ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«JOB SKILLS EXTRACTION FROM JOB LISTINGS AND MAPPING TO KNOWN TAXONOMIES USING DEEP LEARNING TECHNIQUES»

Των φοιτητών

Καζλάρη Ιωάννη (05/2022)

και

Φυλαχτού Χρήστου (16/2022)

Επιβλέπων

Ονοματεπώνυμο: Κ. Διαμαντάρας

Βαθμίδα: Καθηγητής

Τίτλος Δ.Ε.: Job skills extraction from job listings and mapping to known taxonomies using deep learning techniques

Κωδικός Δ.Ε.: 23352

Ονοματεπώνυμο φοιτητών: Καζλάρης Ιωάννης, Φυλαχτός Χρήστος

Ονοματεπώνυμο εισηγητή: Διαμαντάρας Κωνσταντίνος

Ημερομηνία ανάληψης Δ.Ε. 07.12.2023

Ημερομηνία περάτωσης Δ.Ε. 29.06.2024

Βεβαιώνουμε ότι είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων των συγγραφέων, εκ μέρους του Τμήματος.

DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Job skills extraction from job listings and mapping to known taxonomies using deep learning techniques

MSc Thesis

Kazlaris Ioannis, Fylachtos Christos

Supervisor: **Dr. Diamantaras Konstantinos**
Professor, International Hellenic University

Thessaloniki, June 2024

This page is deliberately left blank.

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Αναγνώριση δεξιοτήτων σε κείμενα προσφοράς θέσεων εργασίας και αντιστοίχιση σε γνωστές ταξινομίες με χρήση βαθιάς μάθησης

Μεταπτυχιακή Διπλωματική Εργασία

Καζλάρης Ιωάννης, Φυλαχτός Χρήστος

Επιβλέπων : Δρ. Διαμαντάρας Κωνσταντίνος
Καθηγητής, Διεθνές Πανεπιστήμιο της Ελλάδας

Approved by the following three-member examination committee on 29 Ιουνίου 2024.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Όνομα Επώνυμο Choose an item. ΔΙ.ΠΑ.Ε.	Όνομα Επώνυμο Choose an item. ΔΙ.ΠΑ.Ε.	'Ονομα Επώνυμο Choose an item. ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Ιούνιος 2024

(Υπογραφή 1)

.....

Καζλάρης Ιωάννης

Μηχανικός Πληροφορικής ΑΤΕΙΘ

i.kazlaris@iee.ihu.gr

(Υπογραφή 2)

.....

Φυλαχτός Χρήστος

Μηχανικός Πληροφορικής ΑΤΕΙΘ

chrifyla@iee.ihu.gr

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας για την αμέριστη συμπαράσταση που μας παρείχαν καθ' όλη τη χρονική διάρκεια εκπόνησης της διπλωματικής μας εργασίας καθώς και τον επιβλέποντα καθηγητή μας Δρ. Διαμαντάρα Κωνσταντίνο για τις πολύτιμες συμβουλές και την καθοδήγηση του.

Περίληψη

Η παρούσα μεταπτυχιακή εργασία επικεντρώνεται στον εντοπισμό και την εξαγωγή οντοτήτων που σχετίζονται με δεξιότητες από αγγελίες εργασίας, και στην αντιστοίχιση αυτών των οντοτήτων σε ταξονομίες χρησιμοποιώντας προηγμένες τεχνικές βαθιάς μάθησης. Τα αρχικά κεφάλαια της εργασίας αναλύουν την εξέλιξη των νευρωνικών δικτύων, από τα απλά Perceptron μέχρι τα σύγχρονα νευρωνικά δίκτυα τύπου Transformers, και προσφέρουν μια βιβλιογραφική ανασκόπηση στην αναγνώριση οντοτήτων και σημασιολογικής ομοιότητας στο πεδίο της Φυσικής Επεξεργασίας Γλώσσας, εξετάζοντας διάφορες μεθοδολογίες και τεχνολογίες. Στην ενότητα μεθοδολογίας, περιγράφεται αναλυτικά ο τρόπος με τον οποίο διενεργήθηκε η έρευνα, από τον εντοπισμό και καθαρισμό των δεδομένων έως τη χρήση εργαλείων και βιβλιοθηκών για την επεξεργασία και ανάλυση τους. Επίσης, αναφέρονται οι στατιστικές μετρήσεις και η παραγωγή σχεδιαγραμμάτων για την κατανόηση και παρουσίαση των αποτελεσμάτων, προσδίδοντας σημασία στην πρακτική εφαρμογή των ευρημάτων στον τομέα των ανθρώπινων πόρων και της διαχείρισης εργασίας.

Λέξεις Κλειδιά: <<νευρωνικά δίκτυα, transformers, φυσική επεξεργασία γλώσσας, εξαγωγή και αναγνώριση οντοτήτων, σημασιολογική ομοιότητα, δεξιότητες εργασίας>>

This page is deliberately left blank.

Abstract

This MSc thesis focuses on identifying and extracting skill-related entities from job postings, and mapping these entities to established taxonomies using advanced deep learning techniques. The initial chapters analyze the evolution of neural networks, from the simple Perceptron to the modern Transformer-type neural networks, and offer a literature review in entity recognition and semantic similarity in the field of Natural Language Processing, examining various approaches and technologies. The methodology section describes in detail how the research was conducted, starting from identifying and cleaning the datasets as well as the tools and libraries that have been used to process and analyze it. Statistical measurements and diagrams are also given that further the understanding and presentation of the results, giving importance to the practical application of the findings in the field of human resources and labor management.

Keywords: "Neural networks, transformers, natural language processing, named entity recognition, semantic similarity, job skills"

Η σελίδα αυτή είναι σκόπιμα λευκή.

Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Deep Learning and Job Listings

The field of deep learning pertinent to our master thesis, titled "Job Skills Extraction from Job Listings and Mapping to Known Taxonomies Using Deep Learning," revolves around the application of advanced machine learning techniques to process and analyze textual data from job listings. This specialized area within deep learning focuses on the extraction and classification of unstructured text data into predefined categories or taxonomies. The primary goal is to accurately identify specific job skills required by employers and map these skills onto established skill frameworks. This process not only aids in understanding the demand for certain skills across various industries but also helps in structuring vast amounts of job-related data into actionable insights, facilitating better decision-making for both job seekers and employers.

One of the primary challenges in this field is the inherent complexity of natural language processing (NLP). Text in job listings is often informal and highly contextual, with a significant amount of industry-specific jargon, acronyms, and implicit requirements. The variability in how different job listings describe similar skills adds another layer of difficulty in accurately extracting and categorizing skills. Additionally, the evolving nature of job markets and the continuous emergence of new skills pose a challenge in keeping the taxonomy up-to-date. Deep learning models must be robust enough to handle these nuances and adapt to new patterns in data, ensuring they remain relevant and accurate over time.

Effectively mapping the extracted skills to known taxonomies is another significant challenge. This task involves not just identifying skill names but also understanding the context in which they are used, their relevance to particular job roles, and their relationships with other skills. Accurate mapping requires sophisticated algorithms capable of capturing the semantic meaning and context of skills, which is essential for generating reliable and useful insights. Moreover, the process must account for variations in terminology and usage across different regions and industries, further complicating the task.

To address these challenges, our approach leverages state-of-the-art deep learning techniques, including transformer models and other advanced NLP frameworks, which have shown great promise in handling complex language tasks. These models are trained on vast amounts of data to recognize patterns and make sense of the unstructured text found in job listings. By employing such cutting-edge technology, we aim to develop a robust system that can accurately extract job skills and map them to known taxonomies, ultimately contributing to a more organized and insightful understanding of job market demands.

1.2 Objective of our Thesis

The objective of the thesis is to develop a deep learning-based framework that can automate the extraction of job skills from diverse job listings and intelligently map these skills onto the ESCO taxonomy (European Skills, Competences, Qualifications and Occupations). This involves training deep learning models that can learn from a broad dataset of job descriptions and accurately reflect the complexity and interconnected nature of the job market's skill requirements. Such a framework would not only streamline human resource processes but also enhance educational and training programs by aligning them more closely with market needs.

1.2.1 Contribution

This thesis aims to make significant contributions to the field of natural language processing (NLP) by showcasing the application of transformer models for entity extraction and semantic similarity analysis in the context of job skills. Our methodology has the potential for broader application across various academic fields, such as educational technology and organizational studies, where understanding competencies and skills is crucial. Specifically, the techniques we have developed could serve as a foundation for novel methods or enhancements to existing approaches. Additionally, by collecting and annotating job descriptions for skill extraction, this thesis creates a valuable dataset that can benefit other researchers within the NLP community.

Our thesis also aspires to make meaningful contributions to the field of Human Resources (HR), particularly in automating and refining the process of matching candidates to job openings. By extracting specific skills from job descriptions and aligning them with candidate profiles, our research aims to enhance the speed and efficiency of recruitment workflows. Mapping these extracted skills to established taxonomies, such as ESCO, ISCO, NOC, O*NET, and others, can help HR professionals and organizations craft job descriptions that are more precise and aligned with industry standards. Furthermore, this approach can assist in identifying current skills gaps within their workforce, leading to clearer communication of job requirements and more targeted training and development programs.

Ultimately, our work aims to provide a dual impact: advancing the technical capabilities of NLP in skill extraction and semantic similarity, while simultaneously offering practical solutions to improve HR practices. By bridging these fields, our research has the potential to foster innovation and efficiency in both academia and industry, contributing to a more systematic understanding and utilization of job skills and competencies.

1.3 Thesis layout

Chapter 1 provides a brief historical overview of the evolutionary trajectory of neural networks, starting from the Perceptron and concluding with Transformer-type neural networks.

Chapter 2 conducts a literature review in the field of entity recognition within Natural Language Processing (NLP), analyzing concepts such as rule-based systems, Conditional Random Fields, Support Vector Machines, Hidden Markov Models, RNNs-LSTMs-CNNs, and transformers, and how each of these systems can be utilized in recognizing and extracting entities from texts.

Chapter 3 reviews the field of semantic similarity, offering a brief historical recap of the evolutionary path of neural networks.

Chapter 4 extensively presents the methodology we followed, which includes among others:

- The identification, preprocessing, cleaning and tagging of the datasets used both for the entity recognition subsystem and for the semantic similarity subsystem.
- The extraction of statistical measurements through the Explorative Data Analysis process, as well as the production of diagrams and comparative tables to depict these statistical measurements and other results.
- The tools used (Python, SpaCy libraries and transformer model (based on RoBERTa), NLTK, Label Studio, plotly, matplotlib, Hugging Face sentence transformers such as the all-mpnet-base-v2, the msmarco_distilbert_cos_v5 model as well as Google's sentence-t5-base)

This Master's Thesis concludes with the citation of relevant literature and ends with appendices that include tables, diagrams, as well as ablations studies.

2 From Perceptrons to Transformers

Neural Networks have undergone an evolutionary journey since their inception, morphing from simple Perceptrons capable of executing basic tasks to Transformers that define the frontiers of the research in Artificial Intelligence. This evolution showcases the dynamic nature of AI research, with each leap forward addressing limitations of prior models and opening new avenues of exploration as well as marking a significant shift in the applications of artificial intelligence across various domains. This review will explore this chronological journey, highlighting both the innovative approaches that each of the major architectures contributed to the field of AI as well as the advantages and disadvantages of each implementation.

2.1 Historical Context and Evolution

The concept of neural networks was inspired by the human brain's structure and functioning, aiming to mimic its ability to learn from experience. The initial model, the Perceptron, was introduced by Frank Rosenblatt in 1958, marking the inception of machine learning (ML) models capable of binary classifications based on weighted inputs [1]. The Perceptron's limitations stem from its nature as a linear classifier and its inherent inability to solve non-linear problems (such as the XOR problem) which led to a decreased interest in artificial intelligence for almost a decade.

The renewal of interest appeared with the introduction of backpropagation by Rumelhart, Hinton, and Williams in 1986, which enabled the training of multi-layer Perceptrons (MLPs) by adjusting weights in response to errors, significantly enhancing the learning capabilities of neural networks [2]. This advancement paved the way for the development of a variety of neural network architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which have become fundamental to progressing fields such as computer vision and natural language processing (NLP). What follows is a brief description of the most prominent architectures in Neural Networks which led to the development of Transformers.

2.2 Multilayer Perceptrons (MLPs)

MLPs consist of an input layer, one or more hidden layers, and an output layer. Each layer is comprised of nodes or neurons, with each neuron in one layer connected to every neuron in the next layer, facilitating a forward propagation of inputs through the network [3]. The backpropagation algorithm adjusts the weights of these connections to minimize the error between the predicted and actual outputs, enabling the network to learn. MLPs have significantly contributed to the advancement of artificial intelligence by enabling the modeling of complex, non-linear relationships between inputs and outputs, which was not possible with single-layer Perceptrons.

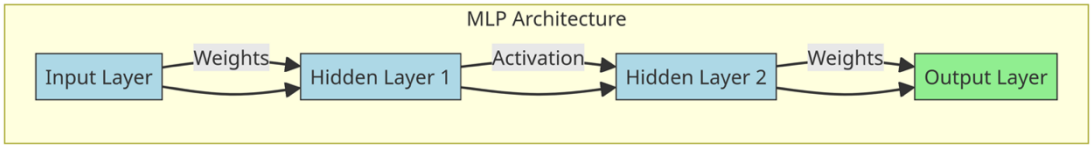

Figure 1: Multilayer Perceptron Architecture

This capability has led to breakthroughs in various fields such as speech recognition, image classification, and natural language processing, marking MLPs as foundational to the development of deep learning. However, MLPs come with their own set of drawbacks, including the tendency to overfit data if not properly regularized, the computational expense associated with training large networks, especially on massive datasets, and the challenge of optimizing their architecture, including the number of layers and nodes, which often requires extensive experimentation and expertise. Additionally, their "black box" nature makes it difficult to interpret the decision-making process of the network, posing challenges in applications where explainability is critical.

2.3 Convolutional Neural Networks (CNNs)

Introduced by LeCun et al. [4], Convolutional Neural Networks (CNNs) are specifically designed to excel in processing spatial data arranged in a grid, especially images. Their unique architecture consists of three key layers: convolutional layers, pooling layers, and fully connected layers. This collaborative interplay allows them to achieve state-of-the-art performance in tasks like image recognition and classification, as demonstrated by Krizhevsky et al. [5]. The first layer, the convolutional layer, utilizes filters to identify and highlight specific features within the input data. Pooling layers (maximum or average pooling) then downsample the extracted information, improving efficiency and focusing on the most significant features. Finally, fully connected layers receive the processed data and use it to classify the image or make predictions. This combination of feature extraction, dimensionality reduction, and interpretation empowers CNNs to excel in various domains, revolutionizing fields like computer vision.

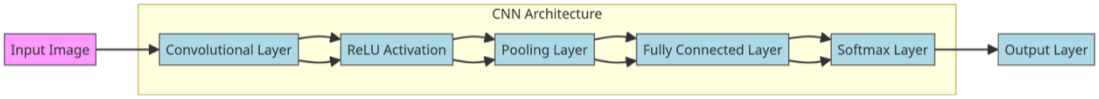

Figure 2: Convolutional Neural Network Architecture

CNNs are highly effective at recognizing patterns due to their ability to capture spatial hierarchies, leading to their use in critical applications like autonomous driving and medical diagnostics. However, they demand significant computational resources for training, posing challenges for smaller projects. Deep convolutional networks which are implemented by stacking in various configurations hundreds of layers are particularly susceptible to the problem of vanishing gradients, where gradients become too small for effective learning in lower layers during backpropagation, making the training of very deep architectures quite challenging without specialized techniques [2]. Due to this complex nature, it is oftentimes difficult to interpret decision-making processes, which can be a drawback in scenarios requiring transparency.

2.4 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a specific type of neural network designed to tackle the challenges of handling sequential data as well as making predictions based on a sequence of frames.

Unlike traditional feedforward networks, where information flows in one direction, RNNs incorporate loops in their architecture, allowing them to retain information from previous steps or states. The recursive nature of this "memory" element makes them particularly adept at tasks like language modeling, where understanding the context of preceding words is crucial, and speech recognition, where the meaning of a sound depends on those preceding it. RNNs have been instrumental in advancing reinforcement learning, where they help agents to make sense of the environment in which they are found, and also in generative models as they can produce sequences of text, music, or speech that are coherent over long stretches.

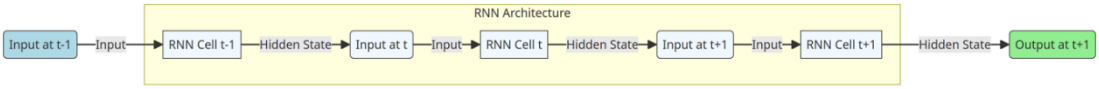

Figure 3: Recurrent Neural Network Architecture

While RNNs offer significant advantages for sequential data, they can also suffer from vanishing and exploding gradients, meaning information either fades away or becomes amplified as it travels through the network, hindering their ability to learn long-term dependencies [2]. To address these challenges, Hochreiter & Schmidhuber have introduced Long Short-Term Memory (LSTM) units [6] whereas Cho et al. [7] proposed Gated Recurrent Units (GRUs). These variations on the RNN architecture incorporate mechanisms to control the flow of information, allowing them to learn and remember information over extended periods, significantly enhancing their capabilities.

2.5 Restricted Boltzmann Machines (RBMs)

Restricted Boltzmann Machines (RBMs) are a type of stochastic neural network that belongs to the family of energy-based models. They are used for dimensionality reduction, classification, regression, collaborative filtering, feature learning, and topic modeling. An RBM consists of two layers: a visible layer that represents the observed variables (data) and a hidden layer that captures latent variables (features). These two layers are fully connected, but there are no connections within a layer, a restriction that simplifies the learning process. This architecture allows RBMs to learn a probability distribution over the input data. The learning in RBMs is performed through a process called contrastive divergence, where the model's parameters are adjusted to minimize the difference between the observed data distribution and the model's distribution [8].

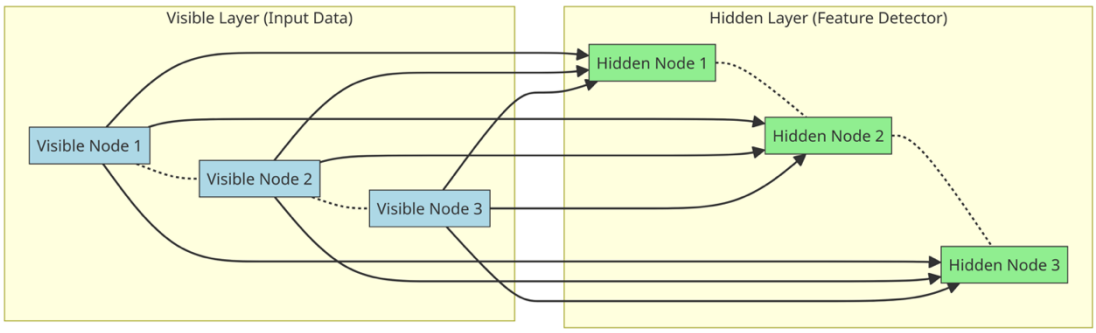

Figure 4: Restricted Boltzmann Machine Architecture

RBMs have been foundational in the development of deeper neural networks, particularly in the pre-training of deep belief networks (DBNs). By stacking multiple RBMs and fine-tuning with

backpropagation, researchers have been able to train deep networks effectively, a process that was challenging in the early days of deep learning due to the vanishing gradient problem. Despite their success in the early 2000s, the popularity of RBMs has waned with the rise of other deep learning architectures, such as CNNs and RNNs, which directly learn from labeled data in a supervised manner and have shown superior performance in tasks like image and speech recognition. Nonetheless, RBMs remain an important part of the neural network landscape, particularly in unsupervised learning scenarios where understanding the underlying distribution of data is crucial [9].

2.6 Generative Adversarial Networks (GANs)

The concept of GANs was introduced by Goodfellow et al [10]. in 2014 and represents a significant advancement in the field of generative models. GANs consist of two distinct neural networks that are trained against each other in a scenario that resembles a game: a generator network, which learns to generate data resembling a particular distribution by mimicking the distribution of a genuine dataset, and a discriminator network, which functions as a judge and learns to distinguish between genuine data and data produced by the generator. During training, the generator tries to produce increasingly realistic data, while the discriminator becomes increasingly better at detecting fakes. This process continues until the generator produces data so close to the real thing that the discriminator can no longer reliably tell the difference, achieving a state known as Nash equilibrium. The architecture of both the generator and discriminator in a GAN can vary, often employing deep neural networks that might include convolutional layers, depending on the application (e.g., image generation).

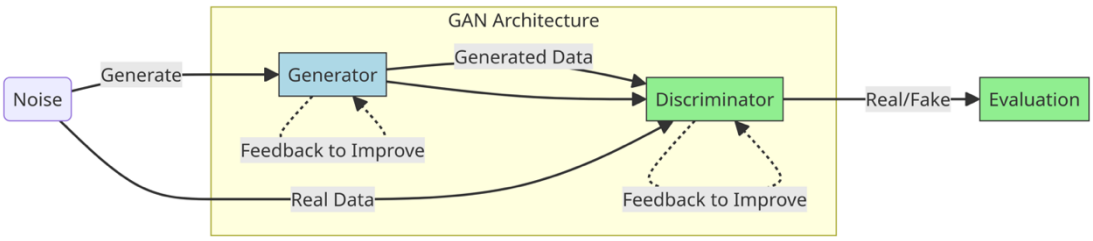

Figure 5: Generative Adversarial Network Architecture

GANs excel at their ability to generate high-quality, realistic data across a variety of domains such as images, videos, and text. This inherent ability of GANs makes them a prime candidate for data augmentation, aiming at improving the performance of machine learning models by generating additional training data, and for unsupervised learning, discovering intricate patterns in data with no labels. Training GANs, however, is notoriously difficult and resource-intensive, often requiring fine-tuning and a careful balance between the generator and discriminator to avoid issues like mode collapse, where the generator produces a limited variety of outputs. Additionally, the potential for misuse in creating deep fakes raises ethical concerns, highlighting the need for responsible use and development of GAN technology to mitigate risks associated with privacy, security, and misinformation.

2.7 Autoencoders

Autoencoders are a type of neural network used for unsupervised learning, primarily for the tasks of data compression, feature extraction, and noise reduction. They work by encoding input data into a compressed representation and then decoding that representation back to a reconstruction of the original input. This process of learning to encode the input data in an efficient and compact form is akin to dimensionality reduction [11]. The network is trained to minimize the difference between the input and

its reconstruction, learning to capture the most important features in the compressed representation. Variants such as Variational Autoencoders (VAEs) and Denoising Autoencoders extend the basic model to generate new data or remove noise from data.

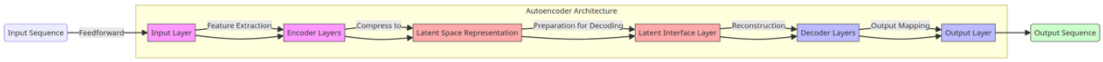

Figure 6: Autoencoder Architecture

One of the main advantages of autoencoders is their ability to learn data-specific compressions, which can result in more efficient representation than generic compression algorithms. They are also capable of removing noise from the input data, making them useful in data preprocessing. However, autoencoders can sometimes be too data-specific, failing to generalize well to unseen data, and the training process can be sensitive to the choice of hyperparameters. Additionally, the loss of information during the compression phase can sometimes lead to suboptimal reconstruction quality, especially if the network has not been properly trained or the architecture is not suitable for the type of data being processed.

2.8 Long Short-Term Memory Networks (LSTMs)

The introduction of Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber [6] presented a groundbreaking solution to the vanishing gradient problem inherent in traditional RNNs. LSTMs are a special kind of RNNs particularly well-suited to sequential data analysis. They feature a unique memory cell that can maintain information in memory for long periods, thanks to gate mechanisms that control the cell's ability to retain or forget information, thereby allowing it to selectively remember patterns over long sequences. This ability to remember information for long periods is critical in applications such as language translation, speech recognition, and time series forecasting where context from much earlier inputs significantly influences the output. By efficiently addressing the vanishing gradient problem, LSTMs maintain robustness in learning over long sequences, making them superior to traditional RNNs for complex sequence modeling tasks.

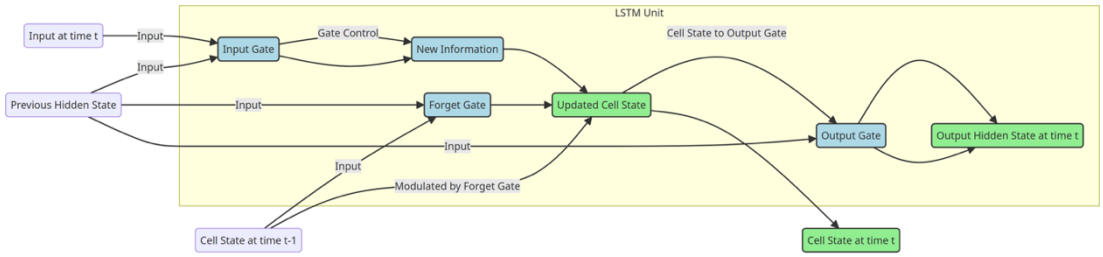

Figure 7: Long Short-Term Memory Network Architecture

It should be noted that the sophistication of LSTMs comes with higher computational costs and longer training times, especially as the sequence length and model complexity increase. Additionally, their intricate internal structure can make them less interpretable than simpler models, posing challenges in applications where understanding the model's decision process is essential. Despite these drawbacks, LSTMs remain a popular choice for tasks requiring the modeling of long-term dependencies, due to their unparalleled ability to capture temporal relationships in data.

2.9 Gated Recurrent Units (GRUs)

GRUs or Gated Recurrent Units are a variant of LSTMs designed to be simpler and more efficient, both in terms of computation and memory usage [7]. They combine the forget and input gates into a single update gate, maintaining the ability to model long-term dependencies while being less complex. GRUs contributed significantly to the progress of artificial intelligence, particularly in the domain of sequential data analysis such as time series prediction, language modeling, and machine translation. By streamlining the architecture of recurrent neural networks, GRUs offer a more efficient alternative to LSTMs, making them suitable for tasks where computational resources are limited or where model complexity needs to be minimized.

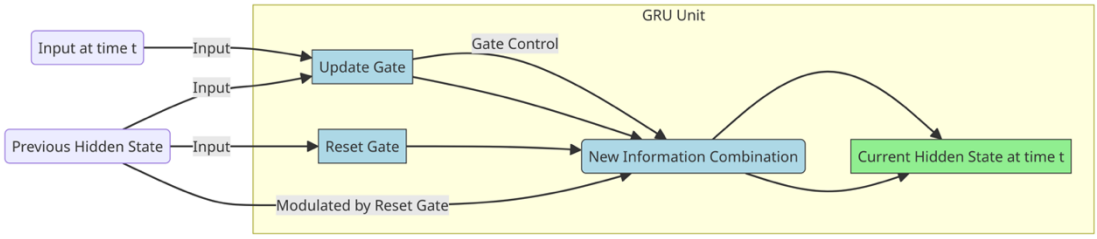

Figure 8: Gated Recurrent Unit Architecture

Despite their simplicity and efficiency, GRUs still retain the capacity to capture long-term dependencies in data, a crucial feature for tasks involving sequential information. Their drawbacks stem from the difficulty in parallelizing the training process due to their sequential nature, which can lead to longer training times for very large datasets. Additionally, while they are generally more efficient than LSTMs, there are specific scenarios where the additional complexity of LSTMs may provide better performance, indicating that the choice between GRUs and LSTMs often depends on the specific requirements and constraints of the task at hand.

2.10 Sequence to Sequence Models (Seq2Seq)

Seq2Seq (Sequence-to-Sequence) models, as introduced by Sutskever, Vinyals, and Le [12], represent a powerful framework for crafting neural networks capable of transforming sequences from one domain to another. Translating English to Greek is one such example, where the model receives the English sentence as input and generates the corresponding Greek sentence as output. This is achieved through the collaboration of two components within the model: the encoder which is used for input processing and the decoder which is used for output generation. The encoder acts like a "reader," processing the input sequence and capturing its essence into a compressed representation. This representation then serves as a bridge, allowing the decoder, acting as a "writer," to generate the output sequence based on the received information. Cho et al. [13] further refined this model by introducing learning phrase representations using an RNN Encoder-Decoder, which significantly advances the capability of Seq2Seq models in handling complex language structures and generating more accurate translations. Both the encoder and decoder can be built using various architectures, including RNNs, LSTMs (Long Short-Term Memory) units, and GRUs (Gated Recurrent Units).

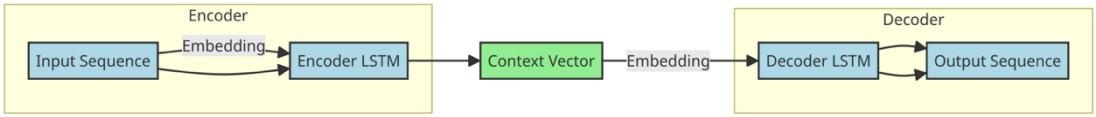

Figure 9: Sequence to Sequence Model Architecture

This structure positions Seq2Seq models as a cornerstone of various Natural Language Processing (NLP) applications, including machine translation, text summarization, and question-answering systems. Despite their effectiveness, these models often come with high computational costs and require extensive datasets for training, which can be prohibitive. Moreover, their complex internal mechanisms can lead to challenges in model interpretability, making it difficult to diagnose and correct errors or biases in the model's predictions.

2.11 Transformers

Transformers [14] are the state of the art in various sequence modeling machine learning tasks, such as language translation. They introduce a new architecture that resolves the issues of no-parallelization, and large distance between tokens playing a role in attention mechanisms in models like RNNs, LSTMs, and GRUs.

The vanilla transformer model uses an encoder-decoder component, where on every step the decoder before calculating the next output, takes as input all the tokens generated so far as an additional parameter to the given input. The encoder consists of 6 layers with identical architecture, in every layer there is a multi-head self-attention component and a fully connected feed-forward network. All layers generate an output of dimensions of 512 so that they can be connected with each other. Similarly, the decoder has 6 identical layers, additionally, it introduces an extra layer that implements a multi-head attention mechanism to the output of the encoder component. The self-attention mechanism here is adjusted, by including masking and shifting the output embeddings by one position, to be able to use only the previous positions while making the attention calculations and then based on that the final prediction of the current position.

The attention mechanism can be illustrated as a matching function between a query and a collection of key-value pairs, all of them being vectors. The output is a weighted sum of values, the weight of each value is calculated by applying the matching function between queries and keys.

Scaled Dot-Product Attention, instead of calculating the attention for each query separately, queries, keys, and values are put into matrices, this enables parallel computation which makes the whole process more efficient. Dot-product attention and additive attention are the most frequently used methods. They have similar complexity since dot-product attention can use matrix multiplication techniques that are computed in parallel. It is faster and requires less space. For small values, both techniques demonstrate similar results. For larger values dot product attention must be scaled down, otherwise they become extremely large and cause the softmax function to produce very small gradients.

Multi-Head Attention is the procedure of performing the attention function multiple times based on different learned representations. This allows the model to capture information on multiple representations for each position, thus being able to identify more complex contexts. Single-head models can only capture a single representation, due to averaging. Usage of attention mechanisms in transformer models occurs in three distinct places:

- Encoder-decoder: it enables every position in the decoder to apply the attention function across the input sequence.
- Encoder self-attention: it enables every position in the encoder to apply the attention function across all the positions in the layer before.

- Decoder self-attention: it enables every position in the decoder to apply the attention function to all decoder positions up to the current one.

- Position-wise Feed-Forward Networks: is an additional layer that fully connects each layer of the decoder or encoder with the subsequent one. Each layer's PFFN has different parameters, with the linear transformations remaining the same.

- Positional Encoding: in order for transformer models to be able to "understand" the order of the tokens in each given sequence they need to be fed with some data related to the position of each token within each sequence. To achieve this, positional encodings were added to the input embeddings before they were sent to the encoder/decoder stacks. To be able to sum the embeddings with the positional embeddings, a common dimension size must be used.

2.12 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) [15], as its name suggests, is based on the transformer model, and uses bidirectional flow on training, thus leading to deeper language representations. BERT implements pre-training and fine-tuning, during the pretraining process, it's trained with unlabeled datasets on various tasks. In the fine-tuning process, it's instantiated with the parameters it learned during pre-training and then it is fine-tuned on labeled datasets that are specific to each downstream application. A notable aspect of BERT is its consistent architecture between different applications, which means that the pre-trained model architecture is very close to the downstream one. BERT's architecture is similar to the original transformer implementation and makes use of bidirectional attention mechanisms. BERT's architecture is illustrated in the figure below.

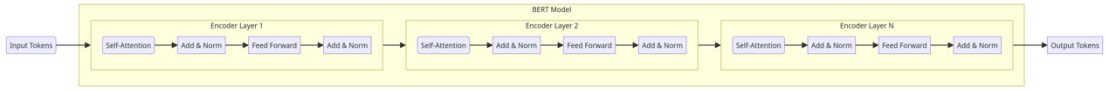

Figure 10: BERT Architecture

BERT is able to use one token sequence to generate representations for either a single sentence or a pair of sentences, thus making it a suitable model for various NLP tasks. In contrast to traditional transformers, BERT does not use one-way language models. Instead, it utilizes two unsupervised functions. The first one is called "masked LM" (MLM), reason suggests that a bidirectional approach would produce better results, compared to a unidirectional approach (e.g. right-to-left, left-to-right). Common language models can only be trained one way, the issue is that bidirectional training would lead to words being able to see themselves. To overcome this obstacle, BERT is randomly masking a percentage of the given input, and then learning to predict the tokens previously masked. The second function is Next Sequence Prediction (NSP), various downstream applications require the model to be able to grasp the connection between two sentences, but language modeling is not able to do that. To train BERT to grasp sentence relationships, it is pre-trained in a binary prediction task, that is easily implemented using monolingual corpora. While training, when choosing pairs of sentences, the target sentence is the correct next sentence at a rate of 50%, and the rest 50% is a randomly selected sentence from the corpus. Although this technique may seem straightforward, its implications are profound in various NLP tasks, such as Question Answering and Natural Language Inference.

Fine-tuning BERT is a relatively straightforward process, by changing the inputs and outputs to the appropriate ones for the specific task, then the self-attention function will enable BERT to adjust the model parameters as needed. A common approach to downstream task that works with text pair, is to get the embeddings of each sentence separately, and then run the bidirectional cross-attention mechanism. BERT improves this paradigm by creating a single embedding for the combined text of both sentences and essentially the self-attention mechanism on the combined text acts as a bidirectional cross-attention on both sentences. Fine-tuning is less computationally heavy compared to the pre-training step.

Feature-based approach can also be implemented using BERT, this is achieved by obtaining the activations from the layers needed and not using fine-tuning. This approach has its benefits, as the transformer encoder mechanism is not always sufficient for every application. Furthermore, the computational cost decreases, because the representations from the dataset are only calculated once and then can be used to experiment with.

Model size plays a significant role in the accuracy and performance of the model, furthermore, model size also plays a role in small-scale tasks, due to the pre-training. When the downstream application has little data to work with, it can greatly benefit from the representations learned when the model was originally trained.

2.13 RoBERTa

The introduction of BERT marked a historic moment in NLP, its innovative use of transformer architecture accompanied by self-supervised pre-training, enabled many NLP tasks to advance. Improving on BERT's success, RoBERTa (Robustly Optimized BERT Approach) [16] improves many aspects, that lead to significant performance boost.

RoBERTa makes various adjustments to the training process. Firstly, it is pre-trained on a much larger dataset, thus it grasps more complex and deeper language patterns and their nuances. Secondly, it implements dynamic masking in contrast to BERT's static one, in this function the model uses different masking patterns during the pre-training phase. Finally, it drops the NSP function during pre-training, simplifying the learning process to improve efficiency.

These improvements make RoBERTa's performance exceptional. It matches or exceeds BERT's performance on many benchmarks, such as GLUE and SQuAD. RoBERTa is performing well in many NLP tasks such as sentiment analysis that can be used in customer feedback, question answering tasks to become more accurate, and can also generate text in a more consistent manner, with a more natural tone and having better grammar. In the fast-paced field of NLP, RoBERTa takes a step forward, improving upon previous approaches and leading to significant gains in language understanding.

2.14 Sentence-BERT

While the BERT model is successful in many NLP tasks, when it comes to sentence embeddings it is very hard to extract them from BERT because it doesn't compute individual sentence embeddings. In order to use BERT to find the most similar pair between N number of sentences, n*(n*1)/2 inference operations are required; therefore, it becomes computationally expensive and as the N grows larger it

becomes unattainable. To overcome this issue, scientists run the sentences one by one through the BERT model to get the corresponding output and then compute a fixed-length embedding by taking the average or by utilizing the CLS token output. Nevertheless, this approach does not generate good sentence embeddings. SBERT [17] addresses these issues by generating fixed-sized embeddings for the sentences using a Siamese network architecture:

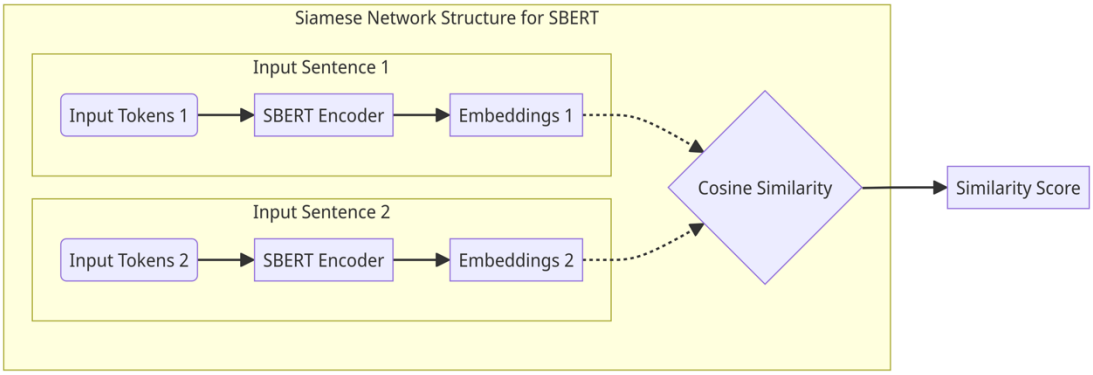

Figure 11: Siamese Network Architecture for SBERT

In SBERT the similarities between sentences are calculated in a computationally efficient manner, using metrics such as cosine similarity or Euclidian distance. This make SBERT suitable for NLP tasks such as clustering and semantic similarity query. For comparison, BERT took 65 hours to find the most similar sentence between 10000 sentences, SBERT took only 5 seconds to calculate the embeddings for each sentence, and then only 0.01 seconds to calculate the cosine similarity. Times can be further improved by implementing indexing.

2.15 Generative Pretrained Transformer (GPT)

GPT [18] is another novel design that as its name suggests is based on Transformer architecture. It is different from the approaches mentioned earlier, as it has no encoder component and it uses only the decoder part of transformers, therefore there is no component with a multi-head cross-attention mechanism. That leaves the decoder with the positional embedding, masked multi-head self-attention, and the feedforward network components, along with any layer normalization, dropout, and concatenation layers.

An important aspect of GPT is that it utilizes the masked multi-head self-attention to find patterns between single words, pairs of words, and it only takes into account the previous words and pairs of words generated, hence the GPT architecture is not bidirectional:

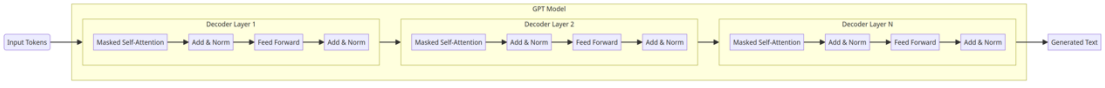

Figure 12: Generative Pretrained Transformer Architecture

Comparing the difference of directionality of the two models (BERT and GPT) we should note that BERT is a bidirectional model that has as its training objective to find masked words in a sentence, whereas GPT is more commonly used as a generative language model, so its objective is to try and find the upcoming word in a incomplete sentence, provided it has access to all previous words up to the

current position. After the new word is predicted, it is added to the current sequence of words and then it is forwarded as input to the next iteration of the process, and this loop is repeated until all sentences are generated completely. It is important to note that GPT can be trained on unlabeled data, meaning that any text is suitable for training, thus it can be trained using the vast amount of data available on the internet.

GPT-3 is an enormous version of GPT, that has a lot of decoder stacks and 175 billion parameters that can be trained, this enables it to train in a huge amount of information from various themes, and topics. It has been able to memorize the texts it was trained on without overfitting, this happened due to the huge number of parameters in conjunction with the vast amount of data from the internet.

3 *Named Entity Recognition*

Named Entity Recognition (NER) [19] is fundamental within NLP, its purpose is to identify and classify named entities found in unlabeled texts. Named entities usually belong to predefined categories, such as locations, persons, names, dates, and job skills among others. NER can go beyond the extraction of NE, it is able to understand the relationships between them, thus providing more refined information to the following tasks. Furthermore, NER is a crucial preprocessing task for various NLP applications and helps to improve the performance of downstream tasks like question-answering, text summarization, and machine translation.

In information extraction systems, NER plays a vital role by locating and labeling the core informational tokens within documents, thus making the construction of knowledge bases more accurate and feasible. It also provides a more accurate and concise text summary. In machine translation, it helps refine the results by identifying NE that might have special translation rules or should remain untouched to preserve their intended meaning within the sentence. In search engines, NER is implemented to help disambiguate the user-submitted query. By understanding if the query is about a specific event, person, or a more general context, the search engine can provide much more relevant results. Additionally, NER is also used to a great extent within chatbots and virtual assistants. It allows them to identify key elements in the user input text, and then return more contextually accurate and meaningful responses.

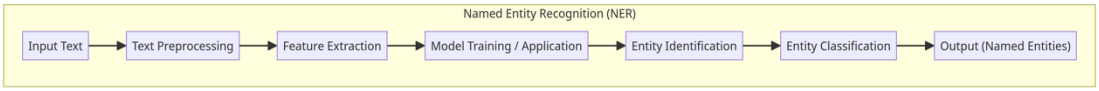

Figure 13: Named Entity Recognition simplified flow diagram

NER is a crucial part of NLP; thus, it has been the subject of rigorous research throughout its history. The initial approaches were heavily dependent on handcrafted rules, gazetteers, and dictionaries. These early systems demanded a large amount of feature engineering and domain specificity; thus, they were unable to adapt to new domains or unseen entities efficiently. Later approaches were based on statistical machine learning architectures, such as hidden Markov Models and Conditional Random Fields, making advancements in the field. These models used annotated datasets to learn patterns in the texts, they were more adaptable and could generalize better to new domains and previously unseen entities. More recently, the deep learning evolution transformed the way NER is approached. Architectures like Recurrent Neural Networks, Long Short-Term Memory Networks, and finally most notably the Transformer were conceptualized and provided state-of-the-art performance in various tasks. These architectures are able to capture complex relationships within the language and can provide better context for each token. Even though extensive progress has been made in the NER field, it is still a very active domain for research, scientists are trying challenges such as ambiguity, limited data for specific domains, and identification of unseen entities among others.

3.1 Rule-based Systems

The first rule-based system [20] was developed in 1995 by Ralph Grishman. This is one of the more classical approaches to NER, it implements handwritten rules that are created by linguistics experts. These systems' function is to parse the given text input and generate a representation, which might be a parse tree or a custom domain specific representation. Rule-based systems use a substantial number of man-made rules in conjunction with dictionaries, and grammatical and syntactic patterns to extract named entities. These kinds of architectures are suitable for domains with small corpora and are also able to capture complex entities in contrast to other architectures. Their most notable drawback is their inability to generalize well, thus they perform poorly in different domains and languages. Furthermore, they are difficult to maintain even with small fluctuations in data format, which significantly increases their maintenance cost.

3.2 Conditional Random Fields

Conditional Random Fields (CRFs) [21] are probabilistic graphical models that combine probability theory with graphs and can perform data segmentation and labeling, like the part-of-speech tagging in NLP. They were designed to improve on previous approaches, by being more efficient and solving other practical problems. In contrast to Hidden Markov Models that model the joint probability of observations and label sequences, CRFs directly model the conditional probability of the label sequence given the observation sequence.

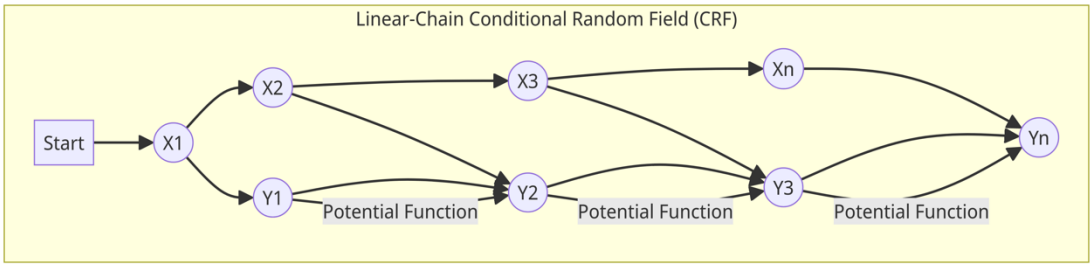

Figure 14: Conditional Random Fields Architecture

CRFs implement functions that model feature interactions between sequential labels. These features can represent a plethora of information, from single words within sentences to broader context or metadata about their linguistic properties. During training, CRFs learn to maximize the conditional probability of correct label sequences in a dataset using methods such as gradient descent or L-BFGS. For inference, CRFs attempt to find the most probable output sequence for each given input sequence, so that the conditional probability is maximized. The most common algorithm used for the inference task is the Viterbi Algorithm.

The advantage of the CRFs is the ability to overcome the label bias problem that exists in many other sequential label approaches. Furthermore, they can handle overlapping features and long-distance relationships between entities, therefore they can capture richer context and interactions between words.

Disadvantages of the CRFs include customization and optimization needed for each specific application requirement. Also, the computational cost needed to train the model, especially as the data size increases, becomes extremely high. Furthermore, they face issues of slow convergence rates and sensitivity to the

selection of features, therefore engineers should carefully consider the feature design for optimal performance.

3.3 Support Vector Machines

Support Vector Machines (SVMs) [22] is a versatile supervised machine learning approach, that has a wide range of applications in classification problems, with NER being one of them. At its core SVMs attempt to find the optimal "hyperplane" to separate different classes within a high-dimensional space of features. This hyperplane is selected by attempting to maximize the margin, which is the distance between the decision boundary and the nearest data of the different classes in the hyperspace, these are also called "support vectors". The maximization of the margin leads to a better generalization of the model. In the simplest form, SVMs can work only on linear separated problems. In the real world though the problems are rarely linear separable. To address this problem SVMs use functions called kernels, that calculate the dot product between non-linear vectors, and then map them within a higher dimension space.

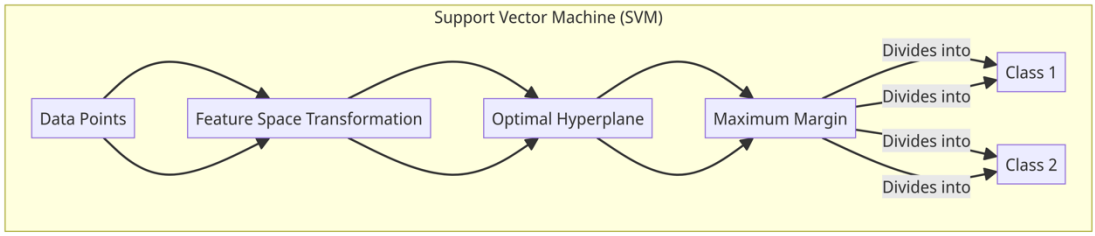

Figure 15: Support Vector Machine Architecture

SVMs have shown two major advantages over other algorithms, like the Hidden Markov Models and decision trees. The first one is that they can generalize very well without being affected by the feature space dimensions. The second one is that they can learn all possible combinations of features with no impact on computational cost, which is achieved by using the kernels mentioned above.

3.4 Hidden Markov Models

Markov models [23] offer a framework for analyzing systems where state transitions play a crucial role. They implement a table consisting of all the probabilities in the problem at hand. This matrix depicts how likely it is for each system change to happen over a period of time. The Markov process enables predictions to be made about the state of a system over time. Hidden Markov Models build upon this concept by introducing a layer of hidden states that indirectly influence observable outputs. The ability to infer these hidden states based on observations is what gives HMMs their power.

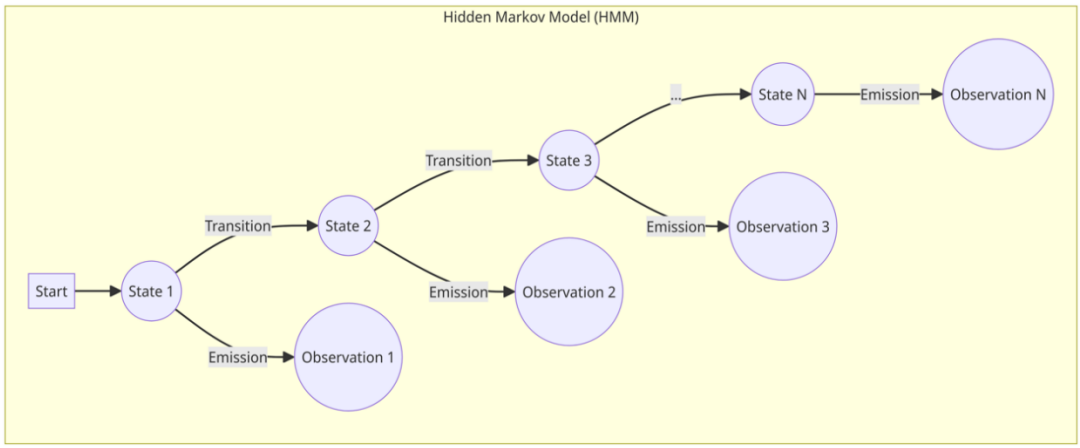

Figure 16: Hidden Markov Model Architecture

Named Entity Recognition (NER) is essential for medical information systems, but identifying biological entities like genes and proteins is extremely complex due to their context-dependent nature. Miller et al. cited in [23] addressed this by using a probabilistic method using hinge-loss Markov random fields (HL-MRF), which combines relative information and word semantics to classify bio-entities with high accuracy and precision. This approach shows promise for disambiguating references within different biomedical contexts.

Arora et al. cited in [23] employed a BiLSTM-CNN model alongside a Markov model to reduce errors in the NER training process. They improved precision by utilizing a custom loss function with a semi-Markov method. This technique effectively balances precision and recall in NER tasks. Lay et al. cited in [23] demonstrated Hidden Markov Models for supervised NER in the Myanmar language. The HMM calculated probabilities for states containing candidate named entities, predicting unobserved entities based on these candidates. While this method yielded strong results, further testing on languages like Arabic or Chinese is needed to fully assess its potential. Similar supervised HMM approaches were applied for both the Bengali and Urdu languages. However, these methods remain somewhat reliant on manual annotation or rule-based systems, highlighting a preference for unsupervised approaches that don't require extensive pre-labeling. Finally, Leaman et al. also cited in [23] used a Semi-Markov model to combine NER with normalization for disease and chemical entity extraction. This model scaled normalization weights to make them independent of lexicon size or text segment length, enhancing the process when dealing with unseen names.

3.5 RNNs – LSTMs – CNNs (NN)

RNNs have been used on various tasks, such as speech recognition and language modeling, producing optimistic results. An RNN model keeps a memory of previously seen tokens in a sequence, and then by using this stored memory it is able to predict the current token based on long-range features. The input layer is a representation of features at a given time step. The dimensions of these features are the same as the feature size and can be in various representation formats, such as dense vectors, one-hot encoding, or sparse features. The difference between RNNs and feedforward networks is that RNNs implement a connection between the previous hidden state and the current one. This connection is called the recurrent layer, and it is responsible for storing history information.

LSTMs are similar to RNNs, but they implement different hidden layer updates, that have application-specific memory cells. This results in better long-range dependencies and solving the vanishing gradient

problem that RNNs have. In tasks like sequence tagging, which is commonly used in NER, access to both previous and next inputs is available, thus a bidirectional LSTM can be utilized. This utilization enables the bidirectional model to use both past features and future features for a given time point. The model is trained using back-propagation through time. This training process is similar to the classic forward and backward technique, with the difference being that in each time step the hidden states must be unfolded. Furthermore, the starting and ending points of data need to be treated specially. Finally, the bidirectional model supports batches, which means that many sentences can be parsed at the same time.

The Bi-LSTM model can produce exceptional results on POS and NER data sets. Furthermore, it does not depend on word embeddings as much compared to other models [24]. Collobert et al. (2011) cited in [25] pioneered a sentence-level approach for NER. Their model processes each word within its broader sentence context. Word embeddings are fed into a convolutional layer, generating local features. Global features, independent of sentence length, are then constructed (often via max-pooling or averaging) for the entire input sequence. These features are decoded to predict NER tags. This architecture has inspired work in domains such as biomedical NER (Yao et al., 2021; Wu et al. 2019) cited in [25].

For efficiency and parallelization, Strubell et al. (2017) cited in [25] introduced Iterated Dilated Convolutional Neural Networks (ID-CNNs). Unlike LSTMs, which scale linearly with sentence length, ID-CNNs allow for fixed-depth processing across the entire text. They achieve significant speedups over Bi-LSTM-CRF models while maintaining strong NER performance.

However, sequential models can overemphasize words appearing later in a sentence. Zhou et al. (2021) cited in [25] tackled this with a hybrid BLSTM-RE model. Bidirectional LSTMs capture long-term dependencies, while CNN extracts higher-level features. Crucially, both the LSTM-generated sentence representation and the CNN's relation representation are taken into account for the final entity prediction [25].

3.6 Transformer-based Models (TB)

In recent years, the TB NER models have dominated the competition, and they are producing state-of-the-art results. In comparison with previous architectures, deep learning and TB models can learn the features on their own, without the need for feature engineering. This reduces the time required to engineer the features and the need for specialized personnel. The TB models can learn from non-linear activation functions, this enables them to learn deeper, more complex, and more nuanced features from the dataset they train upon. Additionally, TB models resolve the RNN's issue of vanishing gradients with the implementation of the attention mechanism, they excel at identifying relationships between words, even if they are far apart in a sentence. This is very helpful in NER where the surrounding context often plays a significant role in determining an entity type. TB model's encoder can outperform the LSTM when it has been trained on very large corpora. Furthermore, another major drawback of RNN decoders is that the current input relies on the output of the previous step, due to this they have limited speed, and they cannot parallelize operations. Transformers overcome this issue by parallelizing, thus gaining improved speed with better infrastructure.

A major drawback of TB models is that they require a vast amount of data to be able to train effectively, and they also require a lot of time and resources. To overcome this issue, a pre-trained version of each

model is used and then it is fine-tuned with data from the specific domain that it will be applied to. This approach produces state-of-the-art results [25].

The implementation of the pre-training step brought many advantages in language representation. Pre-training huge TB models and then making them smaller to apply to various end tasks has become the norm approach. These models are extremely efficient and accurate across many channeling NLP applications, even if the available data is limited. During pre-training the model is trained using a large general-purpose corpus, and then during the fine-tuning step, the model is adjusted to become a better fit for a narrower task or dataset. The typical flow is the following: To begin with, a large more general dataset must be acquired on which the model and its tokenizer will be trained, this will create a more general representation. Then the fine-tuning process will be initiated, the model's weights will be adjusted to fit a more specific downstream task. This workflow enables TB models to become effective even when the available data are extremely sparse, and the model wouldn't be able to train solely on them.

During pre-training both a tokenizer and the TB model are trained. Instead of using fixed dictionaries, language models can directly parse and learn from text. They can do this by using a function called tokenization, in this function, the given text sequence is split into smaller sequences, which then become the input to the TB model, these sequences that the tokenizer outputs are called tokens. Downstream task performance is improved by the use of tokenization because cross-word generalization is possible. The contextualized embeddings created by tokenization are different from the more classic word embeddings, like GloVe and Word2vec. The most prominent tokenization types are SentencePiece, WordPiece, Byte Level Byte Pair Encoding (bBPE), and Byte Pair Encoding (BPE). While the bBPE tokenizer can map all the words of any language, when it is used in a different language than the original one it was trained on, it requires an additional 70% tokens on average. This should be taken into account when the decision is made to select a tokenizer to train a TB model in a given language. To be able to learn generic language representations TB models complete various tasks which can be self-supervised and use pseudo-labeled datasets. These pseudo-labels are generated according to the specific pre-training task. The selected pre-training task should be similar to the downstream task and has to be hard enough for the model to learn deep and rich context and relationships between tokens. Various techniques to corrupt tokens are implemented in the pre-training tasks, such as swapping tokens, masking, and replacing them at a fixed rate. Moreover, they are different based on how each model handles the sequence, for example, if a model is a left-to-right transformer or a bidirectional one. The BERT model uses two special pre-training tasks. The NSP trains the model by making it predict if two sentences are consecutive, and the MLM, trains the model by making it predict masked tokens in the given sentence. Many other pre-training tasks exist with different goals and loss functions that are suitable for a variety of downstream tasks. The usage of pre-training tasks enables the TB models to have a generic language understanding and it has been linked with significantly better ranking performance.

During fine-tuning the model weights are modified with the use of a dataset that is tailored to the downstream task that needs to be addressed. Word representations that were created during the pre-training step with self-supervised learning are now adjusted to better reflect the labeled data from the downstream task. In many cases, some training instability is observed even when the same hyperparameters and learning rates are used. This issue is more noticeable when trying to train a large TB model on narrow datasets. Since the introduction of the BERT model, various workarounds have been developed to address the issue.

TB models face issues when it comes to extended sequences, this is due to the self-attention mechanism making the processing of these sequences computationally and memory expensive. Furthermore, due to their architecture, they have difficulty training on small amounts of data. To resolve these issues, many variants of the original transformer have been developed, these are called X-formers, and they improve various aspects of the original transformer. To improve the efficiency of the model, sparse attention and divide-and-conquer functions have been added to make the model more lightweight. Moreover, to enable the model to generalize more effectively, pre-training on large unlabeled datasets, regularization and structural bias are implemented in some X-formers. Additionally, there are specific variants that have their architecture adjusted to better fit and improve the performance of the specific downstream task [26].

Lothritz et al. (2020) [27] performed an evaluation between commonly used models, such as BERT, RoBERTa, XLNET, CRF, and BiLSTM-CNN-CRF, in the task of FG-NER. Fine-grained NER is a subtask of NER, it has the same objective as NER, with the difference being that it uses a lot more entity types, in some cases, it can go beyond 100 discrete labels. To perform their evaluation the English Wikipedia Named Entity Recognition and Text Categorization (EWNERTC) dataset was used. This dataset consists of 7 million sentences that were annotated and put into categories automatically, separated into 49 different domains. Their findings show that the TB models offer better F1 results compared to CRF and BiLSTM-CNNCRF in the majority of domains. RoBERTa and BERT produce the best results overall, being in the first two places in 36 out of the 49 domains. In the 10 smallest domains XLNet results are worse compared to BiLSTM-CNN-CRF, but in the rest domains it performs slightly worse than BERT and RoBERTa. Furthermore, while TB models excel at F1 score, it is worth investigating the recall and precision results too. CRF model comes first in terms of precision in most cases. TB models perform worse precision-wise than BiLSTM-CNN-CRF in most domains. Regarding recall, TB models consistently outperform the rest of the models. BERT and RoBERTa outperform the CRF and BiLSTM-CNN-CRF by a large margin in almost all the domains. XLNet outperforms them in most of the domains. The tested models showed similar performances, indicating that the domains are either too hard or too easy across all models. An observation can be made here that no model can get a much better language understanding compared to other models in the same domain, so all models face the same issues and challenges. Furthermore, across domains each model's rank is stable, this means that if a model performs well in a domain, it will most likely perform similarly well in the rest of the domains. Only in the four smallest domains does the difference in ranking between models become noticeable, breaking the previous pattern.

3.7 NER Challenges

Data Annotation. NER systems need to train on a large amount of labeled data either during the pre-training process or the fine-tuning process. The process of annotation is still very time-consuming and also has increased costs. This issue is more prominent in languages with scarce resources and in narrow domains where specialized domain experts have to do the annotation. Furthermore, the annotation process faces issues like poor data quality and inconsistencies largely deriving from the ambiguous nature of the language. For example "Empire State" and "Empire State Building" are both marked as Locations in the CoNLL03 dataset, making the boundaries between entities to be mixed. Inconsistencies during the data annotation, like the one mentioned before, lead to poor model performance even if the documents used belong to the same domain. Additionally, to make the situation even worse, it has been reported that nested entities are far more common than expected. The GENIA dataset has 17 percent nested entities, while in the ACE dataset, this figure goes up to 30 percent of sentences. To address this issue, new annotation schemes must be developed. These schemes should apply to both single, nested entities and fine-grained entities, which are entities that can have multiple label tags. Unseen Entities,

such as informal text is another issue NER faces. On datasets that have formal writing, such as news articles, NER performs well. However, when NER is applied to user-generated texts such as tweets and forum posts, NER performance becomes worse, with F-scores averaging close to the 40 percent mark. Informal text becomes challenging to NER due to the fact that it is shorter and noisier compared to formal text. Furthermore, in many applications, the user-generated text is specific to a narrow domain. For example, NER systems are usually applied in e-commerce and customer support. There is an interesting way to evaluate the performance of the model and its robustness, by measuring the model's ability to handle unseen and unusual entities in emerging topics [25].

3.8 NER Future Directions

Advancements in language modeling and the rise of demand in real-word applications, will make NER a prominent topic of research in the future. NER is considered by many researchers to be a pre-processing step for downstream tasks. This means that NER must be adapted and trained to the needs of the downstream application. For example, the types of entities needed for the application and if there is a need for nested entities. Some of these future research topics are presented below:

Fine-grained NER and Boundary Detection. Existing studies focus more heavily on coarse-grained NER in more broad domains, but to be able to support a variety of real-world applications, fine-grained NER research is expected to rise. Fine-grained NER comes with its challenges, such as the greater number of NE used and also the complexity deriving from enabling each named entity to belong to multiple NE types. Researchers should consider NER approaches that annotate both the entity boundaries and entity types. Furthermore, boundary detection could become a separate task. This could make it more robust and make it able to be shared across domains. Furthermore, by detecting boundaries more precisely, error propagation when linking entities to knowledge bases is reduced.

Joint NER and Entity Linking. Entity Linking (EL), more commonly known as NE disambiguation, is the task responsible for identifying the entities that exist in a text by referencing them to a knowledge base e.g. for general domain Wikipedia can be used. EL and NER are considered different tasks in many of the existing studies, exploring approaches that combine NER and EL could be beneficial to the results, due to semantics provided by a knowledge base. This could also reduce the propagation of errors which is common in pipeline settings.

DL-Based NER Scalability. Scaling up Neural-based NER models is still an issue. Furthermore, as the size of the data increases, the parameters grow exponentially, this makes the need for an optimized solution urgent. Some Neural-based NER models have shown great performance, but they were extremely computationally heavy to train, for example, Google BERT was trained on 64 TPUs. When it comes to end users this becomes an issue because they very rarely have access to such powerful equipment, thus they are usually not able to fine-tune these massive models. Researchers should develop approaches that balance scalability with the complexity of the models. Model pruning and compression techniques could be viable options in the direction of reducing the computational needs of the models.

Transfer Learning for NER. Entity-focused applications usually rely on existing NER systems; however, these models are trained on different datasets than the ones that will be used. This leads to poor performance due to the different characteristics each language has and due to annotation differences. Although some studies on how to apply deep transfer learning for NER systems have been conducted, there is a need for more exploration on the topic. Future research can be conducted in directions such as the development of a robust recognizer that can generalize better and work in different domains; NER

learning with zero-shot and few-shot procedures; develop solutions for domain and label mismatch when working across multiple domains.

Toolkits for DL-Based NER. Easy-to-use toolkits should be developed, making the NER process more standardized and accessible. There could be distinct modules for each step: data preprocessing, input representation, context encoder, and decoder, and model performance measuring. These tools would streamline the whole process and would be a great asset for both experts in the field and non-experts [25].

4 *Semantic Similarity*

Semantic similarity is a core concept in Natural Language Processing (NLP) that quantifies the extent to which different linguistic units—such as words, phrases, sentences, paragraphs, or even whole documents—share meaning. Both semantic and lexical similarity fall under the broader term of text similarity. However, unlike lexical similarity, which focuses on surface-level resemblance by analyzing the overlap between words, semantic similarity seeks to capture the deep, conceptual relationships between these units. This differentiation is critical for a range of NLP applications, including but not limited to:

a. *Semantic Search*: This aids in finding meaning-related texts as opposed to simply retrieving documents based on keyword presence. Semantic search can be considered an evolution of lexicon-based information retrieval, enhancing search relevance by understanding the intent behind queries.

b. *Text Summarization*: This involves condensing text while preserving its semantic content, enabling users to quickly grasp the essence of lengthy documents.

c. *Machine Translation*: The goal here is to ensure that the intended meaning is preserved between languages, maintaining the fidelity of the original text across linguistic boundaries.

d. *Sentiment Analysis*: This interprets and classifies the emotional nuances conveyed in a text, providing insights into the underlying sentiment, whether positive, negative, or neutral.

Understanding and computing semantic similarity involves delving into the nature of meaning in language, a topic explored through various theoretical lenses. These can be broadly categorized into:

a. *Cognitive Theories*: These theories explore how humans understand and process language, often drawing on insights from psychology and neuroscience.

b. *Distributional Semantic Theories*: These theories posit that the meaning of a word is defined by its context within large corpora of text, leveraging statistical methods to uncover patterns and relationships.

c. *Formal Theories*: These theories employ logical and mathematical frameworks to model the structure and meaning of language, focusing on precision and formalization.

This final section of our literature review will provide a brief overview of the different theories through which the concept of meaning has been investigated. Additionally, it will offer a more comprehensive review of various research papers on the topic of semantic similarity, with a particular focus on the advancements brought by transformers. Transformers, a type of deep learning model, have revolutionized the field by enabling more nuanced and accurate semantic understanding, significantly enhancing the performance of numerous NLP applications.

4.1 Cognitive Theories

Cognitive theories, particularly those from the field of psycholinguistics, posit that meaning arises from mental representations, suggesting that our understanding of language is rooted in internal cognitive structures and the dynamic interplay of concepts within these structures. This perspective underscores the cognitive processes involved in language perception, information organization, and the establishment of associative connections, leading to the formation of complex mental representations. The linguist J.R. Firth encapsulated this idea with his assertion, "You shall know a word by the company it keeps," [28], which has been fundamental in forming the contextual understanding of linguistic elements and has later been implemented in Large Language Models based on neural networks and transformers.

One of these cognitive theories, more widely known as the Spreading Activation Theory, proposes that similar concepts are linked together and that accessing one concept in memory can trigger the retrieval of closely associated concepts [29], an idea that is conceptually mirrored in the 'vector space model'. In this model, words and concepts are encoded into high-dimensional vectors and their proximity within this multidimensional space reflects their semantic similarity—a notion implemented in models like Word2Vec and GloVe [30]. Complementary to the Spread Activation Theory, the Prototype Theory suggests that the organization and categorization of concepts occur in accordance with their resemblance to prototypes or ideal examples. These categories, however, are not defined by rigid boundaries but rather by typical characteristics that best represent any given category [31, 32].

4.2 Distributional Theories

Distributional theories of semantics are based on the observation that words that appear in analogous contexts often share similar meanings. This concept is reflected in the distributional hypothesis, which posits that the meaning of a word can be closely inferred by the context in which it occurs. According to this theoretical framework, the semantic similarity between words or phrases can be effectively quantified by examining their patterns of co-occurrence within extensive collections of texts, known as corpora. By leveraging statistical analysis of how frequently words appear together across diverse linguistic environments, researchers can construct nuanced models of word semantics that reflect the complex interplay of language in real-world usage [33]. Apart from underlining the significance of context in understanding language, this approach posits that the meaning of words emerges dynamically from their relational patterns rather than being fixed or inherent. Models that implement this theory are called Distributional Semantic Models, the most important of which are:

a. *Latent Semantic Analysis (LSA)*: LSA uses singular value decomposition (SVD) on a term-document matrix to reduce the dimensions of this matrix, thus uncovering latent semantic structures. This process aims at capturing the underlying meanings of words by analyzing their distribution across documents, assuming that words that appear in similar documents have similar meanings [34]. By transforming the original high-dimensional space into a lower-dimensional one, LSA can identify patterns and relationships that are not immediately apparent in the raw data. This dimensionality reduction helps in mitigating issues related to synonymy (different words with similar meanings) and polysemy (same word with multiple meanings). Furthermore, LSA has been effectively applied in various natural language processing tasks, including information retrieval, document classification, and text summarization. The technique's ability to generalize from the observed data to uncover hidden

semantic relationships makes it a powerful tool for understanding and organizing large collections of textual information. The LSA architecture can be seen below:

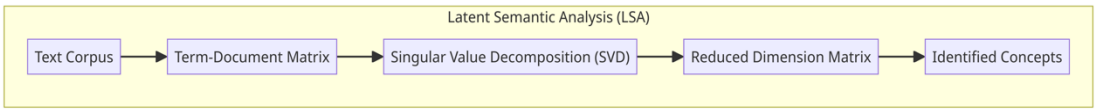

Figure 17: Latent Semantic Architecture

b. *HAL (Hyperspace Analogue to Language)*: HAL constructs a high-dimensional semantic space by examining the co-occurrence frequencies of words within a certain window of context throughout the text. The proximity and frequency of word co-occurrences in this space are used to gauge the strength of their semantic correlation, and in this way, a geometric representation of word meanings is constructed [35]. This method enables HAL to capture both local and global context by considering how words co-occur in close proximity as well as across larger spans of text. The resulting semantic space is highly informative, allowing for nuanced distinctions between words based on their contextual usage. HAL's approach is particularly effective in handling sparse data and can uncover subtle semantic relationships that might be overlooked by other models. Additionally, the high-dimensional nature of the semantic space provides a rich, detailed landscape of word associations, making it useful for various applications such as word sense disambiguation, information retrieval, and enhancing the performance of machine learning algorithms in natural language processing tasks. The HAL architecture can be seen below:

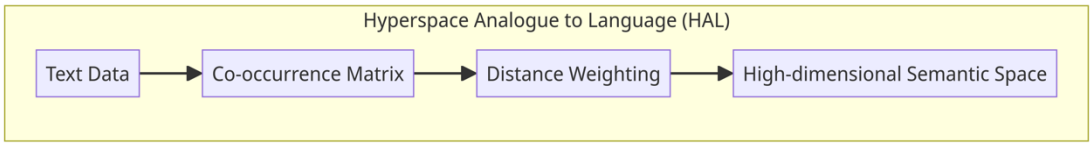

Figure 18: Hyperspace Analogue to Language Architecture

c. *GloVe (Global Vectors for Word Representation)* [36]: Combining the strengths of both the aforementioned global matrix factorization and local context window methods, GloVe generates word embeddings by factorizing a sparse word co-occurrence matrix that reflects how frequently words appear together. This method encompasses both global statistics of word co-occurrences as well as local contextual information, offering a more comprehensive view of word semantics [36]. By leveraging the co-occurrence probabilities, GloVe captures the semantic relationships between words in a more robust manner, as it integrates the benefits of both distributional and count-based approaches. This allows GloVe to produce highly accurate and meaningful word embeddings that are effective in capturing nuances such as analogies and word similarities. Moreover, GloVe's embeddings have demonstrated superior performance across various natural language processing tasks, including named entity recognition, machine translation, and sentiment analysis. The model's ability to efficiently process large text corpora and generate high-quality embeddings has made it a popular choice for researchers and practitioners seeking to enhance the semantic understanding of textual data.The GloVe architecture can be represented diagrammatically thus:

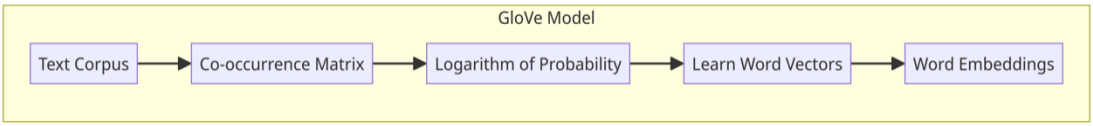

Figure 19: Global Vectors for Word Representation (GloVe) Architecture

4.3 Formal Theories

Formal theories of semantics also offer a systematic and rigorous framework for understanding and analyzing meaning in language. Unlike distributional approaches that derive meaning from usage patterns in large corpora, formal theories use precise mathematical formulae, logic, ontologies as well as computational linguistics to model linguistic phenomena. These models are built on the foundation of logic, ontologies, and computational linguistics [37]. In formal semantic theories, meaning is not an emergent property of linguistic context but is instead encoded in structured representations that specify the relationships between concepts within a logical or ontological framework. These frameworks usually provide strict definitions of terms thereby enabling the assessment of semantic similarity through the evaluation of how closely these structured representations align with each other. Key aspects of formal theories include but are not limited to:

a. *Logical Systems*, where logic is employed to create structured representations of linguistic meaning.

b. *Ontologies*, where the organization of knowledge occurs through hierarchical structures of concepts and categories.

c. *Language Analysis*, which uses formalized rules to define and relate concepts for clear and predictable interpretations of language.

d. *Semantic Similarity and Relatedness*: In formal theories, semantic similarity is often assessed through logical equivalence or the degree to which concepts share properties within an ontology.

The theories mentioned above have been implemented in a variety of ways resulting in the following frameworks:

a. *CYC*: An artificial intelligence project that aims to assemble a comprehensive ontology and knowledge base of everyday common-sense knowledge, allowing the system to perform reasoning as well as semantic understanding [38]. Initiated in the mid-1980s, CYC has been designed to encapsulate the vast array of human knowledge by systematically encoding general truths and facts about the world. The project's extensive ontology includes concepts, relationships, and rules that mimic human understanding, enabling CYC to infer new information, answer questions, and provide contextually relevant insights. By integrating this extensive knowledge base, CYC strives to bridge the gap between human cognitive processes and machine understanding, making significant strides in fields such as natural language processing, automated reasoning, and artificial general intelligence. Its applications span across diverse domains, including robotics, virtual assistants, and data analysis, where robust common-sense reasoning is crucial for effective performance.

b. *WordNet*: Although not a formal theory implementation in the strictest sense, WordNet is a lexical database for English that groups words into sets of synonyms (synsets) and describes the semantic relationships between them. Developed at Princeton University, WordNet includes nouns, verbs, adjectives, and adverbs, organized into hierarchical structures that capture hypernyms, hyponyms, meronyms, and other lexical relations. This organization allows WordNet to provide a rich semantic framework for understanding word meanings and their interconnections. WordNet is often used in semantic analysis and various natural language processing (NLP) tasks, such as word sense disambiguation, information retrieval, and machine translation, to provide insights into the meanings of words and their relations. Its extensive and meticulously curated database serves as a valuable resource for both linguistic research and practical applications, facilitating improved language understanding and processing in computational systems. Additionally, WordNet's structure supports advanced tasks like sentiment analysis and knowledge graph construction, making it a cornerstone tool in the field of computational linguistics [39].

c. *Ontology-Based Systems*: Various domain-specific ontologies have been developed for use in semantic web technologies, knowledge management, and information retrieval. These systems use formal ontologies to represent knowledge within specific fields, enabling precise querying and reasoning over data. By defining a structured framework of concepts and their interrelationships, ontology-based systems facilitate a deeper understanding and organization of domain-specific information. They support interoperability between disparate data sources and enhance the ability to perform complex queries that require an understanding of the underlying semantics. Applications of ontology-based systems include biomedical informatics, where ontologies like the Gene Ontology provide a structured vocabulary for gene products across species, and e-commerce, where product ontologies improve search accuracy and recommendation systems. Furthermore, these systems are integral to the development of intelligent agents and automated reasoning tools, which rely on ontological knowledge to perform tasks such as decision support, context-aware computing, and natural language processing. The precision and clarity provided by ontologies help ensure that data is interpreted consistently, making ontology-based systems a critical component in the advancement of artificial intelligence and data science [40].

d. *Description Logics*: This family of formal knowledge representation languages is extensively used in the Semantic Web to construct and query ontologies, thus providing a framework for reasoning about the concepts and relationships within a specific domain. Description Logics (DL) offer a precise syntax and semantics for defining classes, properties, and individuals, enabling the formalization of domain knowledge in a way that supports automated reasoning. By leveraging DL, one can infer implicit knowledge from explicitly stated facts, perform consistency checking, and classify instances within a hierarchy of concepts. These capabilities are crucial for applications that require a robust and scalable way to manage complex datasets, such as biomedical research, where DL aids in the integration and analysis of diverse biological data, or in enterprise settings, where it enhances data interoperability and decision-making processes. The expressive power of Description Logics, combined with their computational properties, makes them an ideal choice for developing sophisticated ontology-based systems that underpin the Semantic Web, driving advancements in artificial intelligence, data integration, and information retrieval [41].

Cognitive, distributional, and formal theories provide the intellectual foundation for computational methods to measure semantic similarity. Such methods are varied, ranging from word co-occurrence frequencies and TF-IDF (Term Frequency – Inverse Document Frequency), which are based on the presence of terms within documents to deduce relevance and similarity, to vector space models that represent words as points in a geometric space and to more sophisticated deep learning algorithms that use transformers to infer semantic relationships from data.

4.4 Semantic Similarity with Transformers

Transformers have revolutionized the field of natural language processing (NLP), bringing advancements in understanding, processing as well as generating human language. The architecture of transformers enables them to capture semantic relationships within text, which is a key component for a wide array of applications, such as machine translation, summarization, and question-answering systems. Semantic similarity has particularly benefited from transformers since these models leverage self-attention mechanisms to weigh the importance of each word in a sentence in the context of all other words, allowing for a deep understanding of sentence structure and meaning. A simplified representation of the flow in a semantic similarity problem can be seen below:

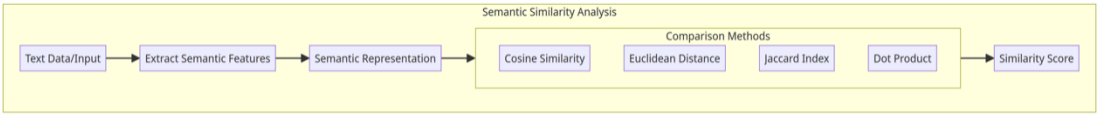

Figure 20: Semantic Similarity Architecture

Models like BERT [15] (Bidirectional Encoder Representations from Transformers) and its variants have set new standards in semantic analysis by being pre-trained on vast corpora of text to learn general language representations and subsequently by being fine-tuned for specific semantic similarity tasks. This has led to significant improvements in performance benchmarks, showcasing the transformer's ability to understand complex language patterns and the subtleties of human language.

In their 2015 paper He et al. [42], explore the use of CNNs for assessing sentence similarity from a multiplicity of perspectives. Their approach implements a modular architecture with two components: sentence modeling and similarity measurement. During sentence modeling, the model extracts features of variable granularity with the help of convolutional filters, variable window sizes, and multiple types of pooling operations whereas similarity measurement is gauged using multiple distance functions, i.e. cosine distance, Euclidean distance as well as element-wise difference. This dual architecture is reminiscent of a ‘siamese’ network where both subnetworks process the sentence in parallel, being followed by the similarity measurement layer and a fully connected layer. The authors conducted evaluation tests on the Microsoft Research Paraphrase Corpus (MSRP) as well as the Sentences Involving Compositional Knowledge (SICK) and Microsoft Video Paraphrase Corpus (MSRVID) datasets. The authors report that their approach seeking to maximize information utilization through the implementation of multiple techniques achieves SOTA results on SemEval semantic relatedness tasks and the Microsoft Research paraphrase identification task.

Kiros et al. (2015) introduce Skip-Thought Vectors [43], an innovative approach to learning universal sentence embeddings that capture the semantic meaning of sentences by predicting their surrounding sentences in a text. Drawing inspiration from the skip-gram model used for word embeddings, this method extends the idea to the sentence level, aiming to encode sentences in such a way that the embeddings can be used to predict adjacent sentences. The Skip-Thought model employs an encoder-decoder architecture, both implemented recurrent neural networks (RNN) with gated recurrent units (GRUs). Training the Skip-Thought model involves a large corpus of books, utilizing the continuous stream of text to provide natural sentence sequences as training data. Kiros et al. demonstrate that the Skip-Thought vectors can be applied to a range of downstream NLP tasks including Semantic Text Similarity without requiring task-specific training data. The authors conduct evaluations on tasks

including semantic relatedness (especially thematic similarity), paraphrase detection, and text classification. The results reveal that Skip-Thought vectors achieve competitive or superior performance compared to other sentence embedding methods. The paper also discusses the scalability of the Skip-Thought model, noting that its performance improves with larger training datasets and they propose potential optimizations, including more sophisticated sampling techniques and parallel processing, to address the demand for significant computational resources during training.

Luong et al. (2015) [44] propose a novel approach that leverages both monolingual corpora and bilingual dictionaries to generate word embeddings that are semantically coherent within each language and aligned across languages. Their methodology involves obtaining bilingual word embeddings using a two-step process. The first part of the training aims at generating monolingual word embeddings for each language independently using large-scale corpora whereas the second part involves introducing a bilingual mapping process that aligns the monolingual embeddings into a common bilingual space. This alignment is made possible with the help of a bilingual dictionary, which provides word pairs that are translations of each other. The authors tested the effectiveness of the generated embeddings on a variety of tasks, such as word similarity, bilingual lexicon induction (which can be understood as the process of creating a bilingual lexicon without direct human intervention), and sentence retrieval across several languages. Their paper also investigates the impact of different sizes of bilingual dictionaries and monolingual corpora on the quality of the embeddings concluding that the quality of monolingual embeddings is largely retained across languages.

Wieting et al. present a different approach to generating sentence embeddings focused on paraphrastic relationships, by creating universal representations that capture meaning regardless of the specific wording [45]. Their study shows that effective sentence embeddings should ideally bring sentences with similar meanings closer together in the embedding space, even if their surface forms differ significantly. The authors propose a model trained on the Paraphrase Database (PPDB), to generate embeddings that emphasize semantic similarity over syntactic similarity. ParaNMT-50M uses a neural machine translation (NMT) framework trained on a corpus of over 50 million English sentence pairs. The core idea is that by understanding how to paraphrase a sentence, the model extracts an abstraction from the specific lexical and syntactic choices to the underlying meaning. Their approach employs a combination of recurrent neural networks (RNNs) with long short-term memory (LSTM) cells arranged in a siamese network architecture, where the model is trained to minimize the distance between embeddings of paraphrased sentence pairs while maximizing the distance between non-paraphrased pairs. This contrastive learning strategy, which is also implemented in the MPNet family of transformer models as we shall later see, is crucial for teaching the model to prioritize semantic content. Wieting et al [45]. conduct extensive evaluations across a variety of downstream NLP tasks, including sentiment analysis, textual entailment, as well as semantic textual similarity. The results demonstrate that ParaNMT-50M outperforms several traditional word embedding models such as GloVe and Word2Vec. Additionally, the authors comment that the diversity and size of the paraphrase dataset play a critical role in the model's ability to learn robust semantic representations. Of particular interest, is their finding that the ParaNMT-50M model achieves notably strong results in tasks involving semantic similarity, suggesting that its training on paraphrastic data directly contributes to its effectiveness in these applications.

Conneau et al. introduce InferSent [46], which generates universal sentence representations by leveraging natural language inference (NLI) data. The implementation of their model utilizes the Stanford Natural Language Inference (SNLI) and the Multi-Genre NLI (MultiNLI) corpora, which contain pairs of sentences annotated with labels indicating their relatedness. The InferSent model is built upon a bidirectional LSTM architecture, with max pooling applied over the LSTM outputs to condense the variable-length sentence representations into a fixed-size vector. The authors experiment with

different pre-trained word embeddings, including GloVe and fastText, while training involves optimizing the model to correctly classify the semantic relationship between sentence pairs. InferSent has been tested on a variety of downstream NLP tasks, such as sentiment analysis, question classification, and semantic textual similarity. The results demonstrate that InferSent significantly outperforms existing methods for sentence representation, including unsupervised approaches like SkipThought and simple word-averaging techniques. Their study suggests that supervised training on NLI data helps the model capture a wide range of semantic properties and that the diversity and size of the NLI datasets are key factors in the model's performance. However, they also underline the limitations of attention mechanisms in this context, since one of their findings is that complex attention mechanisms do not significantly outperform the simpler max pooling strategy, and it is precisely this insight that emphasizes the effectiveness of the chosen architecture in capturing sentence semantics without redundant complexity.

Cer et al. [47] introduce the Universal Sentence Encoder, a model designed to provide high-quality sentence embeddings that comes in two versions of the model: one based on a Transformer architecture and the other on a Deep Averaging Network (DAN): the Transformer model for tasks that require a deep understanding of sentence context and the DAN model which offers a computationally lighter alternative by averaging word and bi-gram embeddings before passing them through a feed-forward network. The Universal Sentence Encoder is trained on web news, forums, and question-answer pages, aiming to capture a wide diversity of linguistic phenomena. Cer et al. [47] conduct evaluations comparing the Transformer and DAN models, which reveal that while the Transformer version generally offers better performance on tasks requiring a deeper understanding of sentence structure, the DAN model provides a more efficient solution for applications where computational resources are limited. The paper also explores the role of training data size and diversity on the quality of the sentence embeddings. The findings of their research suggest that larger and more varied datasets contribute to more generalizable and effective embeddings, which makes the Universal Sentence Encoder a very good candidate for semantic similarity tasks.

The Sentence-BERT (SBERT) model [17] as we have seen, is a modification of the pre-trained BERT and RoBERTa networks that is designed to produce sentence embeddings for various tasks, including Semantic Textual Similarity (STS). The SBERT architecture uses a Siamese network structure and three distinct types of pooling operations which are added to the output of BERT/RoBERTa to derive a fixed-sized sentence embedding. These pooling operations include using the output of the CLS-token, computing the mean of all output vectors (MEAN-strategy, which is the default), and computing a max-over-time of the output vectors (MAX-strategy). Additionally, the Sentence-BERT model employs three distinct objective functions to generate sentence embeddings: The Classification Objective Function, the Regression Objective Function, and the Triplet Objective Function. SBERT has been trained on the SNLI and Multi-Genre NLI datasets and has outperformed average GloVe embeddings, average BERT embeddings, and the BERT CLS-vector, as well as other methods like InferSent and the Universal Sentence Encoder. The results are measured using the Spearman rank correlation between the cosine similarity of sentence representations and the gold labels for various STS tasks.

The MPNet paper by Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu presents a unified model that leverages both MLM and PLM strategies for pre-training, addressing shortcomings in prior approaches to enhance language comprehension and context awareness [48]. This advancement enables MPNet to better understand the intricacies of language, making it a significant contribution to NLP research and applications. The MPNet paper introduces a novel pre-training method that enhances language understanding by integrating ideas from both masked language modeling (MLM) and permuted language modeling (PLM). This approach is designed to capture bidirectional context

effectively. MPNet overcomes limitations in previous models that were either unidirectional or less efficient in context utilization. MPNet's pre-training technique masks some tokens and permutes the order of the remaining ones to train the model in predicting the masked words, considering all possible word orders. This method allows for a better understanding of word context and relationships, leading to improved performance on various downstream NLP tasks. The paper's results demonstrate that MPNet outperforms previous models on several benchmarks, indicating its effectiveness in language representation. Contrastive learning is a method of primarily unsupervised training of large language models that revolves around the principle of pulling similar data points closer together and pushing dissimilar ones apart in the embedding space. By doing so, it significantly improves the model's ability to discern and represent the nuanced semantic relationships between sentences. For instance, the "all-mpnet-base-v2" model, which is one of the models we have used for our semantic similarity experiments, is a sentence transformer model that has been fine-tuned on a large dataset using a contrastive learning objective. This model, based on the MPNet architecture, was fine-tuned to predict which sentences out of a set are pairs, within a dataset of over 1 billion sentence pairs [48]. It's particularly effective for encoding the semantic content of short documents and can be used in tasks such as semantic search and document clustering. The model maps sentences and paragraphs to a 768-dimensional vector space, and by design, it is intended for texts with a maximum sequence length of 384 tokens. For tasks involving longer sequences, further fine-tuning on relevant data may be necessary. However, in the context of SBERT, contrastive learning is implicitly employed through the triplet objective function, where the model is fine-tuned to recognize and adjust to the semantic similarities and differences among sentences. This approach not only refines the sentence embeddings to capture deeper semantic meanings but also facilitates a more accurate measurement of similarity using metrics like cosine similarity. The effectiveness of contrastive learning in semantic similarity tasks has been well-documented, presenting a robust framework for enhancing the quality and applicability of sentence embeddings in diverse NLP tasks [49].

C. Raffel et al. [50] developed the Google T5 (Text-To-Text Transfer Transformer) model that represents a significant evolution in the field of natural language processing that reframes all NLP tasks into a unified text-to-text format where every task is cast as generating text output from text input. This innovative approach simplifies the process of applying the model across a variety of tasks, from translation and summarization to question answering and classification. The architecture of T5 is built upon the traditional transformer model with a few modifications, such as using a relative position encoding and scaling up the size of the model and its training dataset. T5 was pre-trained on a multi-task mixture of unsupervised and supervised tasks, utilizing a colossal dataset compiled from publicly available text. T5 approaches semantic similarity tasks by converting sentences into embeddings or directly generating similarity scores as text output. By leveraging its extensive pre-training, T5 can understand nuanced differences and similarities between sentences, even when they are phrased differently but share similar meanings. This capability makes it highly adept at handling various subtleties in language, thus providing more accurate assessments of sentence similarity compared to traditional models and has provided the best results in our semantic similarity experiments. The T5 model's performance on sentence similarity tasks showcases its ability to generalize well to new texts, making it a powerful tool in the NLP practitioner's toolkit.

J.-J. Decorte et al [51] introduce JobBERT, a neural presentation model for job titles that takes advantage of the augmentation of a pre-trained language model and the co-occurrence information from extracted skill entities. JobBERT’s approach is based on semantic textual similarity where, as we have previously analyzed, the idea is that semantically similar sentences should lie close together in the vector space of representations. This idea is very similar to training a Siamese LSTM model on pairs of similar job titles, although JobBERT is based on the BERT encoder model to generate embeddings and overcomes

previous issues such as the presence of an extended list of labeled and normalized job titles and the continuous effort demanded to update such a list on a regular basis. The authors report that JobBERT leads to considerable improvements compared to generic sentence encoders for the task of job title normalization. Additionally, JobBERT incorporates a multi-task learning approach that simultaneously optimizes for both job title similarity and skill extraction, enhancing its performance and versatility. The model also leverages domain-specific data augmentation techniques to further refine its understanding of job titles across various industries. Furthermore, extensive experiments demonstrate that JobBERT outperforms traditional methods in both accuracy and computational efficiency, making it a robust solution for large-scale job title normalization tasks.

Wang et al. [52] present a novel strategy to overcome the lack of a large collection of high-quality, labeled data with textual similarity scores for semantic textual similarity (STS) tasks by utilizing OpenAI's GPT-4 to generate annotated data. Sim-GPT, however, does not output any similarity scores; rather it synthesizes reliable labeled data using GPT-4, and then a smaller STS model that is based on BERT and RoBERTa models is trained on the synthesized data. Sim-GPT demonstrates state-of-the-art performance on multiple STS benchmarks. The authors highlight that the synthetic data generated by GPT-4 closely mimics the distribution and complexity of real-world data, thereby enhancing the training process. Additionally, Sim-GPT employs a novel data augmentation technique that further improves the robustness of the STS model. The approach is particularly effective in low-resource settings, where obtaining high-quality labeled data is challenging and costly. Extensive evaluations reveal that Sim-GPT not only surpasses traditional methods but also generalizes well across different domains, making it a versatile tool for various STS applications.

4.5 Challenges and Limitations

Transformers have significantly advanced the field of natural language processing, including tasks related to semantic similarity. However, like all models, they come with their own set of challenges and limitations some of which are mentioned below:

Computational Resources: Transformers, especially larger models like GPT-3 or BERT-large, require substantial computational power for both training and inference. However, the sophisticated capabilities of these transformers come at a significant cost in terms of computational resources. Both the training and inference stages of these models require extensive processing power, typically provided by high-performance hardware such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). These specialized hardware components are adept at handling the parallel processing demands of machine learning algorithms, making them indispensable for executing the complex mathematical operations required by large models. The reliance on such high-end computational resources, however, poses a substantial barrier to entry. Researchers and practitioners who lack access to these advanced technologies may find it challenging to engage with the latest developments in AI. This digital divide can hinder innovation, particularly in environments with limited technological infrastructure or in academic institutions with constrained budgets. Moreover, the environmental impact of the extensive energy consumption required to power these advanced computational tools is an emerging concern. The carbon footprint associated with training and maintaining large-scale AI models necessitates a discussion on sustainable AI practices and the development of more energy-efficient computing technologies.

Data Requirements: While transformers benefit from large datasets for pre-training, obtaining such datasets that are sufficiently diverse and unbiased can be a challenge. Smaller or less-resourced languages may not have enough data to benefit from transformer models. assembling such comprehensive datasets is fraught with challenges. One of the primary concerns is ensuring the diversity and representativeness of the data. Datasets must encompass a broad spectrum of language use, demographics, and cultural contexts to avoid biases. Biased data can lead to skewed model behavior, where the AI system might perform well on tasks involving certain types of data but poorly on others, particularly those involving underrepresented groups or topics. Moreover, the issue of data scarcity becomes pronounced for languages that are less resourced. Many of the world's languages are underrepresented in digital form and lack the extensive written or digital resources that are available for languages like English, Chinese, or Spanish. This scarcity limits the ability of transformer models to learn and perform effectively in these languages, thus widening the technological divide between languages and potentially exacerbating cultural erosion. This challenge, however, extends beyond mere quantity. Quality and ethical considerations in dataset compilation involve meticulous curation to avoid the inclusion of harmful or sensitive content. The process of gathering and refining datasets must be governed by strict ethical standards to ensure that the training data does not perpetuate or amplify undesirable biases.

Interpretability: The complex architecture of transformers makes them less interpretable compared to simpler models. Understanding why a transformer model has deemed two sentences as semantically similar can be difficult, which limits the ability to diagnose and correct errors. As we have seen, when a transformer model evaluates semantic similarity between two sentences, it does so through a series of intricate transformations and interactions between embeddings. The decision process involves multiple layers where each input is related to every other input through attention scores. These scores and the subsequent transformations generate a final output that is difficult to backtrack through conventional means. This complexity makes it challenging to pinpoint exactly why the model perceives two sentences as similar, thereby complicating efforts to diagnose or correct errors in the model's reasoning. The lack of interpretability becomes a significant issue in high-stakes applications such as medical diagnosis, legal analysis, or any field where understanding the rationale behind a model's decision is as important as the decision itself. Without a clear understanding of how decisions are made, users may hesitate to rely on these models, which can limit their adoption and integration into critical decision-making processes.

Contextual Overfitting and over-reliance on Context: While transformers handle context well, they can sometimes overfit to the idiosyncrasies of the training data's context, which can result in reduced performance on out-of-distribution texts or in different domains. This specialization can lead to overfitting—where a model is so attuned to the specific contexts and idiosyncrasies of the training data that it performs poorly on texts that are out of distribution or from different domains. For example, a transformer trained predominantly on modern English text may struggle with older forms of English or highly technical documents that deviate from its training context. This overfitting can reduce the model's generalizability, limiting its usefulness across varied applications and resulting in decreased performance where the contextual cues differ significantly from those seen during training. In addition to overfitting, transformers can become overly dependent on contextual cues, prioritizing them over the actual semantics of the text. This issue is particularly evident when dealing with homonyms (words that are spelled the same but have different meanings) or sentences that are structurally similar yet carry different meanings. In such cases, the transformer may incorrectly assess sentences as semantically similar because it relies heavily on the contextual arrangement of words rather than their intrinsic meanings. This can lead to errors in tasks like sentiment analysis, where the precise understanding of

terms in context is crucial, or in language translation, where subtle differences in meaning can alter the interpretation of a text significantly.

Generalization: The ability of transformers to generalize from one domain to another is not always guaranteed. When a transformer model, pre-trained on a specific type of text like news articles or scientific literature, is applied to a different domain such as legal documents or conversational speech, its performance can degrade significantly. This degradation occurs because the linguistic cues and context-specific knowledge that the model has learned may not be applicable or might even be misleading in the new domain. To address these generalization issues, further fine-tuning or domain-specific training is often required. This involves re-training the transformer on a new set of domain-specific data, which can be both time-consuming and computationally expensive. The requirement for additional data collection and training also imposes a significant resource burden, particularly for less-resourced domains where obtaining large, annotated datasets is a challenge. Several strategies can be employed to enhance the generalization capabilities of transformer models. One approach is domain adaptation, where techniques are applied to adjust the model's knowledge to make it more applicable to the target domain without extensive re-training. Another method involves the use of transfer learning, where a model pre-trained on a large, diverse dataset is fine-tuned with a smaller, domain-specific dataset to help it adapt to new contexts more effectively.

5 *Methodology*

This chapter outlines the comprehensive methodology employed in this study, detailing the datasets, tools, and techniques used to achieve our research objectives. Our primary focus is on leveraging advanced NLP tools and datasets to develop robust models for Named Entity Recognition (NER) and semantic similarity tasks. The key components of our methodology include the ESCO taxonomy, the SpaCy library, Label Studio, and several critical datasets: Skillspan, HaHuJobs, and the STS benchmark. We also discuss the preprocessing steps taken to clean the datasets, the implementation of readability indices, and the training and fine-tuning of models for NER and semantic similarity. Below we present a skeleton of our methodology, covering each step from data preprocessing to model training and evaluation:

Datasets

- Skillspan Dataset: A curated dataset containing job postings and skill descriptions.
- HaHuJobs Dataset: Another dataset comprising job postings and relevant skill information from a different region, providing a diverse linguistic base.
- ESCO Taxonomy: The European Skills, Competences, Qualifications, and Occupations (ESCO) taxonomy provides a structured vocabulary for skills and occupations, essential for training semantic similarity models.
- STS Benchmark Dataset: A standard dataset used to evaluate the performance of models on semantic textual similarity tasks.

Tools and Libraries

- SpaCy Library: An open-source software library for advanced NLP in Python, used for training the Roberta model for NER.
- Label Studio: A versatile data labeling tool that facilitated the annotation process for the NER task.
- Scikit-learn is an open-source machine learning library for Python, designed for simple and efficient tools for data mining and data analysis. It provides various classification, regression, and clustering algorithms, built on top of NumPy, SciPy, and matplotlib.
- Kaggle: A platform for data science competitions and datasets, which provided access to some of the datasets used and facilitated collaboration and sharing of insights.
- Python: The primary programming language used throughout the study for implementing various NLP models, preprocessing data, and conducting analyses.
- NLTK (Natural Language Toolkit): A suite of libraries and programs for symbolic and statistical natural language processing, used for text preprocessing and analysis tasks.
- Plotly: A graphing library that makes interactive, publication-quality graphs online, used for visualizing the results of data analysis and model performance.
- Matplotlib: A plotting library for Python and its numerical mathematics extension, NumPy, used for creating static, interactive, and animated visualizations in the study.

Preprocessing

Preprocessing is a crucial step to ensure the quality and reliability of the datasets. The following preprocessing steps were applied to the Skillspan, HaHuJobs, and ESCO datasets:

- Cleaning NaN Values: All rows containing NaN values were removed to ensure data integrity.
- Removing Duplicates: Duplicate entries were identified and removed to prevent redundancy and bias in model training.
- Eliminating Empty Rows: Rows without meaningful content were discarded.
- Outlier Removal: Statistical methods were applied to identify and remove outliers that could skew the results.
- Readability Indices Implementation: The Flesch Reading Ease, Flesch-Kincaid Grade Level, and Gunning Fog Index were calculated to assess the readability of the text in the Skillspan and HaHuJobs datasets. These indices provide insights into the complexity and accessibility of the language used in the datasets.
- Calculation of Bigrams and Trigrams: Bigrams (pairs of consecutive words) and trigrams (triplets of consecutive words) were calculated to identify common multi-word expressions and phrases in the datasets.
- Identification of Most Frequent Common Pairs: The most frequent pairs of words (bigrams) were identified to understand common word associations and patterns in the text.
- Visualization with Wordclouds: The top 50 most common words in each of the datasets were visualized using Wordclouds, providing a visual representation of the most frequent terms and their prominence in the text.

Model Training and Fine-tuning

- NER with Roberta: The SpaCy library's implementation of the Roberta model was trained on a combined dataset (df_ner) created by merging the Skillspan and HaHuJobs datasets. This model was fine-tuned to recognize entities pertinent to skills and occupations.
- Semantic Similarity with HuggingFace Sentence Transformers (available at https://huggingface.co/sentence-transformers): Three models from the sentence-transformers library were utilized for semantic similarity tasks:
    - all_mpnet_base_v2
    - sentence_t5_base
    - msmarco_distilbert_cos_v5

These models were initially fine-tuned on the ESCO dataset to align their embeddings with the structured vocabulary of skills and occupations. Subsequently, a second fine-tuning was conducted using the STS benchmark dataset to enhance the models' performance on general semantic similarity tasks. The Spearman rank correlation was measured to evaluate the effectiveness of the fine-tuning process.

Dimensionality Reduction

To visualize the high-dimensional embeddings generated by the models, Principal Component Analysis (PCA) was applied. This technique reduced the dimensionality of the embeddings from 768 to 3, making it feasible to plot and visually inspect the relationships between different text elements.

This comprehensive methodology integrates the aforementioned datasets, advanced NLP tools, and rigorous preprocessing techniques to develop and fine-tune models for Named Entity Recognition and

semantic similarity. The detailed steps ensure that the models are trained on clean, high-quality data and are capable of performing robustly on real-world tasks.

5.1 The ESCO taxonomy

The ESCO (European Skills, Competences, Qualifications and Occupations) taxonomy is a European multilingual classification and framework aiming at describing, identifying and classifying occupations and skills relevant to the EU labor market, education, and training. ESCO aims to support job mobility across Europe by offering a "common language" on occupations and skills that can be used by a variety of interested stakeholders. This effort is reflected in the ESCO dataset (already at version 1.1.2 at the time of the writing of this thesis and available at https://esco.ec.europa.eu/en/classification/skill_main) provides a hierarchical taxonomy structured around three main pillars: occupations, skills/competences, and qualifications. Each occupation in the dataset is linked to relevant skills and competencies, providing a clear picture of what is required for a person to work in a specific job. The ESCO dataset is continuously updated to reflect the evolving labor market and emerging skills, making it a dynamic tool that responds to the current and future needs of the European labor market.

According to the ESCO implementation manual [53] the mapping of a skill to an ESCO skill is conducted in such a way that this mapping is bidirectional. However, this is not always necessarily the case because:

- The skills pillar does not aim for a complete coverage of all skills
- A skill in ESCO does not necessarily have a broader skill
- There is less similarity between skill classifications than there is between occupational classifications, which makes mapping skills challenging. [53]

ESCO is continuously updated to reflect the changing needs of the job market, making it a dynamic tool that adapts to new skills and occupational trends. Its implementation across various platforms and services, including online job matching and educational resources, demonstrates its role in facilitating a more integrated approach to workforce development and planning in Europe.

5.2 The SpaCy library

SpaCy is an advanced open-source software library for natural language processing (NLP) in Python, designed to handle large-scale, real-world applications with speed and efficiency [54]. It supports a variety of NLP tasks including tokenization, part-of-speech tagging, named entity recognition, dependency parsing, sentence segmentation, text classification, lemmatization, morphological analysis, entity linking, and more. With its linguistically-motivated tokenization, SpaCy is equipped to manage detailed and complex linguistic data efficiently.

One of the notable strengths of SpaCy is its support for 75+ languages and 84 trained pipelines for 25 languages, making it highly versatile for global applications. It incorporates multi-task learning with pretrained transformers like BERT, and offers pretrained word vectors that enhance its capability to understand and generate nuanced language responses. This integration allows SpaCy to achieve state-of-the-art speed and robust, rigorously evaluated accuracy in its operations.

SpaCy is also designed with a production-ready training system that is optimized for performance and productivity, featuring an easy-to-use interface that scales to large volumes of text. It supports deep learning integration using popular frameworks like TensorFlow and PyTorch, making it suitable for advanced NLP applications. Developers can extend SpaCy's functionality with custom components and attributes, and it supports custom models from various frameworks. Furthermore, SpaCy includes built-in visualizers for syntax and named entity recognition (NER), which help in analyzing and understanding text structure and content effectively.

For easy model packaging, deployment, and workflow management, SpaCy provides tools that streamline these processes, making it an ideal choice for developers looking to implement NLP in production environments. Its comprehensive feature set and flexibility make SpaCy a leading choice in the field of natural language processing.

5.3 Label Studio

Label studio is an open-source data labeling platform that was designed to streamline the process of annotating various data types for machine learning tasks [55]. It supports a wide range of tasks, including image classification, object detection, text tagging, and more. This flexibility makes it a valuable tool for researchers and data scientists working on computer vision, natural language processing (NLP), and other AI applications.

One of Label Studio's key advantages is its user-friendly interface that allows users to define custom labeling tasks through a configuration interface, thus making the data annotation process more straightforward. This eliminates the need for complex programming and enables non-technical users to participate in the data annotation process. Additionally, it can be integrated with existing pipelines and work with existing machine learning models. This allows the pre-labeling of data, the online learning of the model as new annotations become available, and active learning when only the most complicated examples are labeled. Finally, Label Studio offers features like data versioning and collaboration tools, facilitating efficient teamwork on large-scale data labeling projects.

5.4 Data Preprocessing for NER

5.4.1 Skillspan Dataset

The Skillspan dataset [56] (available directly from the GitHub repository at https://github.com/kris927b/SkillSpan) is comprised of six (6) CoNNL (Conference on Natural Language Learning) files or three (3) JSON files. To conduct our entity recognition experiments we have used the CoNNL files, which are already tagged with the BIO tagging scheme, where:

- ‘B-Skill’, denotes the first token of an entity that is considered to be a skill
- ‘I-Skill’, denotes any subsequent tokens of that entity
- ‘O’, denotes all other tokens which are not skills

We merged all the CoNNL files, thus obtaining 188,470 rows of data. Following this, we converted the BIO tagging scheme to the specific format requested by the SpaCy library for entity recognition tasks.

The conversion was necessary due to the fact that the BIO tagging scheme uses whole tokens to delimit the beginning and end of an entity whereas the SpaCy library expects the entities to be delimited by indexing the first and the last character of the entity span. A sample of the Skillspan dataset in its original BIO-tagged format can be seen below:

Table 1: The Skillspan dataset in its original BIO-tagged format.

Index	Column 1	Column 2	Column 3
326	handling	B-Skill	O
327	missing/noisy	I-Skill	O
328	data	I-Skill	O
329	and	O	O
…	…	…	…
340	disseminate	B-Skill	O
341	research	I-Skill	O
342	results	I-Skill	O
343	through	O	O
344	technical	O	B-Knowledge
345	reports	O	I-Knowledge
…	…	…	…

In the table above, ‘handling missing/noisy data’ and ‘disseminate research results” are skill entities, whereas ‘technical reports’ is considered a knowledge entity and therefore the respective column was eliminated for our experiments. Extracting the BIO tagging scheme per sentence (as opposed to per token) we condensed the original Skillspan dataset to 11,541 rows of data:

Table 2: Skillspan dataset: BIO tagging scheme per sentence.

Index	Text	BIO
9541	This is your opportunity to join a team of world-class researchers problem solvers and designers who care about people and their needs.	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
338	Guidelines for applicants can be found here.	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

552	As <PROFESSION> <PROFESSION> you enable your team to get the most out their talent and by that deliver best in class solutions in our industry.	['O', 'O', 'O', 'O', 'B-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'O', 'O', 'O', 'B-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'I-Skill', 'O', 'O', 'O', 'O']
5558	Emphasis is on communications skills and teamwork.	['O', 'O', 'O', 'B-Skill', 'I-Skill', 'O', 'B-Skill', 'O']
642	Your advice will potentially change decisions on prices and even on products	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

Following this, we converted the BIO-tagged dataset to the format requested by the SpaCy library. This format comprises a Python dictionary with the keyword ‘entities’, followed by a list of any skill occurrences. The numbers within the parentheses indicate the start and end index of the respective entity. For instance, the phrase ‘Attention to detail’ is an entity starting at index 0 and ending at index 18, as can be seen below:

Table 3: Skillspan dataset with SpaCy entities.

Index	Text	SpaCy_Entities
8811	Senior Java Developer	{'entities': []}
2679	Attention to detail strong organizational skills and an absolute focus on quality of work	{'entities': [(0, 18, 'Skill'), (27, 47, 'Skill'), (65, 88, 'Skill')]}
1974	The successful candidate must be able to provide dynamic leadership in the development of research teaching and research dissemination as well as in working with business and society.	{'entities': [(41, 66, 'Skill'), (75, 97, 'Skill'), (99, 106, 'Skill'), (112, 133, 'Skill'), (149, 181, 'Skill')]}
281	If you are interested we would like to hear from you as soon as possible.	{'entities': []}
1750	From here your path may take you towards extended responsibilities within Application Development IT Delivery or IT Leadership.	{'entities': []}

We then proceeded to:

- remove all rows containing no entities.
- check to see if there are any negative indices within the extracted entities.
- remove all rows with missing values.
- remove duplicates from the column "Text"

thus, finally obtaining a dataset of 2,248 rows. Following this we have been able to calculate the distribution of tokens in the column “Text” as can be seen below:

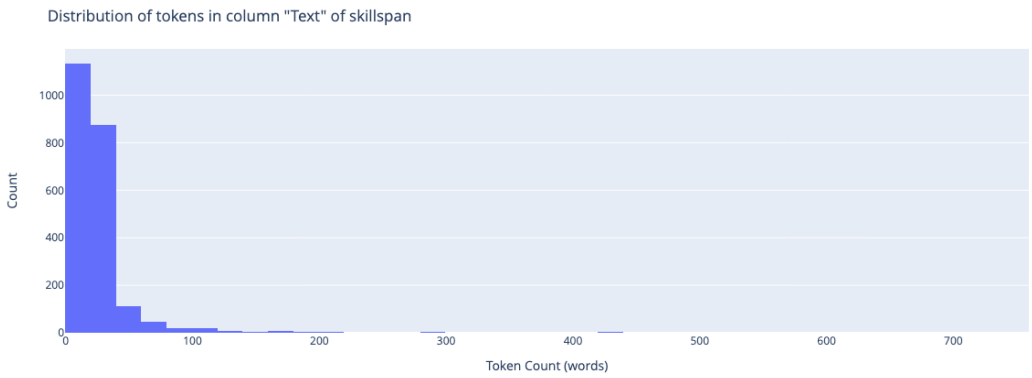

Figure 21: Distribution of tokens in column "Text" of skillspan.

Examining the distribution of Figure 21 it is easy to conclude that there exists a significant number of outliers, some of which may even contain more than 700 tokens as can be seen from the screenshot below:

```
outliers.sample(3)
```

Python

	Text	SpaCy_Entities	Number_of_tokens
1899	<ADDRESS> <ADDRESS> <LOCATION> - <LOCATION> Date posted: 2021-08-07 Likes: 0 Dislikes: 0 Love: 0 Job description: Job type: Full-time Industry: Financial Services Company size: 10k+ people Company type: Public Technologies web-services Job description As an experienced member of our <ORGANIZATION> <ORGANIZATION> <ORGANIZATION> we look first and foremost for people who are passionate around solving business problems through innovation and engineering practices .	{'entities': [(393, 462, 'Skill')]}	62
2081	We'd like to meet someone with the following experience: HTML CSS JavaScript Creating reusable components across micro frontends Identifying site issues and customer journeys using analytics and MVT JavaScript fundamentals and modern syntax Modern JavaScript frameworks Bonus points for React/Redux Node.js for server-side rendering building micro services Building automated acceptance tests Bonus points for Cypress and WebDriver.io frameworks Writing unit tests following TDD Software craftsmanship using engineering fundamentals SOLID KISS DRY and YAGNI writing secure code Contributing to and maintaining an inner source GitHub repo CI / CD pipelines including Docker workflows HTTP and working with CDNs cloud technologies Bonus points for Azure and Kubernetes Building and maintaining user interfaces that support multiple devices and browsers Including accessibility standards compliance and SEO Web application performance best practices Supporting critical applications by using logging and monitoring Working in agile teams Agile & lean practices including XP and pair programming Supporting the engineering community Job benefits: Generous salary discretionary bonus and pension matching Free modern onsite gym plus personal training On-site subsidised beauty salon cafe and coffee bar A bespoke flexible benefits scheme catered to you Best in class Learning & Development schemes and career development 25 days holiday & 1 extra day for your birthday! A dynamic social environment Life insurance free private medical care cycle to work scheme Huge staff discounts and sample sales Company description: About <ORGANIZATION> <ORGANIZATION> is an online retailer for fashion-loving 20-somethings around the world with a purpose to give our customers the confidence to be who they want to be We sell over 85,000 branded and <ORGANIZATION> Brand products through localised app and mobile/desktop web experiences delivering from our fulfilment centres in the <LOCATION> <LOCATION> and <LOCATION> Our propositions help bring our amazing products to almost every country in the world and we serve customers globally with increasingly tailored local experiences: relevant languages payment methods and delivery and return options in over 200 markets and in twelve languages <ORGANIZATION> drives innovation through technology and continues to push the boundaries of online retail In addition to a competitive salary staff discount and a benefits package every employee is offered support and guidance throughout their time at <ORGANIZATION> Personal and professional development is offered to all <ORGANIZATION> people through a range of bespoke and tailored learning solutions including career coaching and planning mentoring team development days and sponsored qualification support <ORGANIZATION> by the numbers £2.7bn revenue in 2018/19 85,000 products20.3m active customers20.2m social media followers2.3bn visits in 2018/1917.9m active mobile app downloads How we work As an online fashion retailer technology is at the core of <ORGANIZATION> The pace at <ORGANIZATION> is fast and our teams and systems have been set up to meet the challenge of delivering a huge portfolio of challenging work simultaneously Over 50 engineering teams own software from cradle to grave allowing independent release of features as well as the opportunity to work with cutting edge tech at massive scale <ORGANIZATION> has fully embraced the cloud and modern DevOps practices to support the velocity and scale of the organisation Most of our engineering effort is focused on creating differential customer experiences either through our digital platform's micro-services innovative machine learning solutions or creating insight to power business decisions Our culture We promote learning Tech Develops a full day of inspirational speakers and events is held on the last Friday of each month and is dedicated to the development of our permanent <ORGANIZATION> tech staff Our in-house <ORGANIZATION> Academy is also on hand to support development and offers access to great online learning tools such as Pluralsight Looop and Good Practice Our regular HackComps offer an opportunity to think about new challenges work with new people and jump out of your comfort zone Focused around a two-day 48hr intensive hack cross-functional teams address a set of business relevant challenges whether it's iterating on an existing experience or creating an entirely new one Some of our hackcomps have led to whole new teams and initiatives being spun-up to work on a brilliant idea that emerged Beyond learning and development we strive to maintain our inclusive and supportive culture Not only does our Women in Tech community play a key role in how <ORGANIZATION> Tech operates it also works to encourage future generations of young women into careers in tech for example by visiting schools and encouraging them to pursue STEM subjects If you want to know more about working in tech at <ORGANIZATION> take a look at our Medium blog here: <CONTACT> <CONTACT>	{'entities': [(77, 127, 'Skill'), (129, 173, 'Skill'), (333, 355, 'Skill'), (357, 391, 'Skill'), (446, 463, 'Skill'), (558, 576, 'Skill'), (578, 639, 'Skill'), (767, 806, 'Skill'), (947, 978, 'Skill'), (1012, 1033, 'Skill'), (1092, 1127, 'Skill')]}	748
931	Job description: The successful candidate will be expected to: Pursue high level research and teaching as well as establishment maintenance and strengthening of the collaboration with other research environments - especially international networks Participate in the consolidation of the Centre of Maritime Health and Society as a strong partner within occupational health and safety and health promotion interventions for the trades the industries and the organizations in the maritime field Develop carry out and publish the results of research projects and disseminate the information among maritime stakeholders Attract external research funding to the centre Manage and support activities of the research projects at the centre Participate in the pre- and postgraduate teaching in global health as well as in relevant public health and social science courses and in relevant maritime education Supervise BSc MSc and PhD students at the <ORGANIZATION> <ORGANIZATION> <ORGANIZATION> <ORGANIZATION> Advise the <ORGANIZATION> <ORGANIZATION> <ORGANIZATION> the <ORGANIZATION> <ORGANIZATION> <ORGANIZATION> and other related organizations on matters of health and safety in the maritime fields .	{'entities': [(70, 88, 'Skill'), (94, 101, 'Skill'), (114, 210, 'Skill'), (493, 554, 'Skill'), (560, 614, 'Skill'), (616, 648, 'Skill'), (664, 692, 'Skill'), (899, 932, 'Skill'), (1001, 1006, 'Skill')]}	167

Figure 22: Sample of outliers from the Skillspan dataset.

To effectively remove outliers from a dataset, we began by calculating the first quartile (Q1) and the third quartile (Q3), which represent the 25th and 75th percentiles of the data, respectively. Next, we determined the interquartile range (IQR) by subtracting Q1 from Q3 (IQR = Q3 - Q1). This IQR measures the spread of the middle 50% of the data. Outliers are then identified as any data points that fall below Q1 - 1.5IQR or above Q3 + 1.5IQR. These outlier data points can be removed to ensure the

dataset more accurately reflects the central tendency and variability of the majority of the data, thus obtaining the following balanced distribution:

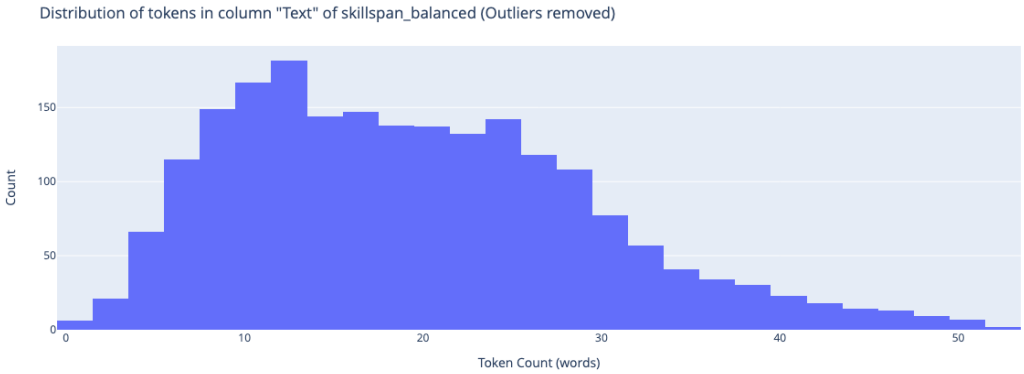

Figure 23: Distribution of tokens in column "Text" of skillspan_balanced (Outliers removed).

5.4.2 HaHuJobs Dataset

HaHuJobs [57] is a cloud-based service designed to collect and manage structured data on the Ethiopian labor market, connecting job seekers in major cities and industrial parks with potential employers. The platform is built as a collection of microservices that offer various functionalities, including data-driven job matching, student tracking, capacity building, and jobseeker assessment. It uses biometric identification and automated matching to ensure the right candidates are paired with the right job openings. The ecosystem caters to job seekers, employers, government agencies, and development partners, addressing data and automation gaps in the labor market. HaHuJobs also features deliverology tracking and Muya, an online career development platform for university and TVET graduates. This platform simplifies the verification of qualifications and tracks recruitment and employment trends. Additionally, it aggregates job vacancies from various sources.

The founders of HaHuJobs have expertise in labor economics and software development, supported by international labor market experts who provide technical advice. They continuously work on enhancing the platform to better serve the needs of the Ethiopian labor market. The HaHuJobs platform is supported by The Oxford Martin Programme on the Future of Development at the University of Oxford. This program aims to enhance economic opportunities in low- and middle-income countries through research and evidence-based strategies.

Due to their inherent complexity, transformer-based models necessitate vast amounts of labeled training data to achieve optimal performance and mitigate the risk of overfitting. To address this challenge and ensure the generalizability of our models, we adopted a human-in-the-loop approach by manually annotating a comprehensive dataset tailored specifically to our task. We utilized Label Studio to facilitate this annotation process, categorizing data with the following labels: “Skill,” “Occupation,” “Qualification,” and “Experience.” For the purposes of our thesis, we focused exclusively on the “Skill” label to train and test the machine learning models. Below is a screenshot of the environment of Label-studio where the annotation processing and the additional tagging of the HaHuJobs dataset took place:

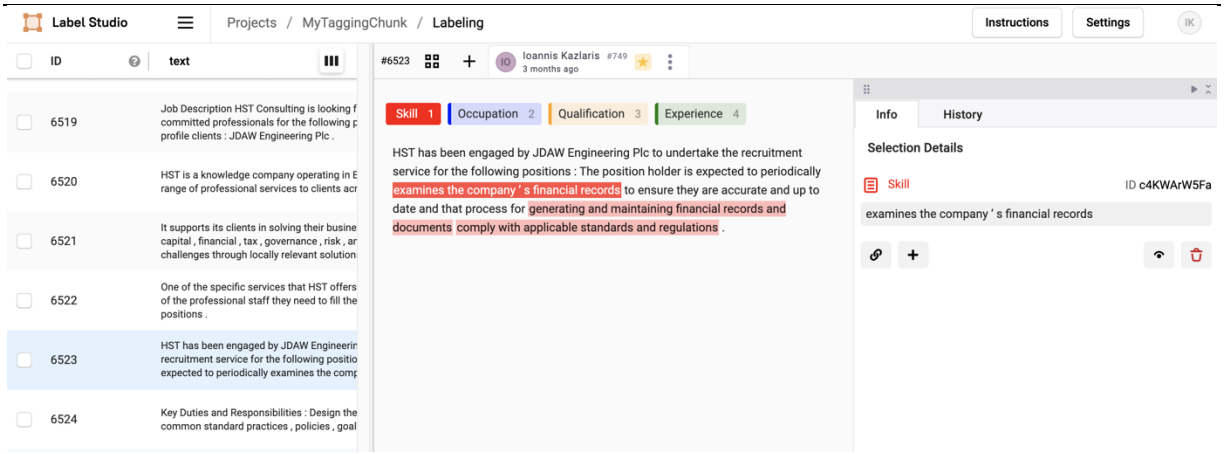

Figure 24: The environment of Label-Studio

After the annotation process, the HaHuJobs dataset comprised 5,608 rows. This dataset underwent a cleansing process to remove NaN values, duplicates, empty rows, and outliers. As a result, we refined the dataset to 3,683 rows. Subsequently, we merged the Skillspan and HaHuJobs datasets, producing a combined dataset of 5,520 rows. This dataset was then loaded into a Pandas DataFrame and named df_ner. This final dataset served as the basis for our Entity Recognition experiments, which were conducted using the SpaCy library. The same preprocessing has also been applied to the HaHuJobs dataset:

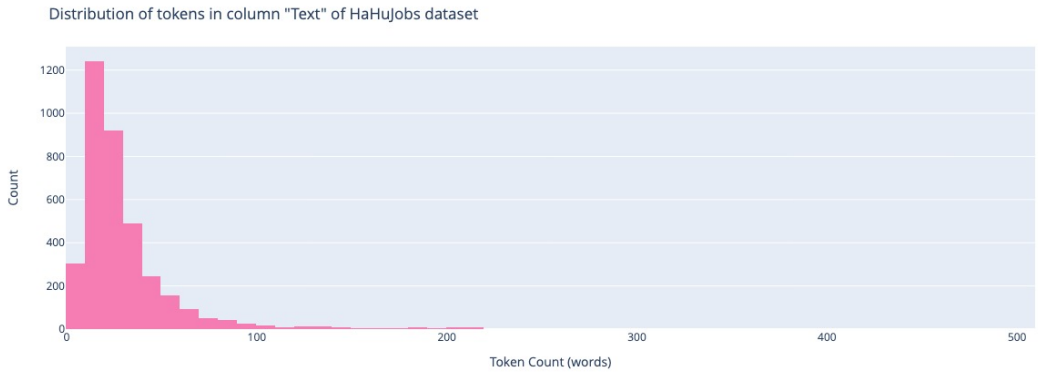

Figure 25: Distribution of tokens in column "Text" of HaHuJobs dataset.

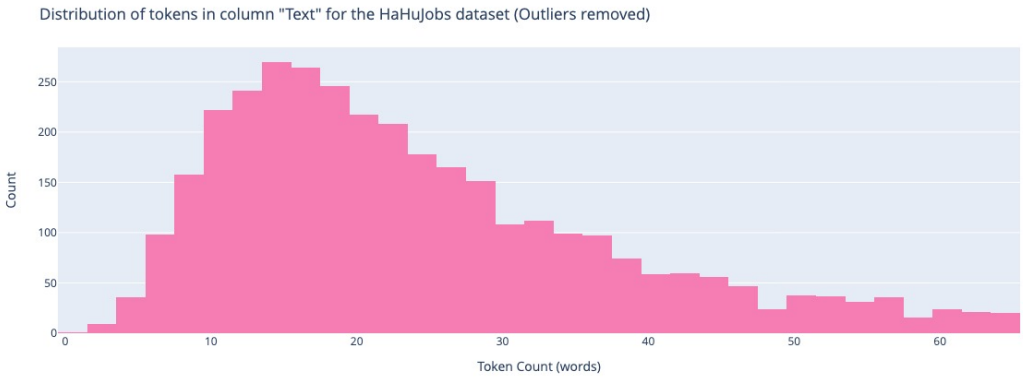

Figure 26: Distribution of tokens in column "Text" for the HaHuJobs dataset (Outliers removed).

After the preprocessing of both datasets, we concatenated them, thus resulting in a dataset consisting of 5520 rows, which we have named df_ner, and it is the dataset we used to conduct our NER experiments. A comparison between the distributions for all the datasets can be seen in the following diagram:

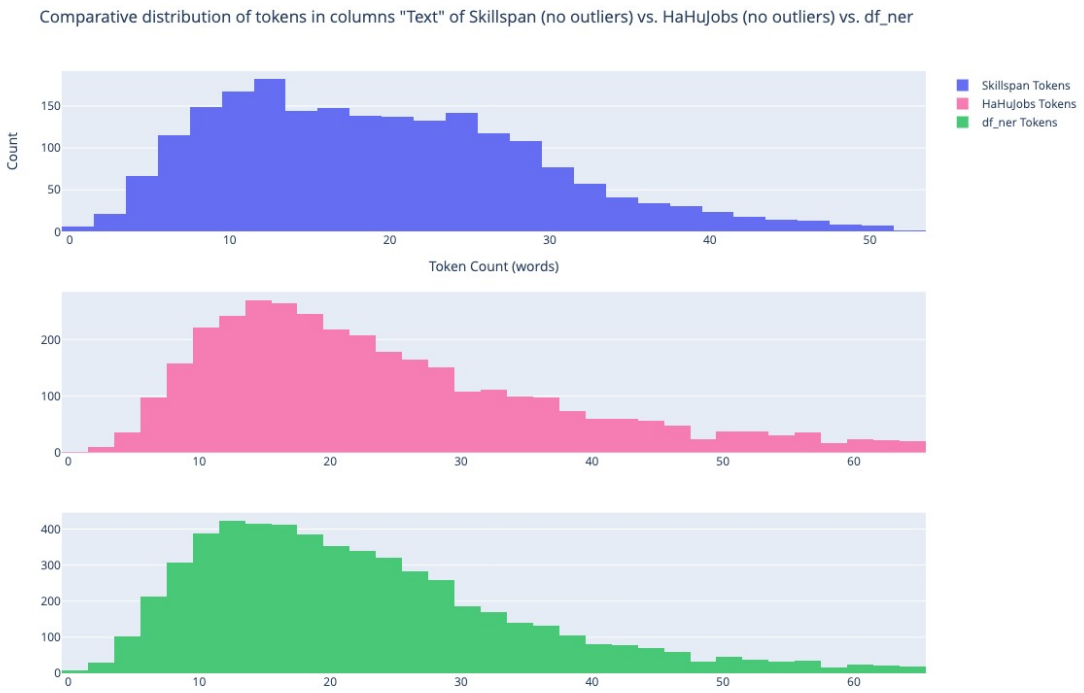

Figure 27: Comparative distribution of tokens in columns "Text" of Skillspan (no outliers) vs. HaHuJobs (no outliers) vs. df_ner.

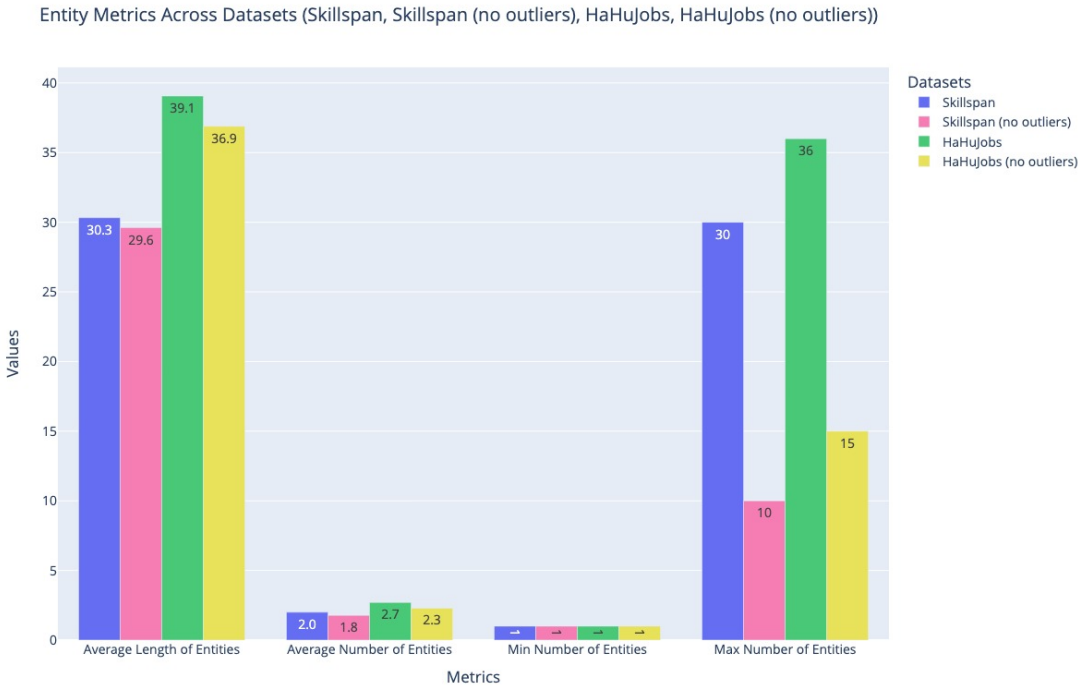

Figure 28: Entity Metrics Across Datasets (Skillspan, Skillspan (no outliers), HaHuJobs, HaHuJobs (no outliers)

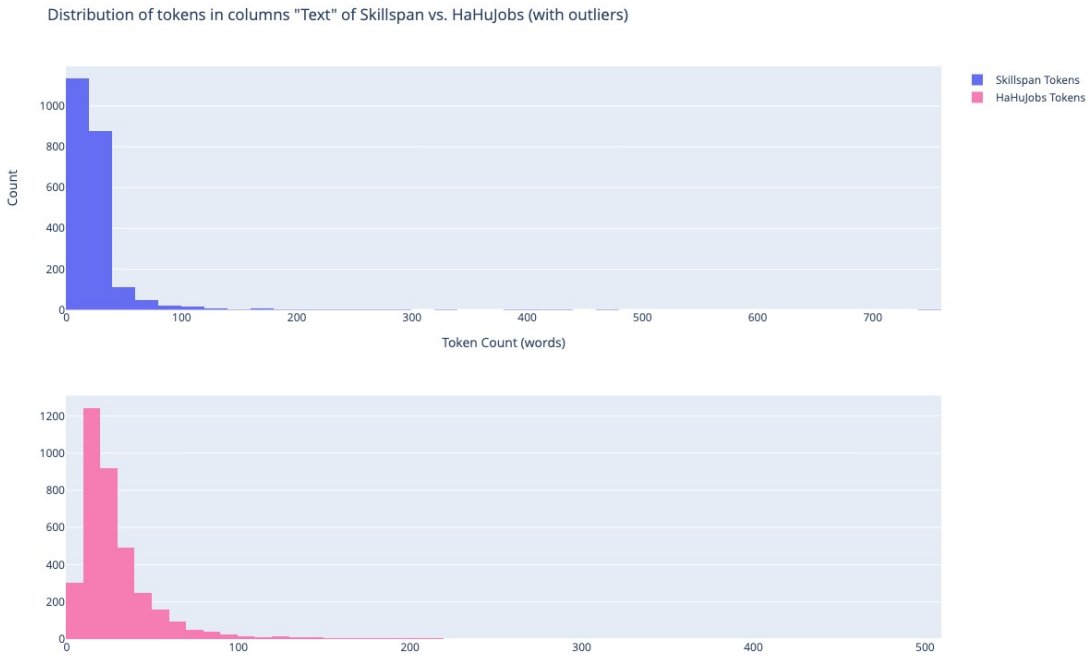

Figure 29: Distribution of tokens in columns "Text" of Skillspan vs. HaHuJobs (with outliers).

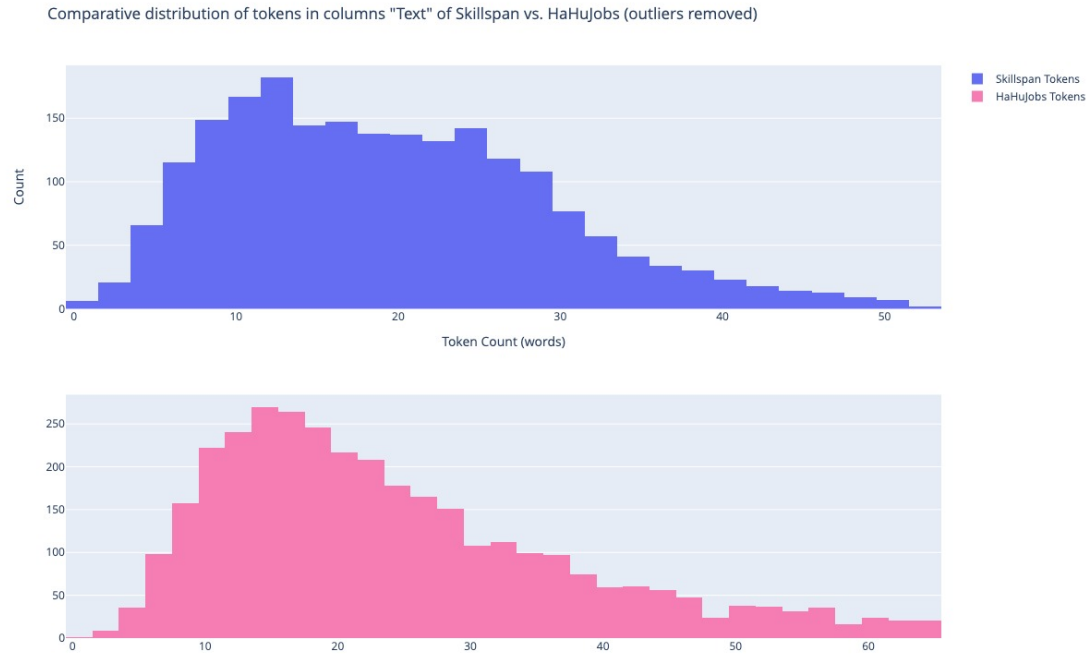

Figure 30: Comparative distribution of tokens in columns "Text" of Skillspan vs. HaHuJobs (outliers removed).

Additionally, in order to get a better understanding of the nature of our datasets, we have generated plots that depict:

- Number of tokens versus the number of entities.
- Number of tokens versus Flesch reading ease.
- Number of tokens versus Flesch Kincaid Grade.

- Number of tokens versus Gunning Fog index.

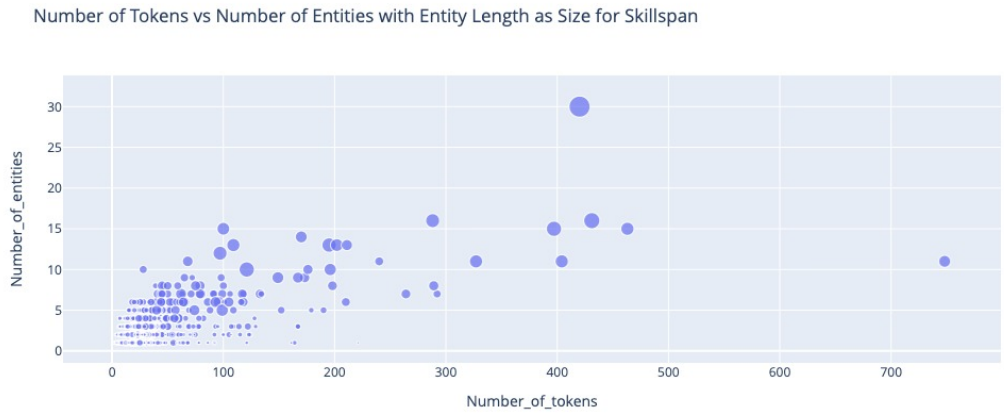

Figure 31: Number of Tokens vs Number of Entities with Entity Length as Size for Skillspan.

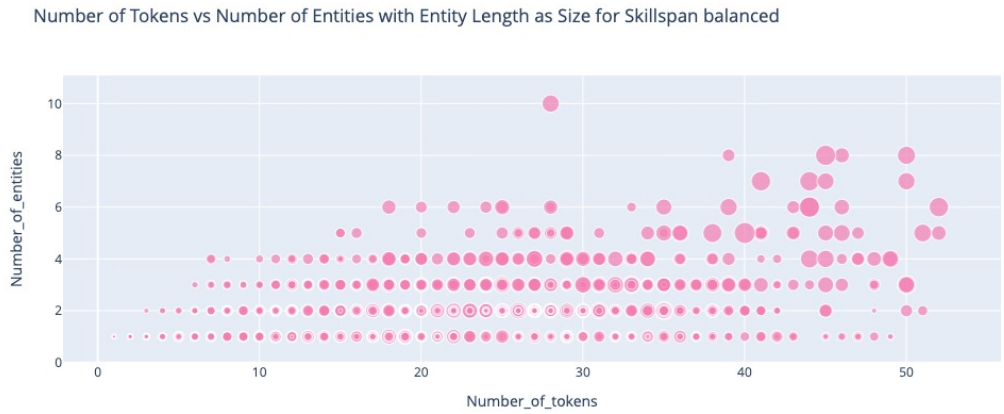

Figure 32: Number of Tokens vs Number of Entities with Entity Length as Size for Skillspan balanced.

Figures 31 and 32 offer valuable insights into the relationship between the number of tokens and the number of extracted entities, with entity length represented by the size of the markers in the dataset Skillspan. More specifically, in figure 31, the dataset shows a wide range of token counts, from very few tokens to over 700 tokens while the number of entities extracted also varies broadly, from 0 to over 30. There are noticeable outliers, particularly where the number of tokens is significantly high (e.g., around 700 tokens), which also corresponds to a relatively high number of entities. Regarding the entity length we observe that larger entity lengths (represented by bigger bubbles) are more dispersed and tend to appear with higher numbers of tokens. This suggests that longer entities are more likely to be found in longer text segments.

In figure 32, the outliers have been removed and we see that the token counts are more tightly clustered, mainly ranging from 0 to around 50 tokens. The number of entities also becomes more concentrated, typically between 0 and 10 entities. The removal of outliers results in a more uniform distribution. There are fewer extreme cases, and the relationship between the number of tokens and the number of entities

becomes clearer. Even without outliers, larger entities are generally associated with a higher number of tokens, but the variation is less pronounced. This indicates a more consistent pattern in how entities are distributed relative to the length of the text.

Figure 32 has a significantly reduced range of token counts and number of entities, providing a clearer view of the typical data distribution without the influence of outliers. The removal of outliers revealed more consistent patterns in the data. For example, it becomes easier to observe how the number of entities increases with the number of tokens. Figure 31 highlights how outliers can skew the perception of data distributions, showing extreme cases that may not represent the typical behavior within the dataset. Removing outliers from the Skillspan dataset provides a clearer and more consistent view of the relationship between the number of tokens and the number of entities extracted. The presence of outliers can significantly distort the interpretation, suggesting more extreme variability than actually exists in the bulk of the data. By focusing on the data without outliers, we gain a more accurate understanding of the typical entity extraction patterns, particularly how entity length correlates with the number of tokens in the text.

Additional insights into the nature of the datasets can be gained by plotting the number of tokens versus the various readability metrics:

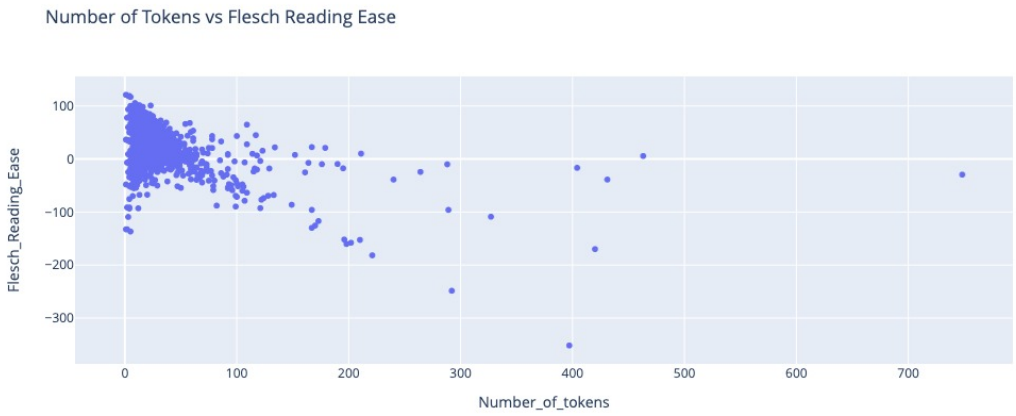

Figure 33: Number of tokens vs Flesch Reading Ease for SkillSpan.

The Flesch Reading Ease scores tend to decrease as the number of tokens increases, especially beyond 100 tokens. The spread of scores is quite wide for shorter texts but becomes more negative for longer texts. Interpretation: This suggests that longer job listings tend to be more difficult to read (lower Flesch Reading Ease scores). Job listings with fewer tokens are more readable, but as the length increases, readability decreases significantly.

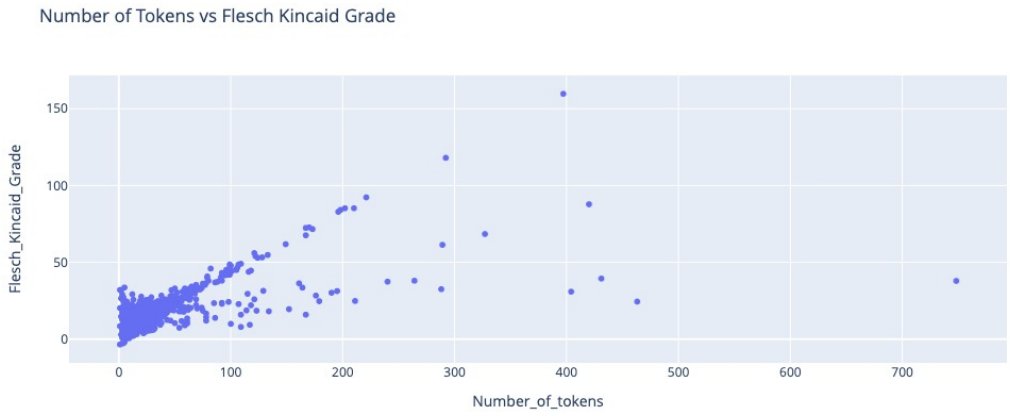

Figure 34: Number of tokens vs Flesch Kincaid Grade for SkillSpan.

The Flesch Kincaid Grade tends to increase as the number of tokens increases. There is a clear upward trend, especially within the first 100 tokens. Interpretation: Longer job listings are generally written at a higher-grade level, meaning they require a higher education level to be easily understood. Shorter job listings are written at a lower grade level and are easier to understand.

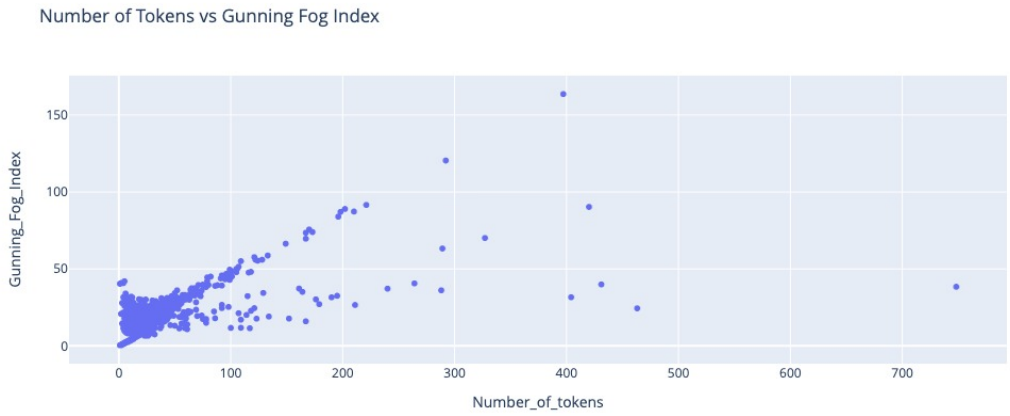

Figure 35: Number of tokens vs Gunning Fog Index for SkillSpan.

Similar to the Flesch Kincaid Grade, the Gunning Fog Index also tends to increase with the number of tokens. The trend is quite evident, showing that longer texts are more complex. Longer job listings are more complex and harder to read according to the Gunning Fog Index. Shorter listings are simpler and more straightforward.

Across all three readability measures, there is a clear trend that longer job listings are more difficult to read. This is consistent with the general understanding that longer texts often use more complex language and structure. Implications for Job Listings: For job listings to be more accessible and readable, it's beneficial to keep them concise. Shorter, more straightforward listings will be easier for a wider

audience to understand. The remainder of the plots can be seen in appendix A (Figures 59-69) so as not to obstruct the flow of our thesis.

Additionally, we have calculated the maximum, minimum and average number of tokens for the SkillSpan and HaHuJobs as can be seen below:

Table 4: Maximum, minimum and average number of tokens among outliers

Dataset	Maximum Number of Tokens	Minimum Number of Tokens	Average Number of Tokens
Skillspan	748	53	116.7
HaHuJobs	500	66	117.73

These metrics indicate a significant variability in the token counts of outliers in the Skillspan dataset. The maximum token count is quite high at 748, suggesting the presence of some very lengthy data points. The minimum token count of 53 shows that even the shortest outliers are relatively sizable. The average token count of 116.7 reflects a moderate level of verbosity among the outliers, implying that, on average, the outliers tend to be longer than typical data points in the dataset.

In the HaHuJobs dataset, the outliers also exhibit notable variability, though the range is slightly narrower compared to the Skillspan dataset. The maximum token count among outliers is 500, indicating that the longest outliers are shorter than those in the Skillspan dataset. The minimum token count is 66, which is higher than the minimum in the Skillspan dataset, suggesting that the shortest outliers in HaHuJobs are longer. The average token count of 117.73 is quite similar to the Skillspan dataset, showing that on average, the verbosity of outliers is consistent across both datasets.

5.4.3 Bigrams and Trigrams

Bigrams and trigrams are types of n-grams, which are contiguous sequences of n items from a given sample of text or speech. Specifically, bigrams consist of two consecutive words, while trigrams consist of three consecutive words. These sequences are used in various natural language processing (NLP) tasks to capture the context and relationships between words in a text. By analyzing bigrams and trigrams, we can gain insights into common phrases and word combinations, which is particularly useful in tasks such as text prediction, sentiment analysis, and machine translation. For example, in text prediction, knowing that "machine learning" is a common bigram can help an algorithm predict "learning" after "machine". Similarly, in sentiment analysis, identifying trigrams like "not at all" can improve the understanding of negations and intensifiers in sentences.

Following our preprocessing we have been able to calculate the top 20 bigrams and top 20 trigrams in the column “Text” of both datasets, namely SkillSpan and HaHuJobs, as can be seen below:

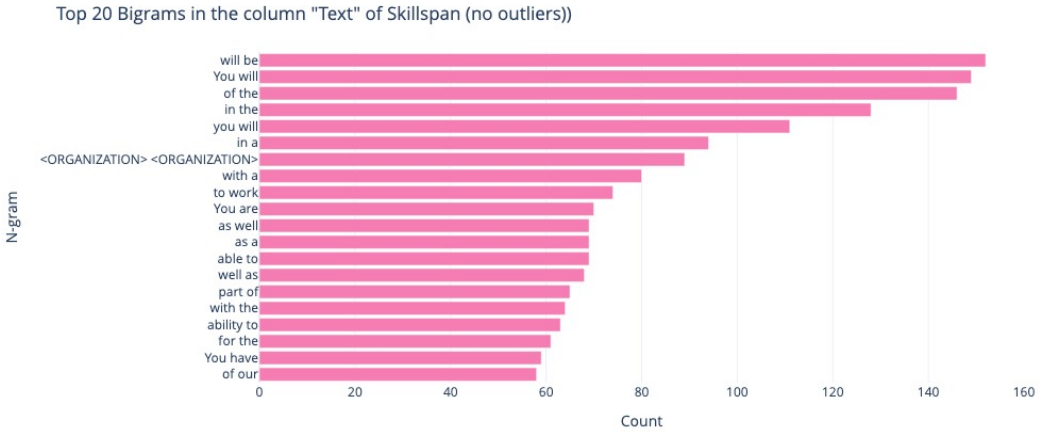

Figure 36: Top 20 Bigrams in the column "Text" of Skillspan (no outliers).

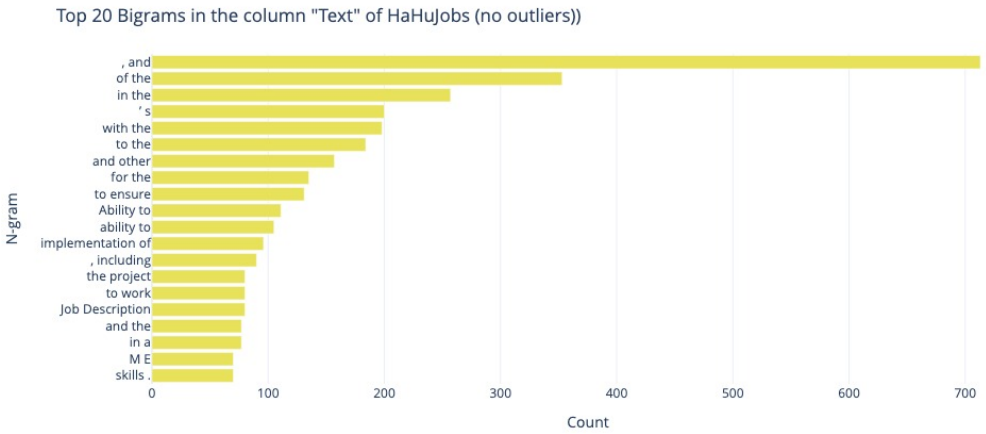

Figure 37: Top 20 Bigrams in the column "Text" of HaHuJobs (no outliers).

Figure 38: Top 20 Trigrams in the column "Text" of Skillspan (no outliers).

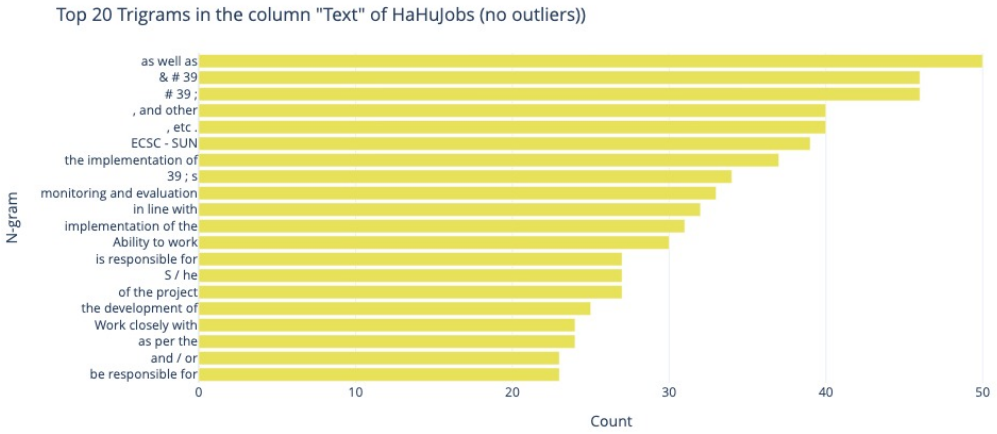

Figure 39: Top 20 Trigrams in the column "Text" of HaHuJobs (no outliers).

5.4.4 Co-occurrence Matrix and Common Pairs

A co-occurrence matrix is a statistical tool used in natural language processing (NLP) to analyze the frequency with which pairs of words appear together in a given corpus or text. This matrix is typically a square grid where both the rows and columns represent unique words from the text. Each cell in the matrix contains a count indicating how often the word corresponding to the row and the word corresponding to the column appear together within a specified window of context, such as within the same sentence or a fixed number of words apart.

Common pairs, or frequently co-occurring word pairs, are identified by examining the entries in the co-occurrence matrix. These pairs are crucial for understanding word relationships and semantic structures within the text. For instance, in a corpus related to technology, the words "machine" and "learning" might frequently co-occur, highlighting their strong association. We have proceeded to calculate the following metrics:

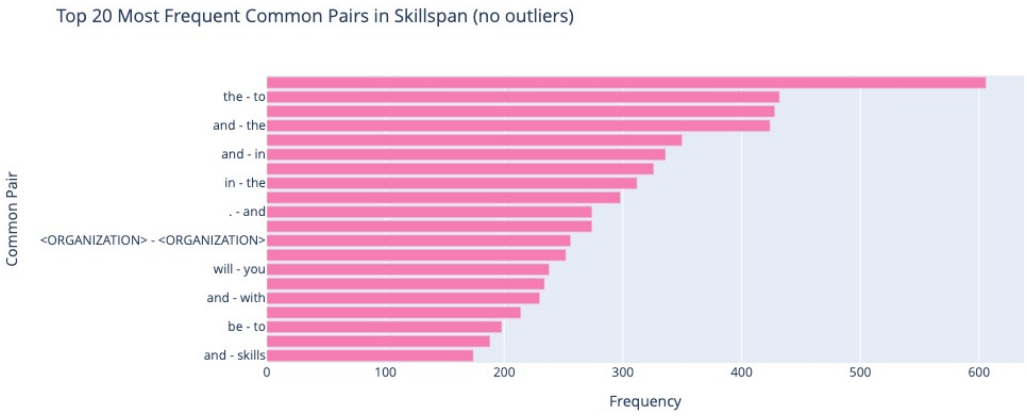

Figure 40: Top 20 Most Frequent Common Pairs in Skillspan (no outliers).

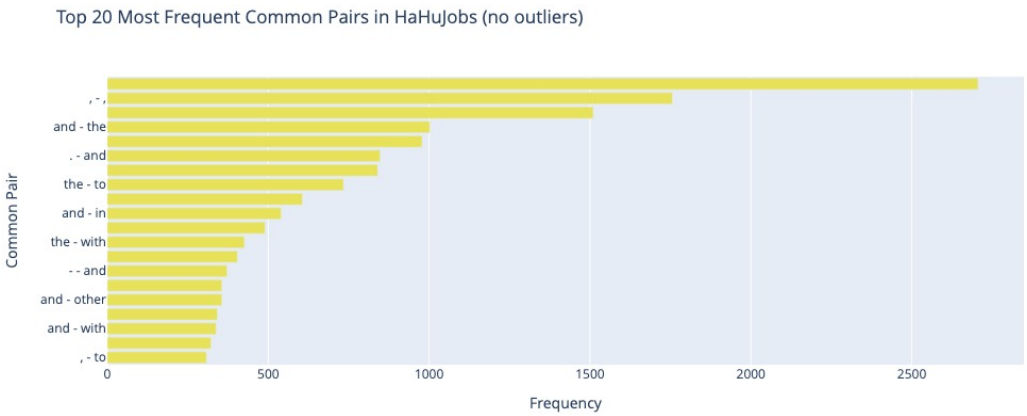

Figure 41: Top 20 Most Frequent Common Pairs in HaHuJobs (no outliers).

5.4.5 Text Readability

Readability metrics are tools used to evaluate how easily a piece of text can be read and understood. These metrics consider factors such as sentence length, word complexity, and overall text structure to assign a readability score. Common readability metrics include the Flesch-Kincaid Grade Level, the Gunning Fog Index, and the SMOG Index, among others. By assessing these elements, readability metrics help writers and researchers ensure their content is accessible to their intended audience, promoting clearer communication and better comprehension. To acquire a better understanding of our datasets readability we used the following metrics:

Flesch Reading Ease Score

The Flesch Reading Ease Score is a quantitative tool designed to assess the readability of a text. Developed by Rudolf Flesch in the 1940s, it evaluates how easy it is to understand a passage in English. The calculation of this score is based on the length of words and sentences, which correlates with the text's complexity. The formula for the Flesch Reading Ease Score is:

$$RE = 206.835 - 1.015\left(\frac{total\ words}{total\ sentences}\right) - 84.6\left(\frac{total\ syllables}{total\ words}\right)$$

This score typically ranges from 0 to 100, where higher scores indicate material that is easier to read. Texts with a score between 90 and 100 are considered suitable for an 11-year-old student, while scores between 0 and 30 are best suited for university graduates.

The usefulness of the Flesch Reading Ease Score extends across various fields, including education, where it helps in selecting texts with appropriate difficulty levels for students, and in marketing, where it assists in crafting messages that are understandable by a target audience. Additionally, it has been incorporated into laws in several U.S. states, mandating a minimum readability level for insurance policies to ensure that they are understandable to a broad audience. The score remains widely used and

is integrated into various word processing software and readability tools, highlighting its enduring relevance in evaluating written content.

Flesch Kincaid Grade Level

The Flesch-Kincaid Grade Level formula is a widely recognized tool used to determine the readability of English texts, tailored specifically to estimate the U.S. school grade level necessary to comprehend the text. Developed by Rudolf Flesch and J. Peter Kincaid, and initially commissioned by the U.S. Navy, this formula was designed to improve the clarity of technical manuals and documentation.

The calculation of the Flesch-Kincaid Grade Level is based on the structure of the English language, particularly the average sentence length (ASL) and the average number of syllables per word (ASW). The formula is:

$$GL = 0.39 * (ASL) + 11.8\,(ASW) - 15.59$$

where:

- ASL is the average sentence length (the number of words divided by the number of sentences)
- ASW is the average number of syllables per word (the number of syllables divided by the number of words)

This formula results in a score that correlates with the U.S. grade levels; for example, a score of 8.0 suggests that the text is suitable for an eighth grader. The simplicity of its application and the clarity of its results make the Flesch-Kincaid Grade Level an effective tool for assessing text suitability in educational settings and ensuring that language in public documents is accessible and clear. Its usage spans various domains, including education, where it helps in crafting age-appropriate literature and textbooks, and in government and legal environments, where it ensures public documents are comprehensible to a wider audience.

Gunning Fog Index

The Gunning Fog Index is a readability test designed to gauge the complexity of English writing. Developed in 1952 by Robert Gunning, an American businessman, the index aims to reduce unnecessary complexity in business writing. The formula uses the average sentence length and the percentage of complex words in a text to calculate the grade level required for comprehension. A complex word is typically defined as one that has three or more syllables, excluding familiar jargon, proper names, and compound words. The formula for the Gunning Fog Index is as follows:

$$GFI = 0.4 * (ASL + PCW)$$

where:

- ASL is the average sentence length (the number of words divided by the number of sentences)
- PCW is the percentage of complex words

This result represents the number of years of formal education a person needs to understand the text on the first reading. For example, a Gunning Fog score of 12 suggests that a twelfth grader (around 17-18 years old) could understand the text. The Gunning Fog Index is particularly useful in settings where clarity and ease of understanding are crucial, such as in legal, medical, and technical writing. By

ensuring texts do not exceed the comprehension levels of their intended audiences, the index helps improve communication efficiency and accessibility. It's a valuable tool for editors, writers, and communicators who aim to reach a broad audience with clear and readable content.

Applying readability metrics offers several significant benefits. Firstly, it ensures that the textual data is accessible and comprehensible to a wider audience, which is crucial for accurate annotation and subsequent machine learning processes. By evaluating and enhancing readability, we can identify and rectify complex, ambiguous, or convoluted language that might hinder understanding or introduce errors. This, in turn, improves the quality of the training data, leading to more effective and reliable machine learning models. Additionally, enhanced readability facilitates better human-in-the-loop annotation, as annotators can more easily and consistently interpret the text, thereby increasing the accuracy and efficiency of the labeling process. Overall, applying readability metrics helps in creating cleaner, more user-friendly datasets, ultimately contributing to the robustness and generalizability of our models. The NLTK library includes readily available methods for evaluating readability using metrics such as the Flesch Reading Ease, the Flesch-Kincaid Grade Level and the Gunning Fog Index. By applying these methods, we compared the readability of text across our datasets and found that the removal of outliers led to increased readability for all datasets. This enhancement in readability contributed to better training outcomes for our RoBERTa transformer, resulting in improved model performance.

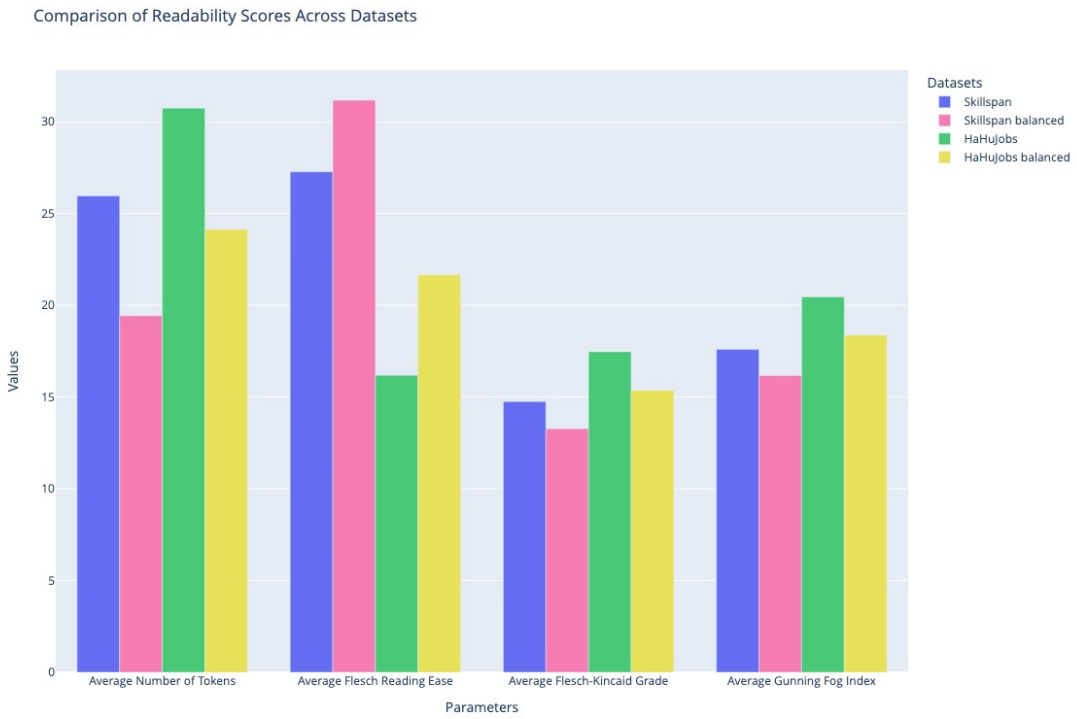

Figure 42: Comparison of Readability Scores Across Datasets

More specifically:

Table 5: Readability metrics for the HaHuJobs dataset (unbalanced, balanced)

Dataset	Average number of tokens	Flesch Reading Ease	Flesch-Kincaid Grade Level	Gunning Fog Index
Skillspan	25.96	27.28	14.76	17.59
Skillspan (no outliers)	19.43	31.17	13.27	16.17
HaHuJobs	30.74	16.19	17.47	20.46
HaHuJobs (no outliers)	24.13	21.66	15.36	18.37

The readability scores indicate that the outliers in the HaHuJobs dataset are significantly more difficult to read than the main body of text. This is evident from the lower Flesch Reading Ease score of 16.19 and higher Flesch-Kincaid Grade Level of 17.47 and Gunning Fog Index of 20.46 for the outliers. Removing outliers improves the readability of the text, as demonstrated by the increased Flesch Reading Ease score of 21.66 and the decreased Flesch-Kincaid Grade Level of 15.36 and Gunning Fog Index of 18.37. The presence of outliers skews the overall readability metrics, making the dataset appear more complex than it actually is. By removing these outliers, the text becomes more accessible, suitable for a slightly lower education level, and easier to understand.

Similarly, in the Skillspan dataset, the impact of outliers on readability is also noticeable. The average number of tokens drops from 25.96 to 19.43 when outliers are removed. This adjustment results in a Flesch Reading Ease score increase from 27.28 to 31.17, and reductions in the Flesch-Kincaid Grade Level from 14.76 to 13.27 and the Gunning Fog Index from 17.59 to 16.17. These changes further illustrate that outliers can significantly inflate the perceived complexity of a dataset. By eliminating these outliers, the readability metrics indicate that the text becomes more readable and understandable, aligning better with a slightly lower education level requirement.

5.4.6 Nervaluate

Nervaluate is an evaluation toolkit designed for assessing Named Entity Recognition (NER) systems. It provides a suite of tools to evaluate the performance of NER models using various metrics. These metrics include exact, strict, partial, and entity type-based evaluations, allowing for a comprehensive analysis of the model's performance. The exact and strict metrics focus on precise matches between predicted and actual entities, whereas the partial metric accounts for partially correct entities, and the entity type-based evaluation assesses the correctness of the entity types identified by the model.

The results presented here can be interpreted as follows:

- Correct indicates the number of entities that the NER system has identified correctly.
- Incorrect refers to entities that were misclassified.
- Partial shows the entities that were partially identified correctly.

- Missed counts the entities that the system failed to identify.
- Spurious represents entities that were incorrectly identified as entities.
- Possible and Actual denote the total number of entities that could be identified and the total number of entities the model identified, respectively.

The precision, recall, and F1-score are crucial metrics:

- Precision measures the accuracy of the identified entities, calculated as the ratio of correctly identified entities to the total identified entities.
- Recall measures the ability of the model to identify all relevant entities, calculated as the ratio of correctly identified entities to the total entities present in the dataset.
- F1-score is the harmonic mean of precision and recall, providing a single metric to evaluate the model's balance between precision and recall.

By examining these metrics, one can determine the strengths and weaknesses of an NER system and identify areas for improvement. High precision with low recall indicates the model is precise but misses many entities, while high recall with low precision indicates the model finds most entities but also makes many mistakes. A balanced F1-score indicates a well-performing model. We have used nervaluate in order to evaluate our NER experiments as can be seen in the ablation studies.

5.4.7 Word Clouds

Word clouds, also known as tag clouds, are visual representations of text data where the size of each word indicates its frequency or importance within a given body of text. Commonly used in natural language processing and data visualization, word clouds help to quickly convey the most prominent terms in a dataset. By displaying words in various sizes, they allow viewers to easily identify key themes and topics. Word clouds are particularly useful for summarizing large volumes of text data, such as survey responses, social media posts, or document corpora. The visual impact of a word cloud can make complex data more accessible and engaging, facilitating the identification of significant patterns and insights at a glance. Despite their simplicity, word clouds are powerful tools for initial exploratory data analysis and for presenting text data in an intuitive and visually appealing way. After removing the outliers from the datasets Skillspan and HaHuJobs, we can use Wordclouds to see a pretty visualization of the 50 most common words for those datasets:

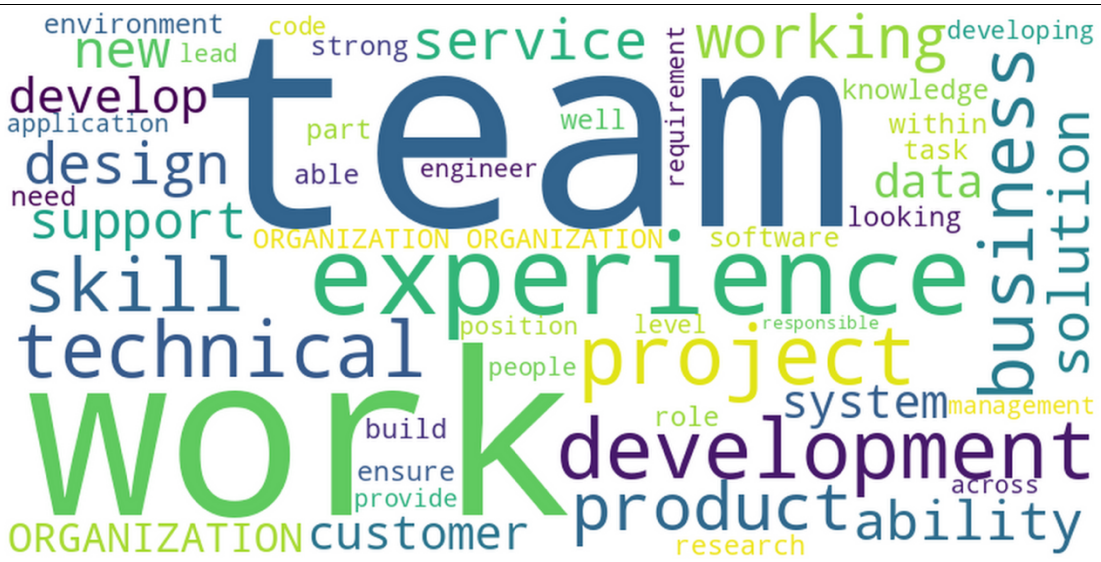

Figure 43: Wordcloud visualization for the dataset Skillspan, no outliers

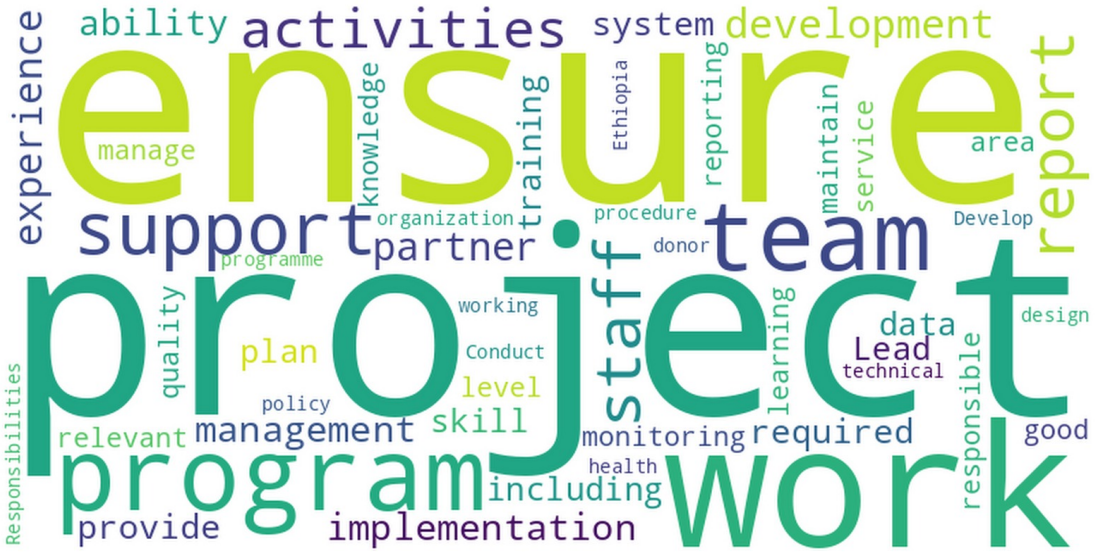

Figure 44: Wordcloud visualization for the dataset HaHuJobs, no outliers.

Figure 45: Wordcloud visualization for the dataset df_ner, no outliers.

5.5 Data Preprocessing for Semantic Similarity

The ESCO (European Skills, Competences, Qualifications, and Occupations) database provides a comprehensive classification system that is pivotal to understanding the European labor market. The esco_skills.csv file, which is available for download from the official ESCO website, contains an extensive dataset with detailed information on a wide array of skills and competencies, meticulously categorized according to the ESCO framework. This dataset is a critical resource for a diverse group of stakeholders, including researchers, policymakers, educators, and employers. It offers a standardized framework that is essential for analyzing the skills landscape within Europe.

With the ESCO dataset, users can delve into the specific skill requirements for various occupations, effectively identify existing skill gaps, and devise informed strategies to address them. Moreover, this data supports a range of labor market initiatives designed to enhance workforce development and facilitate greater mobility across European countries. The ESCO classification aids in aligning educational outcomes with labor market needs, fostering better employment opportunities, and promoting a more agile and responsive workforce. Consequently, it serves as a foundational tool for driving economic growth and social cohesion within the European Union by ensuring that individuals are equipped with the necessary skills to meet current and future job market demands. Here is a screenshot of the ESCO website after searching for the skill “manage musical stuff”:

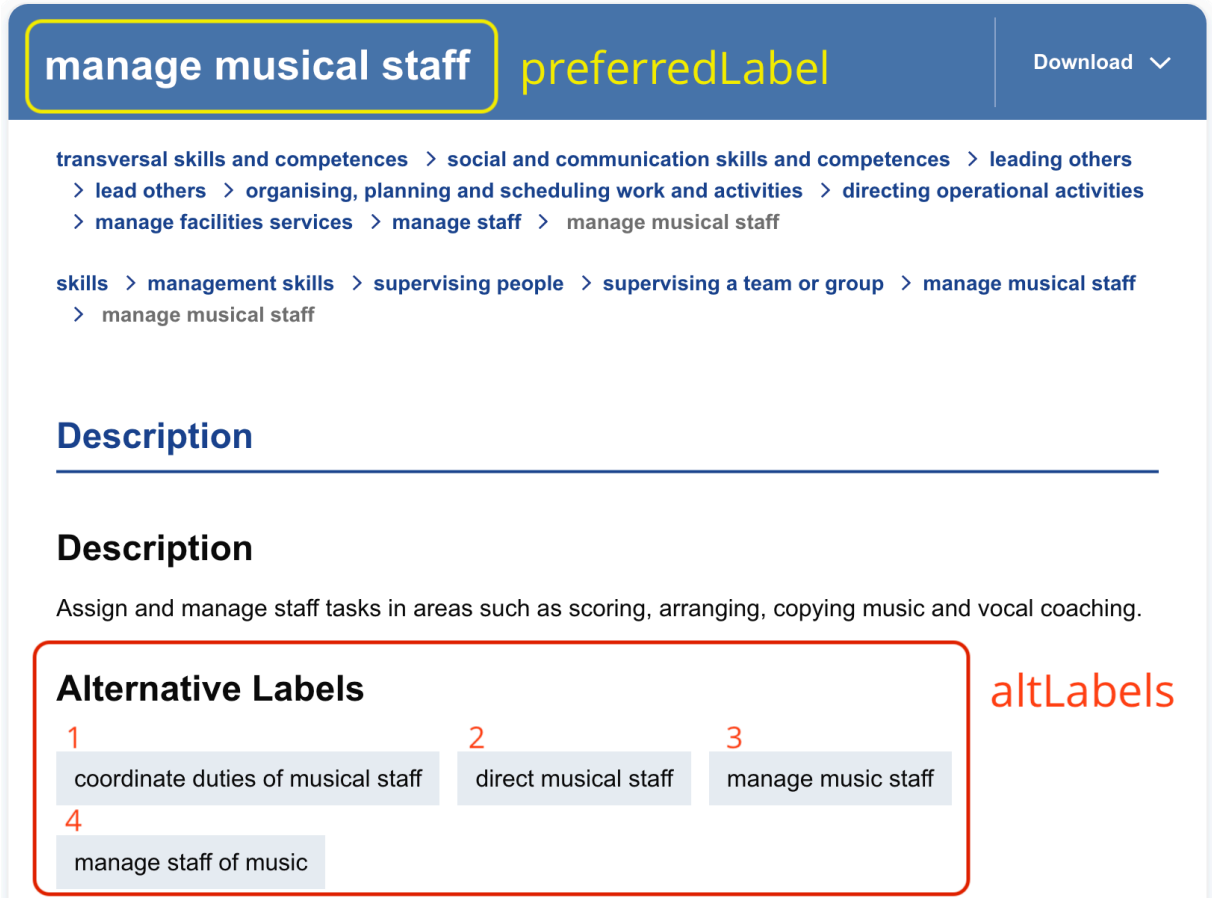

Figure 46: A screenshot of the ESCO website

The screenshot above features a specific skill or competence which we have highlighted in yellow color as the preferred label. Below this preferred label, a series of breadcrumb navigation links provide the hierarchical context of this skill within the broader framework of "transversal skills and competences" and other related categories. The description section explains that this skill involves assigning and managing staff tasks in areas such as scoring, arranging, copying music, and vocal coaching. This provides clarity on the specific responsibilities and activities associated with managing musical staff. Additionally, the screenshot lists several "Alternative Labels" for this skill, which are different terms or phrases that can be used interchangeably with "manage musical staff." These alternative labels include:

- Coordinate duties of musical staff
- Direct musical staff
- Manage music staff
- Manage staff of music

These alternative labels help in understanding the various ways this skill can be described or referred to in different contexts or documentation. Below we can see a sample of the dataset contained in the esco_skills.csv file after loading it to a Pandas DataFrame:

```
esco_skills.sample(3)
```

	conceptType	conceptUri	skillType	reuseLevel	preferredLabel	altLabels	hiddenLabels	status	modifiedDate	scopeNote	definition
5399	KnowledgeSkillCompetence	http://data.europa.eu/esco/skill/623902ce-b595-4de6-8de4-c20a7058c75e	skill/competence	occupation-specific	shine shoes	shine shoes	NaN	released	2016-07-05T08:58:28Z	NaN	NaN
569	KnowledgeSkillCompetence	http://data.europa.eu/esco/skill/0af062de-eb43-41e9-9b96-249e2cd22d26	skill/competence	sector-specific	use markup languages	use XHTML\nuse mark-up languages\nuse HTML\nuse markup languages	NaN	released	2022-07-05T14:37:18.921Z	NaN	NaN
1162	KnowledgeSkillCompetence	http://data.europa.eu/esco/skill/15d09ce6-9a55-4fd9-8853-cd25bea256da	skill/competence	sector-specific	mix wallpaper paste	prepare wallpaper paste\nwallpaper paste mixing\nmix wallpaper paste\nmixing wallpaper paste\npreparing wallpaper paste\nwallpaper paste preparation\nmixing of wallpaper paste\npreparation of wallpaper paste	NaN	released	2016-12-20T18:23:48Z	NaN	NaN

Figure 47: A screenshot of the esco_skills.csv dataset

By carefully examining the columns of the esco_skills.csv file and verifying the details from the ESCO website, we can observe that the ESCO taxonomy assigns a preferred label to each skill, chosen from a selection of alternative labels that can also describe the same skill. This observation provides a significant clue that can be used to construct the dataset that will be used for our semantic similarity experiments: all alternative labels describing a particular skill can be considered synonymous and that implies that they have a semantic similarity of 1. Since the alternative labels are included within a single row, we have split the contents of each row based on the special character '\n' and used the Pandas explode() method to produce as many rows as there are splits in a row. For the purposes of our thesis, we have deemed it necessary to work with a selection of essential columns: 'conceptUri', 'skillType', 'reuseLevel', 'preferredLabel', 'altLabels', and 'description', thus deriving a dataset of 97,675 rows.

```
esco_skills_useful_columns.head(20)
```

	conceptUri	skillType	reuseLevel	preferredLabel	altLabels	description
0	http://data.europa.eu/esco/skill/0005c151-5b5a-4a66-8aac-60e734beb1ab	skill/competence	sector-specific	manage musical staff	manage staff of music	Assign and manage staff tasks in areas such as scoring, arranging, copying music and vocal coaching.
0	http://data.europa.eu/esco/skill/0005c151-5b5a-4a66-8aac-60e734beb1ab	skill/competence	sector-specific	manage musical staff	coordinate duties of musical staff	Assign and manage staff tasks in areas such as scoring, arranging, copying music and vocal coaching.
0	http://data.europa.eu/esco/skill/0005c151-5b5a-4a66-8aac-60e734beb1ab	skill/competence	sector-specific	manage musical staff	manage music staff	Assign and manage staff tasks in areas such as scoring, arranging, copying music and vocal coaching.
0	http://data.europa.eu/esco/skill/0005c151-5b5a-4a66-8aac-60e734beb1ab	skill/competence	sector-specific	manage musical staff	direct musical staff	Assign and manage staff tasks in areas such as scoring, arranging, copying music and vocal coaching.

Figure 48: A screenshot of the simplified esco_skills.csv dataset

The ESCO dataset has been cleansed of NaN values thus resulting in 97,483 rows. We have conducted a mini explorative analysis to gain a better understanding of it:

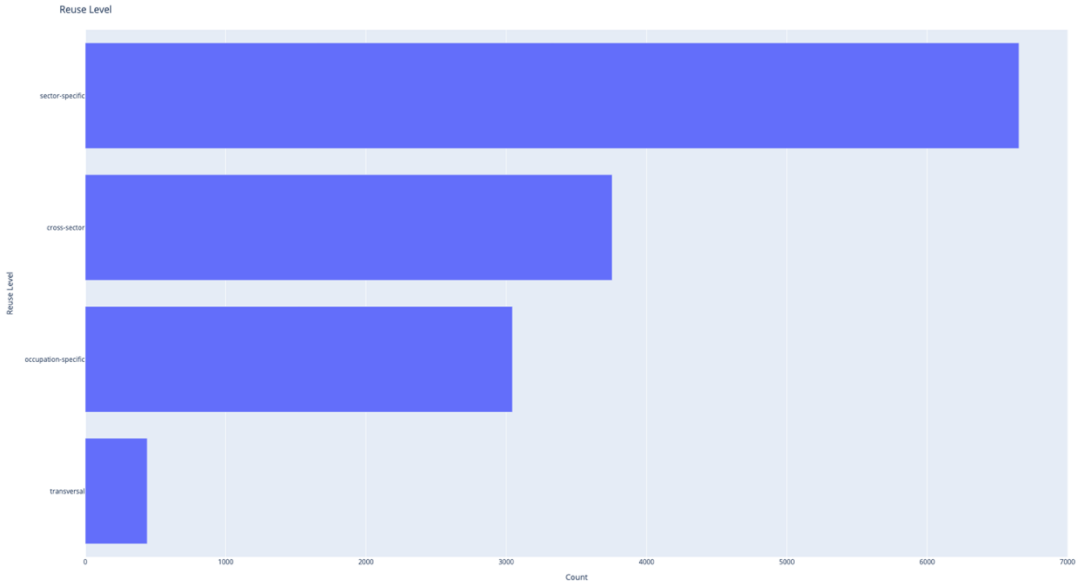

Figure 49: Reuse level vs count.

The bar chart above shows the distribution of skills across different reuse levels within the ESCO (European Skills, Competences, Qualifications, and Occupations) framework.

- Sector-Specific: The sector-specific category has the highest count, approximately 6.600. This suggests that the majority of skills in the ESCO framework are tailored to specific industries, highlighting a strong focus on industry-specific knowledge and competencies. This may reflect the specialized nature of many occupations.

- Cross-Sector: The second highest, with a count around 3800. A significant number of skills are applicable across multiple sectors. This suggests that many competencies are versatile and valuable in various contexts, highlighting the importance of transferable skills in the workforce.

- Occupation-Specific: The third highest, with a count just above 3.000. These skills are unique to specific occupations within sectors, indicating a need for detailed and specialized knowledge for particular job roles. This highlights the granularity of skills required within specific job titles.

- Transversal: The lowest count, approximately 300. Transversal skills, although less numerous, are critical as they apply broadly across all jobs and sectors. These skills include essential soft skills like communication, problem-solving, and teamwork, which are universally valued.

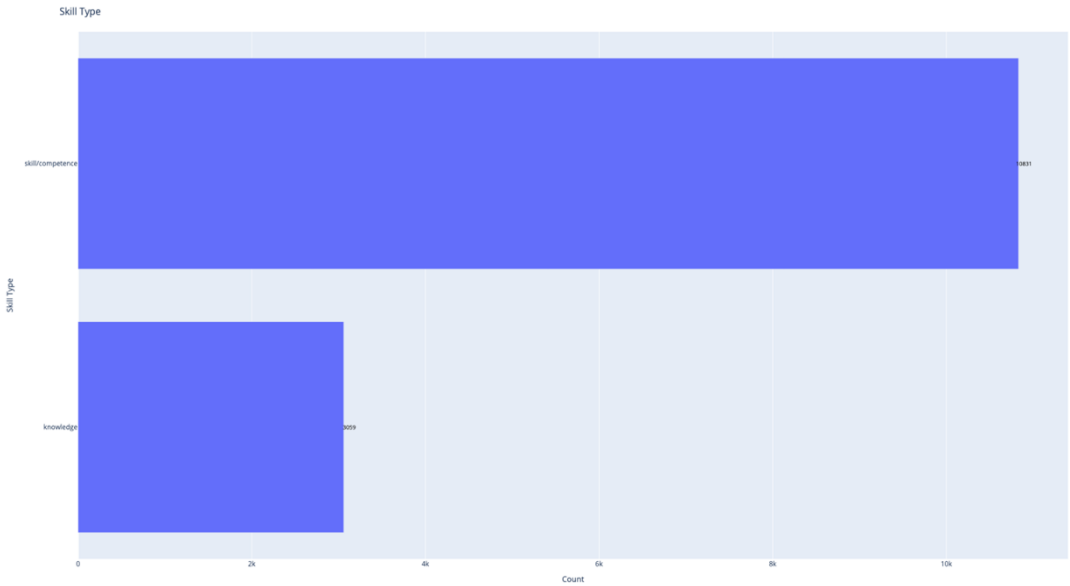

Figure 50: Skill type vs count.

The bar chart above shows the distribution of different skill types within the ESCO framework.

- Skill/Competence: The highest count, exceeding 10000. A large majority of entries in the ESCO database are categorized as skills or competences. This indicates a comprehensive focus on practical abilities and proficiencies that individuals can demonstrate in the workplace. The high count emphasizes the importance of specific skill sets in job descriptions and employment criteria.

- Knowledge: Lower than skills/competences, with a count slightly above 4000. While still substantial, the number of knowledge entries is significantly less than that of skills/competences. This suggests that while theoretical understanding and factual knowledge are important, the ESCO framework places a greater emphasis on actionable skills and competencies that can be directly applied in job tasks.

The ESCO data highlights a strong focus on sector-specific and skill/competence categories, indicating that the framework aims to address detailed and practical aspects of job requirements. The significant presence of cross-sector and transversal skills underscores the need for flexibility and adaptability in the workforce, reflecting the dynamic nature of modern job markets where employees often switch sectors or require a broad set of core skills. The balance between skill/competence and knowledge suggests a holistic approach to workforce development, valuing both practical abilities and theoretical understanding.

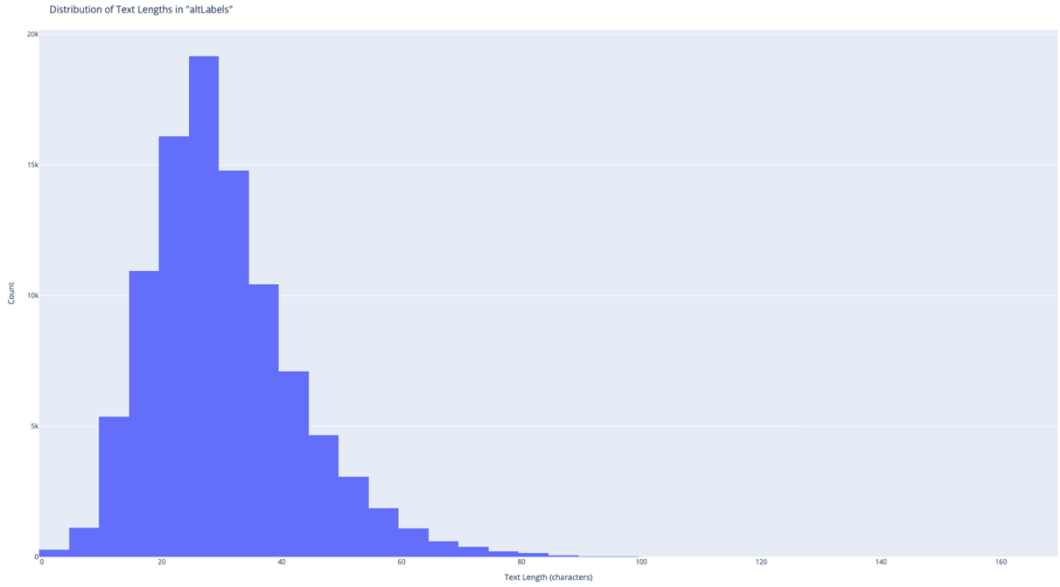

Figure 51: Distribution of Text Lengths in "altLabel".

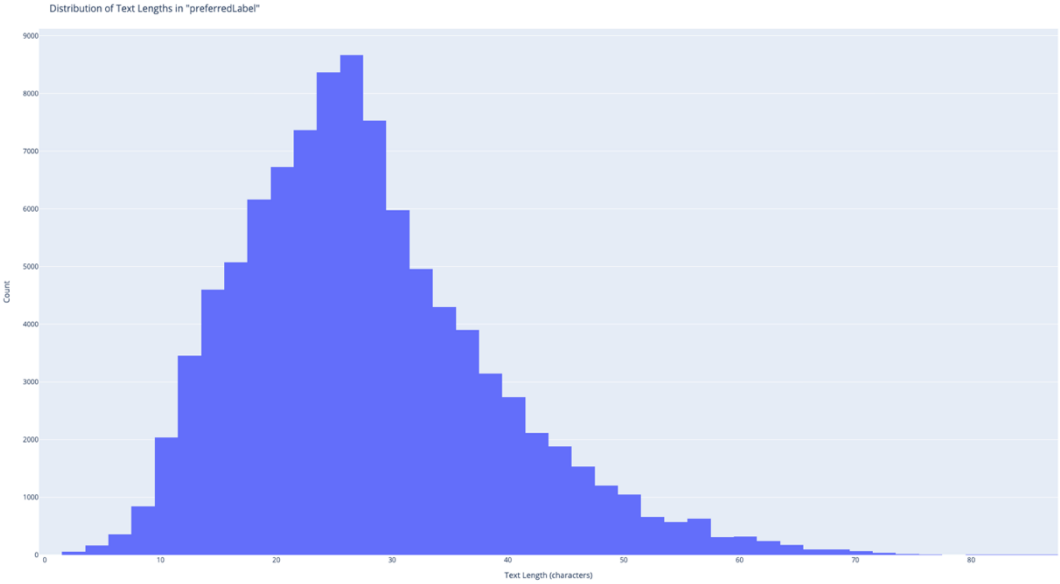

Figure 52: Distribution of Text Lengths in "preferredLabel".

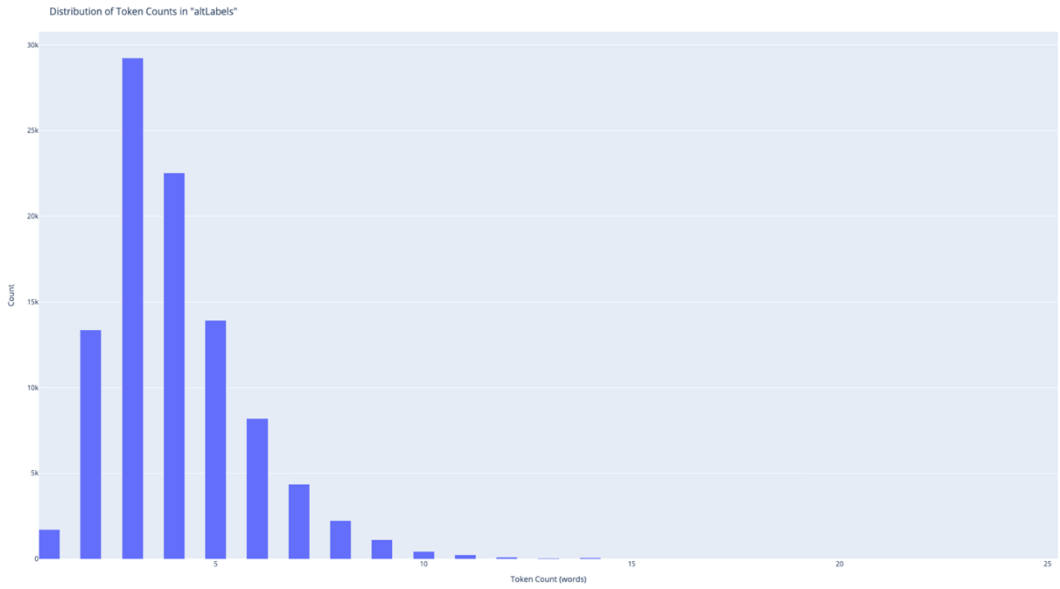

Figure 53: Distribution of Token Counts in "altLabels".

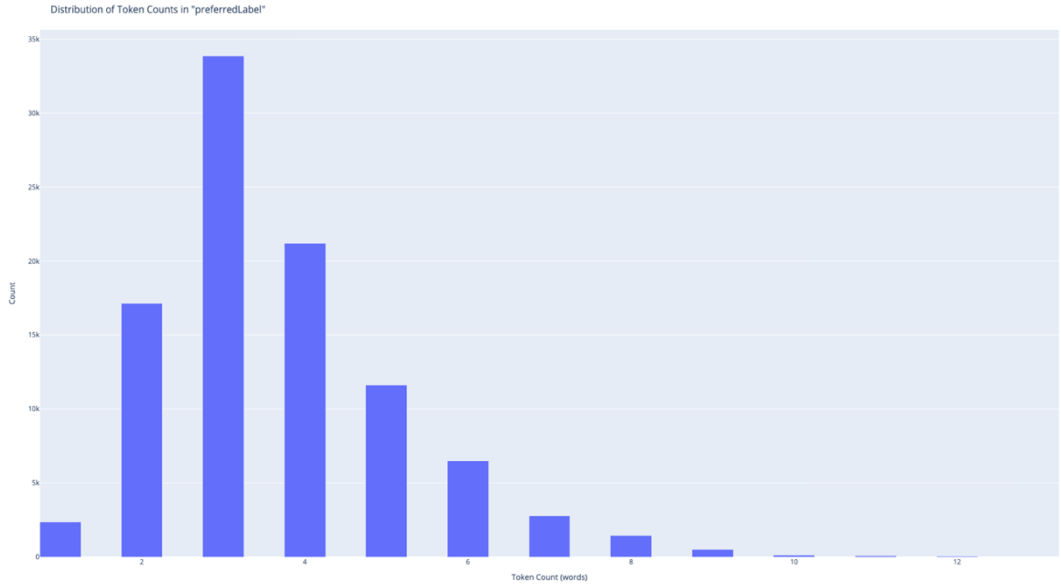

Figure 54: Distribution of Token Counts in "preferredLabel".

For the semantic similarity experiments we have further eliminated all columns of the ESCO dataset except the 'preferredLabel' and 'altLabels' which have provided the sentence pairs necessary for the finetuning of the transformer models. In addition to this, there exist four annotation benchmark files available from ESCO that we have used in order to evaluate the semantic similarity performance:

- house_test_annotations.csv
- house_validation_annotations.csv
- tech_test_annotations.csv
- tech_validations.csv

These four files have also been checked for inconsistencies (‘UNDERSPECIFIED’ and “LABEL NOT PRESENT’ values) and have been concatenated into a single dataset.

We trained three transformer models from Hugging Face: all_mpnet_base_v2, sentence_t5_base, and msmarco_distilbert_cos_v5, to learn the pairs of texts from the columns "span" and "label" of the df_map dataset. Utilizing MultipleNegativesRankingLoss, we conducted our experiments with a batch size of 16, 100 warmup steps, and for 2 epochs.

During finetuning, the transformer models generated embeddings for both the "span" and "label" columns, allowing us to compute the cosine similarity between these embeddings. For visualization, we used the Plotly library with an adjustable threshold of cosine similarity, below which no sentences are shown. This enabled us to filter out less relevant matches and focus on the most pertinent ones.

We calculated top-1, top-3, and top-5 accuracies for all the transformer models, providing a comprehensive evaluation of their performance. Additionally, we identified the most similar sentences to the extracted entities derived from the Named Entity Recognition (NER) subsystem, facilitating a better understanding of the models' effectiveness in real-world applications. These results were crucial for determining the models' capability to match spans to their corresponding labels accurately. Below we can see the results of all models for 5 different job listings. The complete list of inference screenshots can be seen in the Appendix B.

```
results1
```

	Extracted Entities	Most Similar Sentences (mpnet)	Most Similar Sentences (t5)	Most Similar Sentences (marco)	Cosine Similarity (mpnet)	Cosine Similarity (t5)	Cosine Similarity (marco)
0	skilled	craftsmanship	coordination	craft skills	0.716834	0.593310	0.582218
1	experienced	direct movement experience	hire experienced people	recognise and understand the experiences of others	0.557549	0.652838	0.601700
2	provide strategic guidance and solutions	communicate strategic plan	implement strategic management	implement strategic management	0.649818	0.665933	0.715925
3	analytical	analytical chemistry	analytical thinking	think analytically	0.753234	0.844166	0.686099
4	critical thinking abilities	apply critical thinking	apply critical thinking	apply critical thinking	0.727962	0.791307	0.676102
5	communication skills	teach communication skills	teach communication skills	teach communication skills	0.786590	0.852458	0.704675
6	written and verbal, and a	writing methods	verbalise	grammar	0.565642	0.588193	0.610648
7	business analysis tools and software	business intelligence	implement business analysis	software for industry	0.732711	0.789219	0.687279
8	project management	project management	project management	project management	0.950731	0.953679	0.906273
9	lead and execute complex projects	manage projects	perform management of projects	oversee several projects	0.720903	0.690860	0.690312
10	maintaining strict timelines and budgets	manage budgets	ensure budget is up to date	ensure budget is up to date	0.593981	0.643997	0.752734
11	developing and implementing business strategies	develop business plans	develop company strategies	develop business plans	0.728381	0.739135	0.720163
12	data analysis	data analysis	data analysis	data analysis	0.951516	0.962581	0.901822
13	financial modeling	financial calculating	financial forecasting	financial analysis	0.749000	0.721195	0.771972
14	highly regarded	get noticed by people	high reliability theory	quality standards	0.445114	0.517602	0.453635
15	interpersonal skills	personal skills developing	developing personal skills	teamwork methods	0.524053	0.667539	0.548900
16	collaborating with diverse teams and stakeholders	collaborate with different groups	collaborate with stakeholders	collaborate with different groups	0.734852	0.724387	0.711820
17	achieve business objectives	define and validate business needs and goals	imprint visionary goals into the business management	define and validate business needs and goals	0.720063	0.684987	0.716863

Figure 55: Pivot table showing the extracted entities per model and the respective cosine similarity for Job listing 1

```
results2
```

	Extracted Entities	Most Similar Sentences (mpnet)	Most Similar Sentences (t5)	Most Similar Sentences (marco)	Cosine Similarity (mpnet)	Cosine Similarity (t5)	Cosine Similarity (marco)
0	building an \nadvanced NLP-driven chatbot	generate voice interpreting	utilize machine learning	create cloud network	0.326330	0.468596	0.505428
1	leading the development of a sophisticated NLP model	NLP	NLP	design a chp system	0.415940	0.489520	0.451255
2	enhance customer interactions	improve customer interaction	improve customer interaction	improve customer interaction	0.907679	0.844536	0.851484
3	integrating this cutting-edge chatbot into our digital platforms	using internet chat	interact through digital technologies	utilise online communication tools	0.532675	0.483788	0.530198
4	machine learning	utilize machine learning	utilize machine learning	utilize machine learning	0.642539	0.906843	0.676261
5	detail-oriented	Concept-Oriented	attend to detail	attend to detail	0.555968	0.686800	0.551154
6	analytical mindset	analytical thinking	analytical thinking	analytical thinking	0.810641	0.885912	0.759778
7	thriving in problem-solving and complex challenges	cope with demanding challenges	develop solutions to information needs and challenges	solve problems	0.676609	0.536622	0.641807
8	team player	organise a team	work in team	manage the team	0.661912	0.696915	0.600254
9	communicate effectively	communicating effectively and efficiently	communicate efficiently and effectively	communicate efficiently and effectively	0.859713	0.881609	0.842089
10	competitive compensation	fix remuneration	gather compensation payments	ensure competitive prices	0.581253	0.529074	0.577794
11	passionate	delight	delight	inspire others	0.568939	0.656790	0.590432
12	revolutionize e-commerce customer experiences	design a customer experience	design customer experiences	contribute to a customer experience	0.598799	0.551453	0.550363
13	transforming the future \nof e-commerce through technology	e-commerce system	e-commerce systems	combine business technology with user experience	0.592580	0.609911	0.544888

Figure 56: Pivot table showing the extracted entities per model and the respective cosine similarity for Job listing 2

```
results3
```

	Extracted Entities	Most Similar Sentences (mpnet)	Most Similar Sentences (t5)	Most Similar Sentences (marco)	Cosine Similarity (mpnet)	Cosine Similarity (t5)	Cosine Similarity (marco)
0	developing groundbreaking infrastructure \nprojects	foster design of innovative infrastructure	encourage the design of innovative infrastructure in engineering projects	foster innovative infrastructure design	0.659980	0.686577	0.559324
1	planning	perform planning	perform planning	complete planning	0.831963	0.848387	0.799680
2	designing	design	design	design	0.923624	0.917080	0.744952
3	overseeing the construction of our next-generation infrastructure \nsolutions	oversee infrastructure	oversee infrastructure	oversee infrastructure	0.628227	0.590065	0.635371
4	conducting site evaluations	conduct environmental site assessments	conducting environmental site assessments	conduct environmental site assessments	0.821505	0.815775	0.752721
5	developing detailed project plans	prepare project proposals	assess project plans	develop design plans	0.761886	0.746791	0.746161
6	ensuring compliance \nwith industry standards and regulations	follow industrial safety standards	ensure compliance with regulations	adhere to industry best practice in aviation safety requirements	0.741070	0.742493	0.596360
7	using design and analysis software	using specialised design software	design tools for job analysis	use specialized design software	0.760675	0.751793	0.693968
8	AutoCAD	AutoCAD drawing	making AutoCAD drawings	AutoCAD drawing	0.843274	0.881435	0.639588
9	embody precision	attend to precision	attend to precision	attend to precision	0.642590	0.746752	0.751834
10	eye for detail	attending to details	attend to detail	attend to detail	0.655815	0.681684	0.568653
11	commitment to	have commitment	have commitment	demonstrate commitment	0.778339	0.852971	0.851577
12	teamwork	team-working	team-working	teamwork methods	0.924107	0.924227	0.838693
13	communication skills	teach communication skills	teach communication skills	teach communication skills	0.786590	0.852458	0.704675
14	work closely with architects, contractors, and project managers	liaise with architects	coordinate with architects	coordinate with architects	0.690638	0.607155	0.719355
15	professional development	undertake professional development	undertake professional development	undertake professional development	0.835883	0.933298	0.815853
16	competitive compensation	fix remuneration	gather compensation payments	ensure competitive prices	0.581253	0.529074	0.577794
17	passionate	delight	delight	inspire others	0.568939	0.656790	0.590432
18	shaping the future through innovative civil	foster design of innovative infrastructure	encourage the design of innovative infrastructure in engineering projects	shape the future	0.569290	0.642220	0.595151
19	engineering	engineering	engineering	engineering	0.949169	0.982766	0.893169
20	thrive in a collaborative,	work collaboratively	work collaboratively	work collaboratively	0.732793	0.697103	0.777531
21	dynamic setting	interaction design	dynamic positioning systems	interaction design	0.459310	0.552711	0.444515
22	building the infrastructure of tomorrow!	manage infrastructure	setting up of temporary construction site infrastructure	oversee infrastructure	0.594314	0.571290	0.571026

Figure 57: Pivot table showing the extracted entities per model and the respective cosine similarity for Job listing 3

```
results4
```

	Extracted Entities	Most Similar Sentences (mpnet)	Most Similar Sentences (t5)	Most Similar Sentences (marco)	Cosine Similarity (mpnet)	Cosine Similarity (t5)	Cosine Similarity (marco)
0	lead and oversee the operations of our vibrant banking branch	monitor banking activities	oversee corporate bank accounts	directing financial operations	0.550232	0.550939	0.575570
1	driving the branch's success	branch operations	branch operations	branch operations	0.560076	0.465205	0.504536
2	fostering robust financial services	promote financial services	offer financial services	promote financial services	0.687853	0.605865	0.598685
3	ensuring customer satisfaction	ensure customer satisfaction	ensure customer satisfaction	ensure customer satisfaction	0.947798	0.896818	0.876551
4	managing branch staff	manage staff	manage staff	supervising staff	0.657162	0.674765	0.652746
5	developing effective business strategies	develop company's strategies	developing company strategies	develop company's strategies	0.663773	0.732510	0.723549
6	enhancing the overall customer banking experience	deliver better customer experiences	deliver better customer experiences	offer banking services	0.620724	0.609596	0.473333
7	financial management	financial management	financial management	financial management	0.934691	0.949362	0.915771
8	financial software	decision support software	financial data	financial product	0.675927	0.702407	0.675669
9	risk management	risk management	risk management	risk management	0.942863	0.956441	0.905484
10	strategic mindset	strategic thinking	strategic thinking	strategic plan	0.802643	0.885422	0.809195
11	meticulous	work meticulously	work meticulously	work meticulously	0.612878	0.719311	0.779199
12	attention to detail	attend to detail	attend to detail	attend to detail	0.756774	0.794509	0.805627
13	commitment to ethical banking practices	abiding by a business ethical code of conducts	abide by business ethical code of conducts	abide by a business ethical code of conduct	0.586206	0.581077	0.546643
14	Effective leadership and	principles of leadership	demonstrate an exemplary leadership qualities in an organisation	leadership principles	0.784050	0.713153	0.741225
15	team management	managing a team	manage team	managing a team	0.870945	0.906895	0.829183
16	guide and mentor your team	provide mentorship	supervise a team	provide mentorship	0.672687	0.689442	0.672541
17	communication skills	teach communication skills	teach communication skills	teach communication skills	0.786590	0.852458	0.704675
18	interacting with a diverse range of clients, staff, and upper management	consult with clients of business	communicate with clients and their carers	consult with clients of business	0.616304	0.618200	0.567875
19	competitive remuneration	calculate remuneration	calculate remuneration	ensure competitive prices	0.564869	0.644766	0.562556
20	driving positive change	promote social change	bolster positive behaviour	be a driving force	0.730335	0.689864	0.505103
21	financial solutions	financial management	financial planning	financial capabilities	0.706549	0.658077	0.729250
22	excel in a fast-paced	cope with fast-paced situations	cope with fast-paced situations	typing at speed	0.651181	0.664909	0.556673
23	collaborative environment	use online tools to collaborate	work collaboratively	work collaboratively	0.767285	0.782828	0.643438

Figure 58: Pivot table showing the extracted entities per model and the respective cosine similarity for Job listing 4

```
results5
```

	Extracted Entities	Most Similar Sentences (mpnet)	Most Similar Sentences (t5)	Most Similar Sentences (marco)	Cosine Similarity (mpnet)	Cosine Similarity (t5)	Cosine Similarity (marco)
0	motivated	show motivation	show motivation	demonstrate motivation	0.769624	0.822551	0.757394
1	dynamic	motion capture	dynamic positioning systems	kinetics	0.520123	0.634136	0.556814
2	be a key player	practise role	take part in games to ensure player distribution	provide keys	0.566583	0.527487	0.491395
3	driving our vehicle sales	convince customers to purchase vehicles	proactively sell vehicles to customers	vehicle selling	0.654295	0.659508	0.646331
4	providing exceptional customer service	provide outstanding customer service	provide excellence in customer service	ensure customers receive an exceptional service	0.903396	0.905004	0.766059
5	building lasting relationships	building effective relationships	building effective relationships	building effective relationships	0.855965	0.801411	0.778072
6	showcasing our range of cars	manage the presentation of vehicles in dealership	inspect the performance of motor vehicle models	brands of vehicles	0.578999	0.490629	0.562254
7	understanding customer needs	identify customer's needs	identify customer's needs	determine customer's needs	0.894115	0.851235	0.854945
8	providing outstanding customer service	provide outstanding customer service	supply outstanding customer service	provide outstanding customer service	0.939130	0.940275	0.848952
9	sales and CRM software	CRM	CRM	software product sales	0.735697	0.763958	0.645507
10	communication \nand negotiation skills	perform negotiation abilities	perform negotiation abilities	perform negotiation abilities	0.695293	0.765438	0.654981
11	connect with a diverse clientele	build a rapport with people from different cultural backgrounds	engage with customers	interact with clients	0.623678	0.630777	0.691790
12	customer-focused	ensure customer focus	ensure customer focus	be customer oriented	0.785695	0.815370	0.717947
13	honesty	be honest	be honest	be honest	0.728564	0.806340	0.721473
14	integrity	integrity	integrity	integrity	0.932154	0.986378	0.855504
15	work closely with other sales representatives	direct sales teams	oversee sales teams	collaborate with advertising professionals	0.609957	0.597790	0.612349
16	ensure a \nseamless and satisfying customer experience	assure customer satisfaction	guarantee customer satisfaction	ensure customer satisfaction	0.805439	0.784709	0.614456
17	passion for cars	designing automobiles	car handling	vehicle trends	0.522359	0.595485	0.642564
18	thrive in \na fast-paced, customer-oriented environment	cope with fast-paced situations	be customer oriented	be customer oriented	0.618148	0.595151	0.668287

Figure 59: Pivot table showing the extracted entities per model and the respective cosine similarity for Job listing 4

The tables presented show the results of the cosine similarity calculations between the extracted entities and their most similar sentences generated by three transformer models (all_mpnet_base_v2, sentence_t5_base, and msmarco_distilbert_cos_v5) for all five job listings. These results can be interpreted thus:

- Extracted Entities: These are the key phrases or terms identified by the NER subsystem, representing specific skills, attributes, or objectives.
- Most Similar Sentences (mpnet): These are the sentences identified by the all_mpnet_base_v2 model that are most similar to the extracted entities. For example, for the entity "skilled," the most similar sentence is "craftsmanship."
- Most Similar Sentences (t5): These sentences are identified by the sentence_t5_base model. For the entity "skilled," the corresponding sentence is "coordination."
- Most Similar Sentences (marco): These are the sentences identified by the msmarco_distilbert_cos_v5 model. For the entity "skilled," the most similar sentence is "craft skills."
    - Cosine Similarity Scores: These scores indicate the similarity between the embeddings of the extracted entity and the most similar sentence for each model:
    - Cosine Similarity (mpnet): Indicates varying degrees of similarity, with higher scores representing closer matches.
    - Cosine Similarity (t5): Reflects the closeness of the match between the entity and the sentence.
    - Cosine Similarity (marco): Shows the degree of alignment between the entity and the sentence.

The model all_mpnet_base_v2 often produces high similarity scores, indicating strong matches, particularly in cases like "project management" and "data analysis". The model sentence_t5_base also performs well, with some of the highest similarity scores observed in sentences like "apply critical thinking" and "teach communication skills". The model msmarco_distilbert_cos_v5 shows a slightly lower range of similarity scores but still identifies relevant matches, particularly for "maintaining strict timelines and budgets" and "lead and execute complex projects."

All three models demonstrate the ability to find sentences similar to the extracted entities, with varying degrees of success. The all_mpnet_base_v2 model generally provides the highest cosine similarity scores, suggesting it may be the most effective at generating embeddings that closely match the entities. The sentence_t5_base model also shows strong performance, while msmarco_distilbert_cos_v5 offers reasonable matches but with slightly lower similarity scores. These results can help in selecting the most appropriate model for specific tasks involving text similarity and entity matching.

5.6 3D Visualization of embeddings

Principal Component Analysis (PCA) is a statistical method that utilizes orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This method is widely used in exploratory data analysis and for dimensionality reduction. To visualize the embeddings generated by our models, we employed PCA and reduced the dimensions of their embeddings from 768 to 3. To keep the visualizations relatively sparse and easier to examine, we applied an adjustable threshold to filter the embeddings based on their cosine similarity, which is set to 0.40 for the visualizations presented below. Data points represented with a deep color indicate higher cosine similarity values whereas the ones with a pale color indicate lower values. As an example, for the input sentence 'supervise Ph.D. students' which is represented with a red 'X', the 'all_mpnet_base_v2' model achieved a maximum cosine similarity of 0.9209 and the respective ESCO label is "supervise doctoral students":

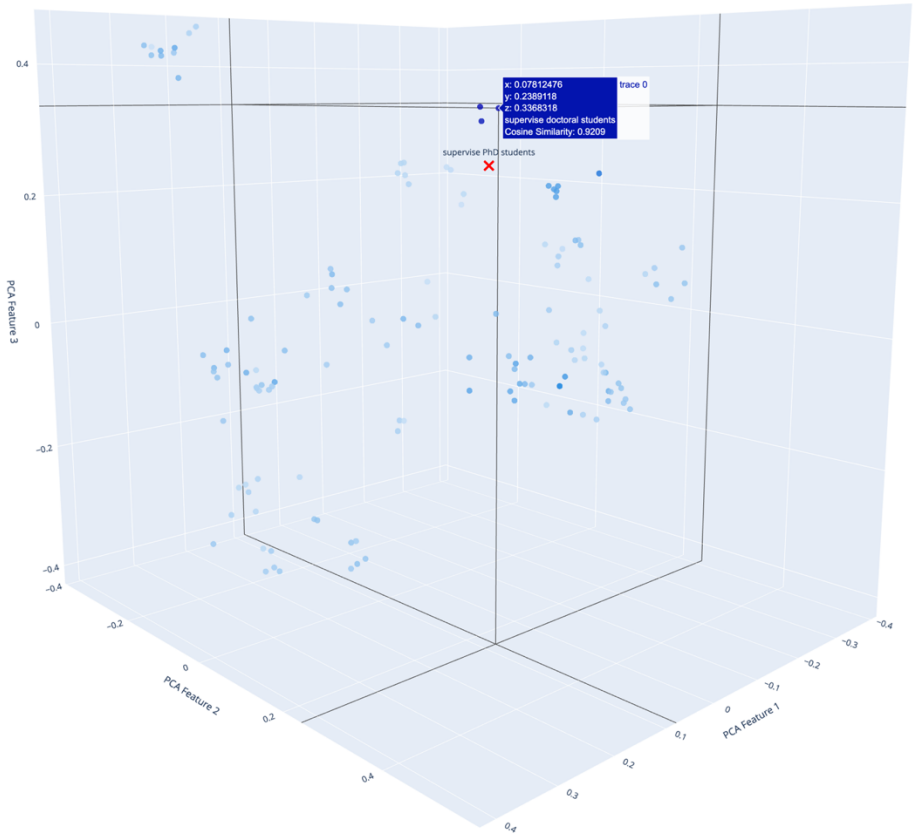

Figure 60: 3D Visualization of the filtered embeddings using PCA for the model 'all_mpnet_base_v2'.

- The sentences that are most similar to "supervise PhD students" are scattered around the red X.
- There is a noticeable cluster of points near the red X, indicating sentences that are semantically similar.
- The tooltip indicates a cosine similarity of around 0.9209 for one of the closest points.

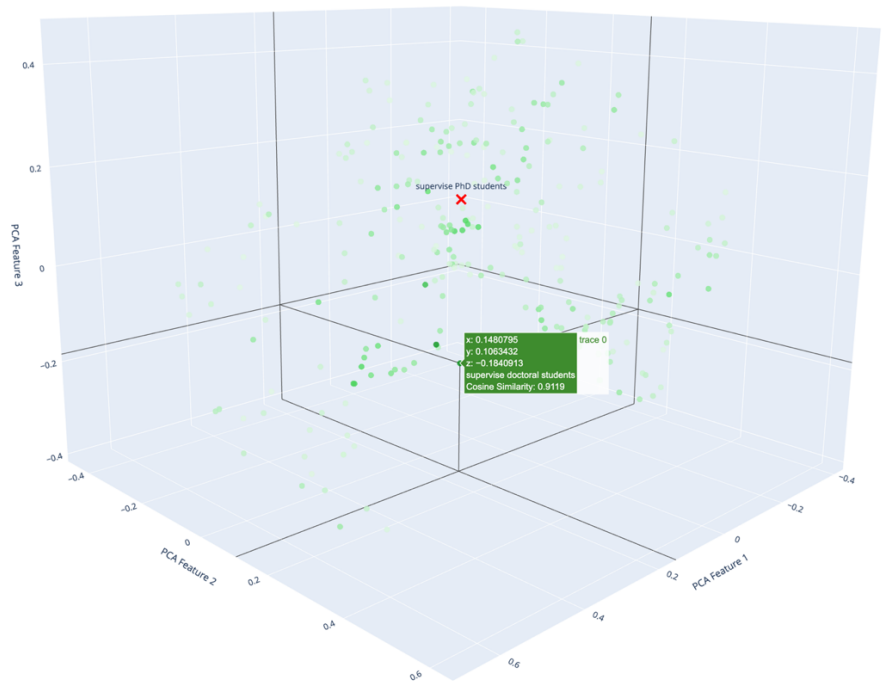

Figure 61: 3D Visualization of the filtered embeddings using PCA for the model 'sentence_t5_base'.

- Similar to the plot for the all_mpnet_base_v2 model, there are points scattered around the red X, representing sentences with high similarity.
- The clustering is slightly more spread out compared to the blue plot.
- The tooltip shows a cosine similarity of around 0.9119 for one of the closest points.

If we examine closely figures 60 and 61, we will notice that in both plots there exists dots representing sentence with a high cosine similarity which are not clustered close to the red X. The reason some relatively close (semantically speaking) points are not clustered around the red X but appear further away can be attributed to the dimensionality reduction process (PCA) and the inherent characteristics of the models. Different models capture semantic similarity in different ways. The all_mpnet_base_v2 and sentence_t5_base models have different training objectives and architectures, which can result in different embedding spaces. When reducing these 768 dimensions to just 3, PCA captures the directions (principal components) that explain the most variance. However, this does not guarantee that the relative positions of all points are preserved. Therefore, some sentences that were close in the original space might not be as close in the reduced space. The transformation from high-dimensional to low-dimensional space can distort distances and angles, meaning that linear relationships are not perfectly retained. This is why semantically similar sentences might appear further apart in the 3D plot. Additionally, PCA aims to maximize the variance captured in the first few principal components. However, some important aspects of the data that contribute to semantic similarity might be captured in components beyond the first three, leading to their underrepresentation in the 3D plot. Given that PCA reduces dimensionality by discarding components that contribute less to variance, some subtle yet significant semantic information might be lost. This causes some points to appear more dispersed or incorrectly positioned relative to the input sentence (red X). The visualizations were generated in Python with the Plotly library and saved directly to disk as HTML files.

5.7 STS Benchmark

STS Benchmark comprises a selection of the English datasets used in the STS tasks organized in the context of SemEval between 2012 and 2017 [58] and is available at: http://ixa2.si.ehu.eus/stswiki. The benchmark comprises 8628 sentence pairs. This is the breakdown according to genres and train-dev-test splits:

Table 6: Categories and Train/Dev/Test splits of the STS Benchmark

Category	Train	Dev	Test	Total
News	3299	500	500	4299
Caption	2000	625	525	3250
Forum	450	375	254	1079
Total	5749	1500	1379	8628

For reference, this is the breakdown according to the original names and task years of the datasets:

Table 7: Categories and task years of the STS Benchmark individual datasets

Genre	File	Years	Train	Dev	Test
News	MSRpar	2012	1000	250	250
News	headlines	2013-16	1999	250	250
News	deft-news	2014	300	0	0
Captions	MSRvid	2012	1000	250	250
Captions	images	2014-15	1000	250	250
Captions	track5.en-en	2017	0	125	125
Forum	deft-forum	2014	450	0	0
Forum	answers-forums	2015	0	375	0
Forum	answer-answer	2016	0	0	254

Given two sentences of text, sentence1 and sentence2, the systems need to compute how similar s1 and s2 are, returning a similarity score between 0 and 5. The dataset comprises naturally occurring pairs of sentences drawn from several domains and genres, annotated by crowdsourcing [59]. Below we can see a screenshot of the STS benchmark dataset:

	score	sentence1	sentence2
0	5.00	A plane is taking off.	An air plane is taking off.
1	3.80	A man is playing a large flute.	A man is playing a flute.
2	3.80	A man is spreading shreded cheese on a pizza.	A man is spreading shredded cheese on an uncooked pizza.
3	2.60	Three men are playing chess.	Two men are playing chess.
4	4.25	A man is playing the cello.	A man seated is playing the cello.
...	...	...	...
5695	0.00	Severe Gales As Storm Clodagh Hits Britain	Merkel pledges NATO solidarity with Latvia
5696	0.00	Dozens of Egyptians hostages taken by Libyan terrorists as revenge for airstrikes	Egyptian boat crash death toll rises as more bodies found in Nile
5697	0.00	President heading to Bahrain	President Xi: China to continue help to fight Ebola
5698	0.00	China, India vow to further bilateral ties	China Scrambles to Reassure Jittery Stock Traders
5699	0.00	Putin spokesman: Doping charges appear unfounded	The Latest on Severe Weather: 1 Dead in Texas After Tornado

5700 rows × 3 columns

Figure 62: A screenshot of the STS Benchmark dataset

Pearson rank correlation, more accurately referred to as Pearson correlation coefficient, is a statistical measure that assesses the strength and direction of the linear relationship between two continuous variables. Here's an elaboration on this concept:

- r = 1 indicates a perfect positive linear relationship.
- r = −1 indicates a perfect negative linear relationship.
- r = 0 indicates no linear relationship.

The formula for calculating the Pearson correlation coefficient is:

$$r = \frac{\sum(x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

and can be interpreted thus:

- Positive Correlation: If r is positive, it means that as one variable increases, the other variable also tends to increase. For instance, height and weight typically exhibit a positive correlation.
- Negative Correlation: If r is negative, it means that as one variable increases, the other variable tends to decrease. An example might be the number of hours spent watching TV and academic performance.
- Magnitude: The closer the value of r is to 1 or -1, the stronger the linear relationship between the variables. Values close to 0 imply a weak linear relationship.

Pearson correlation is commonly used to measure the linear relationship between two variables, but when it comes to semantic similarity, a specialized adaptation of correlation, often leveraging vector representations of text, is typically applied. Semantic similarity measures the degree to which two pieces

of text carry similar meaning, and Pearson correlation can play a role in this context, especially in evaluating the performance of different similarity measures. Pearson correlation can be used to assess the effectiveness of semantic similarity measures by comparing human-annotated similarity scores with computational similarity scores. This is how it works:

- Human Annotation: Obtain a dataset where pairs of texts are annotated with similarity scores by human judges. This dataset serves as a ground truth for semantic similarity.
- Computational Scores: Calculate similarity scores for the same pairs of texts using a chosen semantic similarity measure (e.g., cosine similarity of sentence embeddings).
- Pearson Correlation: Compute the Pearson correlation coefficient between the human-annotated scores and the computational scores. This coefficient indicates how well the computational measure aligns with human judgments.

The STS Benchmark, which we used to fine-tune our models for a second time, is a general-purpose dataset of sentence pairs with human-annotated similarity scores. The only necessary processing was to normalize the STS Benchmark ratings to the range [0…1], instead of the original range [1…5]. This normalization allowed us to calculate the cosine similarity of the sentence pair embeddings directly and obtain the similarity scores for these pairs. Following this, we have used the Embedding Similarity Evaluator module from the sentence_transformers library in order to calculate the Pearson correlation coefficient between the human-annotated scores and the cosine similarity scores, obtaining the following results:

Table 8: Spearman Rank Correlation for the sentence similarity experiments

Model	all_mpnet_base_v2	sentence_t5_base	msmarco_distilbert_cos_v5
Pearson Rank Correlation	87,40%	87,79%	85,05%

indicating a strong alignment and suggesting that embeddings are indeed effective for measuring semantic similarity. As we have mentioned the Pearson correlation coefficient ranges from – 1 to 1. A high positive Pearson correlation coefficient (close to 1) indicates that the computational similarity scores are strongly aligned with human judgments, suggesting the measure is effective at capturing semantic similarity whereas a low or negative correlation coefficient suggests that the computational measure does not align well with human judgments, indicating it may not be capturing semantic similarity accurately. The Pearson correlation coefficient is a valuable metric for evaluating the semantic similarity of various models, ensuring that computational methods align closely with human judgments and improving the accuracy and reliability of various NLP applications. Below we can see a summary of various metrics:

Table 9: Summary of various metrics for all the transformer models.

Model	all_mpnet_base_v2	sentence_t5_base	msmarco_distilbert_cos_v5
Top 1 accuracy	49,25%	50,62%	48,07%
Top 3 accuracy	62,37%	67,02%	59,70%
Top 5 accuracy	69,36%	72,89%	65,44%
Average Cosine Similarity (job listing 1)	70,04%	72,68%	69,10%
Average Cosine Similarity (job listing 2)	62,36%	65,91%	61,95%
Average Cosine Similarity (job listing 3)	72,78%	74,83%	68,14%
Average Cosine Similarity (job listing 4)	71,46%	72,27%	68,14%
Average Cosine Similarity (job listing 5)	72,31%	73,54%	68,88%
Pearson Rank Correlation	87,40%	87,79%	85,05%

6 Challenges and Future Directions

6.1 NER

Named Entity Recognition (NER) systems face significant challenges such as data annotation, handling unseen entities, and managing nested entities. The process of annotating data is time-consuming and costly, particularly for specialized domains. Inconsistencies in annotation and the presence of nested entities further complicate the task. In our experiments, we observed the phenomenon of catastrophic inference when the HaHuJobs dataset was added to perform the NER experiments. The HaHuJobs dataset, tagged by four different people without cross-tagging, showed substantial differences in metrics compared to other datasets, as is evident in the diagrams.

We have observed a significant drop of about 8% in precision and accuracy when we added the first 500 samples of the HaHuJobs dataset. The initial precision and accuracy when the NER experiments were conducted solely using the SkillSpan dataset was about 52%. Feeding more annotated data from the HaHuJobs dataset slowly improved the precision and accuracy until at last we achieved the same precision of about 52% using both datasets. This drop from 52% to 44% can be attributed to several factors. Our explorative data analysis revealed that the SkillSpan and HaHuJobs datasets differ significantly in the number of tokens present in each sentence, the length of the extracted entities, and other statistics. Furthermore, the lack of cross-tagging in the HaHuJobs dataset's annotation process contributed to inconsistencies that exacerbated the issue of catastrophic inference. Relying solely on synthetic data improved the NER metrics but resulted in a significant drop in the number of extracted entities and their variety during inferences due to overfitting. This highlights the importance of using real-world data to train models effectively.

In the context of Named Entity Recognition (NER), entity linking, SpaCy's matcher module, and knowledge graphs can significantly improve performance. Entity linking connects named entities recognized in text to a knowledge base, enhancing the context and accuracy of the NER system. SpaCy’s matcher module provides rule-based matching that can be used alongside machine learning models to improve entity recognition by specifying patterns to identify complex entities. Knowledge graphs can be leveraged to provide contextual information and relationships between entities, which helps in disambiguation and improves the system’s understanding of the text. By incorporating these tools and methodologies, NER systems can achieve higher accuracy, better handle nested entities, and provide more comprehensive entity extraction. While automated metrics and advanced models play a crucial

role in semantic similarity tasks, the inclusion of human judgment and ensemble methods can significantly enhance the accuracy and robustness of the results, ensuring better generalization and handling of complex language patterns.

Future research in NER should focus on fine-grained NER, boundary detection and joint NER. Developing scalable neural-based NER models and exploring transfer learning approaches are essential. Creating easy-to-use toolkits for deep learning-based NER can standardize and streamline the process, making it accessible to a broader audience. Considering the findings, it is essential to ensure datasets are annotated consistently to achieve optimal performance in NER and other NLP tasks. The number of data used to train transformer models is crucial for their performance. Insufficient data can lead to poor generalization and reduced accuracy. Conversely, a large and diverse dataset can enhance the model's ability to understand and process complex language patterns. Addressing these challenges in NER involves not only refining model architectures and methodologies but also ensuring high-quality, consistent data annotation practices. This dual approach will help mitigate issues like catastrophic inference and enhance the overall robustness and reliability of NER systems.

6.2 Semantic Similarity

Semantic similarity involves quantifying the extent to which different linguistic units share meaning. Challenges in this area include capturing deep conceptual relationships and ensuring accurate text summarization, machine translation, and sentiment analysis. Future research should aim at improving models to handle these tasks more effectively. Advancements in language modeling and the rise of demand in real-world applications will make semantic similarity a prominent topic of research in the future. It is also crucial to address the challenges of catastrophic inference and the reliance on synthetic data. The observed phenomenon when adding the HaHuJobs dataset indicates that real-world data, despite its inconsistencies, is vital for training robust models. The differences in tagging and metrics between datasets underscore the need for consistent annotation practices. Additionally, while synthetic data can boost certain metrics, it may not capture the complexity and variability of real-world scenarios, leading to reduced entity extraction performance. To enhance model performance, a balance between synthetic and real data should be maintained, and datasets should be annotated consistently. This approach will ensure better generalization and robustness in handling diverse and complex language patterns.

When using semantic similarity with unsupervised training, metrics like cosine similarity are often employed. Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, providing a metric for how similar two text representations are, based on their direction. It ranges from -1 to 1, where 1 indicates identical directions (and hence high similarity), 0

indicates orthogonality (no similarity), and -1 indicates opposite directions. However, interpreting cosine similarity can be challenging due to high dimensionality and the nuanced nature of semantics. The vectors derived from language models, especially those using transformers, exist in high-dimensional spaces, which can lead to unintuitive results where the similarity score may not align with human judgment. For supervised training, metrics such as Pearson rank correlation are used to assess the relationship between predicted similarity scores and human-annotated similarity scores. The Pearson correlation coefficient measures the linear correlation between two variables, providing insight into how well the model's predictions align with human judgments, but its effectiveness can be dataset-specific and may not fully capture the absolute performance of the model.

Human annotators play a critical role in semantic similarity tasks. Despite advances in automatic evaluation metrics, the nuanced understanding of human language often requires human judgment. Human annotators should have the final say on whether sentence A is indeed similar to sentence B and to what extent, particularly in cases where automated metrics provide conflicting or ambiguous results. Ensuring consistency and quality control in the training data through human annotators helps mitigate issues caused by inconsistent annotations. To leverage the strengths of different transformer models, an ensemble approach can be used. Ensembles combine the predictions of multiple models to improve robustness and accuracy. Aggregation methods such as majority voting, averaging of similarity scores, or using a meta-learner can enhance performance. Confidence weighting can also be applied to assign weights to the predictions of different models based on their confidence or performance on a validation set.

7 References

[1] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," Psychological Review, vol. 65, no. 6, pp. 386-408, 1958.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, pp. 533-536, 1986.

[3] R. Rojas, "Neural Networks: A Systematic Introduction," Springer-Verlag, Berlin, Heidelberg, 1996, pp. 149-183.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems 25, 2012.

[6] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," arXiv preprint arXiv:1406.1078, 2014.

[8] G. E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," Neural Computation, vol. 14, no. 8, pp. 1771-1800, 2002.

[9] Hinton, G. E., Osindero, S., and Teh, Y. W., "A Fast Learning Algorithm for Deep Belief Nets," Neural Computation, vol. 18, no. 7, pp. 1527-1554, 2006.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14), Montreal, Canada, 2014, pp. 2672–2680.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504-507, 2006.

[12] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," NIPS, 2014.

[13] K. Cho et al., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," EMNLP, 2014.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,L. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ in Proc. Adv.Neural Inf. Process. Syst., 2017, pp. 5998–6008.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1810.04805.

[16] Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” arXiv, Jul. 26, 2019. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1907.11692.

[17] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” arXiv, Aug. 27, 2019. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1908.10084.

[18] B. Ghojogh and A. Ghodsi, “Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey,” Open Science Framework, preprint, Dec. 2020. doi: 10.31219/osf.io/m6gcn.

[19] J. Li, A. Sun, J. Han, and C. Li, “A Survey on Deep Learning for Named Entity Recognition,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 1, pp. 50–70, Jan. 2022, doi: 10.1109/TKDE.2020.2981314.

[20] N. Kannaiya Raja et al. “NLP: Rule Based Name Entity Recognition,” IJITEE, vol. 8, no. 11, pp. 4285–4290, Sep. 2019, doi: 10.35940/ijitee.K2047.0981119.

[21] B. Yu and Z. Fan, “A comprehensive review of conditional random fields: variants, hybrids and applications,” Artificial Intelligence Rev, vol. 53, no. 6, pp. 4289–4333, Aug. 2020, doi: 10.1007/s10462-019-09793-6.

[22] A. Ekbal and S. Bandyopadhyay, “Named Entity Recognition using Support Vector Machine: A Language Independent Approach,” 2010.

[23] T. Almutiri, and F. Nadeem, “Markov Models Applications in Natural Language Processing: A Survey,” IJITCS, vol. 14, no. 2, pp. 1–16, Apr. 2022, doi: 10.5815/ijitcs.2022.02.01.

[24] Z. Huang, W. Xu, and K. Yu, “Bidirectional LSTM-CRF Models for Sequence Tagging.” arXiv, Aug. 09, 2015. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1508.01991.

[25] J. Li, A. Sun, J. Han, and C. Li, “A Survey on Deep Learning for Named Entity Recognition,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 1, pp. 50–70, Jan. 2022, doi: 10.1109/TKDE.2020.2981314.

[26] A. Rahali and M. A. Akhloufi, “End-to-End Transformer-Based Models in Textual-Based NLP,” AI, vol. 4, no. 1, pp. 54–110, Jan. 2023, doi: 10.3390/ai4010004.

[27] C. Lothritz, K. Allix, L. Veiber, T. F. Bissyandé, and J. Klein, “Evaluating Pretrained Transformer-based Models on the Task of Fine-Grained Named Entity Recognition,” in Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 3750–3760. doi: 10.18653/v1/2020.coling-main.334.

[28] J. R. Firth, "A Synopsis of Linguistic Theory 1930-1955," in Studies in Linguistic Analysis (Special Volume of the Philological Society), Oxford, UK: Blackwell, 1957, pp. 1–32.

[29] Collins, A. M., and Loftus, E. F., "A Spreading-Activation Theory of Semantic Processing," Psychological Review, vol. 82, no. 6, pp. 407–428, 1975.

[30] T. Mikolov et al., "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[31] Rosch, E., & Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. Cognitive Psychology, 104(4), 192-233.

[32] McClelland, J. L., & Rumelhart, D. E. (1986). Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 2: Psychological and biological models. MIT Press.

[33] Sahlgren, M. (2008). The distributional hypothesis. Italian Journal of Linguistics, 20(1), 33-53.

[34] Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. Discourse Processes, 25(2-3), 259-284.

[35] Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instruments, & Computers, 28(2), 203-208.

[36] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532-1543.

[37] Carnap, R. (1947). Meaning and Necessity: A Study in Semantics and Modal Logic. University of Chicago Press.

[38] Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. Communications of the ACM, 38(11), 33-38.

[39] Miller, G. A. (1995). WordNet: A lexical database for English. Communications of the ACM, 38(11), 39-41.

[40] Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199-220.

[41] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2003). The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press.

[42] He, H., Gimpel, K., & Lin, J. (2015). "Multi-perspective Sentence Similarity Modeling with Convolutional Neural Networks." In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP). This study explores the use of CNNs for assessing sentence similarity from multiple perspectives.

[43] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). "Skip-Thought Vectors." In Advances in Neural Information Processing Systems (NeurIPS). This paper presents a method for learning sentence embeddings that can predict the surrounding sentences in a narrative, contributing to the understanding of semantic similarity.

[44] Luong, M. T., Pham, H., & Manning, C. D. (2015). "Bilingual Word Representations with Monolingual Quality in Mind." In Proceedings of the Workshop on Vector Space Modeling for Natural Language Processing. This work explores bilingual word embeddings and their application in semantic similarity across languages.

[45] Wieting, J., Bansal, M., Gimpel, K., & Livescu, K. (2016). "Towards Universal Paraphrastic Sentence Embeddings." In Proceedings of the International Conference on Learning Representations (ICLR).

[46] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data." In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP).

[47] Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Sung, Y. H. (2018). "Universal Sentence Encoder." arXiv preprint arXiv:1803.11175. The Universal Sentence Encoder is introduced for generating sentence embeddings that can be used in a variety of tasks, including semantic similarity.

[48] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MPNet: Masked and Permuted Pre-training for Language Understanding," arXiv preprint arXiv:2004.09297, 2020.

[49] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," in Proceedings of the 37th International Conference on Machine Learning, 2020.

[50] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," arXiv preprint arXiv:1910.10683, 2019.

[51] J.-J. Decorte, J. Van Hautte, T. Demeester, and C. Develder, "JobBERT: Understanding Job Titles Through Skills," arXiv:2109.09605v1 [cs.CL], 20 Sep 2021. Available at: http://arxiv.org/abs/2109.09605.

[52] S. Wang, B. Cao, S. Zhang, X. Li, J. Li, F. Wu, G. Wang, E. Hovy, "Sim-GPT: Text Similarity via GPT Annotated Data,"

[53] European Union. ESCO implementation manual (European Skills, Competences, Qualifications and Occupations), 2018.

[54] SpaCy, "spaCy: Industrial-strength Natural Language Processing in Python," Explosion, 2023. [Online]. Available: https://spacy.io. [Accessed: May 21, 2024].

[55] Label Studio, "Label Studio: Open Source Data Labeling Tool," Heartex, 2023. [Online]. Available: https://labelstud.io. [Accessed: May 21, 2024].

[56] M. Zhang, K. N. Jensen, S. D. Sonniks, and B. Plank, "SkillSpan: Hard and Soft Skill Extraction from English Job Postings," in Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, United States, Jul. 2022, pp. 4962–4984. Available: https://aclanthology.org/2022.naacl-main.366

[57] HaHuJobs, "HaHuJobs Dataset," 2024. [Online]. Available: http://https://www.hahujobs.io. [Accessed: May 21, 2024].

[58] STS Benchmark: Main English dataset, Semantic Textual Similarity 2012-2017 Dataset. [Online]. Available: http://ixa2.si.ehu.eus/stswiki. [Accessed: May 21, 2024]

[59] Eneko Agirre, Daniel Cer, Mona Diab, Iñigo Lopez-Gazpio, Lucia Specia. Semeval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. Proceedings of SemEval 2017.

8 *Appendix A (Explorative Data Analysis visualizations)*

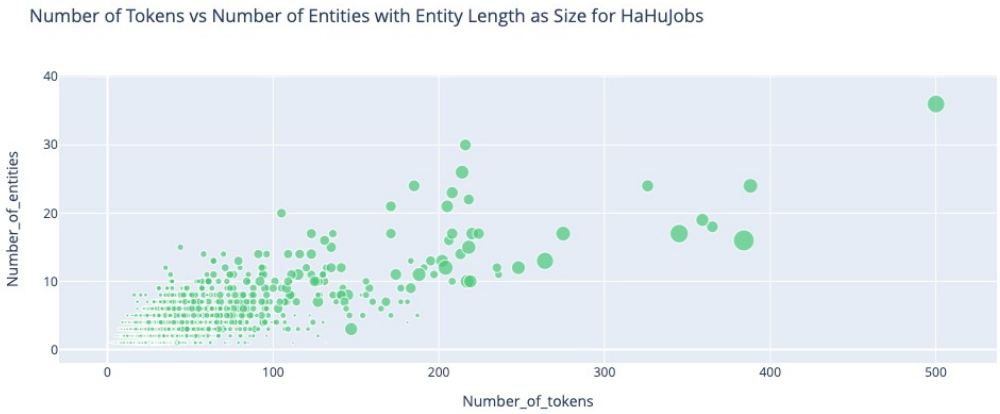

Figure 63: Number of Tokens vs Number of Entities with Entity Length as Size for HaHuJobs.

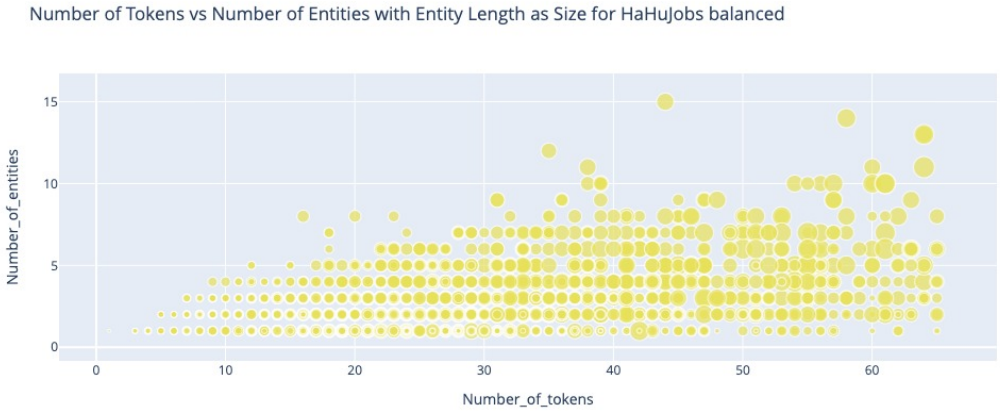

Figure 64: Number of Tokens vs Number of Entities with Entity Length as Size for HaHuJobs balanced.

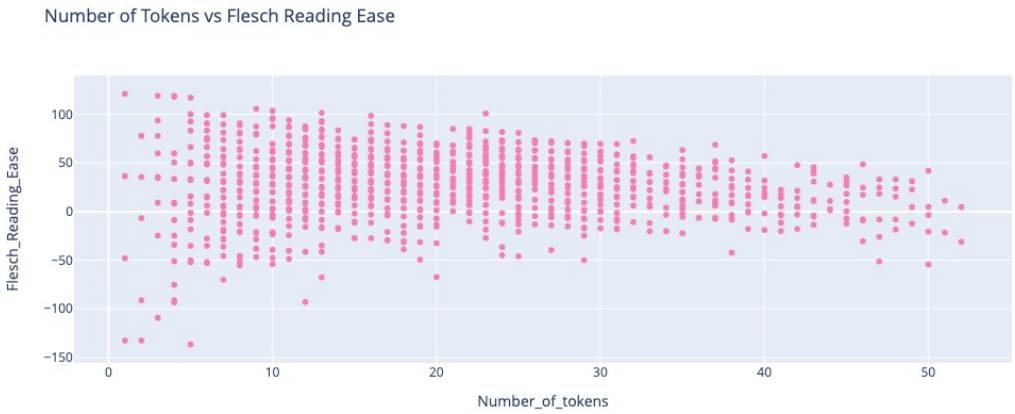

Figure 65: Number of tokens vs Flesch Reading Ease for SkillSpan balanced.

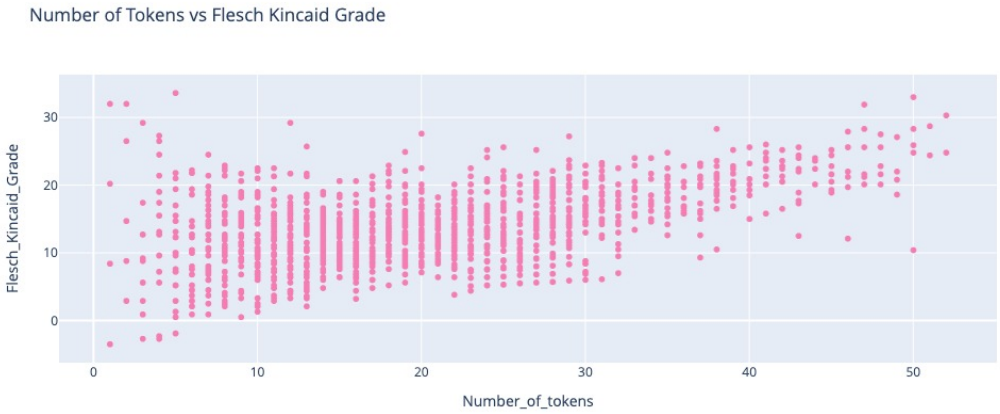

Figure 66: Number of tokens vs Flesch Kincaid Grade for SkillSpan balanced.

Figure 67: Number of tokens vs Gunning Fog Index for SkillSpan balanced.

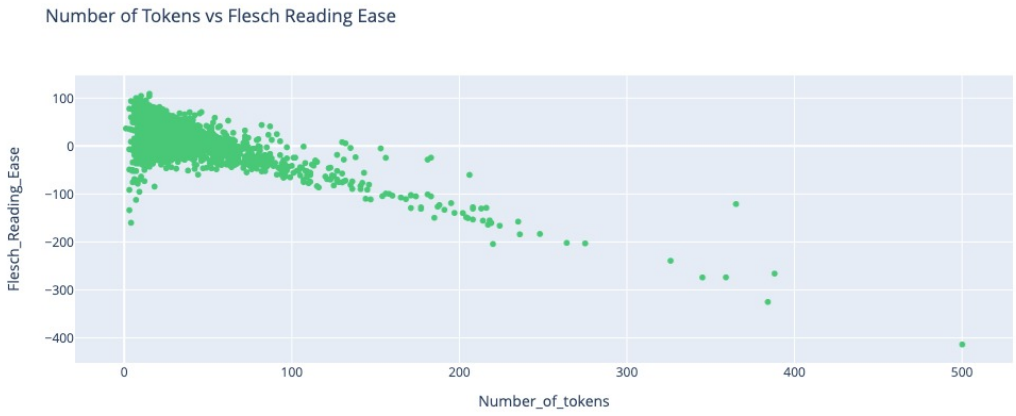

Figure 68: Number of tokens vs Flesch Reading Ease for HaHujobs.

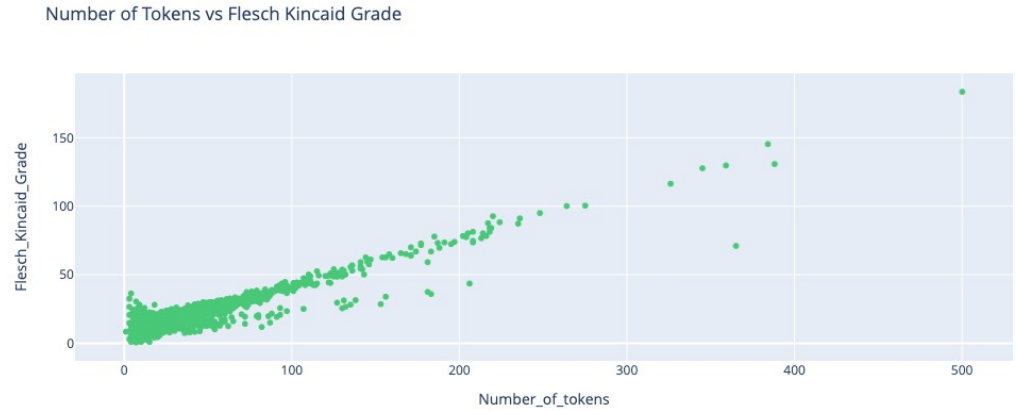

Figure 69: Number of tokens vs Flesch Kincaid Grade for HaHujobs.

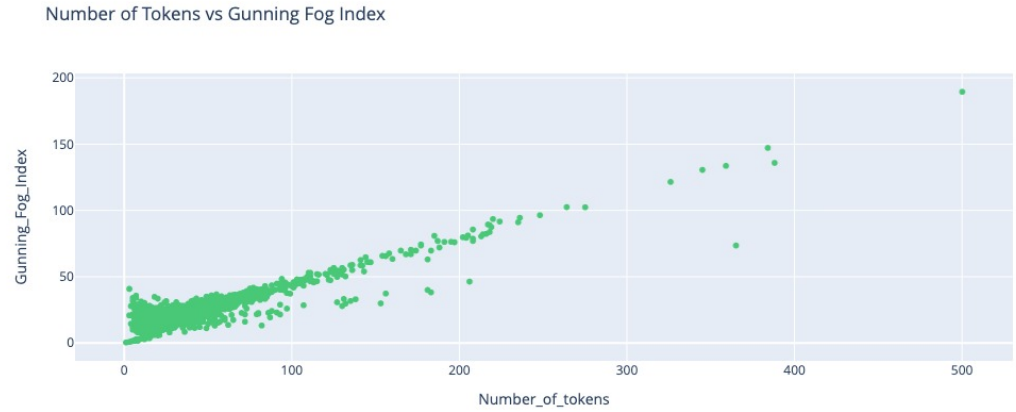

Figure 70: Number of tokens vs Gunning Fog Index for HaHuJobs.

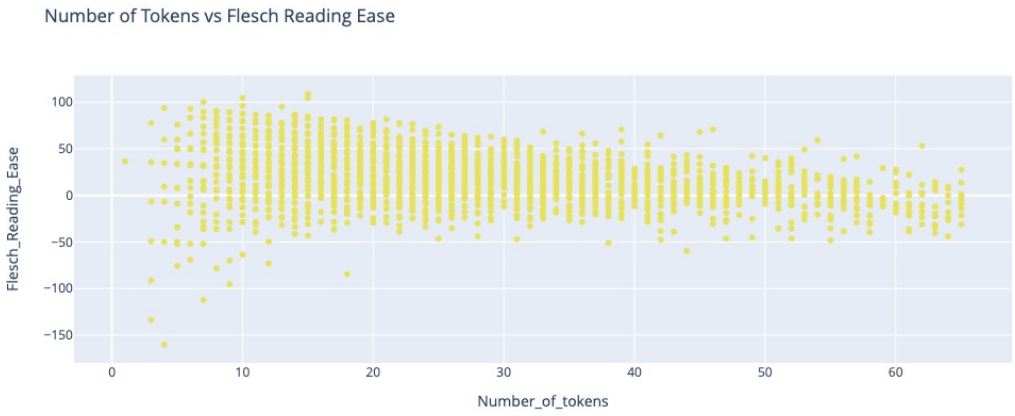

Figure 71: Number of tokens vs Flesch Reading Ease for HaHujobs balanced.

Figure 72: Number of tokens vs Flesch Kincaid Grade for HaHujobs balanced.

Figure 73: Number of tokens vs Gunning Fog Index for HaHujobs balanced.

9 Appendix B (ER Inference Screenshots)

```
# Generate sample job listing with ChatGPT

doc1 = nlp_ner("""
We are currently seeking a highly skilled and experienced Senior Business Consultant to provide strategic guidance and
solutions to our clients. The ideal candidate should hold a Master's degree in Business Administration, Economics, or a related field,
complemented by at least 5 years of proven experience in a business consulting role. Essential skills for this position include advanced
analytical and critical thinking abilities, exceptional communication skills, both written and verbal, and a strong proficiency in business
analysis tools and software. Candidates should demonstrate expertise in project management, with a capacity to lead and execute complex
projects while maintaining strict timelines and budgets. A deep understanding of market trends, business operations, and experience in
developing and implementing business strategies is crucial. Familiarity with digital transformation, data analysis, and financial modeling
will be highly regarded. The role also demands excellent interpersonal skills, as it involves collaborating with diverse teams and
stakeholders to achieve business objectives.
""")

colors = {"SKILL": "linear-gradient(90deg, #53ed28, #b9f5a9)"}
options = {"ents": ["SKILL"], "colors": colors}
spacy.displacy.render(doc1, style = "ent", options = options)
```

We are currently seeking a highly skilled SKILL and experienced Senior Business Consultant to provide strategic guidance and solutions SKILL to our clients. The ideal candidate should hold a Master's degree in Business Administration, Economics, or a related field, complemented by at least 5 years of proven experience in a business consulting role. Essential skills for this position include advanced SKILL analytical SKILL and critical thinking abilities SKILL, exceptional communication skills SKILL, both written and verbal SKILL, and a strong proficiency in business analysis tools SKILL and software. Candidates should demonstrate expertise in project management SKILL, with a capacity to lead and execute complex projects SKILL while maintaining strict timelines and budgets SKILL. A deep understanding of market trends, business operations, and experience in developing and implementing business strategies is crucial SKILL. Familiarity with digital transformation, data analysis SKILL, and financial modeling SKILL will be highly regarded. The role also demands excellent interpersonal skills SKILL, as it involves collaborating with diverse teams and stakeholders SKILL to achieve business objectives SKILL.

Figure 74: Job description for a Business Consultant

```
doc2 = nlp_ner("""Join our innovative e-commerce team as an experienced Python developer, where you'll be instrumental in building an
advanced NLP-driven chatbot. Your primary role will involve leading the development of a sophisticated NLP model to enhance customer interact
Collaborate with a team of skilled programmers, integrating this cutting-edge chatbot into our digital platforms. We're looking for a
professional with a solid background in Python and a portfolio demonstrating proficiency in NLP or similar fields. Familiarity with
machine learning frameworks like TensorFlow or PyTorch, and NLP libraries such as NLTK or spaCy, is essential. Your approach should blend
innovative thinking with a detail-oriented and analytical mindset, thriving in problem-solving and complex challenges.
As a team player, your ability to communicate effectively and work in synergy with others is crucial. We offer a dynamic and supportive
environment, competitive compensation, and opportunities for professional growth. If you're passionate about using Python and NLP to
revolutionize e-commerce customer experiences, we're excited to meet you! Apply now and become a key player in transforming the future
of e-commerce through technology.
""")

colors = {"SKILL": "linear-gradient(90deg, #f25a6b, #f7c6cc)"}
options = {"ents": ["SKILL"], "colors": colors}
spacy.displacy.render(doc2, style = "ent", options = options)
```

Join our innovative e-commerce team as an experienced Python developer, where you'll be instrumental in building an advanced NLP-driven chatbot SKILL. Your primary role will involve leading the development of a sophisticated NLP model SKILL to enhance customer interactions SKILL. Collaborate with a team of skilled programmers, integrating this cutting-edge chatbot into our digital platforms SKILL. We're looking for a professional with a solid background in Python and a portfolio demonstrating proficiency in NLP SKILL or similar fields. Familiarity with machine learning frameworks like TensorFlow or PyTorch, and NLP libraries SKILL such as NLTK or spaCy, is essential. Your approach should blend innovative thinking SKILL with a detail-oriented SKILL and analytical mindset SKILL, thriving in problem-solving and complex challenges SKILL. As a team player SKILL, your ability to communicate effectively and work in synergy with others SKILL is crucial. We offer a dynamic and supportive environment, competitive compensation SKILL, and opportunities for professional growth. If you're passionate about SKILL using Python and NLP to revolutionize e-commerce customer experiences, we're excited to meet you! Apply now and become a key player in transforming the future of e-commerce through technology SKILL.

Figure 75: Job description for a Python developer

```
doc3 = nlp_ner("""We are seeking a dedicated Civil Engineer to join our dynamic team, focused on developing groundbreaking infrastructure
projects. In this role, you'll be instrumental in planning, designing, and overseeing the construction of our next-generation infrastructure
solutions. Your primary responsibilities include conducting site evaluations, developing detailed project plans, and ensuring compliance
with industry standards and regulations.We require a professional with a robust background in civil engineering, demonstrated through a
portfolio of successful projects. Proficiency in using design and analysis software, such as AutoCAD and Civil 3D, is essential. Your
approach should embody precision, a keen eye for detail, and a commitment to sustainable and efficient design practices. Collaborative
teamwork and clear communication skills are vital as you will work closely with architects, contractors, and project managers. We offer a
challenging yet supportive work environment, opportunities for professional development, and competitive compensation. If you're passionate
about shaping the future through innovative civil engineering and thrive in a collaborative, dynamic setting, we invite you to apply.
Join us in building the infrastructure of tomorrow!""")

colors = {"SKILL": "linear-gradient(90deg, #b53e8d, #f2c7e3)"}
options = {"ents": ["SKILL"], "colors": colors}
spacy.displacy.render(doc3, style = "ent", options = options)
```

"We are seeking a dedicated Civil Engineer to join our dynamic team, focused on developing groundbreaking infrastructure projects SKILL . In this role, you'll be instrumental in planning SKILL , designing SKILL , and overseeing the construction of our next SKILL -generation infrastructure solutions. Your primary responsibilities include conducting site evaluations SKILL , developing detailed project plans SKILL , and ensuring compliance with industry standards and regulations SKILL .We require a professional with a robust background in civil engineering, demonstrated through a portfolio of successful projects. Proficiency in using design and analysis software SKILL , such as AutoCAD SKILL and Civil 3D, is essential. Your approach should embody precision SKILL , a keen eye for detail SKILL , and a commitment to sustainable and efficient design practices SKILL . Collaborative teamwork and clear communication skills SKILL are vital as you will work closely with architects, contractors, and project managers. We offer a challenging yet supportive work environment, opportunities SKILL for professional development, and competitive compensation SKILL . If you're passionate SKILL about shaping the future through innovative civil engineering and thrive in a collaborative SKILL , dynamic setting SKILL , we invite you to apply. Join us in building the infrastructure of tomorrow! SKILL

Figure 76: Job description for a Civil Engineer

```
doc4 = nlp_ner("""We are looking for an experienced Bank Manager to lead and oversee the operations of our vibrant banking branch. As a key
figure in driving the branch's success, you will play a crucial role in fostering robust financial services, ensuring customer satisfaction,
and meeting branch objectives. Your core responsibilities will involve managing branch staff, developing effective business strategies, and
enhancing the overall customer banking experience. Candidates must possess a strong background in banking and financial management,
evidenced by a track record of successful leadership and operational excellence. Proficiency in financial software, risk management
principles, and a deep understanding of banking regulations and compliance standards are essential. A strategic mindset, combined with
meticulous attention to detail and a commitment to ethical banking practices, is paramount. Effective leadership and team management skills
are crucial as you will guide and mentor your team towards achieving exceptional performance. Excellent communication skills are vital, as
the role involves interacting with a diverse range of clients, staff, and upper management. We provide a dynamic and supportive work
environment, opportunities for career growth, and competitive remuneration. If you are passionate about driving positive change in the
banking sector, possess a knack for innovative financial solutions, and excel in a fast-paced, collaborative environment, we encourage you
to apply. Join us in shaping the future of banking and delivering outstanding service to our community.""")

colors = {"SKILL": "linear-gradient(90deg, #3f87d9, #a3c1e3)"}
options = {"ents": ["SKILL"], "colors": colors}
spacy.displacy.render(doc4, style = "ent", options = options)
```

We are looking for an experienced Bank Manager to lead and oversee the operations of our vibrant banking SKILL branch. As a key figure in driving the branch's success, you will play a crucial role in fostering robust financial services SKILL , ensuring customer satisfaction SKILL , and meeting branch objectives. Your core responsibilities will involve managing branch staff SKILL , developing effective business strategies SKILL , and enhancing the overall customer banking SKILL experience. Candidates must possess a strong background in banking and financial management, evidenced by a track record of successful leadership and operational SKILL excellence. Proficiency in financial software SKILL , risk management principles SKILL , and a deep understanding of banking regulations and compliance standards are essential. A strategic mindset SKILL , combined with meticulous SKILL attention to detail and a commitment to ethical banking practices SKILL , is paramount. Effective leadership and team management skills SKILL are crucial as you will guide and mentor your team SKILL towards achieving exceptional performance SKILL . Excellent communication skills SKILL are vital, as the role involves interacting SKILL with a diverse range of clients, staff, and upper management SKILL . We provide a dynamic and supportive work environment, opportunities for career growth, and competitive remuneration SKILL . If you are passionate about driving positive change SKILL in the banking sector, possess a knack for innovative financial solutions SKILL , and excel in a fast-paced SKILL , collaborative environment SKILL , we encourage you to apply. Join us in shaping the future of banking and delivering outstanding service SKILL to our community.

Figure 77: Job description for a Bank Manager

```
doc5 = nlp_ner("""We are currently seeking a highly motivated and dynamic Sales Representative to join our car dealership team. In this
role, you will be a key player in driving our vehicle sales, providing exceptional customer service, and building lasting relationships
with clients. Your primary responsibilities will include showcasing our range of cars, understanding customer needs, and guiding them
through the purchasing process. The ideal candidate should have a solid background in sales, preferably in the automotive industry,
demonstrated by a history of achieving sales targets and providing outstanding customer service. Knowledge of different car models,
specifications, and the latest automotive trends is crucial. Proficiency in sales and CRM software is highly desirable. Strong communication
and negotiation skills are essential, as the role requires the ability to connect with a diverse clientele, understand their preferences,
and offer tailored solutions. Your approach should be customer-focused, with an emphasis on honesty, integrity, and building trust. Team
collaboration is vital, as you will work closely with other sales representatives, finance managers, and service departments to ensure a
seamless and satisfying customer experience. We offer a competitive compensation package, including commissions, a supportive work
environment, and opportunities for professional growth and development. If you have a passion for cars, a drive for sales, and thrive in
a fast-paced, customer-oriented environment, we encourage you to apply. Join our team and help us drive the success of our dealership by
matching customers with their ideal vehicles.""")

colors = {"SKILL": "linear-gradient(90deg, #e2ed0e, #edeaca)"}
options = {"ents": ["SKILL"], "colors": colors}
spacy.displacy.render(doc5, style = "ent", options = options)
```

We are currently seeking a highly motivated SKILL and dynamic SKILL Sales Representative to join our car dealership team. In this role, you will be a key player in driving our vehicle sales SKILL , providing exceptional customer service SKILL , and building lasting relationships with clients SKILL . Your primary responsibilities will include showcasing our range of cars SKILL , understanding customer needs SKILL , and guiding them through the purchasing process. The ideal candidate should have a solid background in sales, preferably in the automotive industry, demonstrated by a history of achieving sales targets SKILL and providing outstanding customer service SKILL . Knowledge of different car models, specifications, and the latest automotive trends is crucial. Proficiency in sales and CRM software SKILL is highly desirable. Strong communication and negotiation skills SKILL are essential, as the role requires the ability to connect with a diverse clientele, understand their preferences, and offer tailored solutions. Your approach should be customer-focused SKILL , with an emphasis on honesty SKILL , integrity SKILL , and building trust SKILL . Team collaboration SKILL is vital, as you will work closely with other sales representatives SKILL , finance managers, and service departments to ensure a seamless SKILL and satisfying customer experience SKILL . We offer a competitive compensation package, including commissions SKILL , a supportive work environment, and opportunities for professional growth and development. If you have a passion for cars SKILL , a drive for sales SKILL , and thrive in a fast-paced SKILL , customer-oriented environment SKILL , we encourage you to apply. Join our team and help us drive the success of our dealership by matching customers with their ideal vehicles.

Figure 78: Job description for a Sales Representative for cars

10 *Ablation Studies*

We have conducted our NER experiments using spaCy's internal transformer "en_core_web_trf" (based on the RoBERTa model) and utilized the train_test_split function from the sklearn.model_selection module to divide our dataset into training, development, and test sets. Initially, the dataset training_data_ner was split into two parts: 90% for training (train_data_ner) and 10% as a remaining dataset (remaining_data_ner). This split ensures that the model has sufficient data to learn from during the training phase. Subsequently, the remaining dataset was further divided equally into development (dev_data_ner) and test sets (test_data_ner), each constituting 5% of the original dataset. The development set is used for tuning hyperparameters and validating the model during training, while the test set is reserved for evaluating the final performance of the trained model. The random_state = 42 parameter ensures reproducibility by fixing the random seed. We have used the version 1 of SpaCy's Adam optimizer with an initial learning rate of 0.001, beta1 = 0.9 and beta2 = 0.999. The evaluation frequency was set to 200 and the dropout to 0.1. The complete config.cfg file of the hyperparameters for the finetuning experiments can be seen at the end of the ablation studies.

Experiment 1:

Table 10: Steps: 8000, Patience: 2000, Rows: 5520, No token limit

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0.0	0.0	0.0	0.0	0.0	0.0
0.0	200.0	12.06	17.28	9.27	0.12
0.0	400.0	28.82	36.31	23.89	0.29
0.0	600.0	38.45	43.57	34.4	0.38
0.0	800.0	42.06	51.28	35.65	0.42
1.0	1000.0	43.25	45.98	40.82	0.43
1.0	1200.0	46.0	51.55	41.53	0.46
2.0	1400.0	47.4	48.51	46.35	0.47
3.0	1600.0	47.66	49.06	46.35	
4.0	1800.0	46.9	49.5	44.56	0.47
5.0	2000.0	49.35	51.66	47.24	0.49
6.0	2200.0	46.5	49.5	43.85	0.47
8.0	2400.0	48.45	51.19	45.99	0.48
10.0	2600.0	45.93	47.78	44.21	0.46
11.0	2800.0	48.56	50.78	46.52	0.49
13.0	3000.0	47.71	51.33	44.56	0.48
15.0	3200.0	45.0	48.36	42.07	0.45
17.0	3400.0	48.92	51.79	46.35	0.49
18.0	3600.0	44.84	47.33	42.6	0.45
20.0	3800.0	47.63	50.91	44.74	0.48
22.0	4000.0	45.14	48.47	42.25	0.45

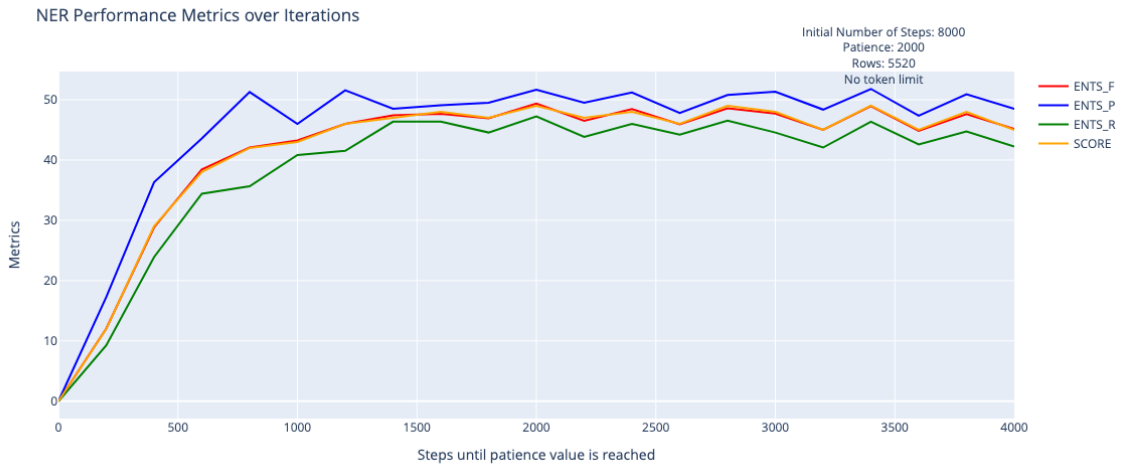

Figure 79: NER Performance Metrics, Steps: 8000, Patience: 2000, Rows: 5520, No token limit

Table 11: Nervaluate metrics for Steps: 8000, Patience: 2000, Rows: 5520, No token limit.

Metric	ent_type	partial	strict	exact
correct	434.0	84.0	84.0	84.0
incorrect	0.0	0.0	350.0	350.0
partial	0.0	350.0	0.0	0.0
missed	155.0	155.0	155.0	155.0
spurious	106.0	106.0	106.0	106.0
possible	589.0	589.0	589.0	589.0
actual	540.0	540.0	540.0	540.0
precision	0.80	0.48	0.15	0.15
recall	0.73	0.44	0.14	0.14
f1	0.76	0.46	0.15	0.15

Table 12: Steps: 16000, Patience: 2000, Rows: 5520, No token limit

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0.0	0.0	0.0	0.0	0.0	0.0
0.0	200.0	17.77	24.61	13.9	0.18
0.0	400.0	28.85	36.41	23.89	0.29
0.0	600.0	37.4	41.76	33.87	0.37
0.0	800.0	37.37	46.19	31.37	0.37
1.0	1000.0	41.96	45.61	38.86	0.42
1.0	1200.0	45.5	52.57	40.11	0.46
2.0	1400.0	48.0	48.98	47.06	0.48
3.0	1600.0	48.57	50.38	46.88	0.49
4.0	1800.0	48.48	50.19	46.88	0.48
5.0	2000.0	48.14	51.63	45.1	0.48
6.0	2200.0	45.5	51.11	41.0	0.45
8.0	2400.0	45.02	48.65	41.89	0.45

10.0	2600.0	46.44	48.92	44.21	0.46
11.0	2800.0	45.2	48.37	42.42	0.45
13.0	3000.0	44.72	48.94	41.18	0.45
15.0	3200.0	46.7	48.83	44.74	0.47
17.0	3400.0	47.32	50.0	44.92	0.47
18.0	3600.0	43.05	46.95	39.75	0.43

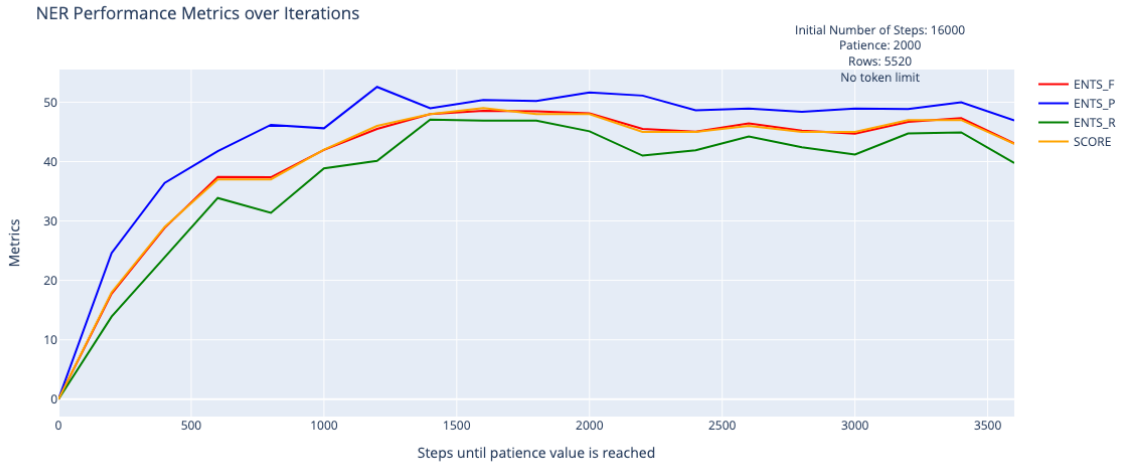

Figure 80: NER Performance Metrics, Steps: 16000, Patience: 2000, Rows: 5520, No token limit

Table 13: Nervaluate metrics for Steps: 16000, Patience: 2000, Rows: 5520, No token limit

Metric	ent_type	partial	strict	exact
correct	447.0	82.0	82.0	82.0
incorrect	0.0	0.0	365.0	365.0
partial	0.0	365.0	0.0	0.0
missed	142.0	142.0	142.0	142.0
spurious	99.0	99.0	99.0	99.0
possible	589.0	589.0	589.0	589.0
actual	546.0	546.0	546.0	546.0
precision	0.82	0.48	0.15	0.15
recall	0.76	0.45	0.14	0.14
f1	0.79	0.47	0.14	0.14

Table 14: Steps: 20000, Patience: 3000, Rows: 5520, No token limit

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0.0	0.0	0.0	0.0	0.0
0	200.0	17.64	20.14	15.69	18.0
0	400.0	22.08	28.53	18.0	22.0
0	600.0	37.25	41.39	33.87	37.0
0	800.0	38.49	48.51	31.91	38.0
1	1000.0	45.56	47.4	43.85	46.0

1	1200.0	47.07	51.04	43.67	47.0
2	1400.0	47.05	48.76	45.45	47.0
3	1600.0	48.98	51.27	46.88	49.0
4	1800.0	47.44	48.78	46.17	47.0
5	2000.0	47.38	48.07	46.7	47.0
6	2200.0	44.65	51.52	39.39	45.0
8	2400.0	47.55	50.5	44.92	48.0
10	2600.0	47.45	50.5	44.74	47.0
11	2800.0	47.66	51.23	44.56	48.0
13	3000.0	46.93	50.0	44.21	47.0
15	3200.0	45.64	49.38	42.42	46.0
17	3400.0	48.5	51.08	46.17	49.0
18	3600.0	47.18	49.9	44.74	47.0
20	3800.0	47.88	52.21	44.21	48.0
22	4000.0	46.91	51.16	43.32	47.0
23	4200.0	49.63	51.84	47.59	50.0
25	4400.0	48.21	52.75	44.39	48.0
27	4600.0	48.92	51.58	46.52	49.0
29	4800.0	46.44	48.92	44.21	46.0
30	5000.0	46.78	49.02	44.74	47.0
32	5200.0	47.7	49.33	46.17	48.0
34	5400.0	47.48	47.91	47.06	47.0
36	5600.0	46.17	49.19	43.49	46.0
37	5800.0	47.48	49.9	45.28	47.0
39	6000.0	46.54	48.92	44.39	47.0
41	6200.0	47.59	50.6	44.92	48.0
42	6400.0	48.14	51.85	44.92	48.0
44	6600.0	48.51	52.72	44.92	49.0
46	6800.0	47.25	49.61	45.1	47.0
48	7000.0	50.56	52.82	48.48	51.0
49	7200.0	46.17	50.64	42.42	46.0
51	7400.0	46.09	50.21	42.6	46.0
53	7600.0	47.48	48.87	46.17	47.0
55	7800.0	46.76	49.4	44.39	47.0
56	8000.0	45.95	48.7	43.49	46.0
58	8200.0	45.65	48.03	43.49	46.0
60	8400.0	46.36	48.72	44.21	46.0
61	8600.0	46.67	50.1	43.67	47.0
63	8800.0	48.3	51.3	45.63	48.0
65	9000.0	47.73	50.91	44.92	48.0
67	9200.0	49.09	49.91	48.31	49.0
68	9400.0	47.34	50.71	44.39	47.0
70	9600.0	44.84	46.89	42.96	45.0
72	9800.0	47.32	50.0	44.92	47.0
74	10000.0	48.89	50.67	47.24	49.0

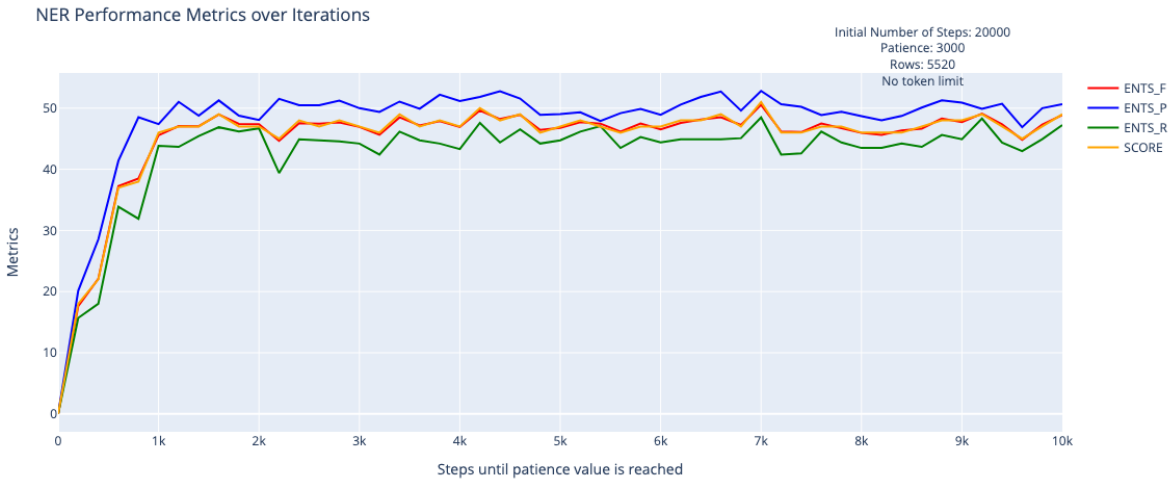

Figure 81: NER Performance Metrics, 5520 rows, no token limit

The dataset with no token limit allows the model to utilize the full context of the text, leading to superior performance. Metrics such as ENTS_F (F1 score), ENTS_P (Precision), ENTS_R (Recall), and SCORE are measured across various epochs and iterations. Over time, the model demonstrates steady improvements in ENTS_F, ENTS_P, and ENTS_R, with the SCORE, representing overall performance, reaching approximately 48%-51% at later stages. This indicates a well-trained model capable of processing longer sentences and capturing more context and relationships between entities, thus enhancing its ability to accurately recognize named entities.

Experiment 2:

Table 15: Steps: 8000, Patience: 2000, Rows: 5156 Token Limit = 45

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0.0	0.0	2.24	1.45	4.88	0.02
0.0	200.0	25.59	30.68	21.95	0.26
0.0	400.0	32.21	37.81	28.05	0.32
0.0	600.0	39.74	42.92	36.99	0.4
1.0	800.0	38.47	41.61	35.77	0.38
1.0	1000.0	39.59	40.17	39.02	0.4
2.0	1200.0	44.49	46.95	42.28	0.44
3.0	1400.0	44.94	49.63	41.06	0.45
3.0	1600.0	44.71	47.7	42.07	0.45
5.0	1800.0	45.05	48.48	42.07	0.45
6.0	2000.0	44.51	45.81	43.29	0.45
8.0	2200.0	44.83	47.71	42.28	0.45
10.0	2400.0	42.87	45.35	40.65	0.43
12.0	2600.0	43.86	45.8	42.07	0.44
14.0	2800.0	43.68	46.15	41.46	0.44
16.0	3000.0	44.1	47.64	41.06	0.44
18.0	3200.0	42.78	44.42	41.26	0.43
20.0	3400.0	42.73	46.63	39.43	0.43
22.0	3600.0	41.77	43.42	40.24	0.42
24.0	3800.0	44.89	46.61	43.29	0.45

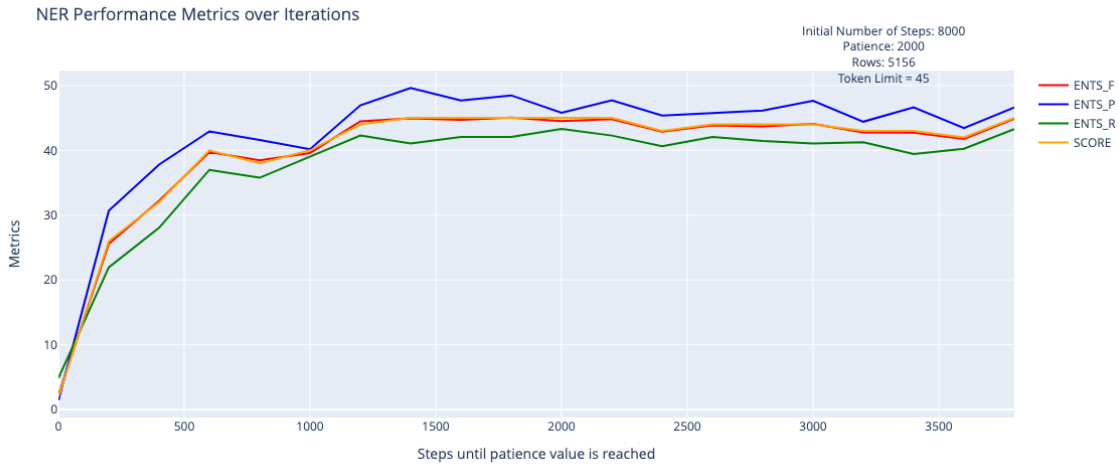

Figure 82: NER Performance Metrics, Steps: 8000, Patience: 2000, Rows: 5156, Token limit = 45

Table 16: Nervaluate metrics for Steps: 8000, Patience: 2000, Rows: 5156, Token limit = 45.

Metric	ent_type	partial	strict	exact
correct	373.0	84.0	84.0	84.0
incorrect	0.0	0.0	289.0	289.0
partial	0.0	289.0	0.0	0.0

missed	130.0	130.0	130.0	130.0
spurious	80.0	80.0	80.0	80.0
possible	503.0	503.0	503.0	503.0
actual	453.0	453.0	453.0	453.0
precision	0.82	0.5	0.19	0.19
recall	0.74	0.45	0.17	0.17
f1	0.78	0.48	0.18	0.18

Table 17: Steps: 16000, Patience: 2000, Rows: 5156 Token Limit = 45

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0.0	0.0	2.24	1.45	4.88	0.02
0.0	200.0	15.96	23.08	12.2	0.16
0.0	400.0	28.71	34.08	24.8	0.29
0.0	600.0	40.93	42.79	39.23	0.37
1.0	800.0	41.13	44.26	38.41	0.41
1.0	1000.0	39.87	41.21	38.62	0.4
2.0	1200.0	46.52	47.56	45.53	0.47
3.0	1400.0	39.45	45.6	34.76	0.39
3.0	1600.0	46.38	53.3	41.06	0.46
5.0	1800.0	43.78	46.36	41.46	0.44
6.0	2000.0	46.88	48.08	45.73	0.47
8.0	2200.0	42.22	46.57	38.62	0.42
10.0	2400.0	42.06	44.55	39.84	0.42
12.0	2600.0	42.4	44.8	40.24	0.42
14.0	2800.0	43.5	47.04	40.45	0.43
16.0	3000.0	41.25	44.01	38.82	0.41
18.0	3200.0	39.02	40.8	37.4	0.39
20.0	3400.0	42.8	44.69	41.06	0.43
22.0	3600.0	43.08	44.84	41.46	0.43
24.0	3800.0	43.24	42.6	43.9	0.43
26.0	4000.0	41.27	44.58	38.41	0.41

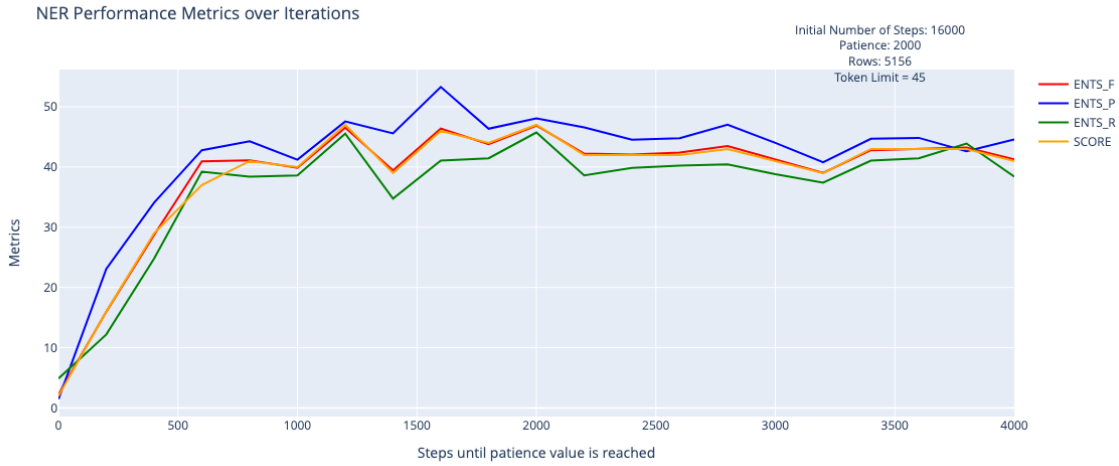

Figure 83: NER Performance Metrics, Steps: 16000, Patience: 2000, Rows: 5156, Token limit = 45

Table 18: Nervaluate metrics for Steps: 16000, Patience: 2000, Rows: 5156, Token limit = 45.

Metric	ent_type	partial	strict	exact
correct	390.0	88.0	88.0	88.0
incorrect	0.0	0.0	302.0	302.0
partial	0.0	302.0	0.0	0.0
missed	113.0	113.0	113.0	113.0
spurious	87.0	87.0	87.0	87.0
possible	503.0	503.0	503.0	503.0
actual	477.0	477.0	477.0	477.0
precision	0.82	0.5	0.18	0.18
recall	0.78	0.48	0.17	0.17
f1	0.8	0.49	0.18	0.18

Table 19: Steps: 20000, Patience: 3000, Rows: 5156, Token limit = 45

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0.0	2.24	1.45	4.88	2.0
0	200.0	21.18	31.1	16.06	21.0
0	400.0	28.88	34.97	24.59	29.0
0	600.0	38.53	42.61	35.16	39.0
1	800.0	37.09	40.58	34.15	37.0
1	1000.0	40.56	39.69	41.46	41.0
2	1200.0	44.51	43.73	45.33	45.0
3	1400.0	40.18	45.83	35.77	40.0
3	1600.0	43.57	45.66	41.67	44.0
5	1800.0	43.64	45.58	41.87	44.0
6	2000.0	45.87	48.53	43.5	46.0
8	2200.0	42.97	46.35	40.04	43.0
10	2400.0	42.69	44.23	41.26	43.0
12	2600.0	44.56	45.45	43.7	45.0
14	2800.0	41.95	44.8	39.43	42.0
16	3000.0	42.49	44.27	40.85	42.0
18	3200.0	43.17	45.03	41.46	43.0
20	3400.0	42.49	44.27	40.85	42.0
22	3600.0	42.48	44.01	41.06	42.0
24	3800.0	41.14	41.22	41.06	41.0
26	4000.0	41.56	44.44	39.02	42.0
28	4200.0	46.17	47.72	44.72	46.0
30	4400.0	43.95	46.0	42.07	44.0
32	4600.0	41.86	42.47	41.26	42.0
34	4800.0	44.97	45.88	44.11	45.0
36	5000.0	44.1	44.94	43.29	44.0
38	5200.0	43.28	46.81	40.24	43.0

40	5400.0	41.91	43.97	40.04	42.0
42	5600.0	42.2	43.19	41.26	42.0
45	5800.0	43.7	46.45	41.26	44.0
47	6000.0	42.07	43.83	40.45	42.0
49	6200.0	43.95	46.0	42.07	44.0
51	6400.0	45.29	45.66	44.92	45.0
53	6600.0	41.39	42.83	40.04	41.0
55	6800.0	41.42	44.14	39.02	41.0
57	7000.0	43.09	45.58	40.85	43.0
59	7200.0	40.94	43.05	39.02	41.0

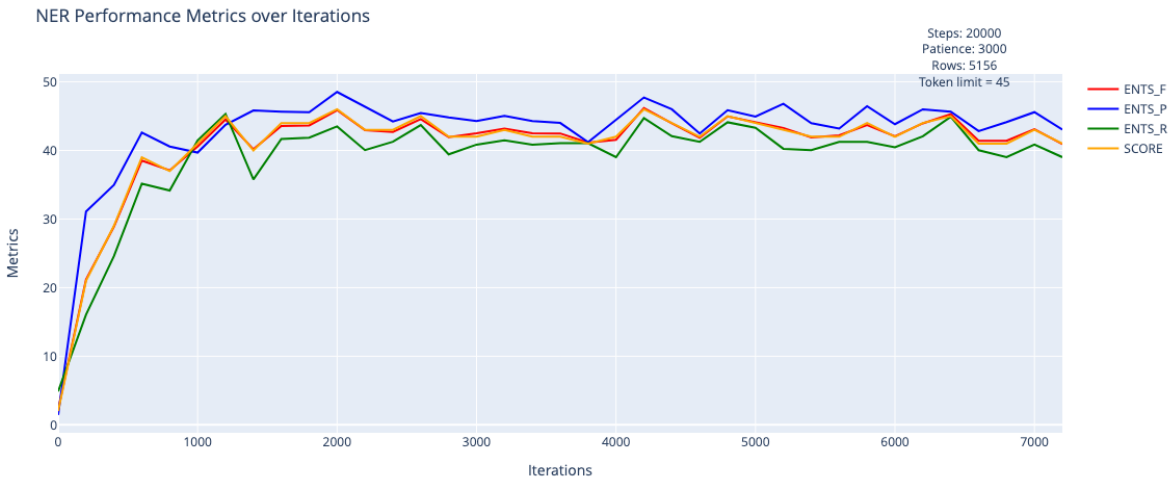

Figure 84: NER Performance Metrics, Steps: 20000, Patience: 3000, Rows: 5156, Token limit = 45

The dataset with a token limit of 45 records metrics similarly to the no token limit dataset, but the performance metrics (ENTS_F, ENTS_P, ENTS_R, SCORE) are slightly lower. The SCORE reaches around 46%-48% at later stages. Imposing a token limit of 45 reduces the context available for the model, which might cause it to miss important relationships and dependencies between entities that span beyond the limit. This slight degradation in performance suggests that, while the token limit simplifies processing and reduces memory requirements, it also trims valuable information essential for accurate entity recognition.

Experiment 3:

Table 20: Steps: 8000, Patience: 2000, Rows: 4236, Token limit = 30

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0.0	0.0	1.91	1.75	2.11	0.02
0.0	200.0	15.75	25.9	11.32	0.16
0.0	400.0	31.67	43.18	25.0	0.32
1.0	600.0	34.92	41.76	30.0	0.35
1.0	800.0	36.15	39.75	33.16	0.36
2.0	1000.0	39.67	41.94	37.63	0.4
3.0	1200.0	44.17	45.53	42.89	0.44
4.0	1400.0	45.26	46.65	43.95	0.45
5.0	1600.0	42.8	44.7	41.05	0.43
7.0	1800.0	45.02	46.74	43.42	0.45
9.0	2000.0	44.98	48.62	41.84	0.45
11.0	2200.0	41.23	46.38	37.11	0.41
14.0	2400.0	44.78	47.49	42.37	0.45
17.0	2600.0	42.52	45.37	40.0	0.43
20.0	2800.0	46.15	48.28	44.21	0.46
23.0	3000.0	45.27	45.82	44.74	0.45
26.0	3200.0	45.36	47.16	43.68	0.45
29.0	3400.0	45.26	46.65	43.95	0.45
32.0	3600.0	44.41	47.32	41.84	0.44
35.0	3800.0	44.38	46.92	42.11	0.44
38.0	4000.0	44.8	49.68	40.79	0.45
41.0	4200.0	44.16	48.14	40.79	0.44
44.0	4400.0	44.69	46.33	43.16	0.45
47.0	4600.0	44.03	44.39	43.68	0.44
50.0	4800.0	44.93	46.86	43.16	0.45

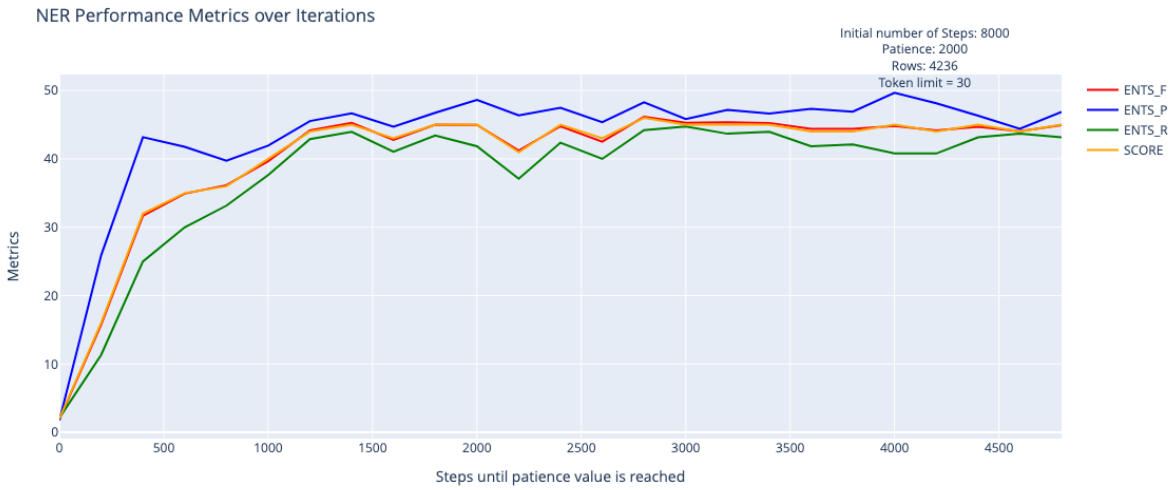

Figure 85: NER Performance Metrics, Steps: 8000, Patience: 2000, Rows: 4236, Token limit = 30

Table 21: Nervaluate metrics for Steps: 8000, Patience: 2000, Rows: 4236, Token limit = 30.

Metric	ent_type	partial	strict	exact
correct	281.0	81.0	81.0	81.0
incorrect	0.0	0.0	200.0	200.0
partial	0.0	200.0	0.0	0.0
missed	97.0	97.0	97.0	97.0
spurious	59.0	59.0	59.0	59.0
possible	378.0	378.0	378.0	378.0
actual	340.0	340.0	340.0	340.0
precision	0.83	0.53	0.24	0.24
recall	0.74	0.48	0.21	0.21
f1	0.78	0.5	0.23	0.23

Table 22: Steps: 16000, Patience: 2000, Rows: 4236, Token limit = 30

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0.0	0.0	1.91	1.75	2.11	0.02
0.0	200.0	12.03	21.05	8.42	0.12
0.0	400.0	33.01	42.86	26.84	0.33
1.0	600.0	37.23	44.81	31.84	0.37
1.0	800.0	35.62	38.14	33.42	0.36
2.0	1000.0	41.23	43.79	38.95	0.41
3.0	1200.0	44.44	45.23	43.68	0.44
4.0	1400.0	44.06	44.72	43.42	0.44
5.0	1600.0	43.69	45.1	42.37	0.44
7.0	1800.0	44.32	46.78	42.11	0.44
9.0	2000.0	44.17	46.13	42.37	0.44
11.0	2200.0	45.63	47.44	43.95	0.46
14.0	2400.0	43.99	45.15	42.89	0.44
17.0	2600.0	46.07	50.47	42.37	0.46
20.0	2800.0	46.35	49.7	43.42	0.46
23.0	3000.0	45.21	45.7	44.74	0.45
26.0	3200.0	44.5	45.94	43.16	0.45
29.0	3400.0	44.87	46.72	43.16	0.45
32.0	3600.0	43.63	44.97	42.37	0.44
35.0	3800.0	44.5	45.94	43.16	0.45
38.0	4000.0	44.54	46.31	42.89	0.45
41.0	4200.0	42.34	44.97	40.0	0.42
44.0	4400.0	45.14	46.39	43.95	0.45
47.0	4600.0	45.28	46.41	44.21	0.45
50.0	4800.0	48.02	49.86	46.32	0.48
53.0	5000.0	48.74	52.1	45.79	0.49
56.0	5200.0	45.57	47.31	43.95	0.46
59.0	5400.0	46.77	48.99	44.74	0.47
62.0	5600.0	46.47	48.98	44.21	0.46

65.0	5800.0	47.03	50.92	43.68	0.47
68.0	6000.0	43.29	45.14	41.58	0.43
71.0	6200.0	47.59	49.86	45.53	0.48
74.0	6400.0	45.29	47.03	43.68	0.45
77.0	6600.0	45.15	47.66	42.89	0.45
80.0	6800.0	43.89	46.47	41.58	0.44
83.0	7000.0	46.13	48.55	43.95	0.46

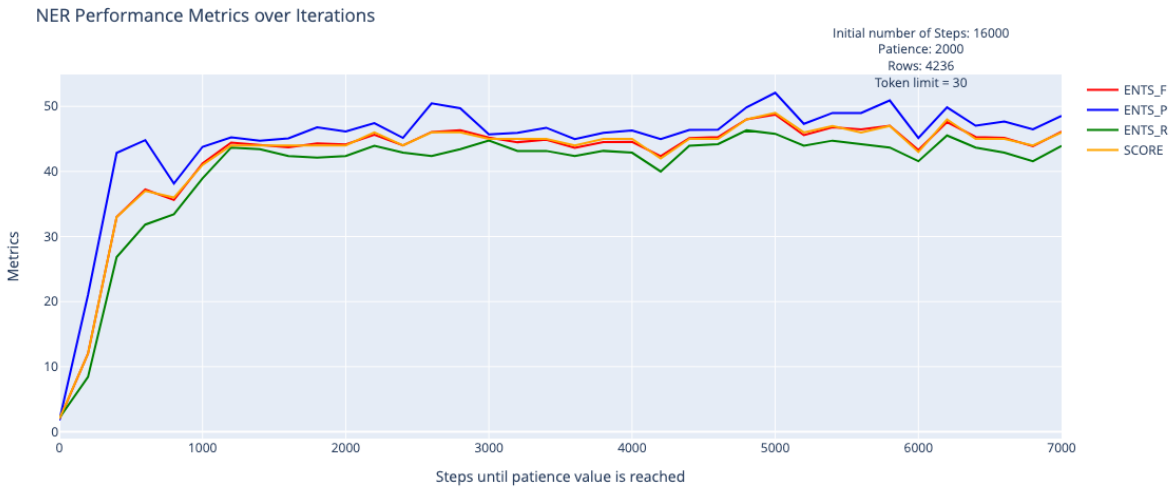

Figure 86: NER Performance Metrics, Steps: 16000, Patience: 2000, Rows: 4236, Token limit = 30

Table 23: Nervaluate metrics for Steps: 16000, Patience: 2000, Rows: 4236, Token limit = 30.

Metric	ent_type	partial	strict	exact
correct	283.0	79.0	79.0	79.0
incorrect	0.0	0.0	204.0	204.0
partial	0.0	204.0	0.0	0.0
missed	95.0	95.0	95.0	95.0
spurious	52.0	52.0	52.0	52.0
possible	378.0	378.0	378.0	378.0
actual	335.0	335.0	335.0	335.0
precision	0.84	0.54	0.24	0.24
recall	0.75	0.48	0.21	0.21
f1	0.79	0.51	0.22	0.22

Table 24: Steps: 20000, Patience: 3000, Rows: 4236, Token limit = 30

Epoch	#	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	1.91	1.75	2.11	2
0	200	16.16	20.32	13.42	16
0	400	35.04	42.64	29.74	35
1	600	38.5	44.91	33.68	38
1	800	36.21	39.87	33.16	36
2	1000	45.25	48.21	42.63	45
3	1200	46.64	46.7	46.58	47
4	1400	45.95	47.22	44.74	46
5	1600	44.93	46.24	43.68	45
7	1800	42.05	43.09	41.05	42
9	2000	47.32	50.91	44.21	47
11	2200	43.94	44.47	43.42	44
14	2400	43.54	43.65	43.42	44
17	2600	47.03	49.56	44.74	47
20	2800	48.25	49.45	47.11	48
23	3000	46.52	47.28	45.79	47
26	3200	46.86	49.85	44.21	47
29	3400	46.28	49.55	43.42	46
32	3600	44.63	46.82	42.63	45
35	3800	44.35	47.18	41.84	44
38	4000	47.89	49.58	46.32	48
41	4200	48.76	51.16	46.58	49
44	4400	45.65	47.19	44.21	46
47	4600	46.39	49.12	43.95	46
50	4800	46.17	48.01	44.47	46
53	5000	45.76	46.83	44.74	46
56	5200	46.59	48.31	45.0	47
59	5400	47.3	48.61	46.05	47
62	5600	45.06	47.79	42.63	45

Figure 87: NER Performance Metrics, Steps: 20000, Patience: 3000, Rows: 4236, Token limit = 30

The dataset with a stricter token limit of 30 records metrics similarly to the other datasets, but shows a further slight reduction in performance compared to the 45-token limit dataset. The SCORE ranges around 44%-47% at later stages. This stricter token limit exacerbates the loss of context, making it more likely for the model to miss important entity relationships that require a broader context. Consequently, the more stringent token limit significantly impacts the model's ability to capture and learn from longer sentences, leading to reduced performance metrics.

NER models benefit from more context, as entities and their relationships often span multiple tokens; however, imposing token limits truncates this context, hindering the model's ability to learn effectively. While token limits simplify model complexity and reduce computational resources, they can detrimentally affect performance, necessitating a careful trade-off between resource efficiency and model accuracy. The quality and size of the training data also play a crucial role, and ensuring that the data is clean and representative can mitigate some of the performance losses due to token limits. Potential solutions include employing techniques such as sliding windows or overlapping contexts to mitigate the loss of information, as well as fine-tuning hyperparameters and using advanced architectures that can handle longer contexts more efficiently.

SpaCy's config.cfg file used for our NER experiments:

```
[paths]
train = null
dev = null
vectors = "en_core_web_trf"
init_tok2vec = null

[system]
gpu_allocator = null
seed = 0

[nlp]
lang = "en"
pipeline = ["tok2vec","ner"]
batch_size = 1000
disabled = []
before_creation = null
after_creation = null
after_pipeline_creation = null
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}
vectors = {"@vectors":"spacy.Vectors.v1"}

[components]

[components.ner]
factory = "ner"
incorrect_spans_key = null
moves = null
scorer = {"@scorers":"spacy.ner_scorer.v1"}
update_with_oracle_cut_size = 100

[components.ner.model]
@architectures = "spacy.TransitionBasedParser.v2"
state_type = "ner"
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = true
nO = null

[components.ner.model.tok2vec]
@architectures = "spacy.Tok2VecListener.v1"
width = ${components.tok2vec.model.encode.width}
upstream = "*"

[components.tok2vec]
factory = "tok2vec"

[components.tok2vec.model]
@architectures = "spacy.Tok2Vec.v2"

[components.tok2vec.model.embed]
@architectures = "spacy.MultiHashEmbed.v2"
width = ${components.tok2vec.model.encode.width}
attrs = ["NORM","PREFIX","SUFFIX","SHAPE"]
rows = [5000,1000,2500,2500]
include_static_vectors = true

[components.tok2vec.model.encode]
@architectures = "spacy.MaxoutWindowEncoder.v2"
width = 256
depth = 8
window_size = 1
maxout_pieces = 3

[corpora]

[corpora.dev]
@readers = "spacy.Corpus.v1"
path = ${paths.dev}
max_length = 0
gold_preproc = false
limit = 0
augmenter = null

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
max_length = 0
gold_preproc = false
limit = 0
augmenter = null

[training]
dev_corpus = "corpora.dev"
train_corpus = "corpora.train"
seed = ${system.seed}
gpu_allocator = ${system.gpu_allocator}
dropout = 0.1
accumulate_gradient = 1
patience = 1600
max_epochs = 0
max_steps = 20000
eval_frequency = 200
frozen_components = []
annotating_components = []
before_to_disk = null
before_update = null

[training.batcher]
@batchers = "spacy.batch_by_words.v1"
```

```
discard_oversize = false
tolerance = 0.2
get_length = null

[training.batcher.size]
@schedules = "compounding.v1"
start = 100
stop = 1000
compound = 1.001
t = 0.0

[training.logger]
@loggers = "spacy.ConsoleLogger.v1"
progress_bar = false

[training.optimizer]
@optimizers = "Adam.v1"
beta1 = 0.9
beta2 = 0.999
L2_is_weight_decay = true
L2 = 0.01
grad_clip = 1.0
use_averages = false
eps = 0.00000001
learn_rate = 0.001

[training.score_weights]
ents_f = 1.0
ents_p = 0.0
ents_r = 0.0
ents_per_type = null

[pretraining]

[initialize]
vectors = ${paths.vectors}
init_tok2vec = ${paths.init_tok2vec}
vocab_data = null
lookups = null
before_init = null
after_init = null

[initialize.components]

[initialize.tokenizer]
```