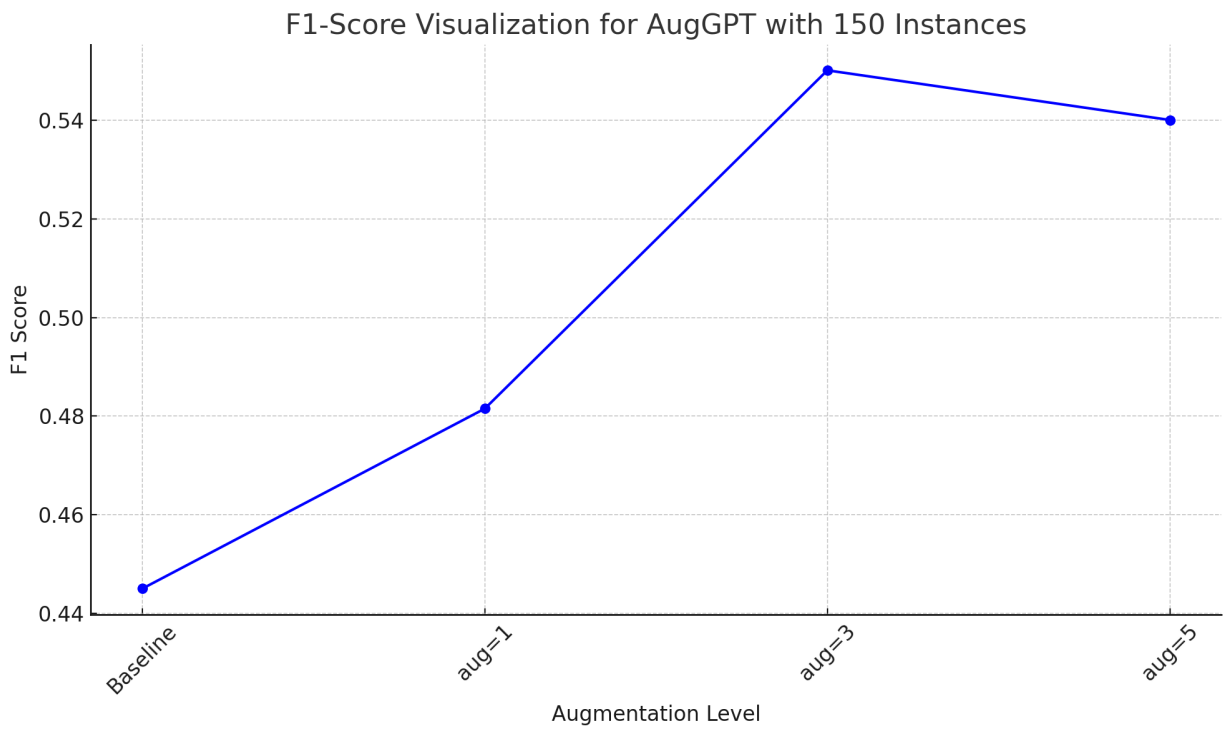


Figure 6.8: F1-score comparison with a line chart for AugGPT with a medium-sized dataset

6.4.3 Large training set

In the AugGPT experiment with a large-sized dataset, the impact of different augmentation levels on F1-scores was analyzed. The outcomes are illustrated in Figure 6.9 and Table 6.12. Here fluctuations in F1-scores were also observed across various parameter settings. At an augmentation level of 1, the F1-score saw a slight increase to 0.643, a modest improvement of 0.60% over the baseline. The most significant change was observed at an augmentation level of 3, where the F1-score rose to 0.700, which resulted in a 9.55% improvement compared to the baseline. Further increasing the augmentation to level 5 resulted in an F1-score of 0.645, which is a 0.95% improvement over the baseline. Although higher than the baseline, this level of augmentation did not result in as significant an improvement as seen at level 3.

AugGPT Augmentation Methods	**F1-score**	**Percentage Change**
Baseline	0.639	-
Augmentation Level=1	0.643	0.60%
Augmentation Level=3	0.700	9.55%
Augmentation Level=5	0.645	0.95%
Average	**0.663**	**3.68%**

Table 6.12: This table displays F1-scores and their percentage increases across different levels of augmentation in the AugGPT method. The table illustrates the impact each parameter setting had on a large training set compared to the baseline.

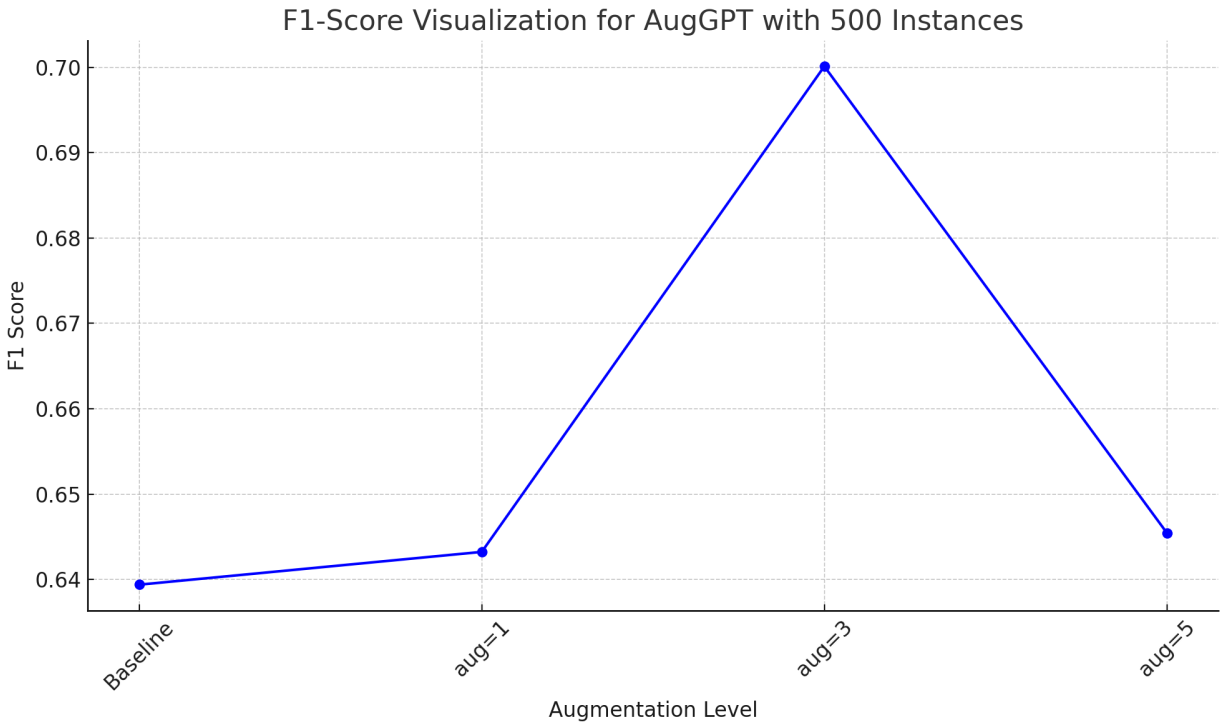


Figure 6.9: F1-score comparison with a line chart for AugGPT with a large-sized dataset

6.5 Method comparisons

For our thesis, we evaluated the effectiveness of three data augmentation techniques: Synonym Replacement, Mention Replacement, and AugGPT across various sizes of training sets. The results are detailed in Table 6.13 and Figure 6.10.

For small training sets, synonym replacement proved to be the most effective method, enhancing the baseline F1-score from 0.384 to 0.409. On the other hand, mention replacement was the least effective in this setting, actually decreasing the F1-score from the baseline. AugGPT was the second most effective, with a slightly less increase than what synonym replacement had. In medium-sized training sets, synonym replacement was shown to also be the most efficient, increasing the F1-score from 0.445 to 0.538. AugGPT was again the second most effective data augmentation method for the medium-sized training set, with a slightly less increase than what synonym replacement had. Conversely, mention replacement had a minimal positive impact in this setting, indicating that its benefits are not as useful as those of synonym replacement, and AugGPT medium training setting. For large training sets, synonym replacement again stood out, raising the F1-score from 0.639 to 0.728. In contrast, AugGPT was the least effective in large training settings, while still effective, and had a comparatively lesser impact.

Based on the results, synonym replacement was the most effective method for data augmentation in low-resource settings of different sizes, with the highest average increase.

	Small	Medium	Large	Avg.% Increase across datasets	Avg. Δ
Baseline	0.384	0.445	0.639	-	-
Synonym Replacement	0.409	0.538	0.728	13.80%	0.069
Mention Replacement	0.343	0.452	0.675	-1.19 %	0.00067
AugGPT	0.397	0.524	0.663	8.27 %	0.0387

Table 6.13: Combined F1-score Comparison for synonym replacement, mention replacement, and AugGPT across different sizes of training sets, showing average increases and delta changes.

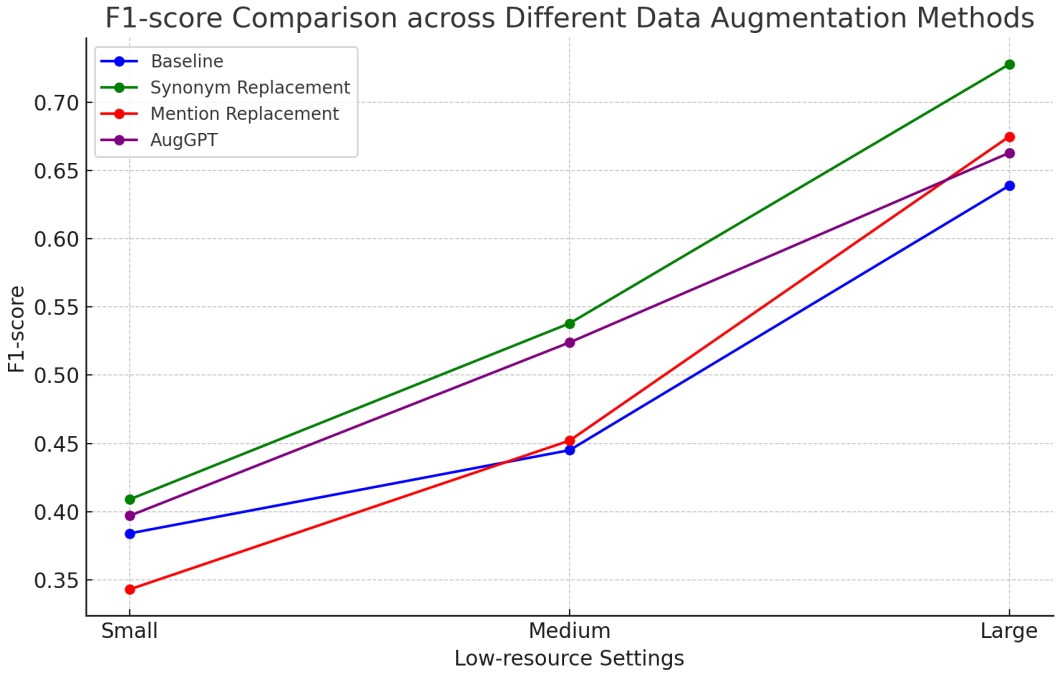


Figure 6.10: Line chart for combined F1-score Comparison for synonym replacement, mention replacement, and AugGPT

6.6 Cosine similiarty

For this section, we will display our findings on the impact of our augmentation method on the model's performance on our three low-resource setting datasets, which was accomplished by cosine similarity.

6.6.1 Small training set

Figure 6.11 illustrates a scatter plot with a small-sized dataset, where each point represents the performance of a different method and parameter setting in terms of cosine similarity and F1-score. In this context, synonym replacement (SYNONYM) appears to be the most effective method, achieving the highest scores in both cosine similarity, with an average of 0.937, and F1-score with an average of 0.409.

On the other hand, AugGPT, which uses a moderate level of text alteration, ranks second in the f1-score average. With an average cosine similarity of 0.847 and an F1-score of 0.397. However, mention replacement (Mention) has the lowest F1-score average, scoring the lowest with an average cosine similarity of 0.652 and an F1-Score of 0.343.

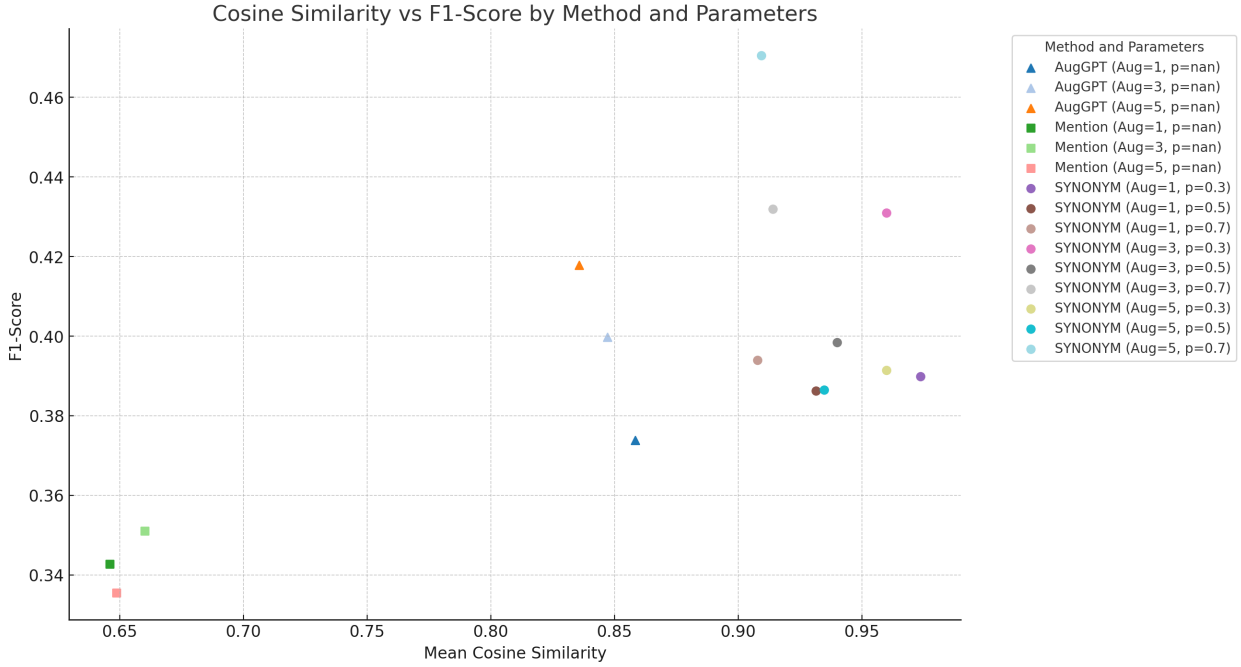


Figure 6.11: Cosine similarity for 50 instances

6.6.2 Medium training set

Figure 6.12, illustrating cosine similarity vs. F1-score with a medium-sized dataset, illustrates a scatter plot where each point represents the performance of a different method and parameter setting. From the data, the synonym replacement method appears to have the highest F1-score average, showing the highest mean cosine similarity score at 0.938 and also the highest average F1-score at 0.538. The mentioned

replacement method demonstrates the lowest performance with cosine similarity scores averaging at 0.666 and F1-scores at 0.452. AugGPT holds the second-highest scores in both cosine similarity and F1-score averages, with 0.875 and 0.524 respectively, only slightly behind the method with the highest average F1-score.

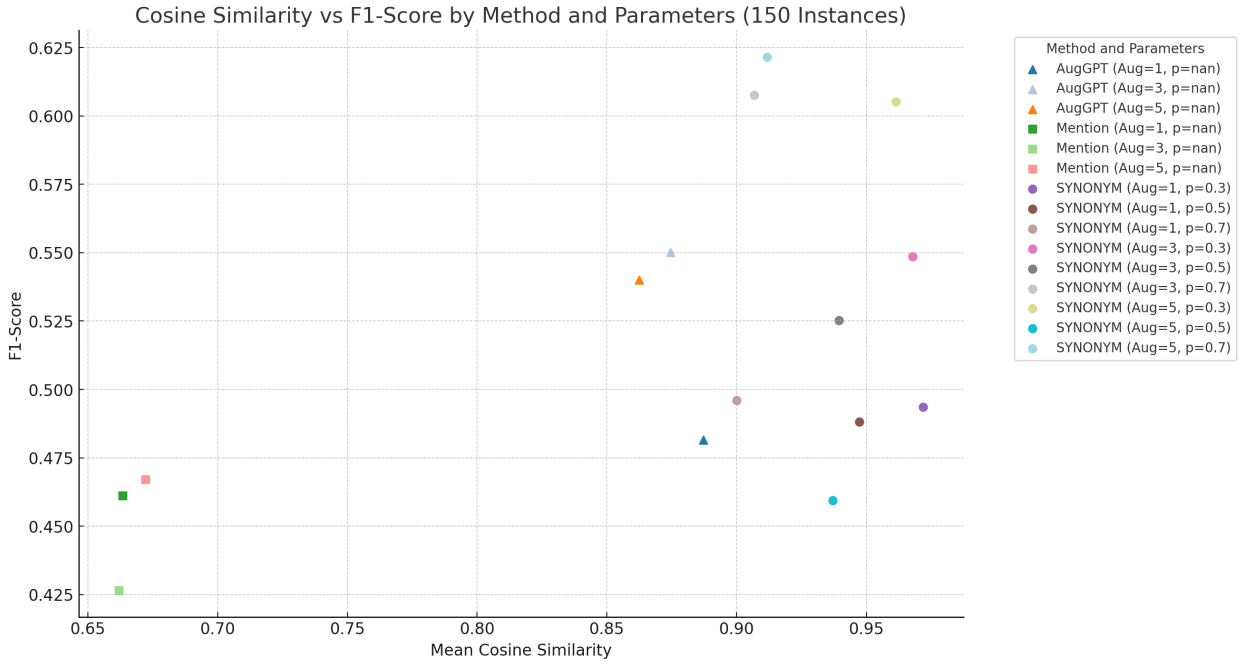


Figure 6.12: Cosine similarity for 150 instances

6.6.3 Large training set

Figure 6.13 presents a scatter plot with the performance of different methods and parameter settings, each represented by a point. Synonym replacement appears to also be the most effective method with a large-sized dataset, illustrating the highest average F1-score of 0.728 and the highest mean cosine similarity at around 0.944. On the other hand, mention replacement, despite its significantly lower mean cosine similarity score of about 0.639, manages to achieve an average F1-score of approximately 0.675. AugGPT achieves an average F1-score of around 0.656, which is quite close to that of mention replacement. However, it does so with a higher cosine similarity mean score of approximately 0.877.

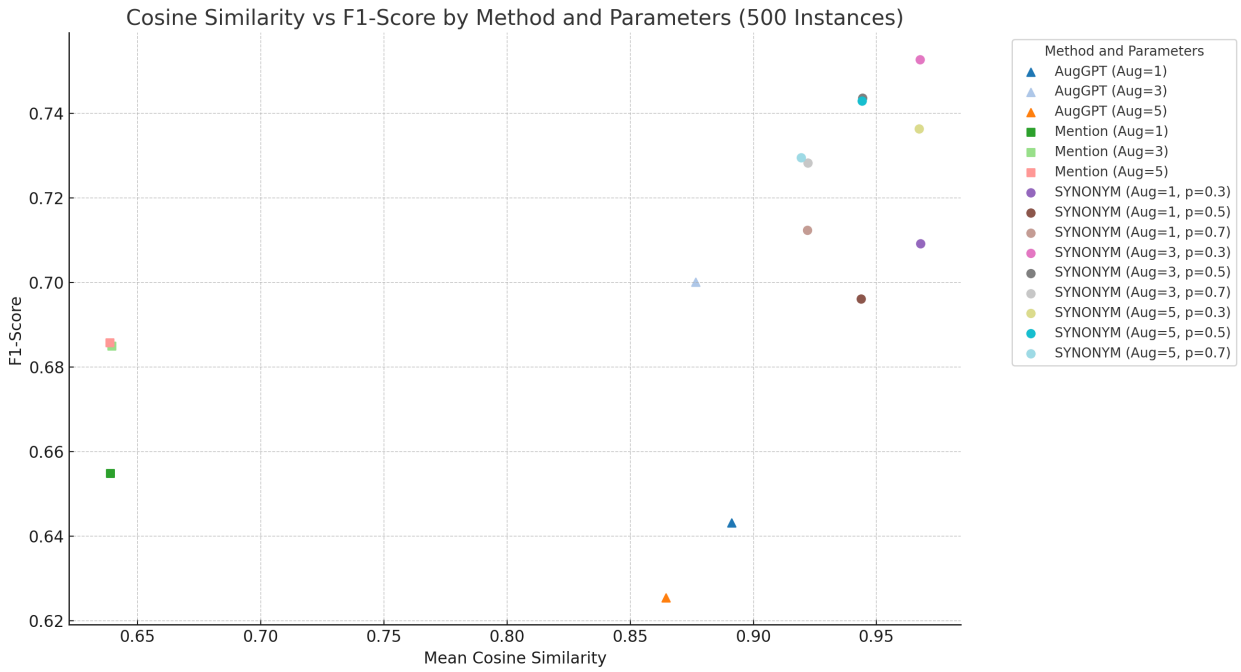


Figure 6.13: Cosine similarity for 500 instances

7

Discussion

This chapter will discuss the thesis, explaining the reasoning behind our decisions and the challenges encountered during the thesis.

7.0.1 Data augmentation in various low-resource settings

In this study, we explored the impact of different data augmentation methods on NER performance across various low-resource settings, categorized as small (50 instances), medium (150 instances), and large (500 instances). The aim was to identify which method is most effective in enhancing the accuracy of models when data is limited. The study illustrates that the effectiveness of data augmentation is connected to the size of the low-resource setting. While augmentation can significantly improve model performance in low-resource settings, the choice of the method needs to be chosen wisely for optimal results. It is also important to note that training on such limited sample sizes in the baseline scenarios can lead to variability in results. This variability can derive from the model's small exposure to diverse data, potentially leading to overfitting or underfitting. This factor is crucial to consider in evaluating the performance and robustness of NER models in low-resource situations.

Synonym replacement emerged as the most consistently effective method across all settings. The success of the method can be derived from a balance between the introduction of novel lexical variations and the preservation of the semantic content of the data. AugGPT, with its GPT-driven contextual augmentations, also shows promise, especially in settings that are small and medium. Mention replacement's mixed results highlight its dependency on dataset size and the potential risks of introducing less relevance for the KB-BERT model. We noticed that in a small setting, replacing synonyms has enhanced the model's performance. There was a 6.48% increase in F1-score, which shows that replacing words with synonyms from a rich source like Språkbanken's lexicon Swesaurus[42] can provide new variations, which helps models like KB-BERT to learn better in environments with limited resources. On the contrary, the mention replacement method resulted in a 10.68% decrease in F1-score. This could be due to the method introducing nonessential variations, which can confuse the model in such a small dataset, some examples can be seen in Appendix A. However, to fully understand the impact of these variations, a more detailed augmentation evaluation is needed on the quality of the new data. AugGPT showed a subtle improvement, indicating that contextually richer augmentations from GPT models can be beneficial, through small data datasets. In the

medium setting, the benefits of augmentation became more clear. Synonym replacement improved our model with a significant 21.00% increase in the F1-score. The improvement could be attributed to its ability to introduce meaningful variability without significantly altering the original data context, which seems essential for this setting-sized dataset. AugGPT also performed well, with a 17.74% increase, possibly due to its ability to generate contextually relevant data enhancements. The slight improvement seen with mention replacement suggests that as the dataset size increases, this method becomes more effective due to more significant variation in the dataset. For the large setting, all methods showed improvement over the raw dataset, but with a less noticeable increase. This result is expected as larger datasets already provided a significant amount of information for the model to learn from. Thus the marginal advantage of augmentation decreases. Synonym replacement again had the most significant improvements. The lesser improvements seen with AugGPT indicate that while beneficial, their impact is more damped in larger datasets, possibly due to the size of the dataset. Mention replacement in large resource-setting datasets, shows significant improvement compared to the previous resource-settings, thereby strengthening our hypothesis about its increased effectiveness in correlation with the expansion of dataset size.

When it comes to the exploration of optimal parameters for data augmentation methods, we couldn't determine a conclusion on what parameters are most promising, or regarding the spikes observed in the results for different parameters. The result was sometimes fluctuated in performance. A theory for some of the fluctuating results could be due to the inherent randomness in the methods applied. For instance, in synonym replacement, the method involves selecting a random synonym for the selected word. This process may potentially result in a synonym that is contextually wrong with the sentence's meaning. Such an anomaly could lead to a decline in the model's performance. Similarly, in mention replacement, where a word is replaced with another of the same entity class, the random selection could lead to not contextually suitable replacements. Such random replacements can significantly affect the data's quality and, consequently, the model's performance. Also upon a small inspection of the SWE-NERC data, some of the labels were found to be questionable. This is visible in Appendix A. The questionable labels could affect the performance of the replacement method mentioned. This is because the new replacement words may not fit the context correctly. The AugGPT method has also this issue of reliability due to the unpredictability of its outputs. The generative model does not always guarantee relevant augmentations, which derives from the inability to control the model's output, leading to potential insufficient augmentation. For example, an entity that should remain intact might be modified by the AugGPT method, causing it to lose its class label mapping. Thus, the new modified entity becomes an "O-class", which was described in section 5.2.1.3. This can also be seen in Appendix A.

7.0.2 NER AugGPT and traditional techniques

Another research question was to compare the effectiveness of AugGPT, a cutting-edge data augmentation method, to the more traditional methods of synonym re-

placement and mention replacement for Swedish NER. This comparison aimed to understand the relative efficacy of advanced AI-driven methods against established, simpler techniques in the context of a language-specific task.

The general result indicated that while AugGPT showed promise in certain aspects, traditional methods like synonym replacement often outperformed it in terms of overall reliability. Specifically, synonym replacement demonstrated a better performance in enhancing the NER model's F1-score. This can be due to its straightforward approach of substituting words with their synonyms, which introduces variability without significantly altering the context or semantic meaning of the sentences. Such a method is particularly effective in a language-specific task like Swedish NER, where maintaining the integrity of linguistic structures is crucial.

In contrast, AugGPT sometimes proved to be less reliable. For instance, as mentioned earlier, AugGPT can alter an entity in such a way that it no longer retains its original class label, resulting in the entity being reclassified as an 'O-class.', since we couldn't map it to the previous label. This unreliability can be due to the method's reliance on large language models that are not always fine-tuned to specific languages like Swedish and its lack of domain-specific knowledge. Moreover, the generative nature of AugGPT means it can create entirely new sentence structures or phrases that may not always align with the specific requirements of our tasks.

7.0.3 Cosine similarity versus F1-score

In analyzing data augmentation effects on various low-resource settings, a key focus was to understand how different degrees of cosine-similarity from the augmented dataset influence the F1-score metrics.

In small and medium low-resource settings, a notable positive correlation was observed between cosine similarity and the F1-score, which suggests that in these settings, where data is limited, maintaining a higher degree of similarity to the original dataset contributes positively to the F1-score. The observed correlation might derive from the fact that in smaller datasets, each data point yields a more significant impact on the model's learning process. Therefore, augmentations that closely resemble the original data help reinforce patterns, leading to better model performance as reflected in the F1-score. However, as the dataset size increased, the influence of cosine similarity on the F1-score began to diminish. This was particularly noticeable in the case of mention replacement in larger datasets. Despite having a lower cosine similarity score than AugGPT, the mentioned replacement had a higher F1-score. One hypothesis, for this cause could be due to overfitting. In small datasets, models are at a higher risk of overfitting to the limited data available. Hence, augmentations that closely relate to the original data can be beneficial if the new augmented data doesn't reflect the real-world data, which is tested with the test set. In contrast, in larger datasets, the risk of overfitting poor data is lower due to the variety and volume of data. Here, introducing more diverse data might not harm the model's generalization ability. Thus, not greatly affecting the F1-score.

8

Conclusion

This chapter will include a conclusion of the thesis, summarizing our findings, which will end with suggestions for future work and improvements.

8.1 Conclusion

This study aimed to investigate the effectiveness of different data augmentation methods in low-resource settings, focusing on three key aspects: identifying the most effective augmentation method across various low-resource settings, comparing the performance of the advanced AugGPT method with traditional techniques in enhancing Swedish NER model, and understanding the impact of data augmentation on F1-score metrics. Our findings revealed that synonym replacement was the most effective data augmentation method in various low-resource settings, achieving the highest F1-score increase in all of the settings. Meanwhile, AugGPT achieved the second-highest average F1-score, while mention replacement achieved the lowest average F1-score. We noted as the data volume increased, mention replacement showed notable improvements in larger datasets. When it comes to determining the optimal parameters for our data augmentation methods, we couldn't identify which parameters were the most optimal. In comparing AugGPT with traditional methods, we observed that AugGPT did not outperform synonym replacement in average F1-score across all datasets, but it outperformed mention replacement. This leads to the possibility that AugGPT might be a more effective data augmentation method compared to some traditional approaches. Our analysis for the final research question revealed that there might be a correlation between F1-score and cosine similarity in small and medium-sized low-resource datasets, but this correlation diminished in larger datasets.

8.2 Future work

In future work, it would be beneficial to further investigate the efficacy of data augmentation methods by applying each augmentation multiple times. This approach would provide a further understanding of their effectiveness. Our thesis applied each data augmentation method only once, but given that these methods can yield varying outputs, testing each method multiple times would likely yield more thorough insights.

Another future work could be to explore the method AugGPT with alternative prompting strategies and apply it to newer models such as GPT-4 or GPT-4 Turbo. The exploration of AugGPT with newer models like GPT-4 and GPT-4 Turbo could reveal new insights, given that these models are more complex and have shown to be more powerful [51].

In my thesis, I used ChatGPT as a tool for data augmentation, creating enhanced datasets to improve my NER model. However, an interesting direction for future work could be exploring the direct deployment of GPT models in making predictions for NER tasks. Due to time constraints, this potential application was not investigated in the current study.

Experimenting with different ways of prompting implies trying various types of questions or instructions to explore how the models respond when using ChatGPT for tasks like generating language. This approach could show new and more promising ways to use data augmentation with these advanced models. Additionally, there is a need for more extended research to determine if approaches like AugGPT could be suitable as data augmentation methods for NER tasks, this can be done by investigating different approaches. Furthermore, for this thesis, we have set some limitations in the scope of NLP models and datasets explored. Future studies should aim to address this by testing a variety of ML models. By expanding the range of models tested, future research can show whether the observed result of data augmentation is specific to a model or is generalized. The same principle also applies when experimenting with various datasets. There is also a need to experiment with a range of data augmentation techniques, which could uncover even more insights into which methods are most effective for Swedish language datasets, particularly in low-resource settings.

Bibliography

[1] X. Dai and H. Adel, "An analysis of simple data augmentation for named entity recognition," *arXiv preprint arXiv:2010.11683*, 2020.

[2] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[3] S. Y. Feng, V. Gangal, J. Wei, *et al.*, "A survey of data augmentation approaches for NLP," *arXiv preprint arXiv:2105.03075*, 2021.

[4] H. Dai, Z. Liu, W. Liao, *et al.*, "AugGPT: Leveraging ChatGPT for text data augmentation," *arXiv preprint arXiv:2302.13007*, 2023.

[5] *Swe-NERC*, `https://spraakbanken.gu.se/lb/resurser/swe-nerc/`, Accessed: 2023-07-18, 2023.

[6] P. P. Ray, "ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 121–154, 2023, ISSN: 2667-3452. DOI: `https://doi.org/10.1016/j.iotcps.2023.04.003`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S266734522300024X`.

[7] M. Malmsten, L. Börjeson, and C. Haffenden, "Playing with words at the national library of Sweden–making a Swedish BERT," *arXiv preprint arXiv:2007.01658*, 2020.

[8] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018.

[9] W. H. Delashmit, M. T. Manry, *et al.*, "Recent developments in multilayer perceptron neural networks," in *Proceedings of the seventh annual memphis area engineering and science conference, MAESC*, 2005, pp. 1–15.

[10] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," 2020.

[11] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," *Artificial intelligence review*, vol. 55, no. 1, pp. 291–322, 2022.

[12] Z. Xu, A. M. Dai, J. Kemp, and L. Metz, "Learning an adaptive learning rate schedule," *arXiv preprint arXiv:1909.09712*, 2019.

[13] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, SPIE, vol. 11006, 2019, pp. 369–386.

[14] K. C. Luk, J. E. Ball, and A. Sharma, "A study of optimal model lag and spatial inputs to artificial neural network for rainfall forecasting," *Journal of Hydrology*, vol. 227, no. 1-4, pp. 56–65, 2000.

[15] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[16] P. P. Ray, "ChatGPT : A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope," *Internet of Things and Cyber-Physical Systems*, 2023.

[17] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, 2021.

[18] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[19] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.

[20] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] S. R. Kundeti, J. Vijayananda, S. Mujjiga, and M. Kalyan, "Clinical named entity recognition: Challenges and opportunities," in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 1937–1945.

[23] R. Sharnagat, "Named entity recognition: A literature survey," *Center For Indian Language Technology*, pp. 1–27, 2014.

[24] T. Eftimov, B. Koroui Seljak, and P. Koroec, "A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations," *PloS one*, vol. 12, no. 6, e0179488, 2017.

[25] R. E. Vlas and W. N. Robinson, "Two rule-based natural language strategies for requirements discovery and classification in open source software development projects," *Journal of management information systems*, vol. 28, no. 4, pp. 11–38, 2012.

[26] B. Jehangir, S. Radhakrishnan, and R. Agarwal, "A survey on named entity recognitiondatasets, tools, and methodologies," *Natural Language Processing Journal*, vol. 3, p. 100 017, 2023.

[27] M. Abulaish and A. K. Sah, "A text data augmentation approach for improving the performance of cnn," in *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, IEEE, 2019, pp. 625–630.

[28] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *2018 IEEE symposium series on computational intelligence (SSCI)*, IEEE, 2018, pp. 1542–1547.

[29] C. Coulombe, "Text data augmentation made simple by leveraging NLP cloud apis," *arXiv preprint arXiv:1812.04718*, 2018.

[30] D. Pluec and J. najder, "Data augmentation for neural NLP," *arXiv preprint arXiv:2302.11412*, 2023.

[31] B. Li, Y. Hou, and W. Che, "Data augmentation approaches in natural language processing: A survey," *AI Open*, vol. 3, pp. 71–90, 2022.

[32] D. R. Beddiar, M. S. Jahan, and M. Oussalah, "Data expansion using back translation and paraphrasing for hate speech detection," *Online Social Networks and Media*, vol. 24, p. 100 153, 2021.

[33] T. Ek, C. Kirkegaard, H. Jonsson, and P. Nugues, "Named entity recognition for short text messages," *Procedia-Social and Behavioral Sciences*, vol. 27, pp. 178–187, 2011.

[34] L. Ahrenberg, J. Frid, and L.-J. Olsson, "A new resource for Swedish named-entity recognition," 2020.

[35] *Swedish BERT models*, `https://github.com/Kungbib/swedish-bert-models`, Accessed: 2023-07-18, 2023.

[36] G. G. ahin, "To augment or not to augment? a comparative study on text augmentation techniques for low-resource NLP," *Computational Linguistics*, vol. 48, no. 1, pp. 5–42, 2022.

[37] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," *arXiv preprint arXiv:1901.11196*, 2019.

[38] R. Erd, L. Feddoul, C. Lachenmaier, and M. J. Mauch, *Evaluation of data augmentation for named entity recognition in the german legal domain*, 2022.

[39] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "Flair: An easy-to-use framework for state-of-the-art NLP," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)*, 2019, pp. 54–59.

[40] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

[41] L. Tunstall, L. Von Werra, and T. Wolf, *Natural language processing with transformers*. " O'Reilly Media, Inc.", 2022.

[42] L. Borin and M. Forsberg, "From the peoples synonym dictionary to fuzzy synsets-first steps," in *Proceedings of the LREC 2010 workshop Semantic relations. Theory and Applications*, 2010, pp. 18–25.

[43] Synonymer API, *Synonymer api*, `https://synonymord.se/api/`, Accessed: 2023-11-15, 2023.

[44] D. Kokkinakis, J. Niemi, S. Hardwick, K. Lindén, and L. Borin, "HFST-SweNER — a new NER resource for Swedish," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, N. Calzolari, K. Choukri, T. Declerck, *et al.*, Eds., Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 2537–2543. [Online]. Available: `http://www.lrec-conf.org/proceedings/lrec2014/pdf/391_Paper.pdf`.

[45] D. Kokkinakis, "Evaluating the coverage of three controlled health vocabularies with focus on findings, signs and symptoms," *NEALT Proceedings Series, editor, NODALIDA*, vol. 12, pp. 27–31, 2011.

[46] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in PyTorch," 2017.

[47] T. Wolf, L. Debut, V. Sanh, *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[48] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[49] PyPI, *Seqeval*, `https://pypi.org/project/seqeval/0.0.10/`, version 0.0.10, Accessed: 2023-11-15, 2019.

[50] F. Rahutomo, T. Kitasuka, and M. Aritsugi, "Semantic cosine similarity," in *The 7th international student conference on advanced science and technology ICAST*, vol. 4, 2012, p. 1.

[51] OpenAI, *Gpt-4 technical report*, 2023. arXiv: `2303.08774 [cs.CL]`.

A

Appendix Augmentation examples

Here are the results of our augmentation on four different sentence examples with our 3 augmentation methods.

Sentence	**Labels**
Original	
från och med imorn blir de roligare inlägg, musik, mode och ja ni fattar !	['O', 'O', 'O', 'TME', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Augmentation Methods	
Mention replacement från och med augusti blir de roligare inlägg, musik, mode och ja ni fattar !	['O', 'O', 'O', 'TME', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
AugGPT Imorgon är dagen då inläggen kommer att bli roligare med inslag av musik, mode och mycket annat - ni fattar!	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Synonym replacement with p=0.3 av och med imorn blir de roligare inlägg, musik, mode och ja ni fattar !	['O', 'O', 'O', 'TME', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Synonym replacement with p=0.5 av samt tillsammans imorn blir dom roligare kommentar, musik, mode och ja du fattar !	['O', 'O', 'O', 'TME', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Synonym replacement with p=0.7 av samt tillsammans imorn blir dom roligare kommentar, musik, mode samt givetvis ni fattar !	['O', 'O', 'O', 'TME', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

Table A.1: Data Augmentation for Original Sentence 1

Sentence	**Labels**
Original 2	
Äldre barn kan få vanliga näsdroppar.	['O', 'O', 'O', 'O', 'MNT', 'MNT', 'O']
Augmentation Methods	
Mention replacement Äldre barn kan få tenutexkurer agglutinationstest.	['O', 'O', 'O', 'O', 'MNT', 'MNT', 'O']
AugGPT Äldre barn kan ofta använda vanliga näsdroppar för att lindra förkylningssymtom.	['O', 'O', 'O', 'O', 'MNT', 'MNT', 'O', 'O', 'O']
Synonym replacement with p=0.3 Äldre barn kan få vanliga näsdroppar.	['O', 'O', 'O', 'O', 'MNT', 'MNT', 'O']
Synonym replacement with p=0.5 Äldre unge kan ringa vanliga näsdroppar.	['O', 'O', 'O', 'O', 'MNT', 'MNT', 'O']
Synonym replacement with p=0.7 Äldre barnunge kan erhålla vanliga näsdroppar.	['O', 'O', 'O', 'O', 'MNT', 'MNT', 'O']

Table A.2: Data Augmentation for Original Sentence 2

Sentence	**Labels**
Original 3	
I Sverige får föräldrar bekosta rotavirusvaccinationen själva.	['O', 'LOC', 'O', 'O', 'O', 'MNT', 'O', 'O']
Augmentation Methods	
Mention replacement I Kristianstad får föräldrar bekosta pleurostomi själva.	['O', 'LOC', 'O', 'O', 'O', 'MNT', 'O', 'O']
AugGPT Det är vanligt i Sverige att föräldrar själva får stå för kostnaden för rotavirusvaccinationen.	['O', 'O', 'O', 'O', 'LOC', 'O', 'O', 'O', 'O', 'O', 'O', 'O' 'MNT', 'O']
Synonym replacement with p=0.3 I Sverige får föräldrar bekosta rotavirusvaccinationen solo.	['O', 'LOC', 'O', 'O', 'O', 'MNT', 'O', 'O']
Synonym replacement with p=0.5 I Sverige får föräldrar bekosta rotavirusvaccinationen solo.	['O', 'LOC', 'O', 'O', 'O', 'MNT', 'O', 'O']
Synonym replacement with p=0.7 inom Sverige får föräldrar bekosta rotavirusvaccinationen solo.	['O', 'LOC', 'O', 'O', 'O', 'MNT', 'O', 'O']

Table A.3: Data Augmentation for Original Sentence 3

B

Appendix AugGPT Prompt

Prompt:
Skriv om meningen "X" gånger och behåll orden [entity 1, entity 2, etc...]. Numerera.

Here, X is the number of times you want to augment a sentence, and the words inside the "[]" are the entities, which should be preserved.