universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Illuminating What Multilingual PLMs Know: A Probe via Linear Transformation

verfasst von | submitted by

Nastasia Mazur BA

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 587
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Joint-Masterstudium Multilingual Technologies
Betreut von | Supervisor:	Assoz. Prof. Mag. Dr. Dagmar Gromann BSc

Acknowledgements

This master's thesis would not have been possible without the support of remarkable people, to whom I owe my heartfelt appreciation.

First and foremost, I would like to express my deepest gratitude to my supervisor, Assoz. Prof. Mag. Dr. Dagmar Gromann, BSc, for her invaluable support, insightful feedback, and constant encouragement. Her expertise, patience, and dedication have greatly contributed to the development of this thesis, and I feel fortunate to have worked under her guidance. Her unwavering belief that I could find my way through the complexity of this research gave me the courage to keep going when the path seemed uncertain. Her advice and guidance have shaped not only this thesis but also my growth as a researcher. I would also like to acknowledge the dedication she has put into the Multilingual Technologies program, which has provided invaluable opportunities for its students to learn and to contribute to a vibrant research community.

My sincere thanks also go to Mr. Liad Magen, MSc, whose contagious enthusiasm, deep passion for teaching, and generosity in sharing his experience have been a lasting source of inspiration. He created an atmosphere of openness and trust that allowed us to explore, to make mistakes, and to never forget to be ourselves and stay creative. I will always be grateful for the confidence he helped me find and for the fresh perspectives he continually brought to his lectures.

I am thankful to my colleagues and friends in the Multilingual Technologies program, whose discussions, feedback, and companionship have made the long hours of research and writing far more rewarding. To MLT25, being your class representative was an unforgettable experience.

To my family, who may not always understand exactly what I do, but who have always known that it matters to me—and that has always been enough. Finally, to my partner, Taulant, I am thankful for his patience and love, for reminding me of balance and life beyond research, and of who I am beyond this work.

The experience of writing this thesis has taught me that knowledge is not created in isolation, but through the generous sharing of ideas, time, and support. This work is dedicated to all those who supported and inspired me along the way.

Abstract

Most research on factual knowledge retrieval has concentrated on models trained predominantly in English; however, investigations into multilingual settings remain comparatively limited. This study addresses this gap by investigating the representation of factual knowledge in multilingual Pre-trained Language Models (PLMs) through the extension of probing methodologies to a multilingual context. The exploration focuses on how such models encode factual information within their vector spaces, using cosine similarities computed between entity pairs from multilingual knowledge-graph embeddings as the reference, and applying linear transformation techniques to assess the preservation and alignment of factual associations across multiple languages. The contributions comprise an extension of probing-based methodologies to multilingual factual knowledge analysis, an evaluation of knowledge encoding in three languages, and a delineation of challenges and opportunities for improving multilingual knowledge representation. Moreover, the findings provide a deeper understanding of the multilingual PLMs' internal representation of factual knowledge, providing a foundation for research in multilingual Natural Language Processing (NLP).

Zusammenfassung

Die meisten Forschungsarbeiten zum Abruf faktischen Wissens konzentrieren sich auf Modelle, die überwiegend auf Englisch trainiert wurden, während Untersuchungen in mehrsprachigen Kontexten vergleichsweise selten sind. Diese Masterarbeit zielt darauf ab, diese Lücke zu schließen, indem sie die Repräsentation faktischen Wissens in mehrsprachigen vortrainierten Sprachmodellen (engl. Pre-trained Language Models, PLMs) untersucht und bestehende Probing-Methoden auf einen mehrsprachigen Kontext ausweitet. Der Fokus liegt darauf, wie solche Modelle faktische Informationen in ihren Vektorräumen kodieren, wobei Kosinus-Ähnlichkeiten zwischen Entitätspaaren aus mehrsprachigen Knowledge Graph Embeddings als Referenz herangezogen werden. Zur Analyse werden lineare Transformationen eingesetzt, um die Erhaltung und das Alignment faktischer Relationen über mehrere Sprachen hinweg zu bewerten. Die Beiträge dieser Arbeit umfassen die Erweiterung von Probing-basierten Ansätzen zur Analyse mehrsprachigen faktischen Wissens, die Evaluation der Wissenskodierung in drei Sprachen sowie die Darstellung von Herausforderungen und Potenzialen zur Weiterentwicklung der mehrsprachigen Wissensrepräsentation. Darüber hinaus geben die Ergebnisse einen tieferen Einblick in die interne Repräsentation faktischen Wissens in mehrsprachigen PLMs und bilden eine Grundlage für weiterführende Forschung im Bereich der mehrsprachigen natürlichen Sprachverarbeitung.

Contents

List of Tables

List of Figures

1 Introduction

Pre-trained Language Models (PLMs) have demonstrated remarkable capabilities in encoding not only linguistic patterns but also factual knowledge within their internal parameters, acquired through exposure to large-scale corpora during pretraining (Petroni et al., 2019; Roberts et al., 2020). Extensive research has focused on examining these models' ability to recall factual knowledge, primarily by probing tasks to assess their responses to specific prompts (Davison et al., 2019; Shin et al., 2020; Bouraoui et al., 2020; Brown et al., 2020b; Alghanmi et al., 2021; Peng et al., 2022). However, the internal representation of such factual knowledge remains opaque, distributed across high-dimensional vector spaces and not explicitly stored in interpretable forms (Jiang et al., 2020a; Bouraoui et al., 2020; Shin et al., 2020). Moreover, the majority of these investigations are confined to English PLMs, leaving the factual knowledge representations of multilingual PLMs comparatively underexamined (Kassner et al., 2021; Yin et al., 2022a). Additionally, existing evaluation approaches are largely output-based, focusing on the model's predictions rather than interrogating the underlying vector space representations in which this knowledge is embedded.

The recent work by Munda et al. (2025) introduced an alternative, representation-level probing method for English PLMs. By aligning PLM-based entity embeddings with Knowledge Graph Embeddings (KGEs) through a supervised linear transformation, their approach enabled the analysis of how well factual associations are preserved in the PLMs' latent space. While the method provides a promising framework for investigating factual knowledge, its scope remains limited, as it focuses on English monolingual models, relies on language agnostic knowledge graph embeddings, and does not account for multilingual variability in knowledge encoding.

To address the existing limitations this thesis extends the methodology propose by Munda et al. (2025) to the multilingual context. Specifically, the study investigates whether the hidden vector spaces of multilingual PLMs encode factual associations that are consistent with multilingual knowledge graphs. To construct the supervision signal for training these transformations, bilingual knowledge graph embeddings were obtained using MTransE (Chen et al., 2018), a translation-based model that supports multilingual entity representations. The cosine similarity scores between entities serve as ground truth for learning a linear transformation that projects PLM embeddings into a space that reflects factual proximity. Furthermore,

separate labeled datasets in English, German, and Russian were constructed using Wikidata5M (Wang et al., 2021) dataset, ensuring comparability across languages while preserving language-specific relational structure.

The scientific relevance of this study lies in its methodological shift from probing outputs to probing latent representations of multilingual PLMs. In contrast to prior research that evaluates factual knowledge through classification accuracy or masked language modeling, the present approach offers a direct measure of how well factual similarity is preserved in the embedding space. Moreover, by comparing multiple multilingual PLMs with differing transformer architectures, the study sheds light on architectural factors that influence cross-lingual factual encoding. Understanding whether and how these models preserve cross-lingual factual associations at the representation level is particularly relevant for applications in cross-lingual transfer, multilingual knowledge-intensive tasks, and language-specific factual retrieval.

Since multilingual PLMs are trained on multilingual corpora, they are inherently exposed to language-specific syntactic and semantic patterns as well as factual content in diverse linguistic contexts. This exposure makes them highly effective for downstream tasks involving multiple languages. However, it also raises important questions regarding the consistency and robustness of factual knowledge representations across languages. Understanding whether and how these models preserve cross-lingual factual associations at the representation level is essential for applications in cross-lingual transfer, multilingual knowledge-intensive tasks, and language-specific factual retrieval.

To address this gap, the present master's thesis extends a methodological framework designed to answer the following research question:

> *To what extent can applying vector space linear transformations to multilingual pre-trained language models enhance their ability to represent factual knowledge, as structured within multilingual knowledge graphs?*

The contribution of this study is threefold. First, it expands the study of factual knowledge in PLMs from an English-centric to a multilingual setting by constructing parallel datasets and evaluating models across linguistically diverse contexts. Second, it proposes a novel combination of multilingual KG embeddings and linear transformation training to probe internal knowledge representations without relying on prompt-based evaluation. Third, the study conducts a comparative analysis across three multilingual PLMs with different transformer architectures (encoder-only, decoder-only, and encoder-decoder), providing insight into how model design influences the encoding of factual knowledge across languages. By visualizing embedding spaces before and after transformation, and analyzing the behavior of selected subject-object pairs, the study also offers a qualitative perspective on how factual alignment manifests within the learned vector spaces.

The code and data underlying this study have been published on GitHub[1] to support transparency and reproducibility and encourage future research. Furthermore, the trained linear transformation models are available on Hugging Face[2].

[1] https://github.com/NastasiaMazur/Multilingual_KG_LinearTransformation.git
[2] https://huggingface.co/NastasiaM/

2 Theoretical Background

This chapter establishes the theoretical foundations required for the methodology presented in Chapter 4. Section 2.1 provides an overview of deep learning, beginning with the principal learning paradigms in machine learning and their defining characteristics. The discussion proceeds to the fundamental components of artificial neural networks before progressing to more advanced architectures, with particular emphasis on the Transformer model, which is foundational to many state-of-the-art natural language processing systems. Section 2.2 establishes the foundation for understanding vector-based representations of language, beginning with fundamental statistical approaches and extending to global word embeddings and contextual embeddings, as well as multilingual and cross-lingual embeddings, which enable semantic transfer across languages. Section 2.3 is devoted to language models, introducing the concept of language modeling and outlining its relevance to NLP. It proceeds from statistical approaches to contemporary neural architectures, and further addresses multilingual language models. Moreover, it presents explainability techniques, which provide insights into both the learned representations and model outputs. Section 2.4 further investigates knowledge graphs, beginning with methods for representing structured knowledge and moving to embedding techniques for graph nodes and relations. It covers both translational distance models and semantic matching models. Finally, Section 2.5 concludes with an with a formal definition of linear spaces, a fundamental concept in deep learning and modern NLP. It further introduces linear transformations and established methods for computing vector-based similarity.

2.1 Deep Learning

Deep Learning (DL), a subfield of Machine Learning (ML), builds upon artificial neural networks, computational architectures inspired by the interconnected neurons of the human brain, to automatically learn hierarchical feature representations from data. Unlike traditional machine learning methods, which often depend on handcrafted features requiring domain expertise and extensive pre-processing, deep learning models acquire these representations directly through multiple layers of interconnected processing units. The term “deep” denotes the presence of many hidden layers, enabling the progressive extraction of increasingly abstract and

complex patterns as information flows from the input to the output. This depth grants deep learning a significantly greater representational capacity than shallow architectures, which typically contain only one or two hidden layers. In the context of natural language processing, deep learning has been particularly impactful, as it allows models to automatically derive semantic, syntactic, and contextual features from large text corpora, thereby achieving state-of-the-art performance across a wide range of tasks. This section presents the main learning paradigms relevant to both machine learning and deep learning, ranging from classical supervised learning to the more recent paradigm of self-supervised learning. Additionally, it covers the fundamental principles of neural networks and discusses the Transformer architecture, which serves as a foundational framework for developing advanced Pre-trained Language Models (PLMs) in the field of Natural Language Processing (NLP).

2.1.1 Learning Paradigms in Machine Learning

Machine Learning provides a framework for addressing problems that are infeasible to solve through manually crafted rules or explicitly programmed instructions. Instead of relying on human-designed heuristics, machine learning systems learn patterns and generalizations from data, enabling them to handle complex and high-dimensional tasks that cannot be effectively tackled with traditional programming approaches (Goodfellow et al., 2016).

Approaches within the ML field vary in how they extract patterns from data, depending on the type and availability of supervision during training. The learning paradigm defines the structure of a machine learning task by specifying the relationship between inputs, outputs, and the type of supervision or feedback provided during training. It influences not only the methodological design of model training, but also which types of problems can be addressed and what assumptions are made about the data. Existing machine learning algorithms are generally classified into two main categories: supervised and unsupervised learning (Goodfellow et al., 2016).

Supervised learning is a paradigm in machine learning in which a model is trained on a labeled dataset consisting of input-output pairs. In this framework, each input is associated with a corresponding output label, and the model is trained to learn a function that maps inputs to outputs by adjusting its parameters to minimize the discrepancy between predicted and true labels. The training data typically takes the form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where x_i represents an input instance from the input space $\mathcal{X}$, and y_i is the corresponding label from the output space $\mathcal{Y}$. Each input x_i is commonly represented as a feature vector, where individual features capture relevant properties or characteristics of the data that are informative for predicting the associated output.

The goal of supervised learning is to approximate a function $f : \mathcal{X} \to \mathcal{Y}$ that generalizes well to unseen data. Often, this function is defined as a scoring function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and predictions are obtained by selecting the label that maximizes the score for a given input, as shown in Equation 1.

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} f(x, y). \tag{1}$$

During training, the model's parameters are iteratively adjusted to minimize a loss function, such as cross-entropy for classification or Mean Squared Error (MSE) for regression. Once training is complete, the learned function should generalize well to unseen data, a property typically assessed through validation or testing on held-out examples. However, models may suffer from underfitting or overfitting. Underfitting occurs when a model is too simplistic to capture the underlying data patterns, resulting in high error on both training and unseen examples. Overfitting, on the other hand, arises when a model adapts too closely to the training data, including noise or irrelevant details, leading to poor generalization. These issues can be mitigated through techniques, such as regularization, cross-validation, and early stopping. Supervised learning is widely used in applications ranging from image classification to regression-based forecasting, demonstrating its versatility and strong empirical performance .

Unsupervised learning offers an alternative paradigm in machine learning, particularly suitable when labeled data are scarce, prohibitively expensive, or impractical to obtain. Whereas supervised learning requires labeled input-output pairs, unsupervised learning aims to uncover hidden patterns from input data alone. For instance, in clustering tasks, the objective is to group a dataset $X = \{x_1, x_2, \ldots, x_n\}$ into k distinct clusters by constructing a clustering function $C : X \to \{1, 2, \ldots, k\}$. This function assigns each data point to a specific cluster based solely on similarity or dissimilarity metrics, thereby revealing inherent groupings without external labels (Eisenstein, 2018).

Another prominent unsupervised technique is dimensionality reduction. The aim is to project data from a high-dimensional space $\mathbb{R}^m$ into a lower-dimensional subspace $\mathbb{R}^d$, where $d < m$, in such a way that essential structural properties of the data are preserved (Kokiopoulou et al., 2011). Such a transformation not only simplifies the data structure but also preserves its intrinsic distribution and variance, making it more tractable for visualization and further analysis. Algorithms, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-distributed Stochastic Neighbor Embedding (t-SNE) are widely used for this purpose. These unsupervised methods facilitate the extraction of meaningful representations and latent structures from raw data, providing a foundation for subsequent tasks, such as self-supervised learning and other advanced machine learning applications.

Another category, referred to as **self-supervised learning**, can be regarded as a subclass of unsupervised learning, as it operates without reliance on manually annotated labels (Liu et al., 2021a). However, while unsupervised learning operates without labels, self-supervised learning relies on labels that are derived algorithmically from the input data, rather than being manually annotated by humans (Ohri and Kumar, 2021). Self-supervised learning represents a methodological approach that constructs labels from the data itself, thereby integrating principles of both supervised and unsupervised learning (Rani et al., 2023).

Mathematically, self-supervised learning involves optimizing a loss function $\mathcal{L}$ defined over pseudo-labels $\tilde{y}$ generated from the input data x, such that the model learns a function $f_\theta(x) \approx \tilde{y}$. The quality of the learned representations depends heavily on the design of the pretext task and its alignment with downstream objectives.

In NLP, self-supervised learning underlies the training of most large-scale language models. A prominent example is Masked Language Modeling (MLM), as employed in Bidirectional Encoder Representations from Transformers (BERT)(Devlin et al., 2019), where the model learns to predict randomly masked tokens in a sentence given their context. This encourages the model to encode syntactic and semantic relationships. In contrast, autoregressive objectives, such as those used in Generative Pre-trained Transformer (GPT)-style models, involve predicting the next token in a sequence, promoting temporally grounded representations.

While supervised, unsupervised, and self-supervised learning all rely on static datasets, another paradigm known as **Reinforcement Learning (RL)** follows a fundamentally different approach. Instead of learning from labeled examples or patterns in data, RL is grounded in principles from behavioral psychology and models learning as a process of trial and error (Sutton and Barto, 1998). An agent interacts with an environment and learns to make decisions by receiving feedback in the form of rewards or penalties.

In RL, the agent's behavior is shaped not by predefined labels but through dynamic interaction. Each action taken affects the future state of the environment, making the learning process sequential and state-dependent. This interaction forms a feedback loop between the agent and its environment, which distinguishes RL from the static data setting of other paradigms (Goodfellow et al., 2016). A central challenge in this process is the exploration–exploitation trade-off: the agent must balance using its current knowledge to maximize reward, referred to as exploitation, with trying new actions to gain additional information, known as exploration. This trade-off does not exist in supervised learning, where the correct output is always provided and no exploratory behavior is required (Goodfellow et al., 2016).

In NLP, reinforcement learning is particularly valuable in settings where direct supervision is either unavailable or insufficient. For instance, in dialogue systems, RL

enables models to optimize long-term conversational goals, such as user satisfaction or task completion (Li et al., 2016). More recently, Reinforcement Learning from Human Feedback (RLHF) has been used to align large language models with human preferences. Here, human ratings guide the learning process, helping to fine-tune models to produce more appropriate, safe, and helpful responses (Ouyang et al., 2022). In addition to alignment, reinforcement learning has also been explored as a means to improve factual reasoning in large language models. One relevant approach is Chain-of-Thought Reinforcement Learning (CoT-RL), which incorporates multi-step reasoning into the training objective by assigning rewards not only based on final outputs, but also on the quality and factual accuracy of intermediate reasoning steps. This method uses chain-of-thought prompts to structure the model's reasoning process, encouraging the generation of coherent and factually grounded inferences (Yao et al., 2023).

2.1.2 Artificial Neural Networks

Neural Networks (NNs), often referred to as Artificial Neural Networks (ANNs), represent a class of machine learning models that draw inspiration from the architecture and operational principles of biological neural systems(McCulloch and Pitts, 1943). ANNs are composed of individual processing units, called artificial neurons, which are organized into layers. Similar to biological neurons transmitting electrical impulses through synapses, neurons in an ANN transmit information to connected neurons through numerical transformations. Each connection between two neurons has an associated numerical value, referred to as a weight, which serves as a trainable parameter that is updated during the training.

Connections between neurons function by processing input values, either taken directly from the input data in the first layer or received from preceding neurons in hidden layers, scaling them by associated weights, and transmitting the result to the next neuron. In an ANN, each neuron typically receives inputs from multiple connected neurons, and its value is determined through two mathematical operations. First, a weighted sum of all incoming values is computed, often with an added trainable bias term, which allows the neuron to shift the activation threshold up or down independently of the input values. Second, this sum undergoes a non-linear transformation, known as the **activation function**, which defines the neuron's final output. Introducing non-linearity is necessary because it allows the network to approximate complex relationships between inputs and outputs.

A simple single-layer artificial neural network applies a linear transformation to the input vector, followed by a non-linear activation function, as shown in Equation 2.

$$\hat{y} = f\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} + b\right) \tag{2}$$

Here, $\mathbf{x} \in \mathbb{R}^n$ denotes the input vector, $\boldsymbol{\theta}$ is the learnable parameter space, b is a bias term, and $f(\cdot)$ represents the activation function (Goodfellow et al., 2016).

Early research on the perceptron (Rosenblatt, 1958), which extended the concept of the McCulloch-Pitts neuron (McCulloch and Pitts, 1943), revealed fundamental limitations of purely linear models, particularly their inability to solve problems that are not linearly separable, such as the XOR problem (Minsky and Papert, 1969). This finding underscored the importance of incorporating non-linear activation functions, which allow neural networks to approximate complex, non-linear relationships between input and output. Without non-linearities, a multi-layer network behaves equivalently to a single-layer model since linear transformations alone do not alter the representational space across layers, as the composition of linear functions remains linear. Therefore, it would not make a difference whether the network comprises one or many layers; no additional representational power would be gained. The inclusion of non-linear activation functions, such as the sigmoid, hyperbolic tangent (tanh), or Rectified Linear Unit (ReLU), is therefore essential. These functions enable the model to learn hierarchical and interconnected representations across layers, which is essential for the success of modern deep learning architectures.

The sigmoid function, shown in Equation 3, maps any real valued input to a range between 0 and 1, making it suitable for probabilistic interpretations.

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

The hyperbolic tangent (tanh) function, shown in Equation 4, maps the input to the interval from -1 to 1.

$$f(x) = \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}, \tag{4}$$

The Rectified Linear Unit (ReLU) function, given in Equation 5, applies a non-linear transformation by preserving positive inputs while setting negative inputs to zero.

$$f(x) = \mathrm{ReLU}(x) = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{if } x \leq 0 \end{cases} \tag{5}$$

Feedforward Neural Networks (FFNNs), also known as Multilayer Perceptrons (MLPs) when composed of multiple fully connected layers, constitute a fundamental class of neural architectures used to approximate mathematical mappings from input data to target outputs (Dong et al., 2014). FFNNs consist of a series of layers, each performing specific transformations on the data it receives

before passing the result to the next layer. The entire network can be described as a composition of functions, as shown in Equation 6.

$$f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right), \tag{6}$$

Here, each $f^{(n)}$ denotes a transformation performed by the n-th layer. The first layer processes the input x, while the final layer produces the network's output. Intermediate layers, known as hidden layers, determine the network's depth. Networks with two or more hidden layers are referred to as deep neural networks.

FFNNs are characterized by a strictly unidirectional flow of information during inference, where data propagates forward from input to output layers without any feedback connections. This distinguishes FFNNs from Recurrent Neural Networks (RNNs), which incorporate feedback loops to model sequential dependencies. While FFNNs lack recurrent connections, their parameters are updated during training using backpropagation, as explained in detail below.

Each hidden layer within an FFNN applies a linear transformation to its input, followed by a non-linear activation function. Mathematically, this transformation can be expressed as Equation 7.

$$a_n(x) = f\left(W_n a_{n-1}(x) + b_n\right) \tag{7}$$

In Equation 7, W_n is the weight matrix, b_n is the bias vector, and f is a non-linear activation function, such as ReLU or sigmoid. The output a_n serves as the input to the subsequent layer, allowing the network to gradually refine the data representation as it passes through the network.

The architecture and performance of deep feedforward networks depend on several hyperparameters, including the number of layers, the number of neurons per layer, the choice of activation functions, and the optimization algorithm (Goodfellow et al., 2016). During training, the network's parameters are iteratively adjusted to minimize a predefined loss function using backpropagation, which computes gradients of the loss with respect to the model parameters by applying the chain rule of calculus (Goodfellow et al., 2016). The process begins by computing the loss at the output layer, which quantifies the difference between the predicted and target values. Gradients of this loss are then recursively propagated through the layers, allowing each parameter to be updated in a direction that reduces the overall error.

Gradient-based optimization algorithms, such as Stochastic Gradient Descent (SGD), update parameters according to Equation 8, following the formulation presented in Jurafsky and Martin (2023).

$$\theta^{s+1} = \theta^s - \eta \frac{\partial\left[-\log p(w_t \mid w_{t-1}, \ldots, w_{t-n+1})\right]}{\partial\theta} \tag{8}$$

Here, θ^s denotes the parameter vector at optimization step s, and η is the learning rate, which modulates the step size during the update. The term inside the derivative corresponds to the negative log-likelihood of the target word w_t given its preceding context, as commonly used in neural language modeling. The gradient $\frac{\partial}{\partial\theta}$ quantifies the direction and magnitude of change required to reduce the prediction error. This formulation reflects the broader principle of maximum likelihood estimation, where parameters are optimized to assign higher probability to observed sequences.

Modern deep learning often employs adaptive optimizers, such as RMSprop (Shi et al., 2021) and Adam (Kingma and Ba, 2014). RMSprop adapts each parameter's learning rate by tracking the second moment of the gradient (Shi et al., 2021), while Adam additionally estimates both the first and second moments (Kingma and Ba, 2014). These adaptive techniques enhance convergence stability, especially in deep networks.

Recurrent Neural Networks (RNNs) are a type of deep neural networks commonly used for processing sequential data (Goodfellow et al., 2016). At each time step t, the network receives the current input x_t and updates its hidden state $\mathbf{h}_{\langle t\rangle}$ based on both x_t and the previous hidden state $\mathbf{h}_{\langle t-1\rangle}$. This recurrent structure enables the network to maintain a form of memory over previous time steps, allowing it to capture temporal dependencies within sequences. The update is obtained using a non-linear activation function, as shown in Equation 9.

$$\mathbf{h}_{\langle t\rangle} = f\left(\mathbf{h}_{\langle t-1\rangle}, x_t\right) \tag{9}$$

A fundamental difference between RNNs and feedforward neural networks lies in how information flows through the network. In feedforward networks, data passes through layers in a strictly forward manner, without feedback loops. In contrast, RNNs introduce cyclic connections, feeding information from earlier time steps back into the network. This enables them to process variable-length sequences and capture temporal dependencies over time.

Despite their ability to model sequential data, RNNs are known to suffer from the vanishing gradient and exploding gradient problems during training (Bengio et al., 1994). These issues arise when gradients, computed using algorithms, such as Backpropagation Through Time (BPTT) (Werbos, 1990), either decay exponentially towards zero, as they flow across long sequences, or grow uncontrollably large. When gradients vanish, the weight updates become negligibly small, preventing the network from effectively learning long-range dependencies, while exploding gradients cause excessively large updates, disrupting the optimization process and preventing the network from reaching a stable set of parameters. To address these limitations, more advanced architectures, such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRUs)

(Cho et al., 2014), were introduced, incorporating gating mechanisms to better regulate the flow of information and improve gradient stability.

Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks extend traditional RNNs by incorporating a gating mechanism that helps mitigate the vanishing gradient problem. Each LSTM cell contains an input gate, forget gate, and output gate, which jointly regulate the flow of information into, within, and out of the cell state. The input gate controls the addition of new information, the forget gate modulates the removal of irrelevant content, and the output gate determines which information influences the next hidden state. This structure allows LSTMs to maintain long-range dependencies while preserving stable gradients during backpropagation, making them particularly effective for learning from long sequences. As a result, LSTMs have been widely adopted in tasks, such as language modeling, speech recognition, and machine translation.

Gated Recurrent Units (GRUs) (Cho et al., 2014) represent a simplified version of LSTM networks, designed to capture long-term dependencies while reducing computational complexity. Unlike LSTMs, GRUs combine the forget and input gates into a single update gate and replace the cell state and output gate with a reset gate. Specifically, the update gate controls how much previous information should be retained or updated, while the reset gate modulates the extent to which past hidden states influence the current hidden state. By reducing the gating mechanisms from three to two and eliminating the separate cell state, GRUs achieve comparable performance to LSTMs with fewer parameters, thus enhancing computational efficiency and accelerating training convergence. Consequently, GRUs have become prominent alternatives to LSTMs for sequence modeling tasks where computational resources are limited, but strong performance is still required.

Although recurrent architectures capture temporal dependencies in sequential data, they can present limitations when parallelization is required. **Convolutional Neural Networks (CNNs)** (Lecun et al., 1998) offer an alternative for modeling local dependencies through hierarchical feature extraction. Originally developed for computer vision (Lecun et al., 1998), CNNs have also been successfully applied to sequence modeling tasks, including text classification (Kim, 2014) and machine translation (Gehring et al., 2017).

CNNs process structured inputs by applying filters, also known as kernels, over fixed-size subsequences of the input. These filters slide over the input, identifying local patterns, such as edges in images or n-gram-like structures in text. Unlike fully connected networks, where each unit is connected to the entire input, CNNs use parameter sharing, applying the same filter at every position. This allows the model to recognize the same pattern regardless of its position, while also reducing the number of trainable parameters. As a result, CNNs are more efficient and tend to generalize better than fully connected architectures.

The core operation of a CNN is the convolution, which for one-dimensional input can be expressed as Equation 10.

$$y_i = \sum_{j=1}^{k} w_j x_{i+j-1} + b \tag{10}$$

Here, x is the input, w is a filter of size k, b is a bias term, and y_i is the filter response at position i. Multiple filters are typically applied, each detecting different types of patterns, and the resulting feature maps capture the presence of these patterns across the input. To enable the model to capture complex patterns, each convolutional layer is followed by a non-linear activation function, such as ReLU, which introduces non-linearity into the network. CNNs often also apply pooling layers, which aggregate information across nearby positions, reducing the spatial dimensions of the feature maps while retaining essential information.

Although CNNs are less effective at capturing long-range dependencies directly, stacking multiple convolutional layers increases the receptive field and allows the model to incorporate broader contextual information. Nevertheless, CNNs are limited in their ability to model complex alignments between input and output sequences. To address this limitation, architectures that explicitly separate the encoding and decoding stages have been proposed to better handle variable-length sequences and structural mismatches between source and target representations.

However, CNNs are not well-suited for tasks that require modeling complex alignments between input and output sequences. To address this limitation, architectures that explicitly separate the encoding and decoding stages have been proposed to better handle variable-length sequences and structural mismatches between source and target representations.

Encoder-decoder models provide a flexible framework for modeling tasks that involve transforming one sequence into another, even when the input and output differ in sentence length (Sutskever et al., 2014; Bahdanau et al., 2015) or linguistic properties (Johnson et al., 2017). This architectural design is particularly well-suited for Sequence-to-Sequence (Seq2Seq) tasks, such as machine translation, summarization, and dialogue generation, where the goal is to map an input sequence to an output sequence.

The encoder processes the input sequence, often segmented into subword units using techniques, such as Byte Pair Encoding (BPE) (Sennrich et al., 2016) or WordPiece (Wu et al., 2016), and transforms it into a sequence of context-dependent embeddings that encode the structure and meaning of the input. These vectors form a contextual representation that encodes both the content and dependencies present in the input.

The decoder then generates the output sequence in an autoregressive manner (Sutskever et al., 2014), producing one token at a time. Each token is conditioned on

both the previously generated tokens and the encoder's contextual representations. This stepwise generation allows the decoder to dynamically produce sequences of varying lengths while ensuring that each token reflects both the global context provided by the encoder and the local dependencies within the output itself. The encoder-decoder architecture thus forms the foundation for many modern approaches to sequence transformation tasks in NLP.

2.1.3 Transformer Architecture

The originally proposed introduction of the Transformer architecture (Vaswani et al., 2017) had a significant impact on the field of speech and language processing (Jurafsky and Martin, 2023), fundamentally reshaping the design and training of language models (Guimarães et al., 2024). Initially developed for sequence-to-sequence tasks, the Transformer was designed to mitigate the challenges RNNs and CNNs encounter in representing global sequence structure. While architectures, such as LSTMs and GRUs, improved upon the shortcomings of the simple RNNs, they remained constrained by the inherently sequential nature of their processing. In contrast, the Transformer introduced self-attention mechanisms that model dependencies between all tokens in a sequence at once. This design enables parallel processing during training and reduces computational costs significantly.

Encoder-Decoder Structure. The Transformer consists of two primary components: the encoder and the decoder. The encoder converts input sequences into a continuous representation, often called the hidden state or context, which is subsequently used by the decoder to generate an output sequence autoregressively (Sutskever et al., 2014). The original Transformer model consists of a stack of six identical encoder layers and a stack of six identical decoder layers (Vaswani et al., 2017), although later adaptations have modified this depth (Raffel et al., 2020). Each encoder layer consists of a self-attention mechanism, a position-wise feedforward network, and residual connections followed by layer normalization. Similarly, each decoder layer contains an additional encoder-decoder attention mechanism to focus on relevant encoder outputs during generation. When only the encoder component is used, the model learns contextualized representations of input sequences, making it effective for tasks, such as text classification and named entity recognition (Devlin et al., 2019), where a global understanding of the input is essential. Alternatively, using only the decoder component enables the model to generate sequences autoregressively, predicting each token conditioned on previously generated tokens, which is particularly advantageous for language modeling, creative text generation, and open-ended dialogue systems (Radford et al., 2018).

Self-Attention and Multi-Head Attention. One of the fundamental components of the Transformer architecture that enables parallel processing of sequences is

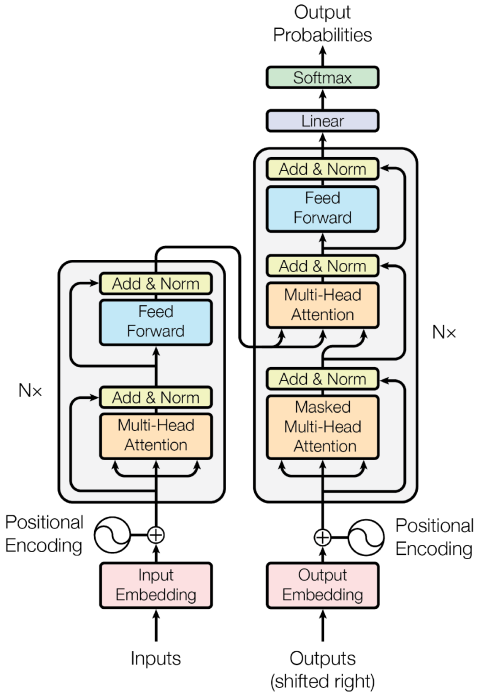


Figure 1: The Transformer architecture with stacked encoder and decoder layers, multi-head attention, and position-wise feedforward networks. (Vaswani et al., 2017, 3)

the use of self-attention mechanisms. The attention mechanism (Bahdanau et al., 2015) allows each token in an input sequence to attend to other tokens, enabling the model to capture context-dependent relationships by computing interactions between all token pairs. This process is designed to weigh the importance of different tokens dynamically, rather than relying on fixed-size contexts or sequential processing.

A type of attention, known as self-attention, is employed when the same sequence is used to compute all inputs—queries, keys, and values (Vaswani et al., 2017). For each input token, three distinct vectors are derived: the query (Q), the key (K), and the value (V). These vectors are obtained through learned linear transformations applied to the input embeddings. Formally, given an input matrix $X \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the model's hidden dimensionality, the following projections are computed, as shown in Equation 11.

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \tag{11}$$

Here, W^Q, W^K, and $W^V \in \mathbb{R}^{d \times d_k}$ are learned parameter matrices. Each row of Q, K, and V corresponds to a transformed representation of an input token.

The goal is to determine, for each query, which keys to attend to and how much attention to assign.

Scaled dot-product attention. The scaled dot-product attention mechanism operates by first computing the similarity between each query and all keys using the dot product. To prevent excessively large values that can lead to vanishing gradients in the softmax function, the result is scaled by the square root of the key dimensionality, $\sqrt{d_k}$. This operation is defined in Equation 12.

$$\text{score}(q_i, k_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}} \tag{12}$$

The resulting scores indicate the extent to which a token should attend to others. To convert the raw scores into a probability distribution, the softmax function is applied. This function normalizes the scores across all tokens, producing attention weights that are non-negative and sum to one, as shown in Equation 13.

$$\alpha_{ij} = \frac{\exp(q_i \cdot k_j / \sqrt{d_k})}{\sum_{j'} \exp(q_i \cdot k_{j'} / \sqrt{d_k})} \tag{13}$$

Once normalized, the attention weights are used to compute a weighted average over the value vectors. This step is formally defined in Equation 14.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \tag{14}$$

The result is a matrix of output representations, where each row corresponds to a token that has attended to the rest of the sequence according to the attention weights. This process allows each token to encode both its own identity and relevant contextual information from other tokens.

To enhance the representational capacity of the model, multi-headed attention is employed. Rather than relying on a single attention computation, the model constructs multiple attention heads, each with its own set of projections for Q, K, and V. Each head independently captures different types of relations among tokens. This process is defined in Equation 15.

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{15}$$

Each projection matrix W_i^Q, W_i^K, and W_i^V is distinct and trainable. The outputs from all attention heads are first concatenated and then passed through a final linear transformation, as shown in Equation 16.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O \tag{16}$$

Here, $W^O \in \mathbb{R}^{hd_k \times d}$ is another learned parameter matrix. This structure enables the model to capture multiple, complementary attention patterns in parallel, with each head potentially focusing on different linguistic features and mapping them to distinct subspaces of the output representation (Stahlberg, 2020).

Given that the Transformer operates on entire sequences in parallel and does not employ recurrent or convolutional mechanisms, it lacks an inherent mechanism for encoding the sequential order of tokens. To address the absence of order information in the attention mechanism, Vaswani et al. (2017) introduced **positional encodings**. These are continuous vectors that are combined with the input embeddings to provide the model with information about token positions. The Transformer employs sinusoidal positional encodings, defined in Equations 17 and 18.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{17}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{18}$$

Here, *pos* represents the token position and i denotes the dimension index. These encodings allow the model to generalize to sequences of varying lengths without requiring explicit position embeddings learned from data.

Each Transformer layer also includes a **Position-wise Feedforward Network**, which applies the same transformation independently to each position in the sequence. This component comprises two successive linear transformations, separated by a non-linear activation function. The first transformation expands the dimensionality of the input, allowing the model to learn richer intermediate representations. The second transformation reduces the dimensionality back to the model's original hidden size, ensuring compatibility with subsequent layers. The standard formulation is given in Equation 19.

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{19}$$

Here, W_1 and W_2 are learned weight matrices, and b_1 and b_2 are bias terms. This component allows the network to learn complex transformations beyond what self-attention provides.

Dimensionality and Projection Matrices. To ensure consistent dimensionality throughout the network, the Transformer employs projection matrices, including W_O, which is applied to reshape the output of the attention head. The input to attention x_i and the output from attention a_i both have the same dimensionality $[1 \times d]$. The model dimensionality d is maintained across all Transformer blocks, including intermediate vectors, which allows the network to remain modular. The query and key vectors both have a dimensionality of $[1 \times d_k]$, enabling the computation of dot-product attention to produce a scalar value. Meanwhile, the value

vectors have a separate dimension d_v. The transformation matrices W_Q, W_K, and W_V have shapes $[d \times d_k]$, $[d \times d_k]$, and $[d \times d_v]$ respectively, and the resulting output of each attention head has a shape of $[1 \times d_v]$. To ensure the output shape is consistent with the input, the projection matrix W_O has a shape of $[d_v \times d]$. In the original Transformer model, d was set to 512, while both d_k and d_v were set to 64.

2.2 Vector Representations for Language

Machine learning algorithms are unable to understand text in the same way humans do, as they require numerical representations to effectively process human language. To enable machines to understand and work with text, various methods have been developed to transform linguistic data into numerical representations, also called embeddings, which will be discussed in this chapter.

The need for vector representations arose in response to the limitations of earlier methods, such as rule-based systems and symbolic meaning representations, which struggled to manage the vast complexity and inherent ambiguity of natural language. In contrast, vector representations offer a more flexible and powerful solution. By representing words as high-dimensional vectors, their meanings can be captured based on their relationships with other words in large corpora.

2.2.1 Count-based Word Representations

Count-based models, also known as distributional models, represent some of the earliest techniques for generating word representations. These models build on the assumption that words that appear in similar contexts tend to share similar meanings (Firth, 1957). In a count-based framework, word representations are derived from the frequency of co-occurrences of words within a specified context window, typically in large corpora of text. While numerous methods are classified as count-based word representations, this section focuses on the Bag-of-Words (BoW) model and Term Frequency-Inverse Document Frequency (TF-IDF). Other count-based methods, such as Latent Semantic Analysis (LSA), introduced by Deerwester et al. (1990), and Latent Dirichlet Allocation (LDA), proposed by Blei et al. (2003), also belong to this category, relying on different principles to derive word representations. However, these methods go beyond the scope of this thesis and are therefore not discussed in detail.

The Bag-of-Words (BoW) model is a method that represents a document as an unordered collection, a so-called bag of words, disregarding word order and syntax. In BoW, a vocabulary is built from all unique words in a corpus, and each document is represented, where each element corresponds to the frequency count of a specific word from the vocabulary in that document. The BoW is effective

for tasks where syntactic or semantic understanding is not essential. However, it ignores word order, which limits its ability to capture syntactic structures and the relationships between words. This lack of structure makes it unsuitable for complex tasks, such as word sense disambiguation. Additionally, BoW leads to sparse, high-dimensional vectors when documents contain only a small subset of vocabulary, resulting in computational inefficiencies, especially with large corpora.

Term Frequency-Inverse Document Frequency (TF-IDF) measures how significant a word is in a document by considering both its frequency in that document and its rarity across the corpus. It enhances the BoW model by giving more weight to terms that are frequent within a document but rare across the corpus. TF-IDF consists of two parts: Term Frequency (TF) and Inverse Document Frequency (IDF).

TF is calculated by taking the raw count of a word $f_{t,d}$ in a document d, and normalizing it by the total count of words in the document as shown in Equation 20.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{20}$$

IDF assesses the importance of a word across all documents. It down-weights terms that appear frequently across many documents and assigns higher weights to terms that are rare. IDF is computed as shown in Equation 21

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}, \tag{21}$$

where N is the total number of documents, and $|\{d \in D : t \in d\}|$ represents the number of documents containing the term t.

The final **TF-IDF** score is obtained by multiplying the TF and IDF values:

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \tag{22}$$

The method gives more importance to terms that appear frequently in a document but are less common across the entire corpus.

2.2.2 Global Word Embeddings

Global word embeddings represent a significant advancement of traditional models, such as BoW and TF-IDF. Unlike these earlier methods, which rely on sparse representations, **word2vec**, introduced by Mikolov et al. (2013), produces dense, continuous vector representations that capture nuanced semantic relationships based on word contexts. The model learns word representations by predicting a word from its surrounding context or vice versa within a fixed-size sliding window.

Mikolov et al. (2013) proposed two model architectures to compute continuous word vector representations, which are Continuous Bag-of-Words (CBOW) and skip-gram.

The **Continuous Bag-of-Words (CBOW)** model predicts a target word based on its surrounding context words within a fixed-size window around the target word. In this approach, each word in the context window is initially represented as a one-hot vector, a sparse vector of dimensionality equal to the vocabulary size $|V|$, with a single active entry corresponding to the index of the given word. Each one-hot vector is then projected into a lower-dimensional embedding space. The resulting embeddings are averaged and passed through a linear transformation followed by a softmax layer, which predicts the target word, as shown in Figure 2.

$$\mathcal{L}_{CBOW} = -\sum_{t=1}^{T} \log\ P(w_t \mid C_t) \tag{23}$$

In Equation 23 T represents the total number of words in the corpus, w_t is the predicted word, and C_t is the context window around w_t. The probability $P(w_t|C_t)$ represents the likelihood of predicting the target word w_t given its context words C_t.

The CBOW model is computationally efficient, particularly when working with large corpora that contain many high-frequency words. By summarizing the context into a single vector representation, CBOW can process frequent words quickly, making it well-suited for tasks where common words dominate. However, its ability to capture the semantics of rare words may be constrained. While CBOW excels at modeling general patterns in the data, it may struggle with representing specialized or less frequent vocabulary.

The **skip-gram** model, in contrast to CBOW, reverses the prediction task by using a target word to predict its surrounding context words, as allustrated in Figure 2. Given a target word w_t, the model predicts the context words w_{t+j} within a specified window size m. The objective is to minimize the negative log-likelihood of the context words, as expressed in Equation 24.

$$\mathcal{L}_{Skip-gram} = -\sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log\ P(w_{t+j} \mid w_t) \tag{24}$$

Here, T is the total number of words in the corpus and $P(w_{t+j} \mid w_t)$ represents the conditional probability of the context word w_{t+j} given the target word w_t. The probability $P(w_{t+j} \mid w_t)$ is modeled using the softmax function as shown in Equation 25.

$$P(o \mid c) = \frac{\exp\ (\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{w \in V} \exp\ (u_w^T \mathbf{v}_c)} \tag{25}$$

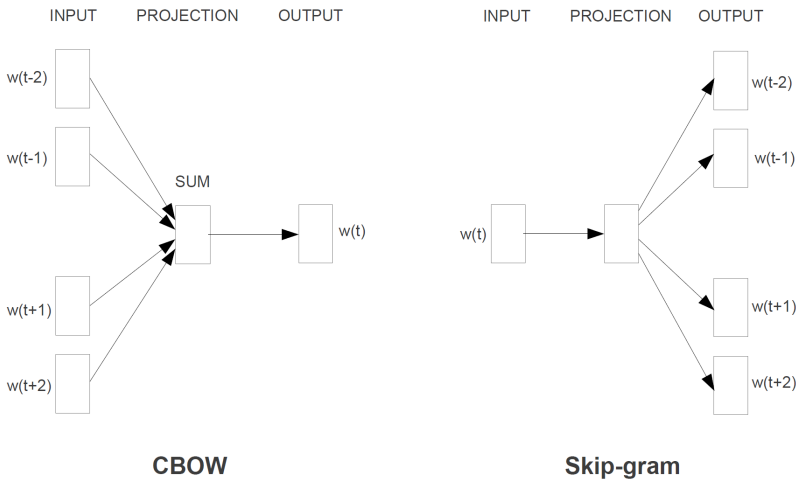


Figure 2: The CBoW model (left) predicts a target word from its surrounding context, and the skip-gram model (right) predicts surrounding context words given a target word. (Mikolov et al., 2013, 5)

In Equation 25, $\mathbf{v}_c$ is the vector for the context word c, and $\mathbf{u}_o$ represents the vector for the target word o. V denotes the entire vocabulary of the model, i.e., the set of all words present in the training corpus. The softmax function normalizes the similarity between the target and context word vectors over the entire vocabulary, ensuring that the computed values sum to one, forming a valid probability distribution.

By training on a large corpus, the skip-gram model learns to map words to vector representations that capture semantic relationships between words. The primary advantage of this approach is its ability to model relationships for rare and infrequent words. However, this model requires more computational resources compared to CBOW due to the larger number of predictions required for each word, leading to slower training times.

FastText (Bojanowski et al., 2017) extends the word2vec model by representing words as collections of character n-grams rather than individual words. This allows fastText to generate more effective embeddings for rare or out-of-vocabulary words, as it can break down words into subword units, capturing morphological features. FastText can also learn embeddings for words that were not present in the training data by constructing embeddings for their subword components. Its ability to handle out-of-vocabulary words makes it a valuable tool for multilingual and low-resource settings.

Another prediction-based model is **GloVe** (Pennington et al., 2014), which is

designed to learn word vectors using global statistical information derived from a corpus. The core idea is that the frequency with which words appear together in a given context provides meaningful semantic relationships. In contrast to local context-based models, such as word2vec, which predict surrounding words from a target word, GloVe uses global co-occurrence statistics to learn word representations that capture the underlying structure of the language.

The GloVe model creates a word co-occurrence matrix, in which each element X_{ij} represents the frequency with which word i co-occurs with word j. To reduce the dimensionality of this matrix, GloVe uses a factorization approach, learning two low-dimensional vectors for each word: one for the word itself and another for the context in which it appears. These vectors are learned by minimizing a cost function that attempts to approximate the log of the co-occurrence probabilities.

The objective function of the GloVe model is detailed in Equation 26.

$$\mathcal{L}_{GloVe} = \sum_{i,j=1}^{V} f(X_{ij}) \left(\mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \tag{26}$$

Here, V represents the vocabulary size, $\mathbf{w}_i$ is the vector for word i, and $\tilde{\mathbf{w}}_j$ is the vector for context word j. b_i and $\tilde{b}_j$ are bias terms for the word and context word, respectively. X_{ij} is the co-occurrence count between words i and j, and $f(X_{ij})$ is a weighting function that adjusts the importance of each co-occurrence count. This function is often defined as shown in Equation 27.

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{27}$$

In Equation 27, $x_{\max}$ is a threshold that limits the influence of very frequent co-occurrences, preventing them from dominating the learning process, and α is an exponent that controls how the weighting function scales. The values $x_{\max} = 100$ and $\alpha = 3/4$ were empirically chosen to balance rare and frequent co-occurrences (Pennington et al., 2014).

The objective function aims to minimize the difference between the predicted and actual log-transformed co-occurrence counts, allowing the model to learn meaningful, dense word vectors. Thus, GloVe provides a way of learning word embeddings that incorporate both local context information and global corpus-wide statistics, producing more robust word representations compared to purely context-based models.

2.2.3 Contextual Word Embeddings

Global word embeddings capture semantic relationships by representing words as static vectors, derived from their co-occurrence patterns within large corpora.

However, these models are constrained by their inability to distinguish between different meanings of a word depending on the context. For instance, the word *bat* would be represented by the same vector whether it refers to a flying mammal or a piece of sports equipment. To address this issue, contextual word embeddings were introduced. These models rely on more sophisticated architectures and can dynamically adjust the representation of words based on the surrounding text. This allows the model to capture subtle differences in meaning based on context, providing more accurate and nuanced word representations that are crucial for complex NLP tasks.

Context2vec (Melamud et al., 2016) is an unsupervised neural model designed to learn task-independent, contextual embeddings for words based on the surrounding text. Unlike word2vec's CBOW architecture (Mikolov et al., 2013), which represents context as the simple average of word embeddings in a fixed window, context2vec uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to model context more effectively. The model processes the context from both left-to-right and right-to-left directions, concatenating the outputs of two separate LSTM networks and enabling to capture long-range dependencies and the richer structure of a sentence. The model learns embeddings by minimizing a negative sampling objective, similar to word2vec, but with the entire surrounding text rather than a fixed window of words.

Embeddings from Language Models (ELMo), introduced by Peters et al. (2018a), uses a bidirectional Long-Short Term Memory (LSTM) network, which is trained using a method where two language models work together, learning word embeddings from both the left and right context. Unlike context2vec, which concatenates the outputs of two separate LSTMs, ELMo combines the representations from multiple layers of the bidirectional LSTM, and computes a weighted linear combination of these layers to create task-specific embeddings. ELMo representations capture rich, hierarchical context dependencies by incorporating all internal LSTM states, improving performance across a range of NLP tasks, such as question answering, coreference resolution and sentiment analysis (Peters et al., 2018a).

The model jointly maximizes the log-likelihood for both the forward and backward directions using Equation 28:

$$\begin{aligned}\mathcal{L}_{ELMo} = \sum_{k=1}^{N} (&\log p\left(t_k \mid t_1, \ldots, t_{k-1}; \theta_x, -\theta_{LSTM}, \theta_s\right) \\ &+ \log p\left(t_k \mid t_{k+1}, \ldots, t_N; \theta_x, \theta_{LSTM}, \theta_s\right))\end{aligned} \tag{28}$$

Here, $p\left(t_k \mid t_1, \ldots, t_{k-1}; \theta_x, -\theta_{LSTM}, \theta_s\right)$ represents the probability of predicting token t_k given its left context, while $p\left(t_k \mid t_{k+1}, \ldots, t_N; \theta_x, \theta_{LSTM}, \theta_s\right)$ represents the probability of predicting t_k given its right context. The parameters for both the token representation θ_x and the softmax layer θ_s are tied across both directions,

but the LSTM parameters for the forward and backward directions remain separate. This structure enables the learning of contextualized embeddings that integrate information from both preceding and subsequent tokens for each word.

The **Transformer** architecture (Vaswani et al., 2017) represents a significant advancement over earlier models, such as LSTMs, for learning contextual embeddings. Unlike LSTMs, which require sequential processing and are computationally expensive to train, Transformers rely on a **self-attention** mechanism that allows for parallelization, significantly improving training efficiency, especially on large datasets. At the core of the Transformer lies the self-attention mechanism, which dynamically adjusts token representations based on their relationships to all other tokens in the sequence. Each token is first mapped to three vectors: a query, a key, and a value. The attention mechanism then computes the relevance of each token to every other token by taking the dot product between queries and keys. The resulting scores are then normalized using a softmax function, converting them into attention weights that indicate the importance of each token in the sequence. These scores are used to weight the corresponding value vectors, generating a refined representation that incorporates contextual information from the entire sequence. By stacking multiple self-attention layers, Transformers iteratively refine token embeddings, enabling them to capture complex dependencies across sentences. As a result, the final contextual embedding of each token captures its relationship to every other token in the input, making it more flexible and adaptable to different contexts compared to static word embeddings.

The **Bidirectional Encoder Representations from Transformers (BERT)** model (Devlin et al., 2019) generates contextual embeddings by applying a bidirectional Transformer architecture, allowing it to learn deep, context-sensitive representations for each token in a sequence. Unlike traditional models, BERT is pre-trained using a Masked Language Model (MLM) method where a proportion of the input tokens is masked, and the model is required to predict the original values of these hidden tokens based solely on their surrounding context. This approach enables BERT to incorporate both left and right context for each token, overcoming the unidirectional constraints of previous models. Additionally, BERT is pre-trained with a Next Sentence Prediction (NSP) task, which trains the model to predict whether one sentence logically follows another. This helps the model learn relationships between sentence pairs, improving its ability to handle tasks that require understanding sentence-level dependencies, such as question answering and natural language inference. BERT's bidirectional nature, combined with these pre-training objectives, enables it to generate highly nuanced contextual embeddings that capture richer dependencies across both individual tokens and sentence pairs. However, the NSP task was omitted in later models building on BERT, such as RoBERTa (Liu et al., 2021b) and XLM-RoBERTa (Conneau et al., 2020), since

removing it matched or even outperformed performance on downstream tasks.

Generative Pre-trained Transformer (GPT) models (Radford et al., 2018) are autoregressive language models that predict the next token in a sequence given the previous ones. The GPT architecture is based on the Transformer decoder, which consists of self-attention mechanisms and feed-forward neural networks. These models are trained on large corpora using a causal (unidirectional) language modeling objective, where each token's representation is computed by attending to all preceding tokens in the sequence. Unlike bidirectional models, such as BERT, which consider both the left and right context, GPT models only rely on past tokens for context. This makes GPT models particularly strong in generative tasks, such as text completion, text generation, and dialogue systems, where the model must produce coherent and contextually appropriate text based on previous inputs. The contextual embeddings from GPT models are learned at each token's position in the sequence, reflecting the relationship to all preceding tokens.

Large Language Model Meta AI (LLaMA) models (Touvron et al., 2023), are based on the Transformer architecture and designed for efficient large-scale training. These models combine both autoregressive and bidirectional learning, offering greater versatility than models, such as GPT and BERT. LLaMA is trained with an MLM task, similar to BERT, and the model predicts masked tokens based on surrounding context, allowing it to capture bidirectional relationships. Simultaneously, LLaMA also learns autoregressively, such as GPT, by predicting the next token in a sequence given previous tokens. This hybrid approach enables LLaMA models to excel at both understanding tasks, requiring bidirectional context, and generation tasks, such as text completion or generation.

2.2.4 Multilingual and Cross-lingual Word Embeddings

Multilingual word embeddings refer to numerical representations learned for words in different languages, without any explicit mapping or association between the words across the languages. In these models, word embeddings are trained independently for each language, typically in monolingual vector spaces, with no consideration of how words in different languages may correspond to one another. This means that while the embedding spaces are structured for each language, there is no cross-lingual alignment, and words with similar meanings in different languages may not be represented by similar vectors. Consequently, these embeddings cannot be directly used for tasks, such as machine translation or cross-lingual information retrieval, as the lack of alignment hampers the ability to compare or transfer information across languages.

Multilingual embeddings are often learned by training separate monolingual models for each language, such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2017), or by using large-scale multilin-

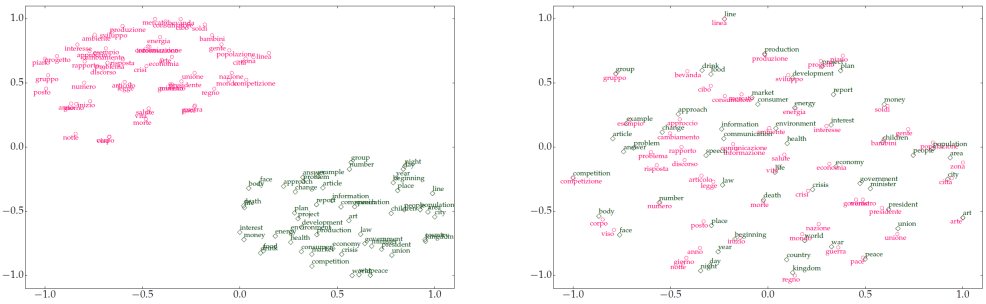


Figure 3: Visual representation of unaligned monolingual word embeddings (left) and their projection into a shared cross-lingual embedding space (right). (Ruder et al., 2019, 570)

gual Pre-trained Language Models (PLMs), such as mBERT (Devlin et al., 2019) or XLM-RoBERTa (Conneau et al., 2020). However, since these embeddings are language-specific, any tasks that require cross-lingual understanding or comparison usually require additional techniques for alignment between the languages.

Following the skip-gram approach (Mikolov et al., 2013), Luong et al. (2015) introduced the **Bilingual Skip-Gram (BiSkip)** model to learn cross-lingual word embeddings. This model extends the monolingual skip-gram by introducing a mechanism that ties two monolingual embedding spaces together. The model requires word alignments across languages. For a word w_1 in language l_1 and its aligned counterpart w_2 in language l_2, BiSkip predicts the neighbors of w_2 using w_1, and vice versa. This approach not only aligns embeddings across languages efficiently, offering the advantage of fast training time, but also preserves the clustering structures of words within each language (Luong et al., 2015).

In contrast, **cross-lingual word embeddings** aim to align the vector spaces of different languages such that words with similar meanings across languages are represented by similar vectors. The key goal of cross-lingual embeddings is to enable the model to understand the relationship between words in different languages, making it possible to transfer knowledge across languages without the need for explicit translation or labeled parallel data. One of the main challenges in cross-lingual embedding learning is ensuring that the representations of words across different languages are aligned in such a way that words with equivalent meanings in different languages are mapped to similar vector locations in the shared space. This alignment can be achieved through several techniques, which can be broadly classified into joint training and projection-based alignment methods.

Joint training of vector space involves learning a shared vector space for multiple languages simultaneously. This approach allows embeddings for different languages to be trained together, ensuring that semantic similarity is captured

across languages from the outset. Some well-known methods that adopt joint training include methods originally designed for monolingual embeddings, which have been extended for multilingual training by learning embeddings for words from different languages in a unified space. For example, word2vec can be trained on a multilingual corpus, where words from different languages are mapped into a single vector space. Similarly, fastText can learn subword-level representations, allowing for more robust cross-lingual embeddings, especially for morphologically rich languages.

Another approach involves **Transformer-based Language Models (LMs)**, such as mBERT (Devlin et al., 2019) or XLM-RoBERTa (Conneau et al., 2020), that have been trained on multilingual corpora. These models generate contextualized embeddings that capture cross-lingual relationships by conditioning the learning of word representations on the surrounding multilingual context. They can generate effective cross-lingual embeddings by jointly training on large multilingual corpora, making them highly effective for downstream tasks involving multiple languages.

Projection-based alignment, also known as cross-lingual mapping, is a method that aligns pre-trained monolingual word embeddings from different languages into a shared vector space (Conneau et al., 2018). This approach does not require joint training of the embeddings, making it a flexible alternative to methods that train embeddings from scratch. Instead, the objective is to align existing word vectors in such a way that semantically similar words from different languages are represented by similar vectors in the shared space, similar to joint training methods. The primary advantage of projection-based alignment is that it does not necessitate parallel data, such as aligned corpora, for training. Instead, it relies on parallel corpora or word-level alignments to guide the mapping process. This makes it particularly valuable in resource-scarce settings where obtaining large-scale parallel datasets may be difficult. Figure 3 illustrates this alignment process by comparing monolingual embeddings with their aligned representations in a shared cross-lingual space.

As described by Glavaš et al. (2019), the process of projection-based alignment begins by constructing a seed translation dictionary, which can be created either through supervised or unsupervised methods. This dictionary provides the initial mappings between words in different languages. Once the seed dictionary is established, the next step involves aligning the monolingual vector spaces for each language, typically using methods that find commonalities between them. This alignment can be achieved through techniques, such as linear transformations or more complex procedures, such as adversarial training. To refine the alignment, projection matrices are learned to map the word embeddings from the individual monolingual spaces into a shared cross-lingual vector space. Conneau et al. (2018) proposed an unsupervised post-hoc alignment method, where they align English

and Italian embeddings by learning a rotation matrix through adversarial training. This process is further enhanced by using frequent words as anchor points and expanding areas of high density to improve the alignment. This iterative approach results in better alignment of word vectors between languages.

2.3 Language Models

Language modeling has gained considerable attention due to its ability to capture patterns in natural language directly from data. While traditional approaches often rely on manually designed rules or features, modern language models learn these patterns automatically by analyzing large collections of text. This makes them scalable and adaptable for a wide range of language tasks. A key challenge, however, is developing models that can account for the complexity, variability, and ambiguity inherent in human language (Jurafsky and Martin, 2023).Unlike formal languages with fixed syntactic rules, natural language is dynamic, evolving over time and influenced by sociocultural and cognitive factors (Jackendoff, 2002). This complexity makes it substantially more difficult to design computational models that can capture both the structured and changing aspects of human language.

Human language is characterized by rich syntactic structures, context-dependent semantics, and pragmatic subtleties, making it difficult to represent and generate through purely deterministic or rule-based approaches (Chomsky, 1957; Manning and Schutze, 1999). Efforts to capture the statistical properties and structural complexity of natural language have led to the development of language models, which estimate probability distributions over word sequences and provide a principled framework for modeling linguistic phenomena (Jelinek, 1990; Bengio et al., 2003). Language models have since become foundational components of modern NLP systems, effectively addressing the complexity of tasks, such as machine translation, speech recognition, and text generation (Jurafsky and Martin, 2023).

2.3.1 Statistical Approaches

Statistical Language Models (SLMs) provide a probabilistic framework for modeling natural language by estimating the likelihood of word sequences based on their frequency in large corpora (Manning and Schutze, 1999; Jurafsky and Martin, 2023). These models are built on the assumption that natural language exhibits statistical regularities, which can be captured through probability distributions over sequences of words. SLMs have been instrumental in addressing the complexity of language and have played a foundational role in NLP applications, such as speech recognition, machine translation, and text generation.

A common approach to estimating the probability of a word given its preceding

context relies on the assumption that language exhibits local dependencies, where the likelihood of a word depends primarily on its recent history. More formally, given a history h, the probability of the next word w is denoted as $P(w \mid h)$. A straightforward method to estimate this probability is to use relative frequency counts obtained from a large corpus. This involves computing how often the sequence h is followed by the word w, and dividing this count by the total number of occurrences of the history h. The resulting estimate is shown in Equation 29.

$$P(w \mid h) = \frac{C(hw)}{C(h)} \tag{29}$$

Here, $C(hw)$ denotes the count of the sequence consisting of the history h followed by the word w, and $C(h)$ represents the count of the history h alone.

However, this approach faces a fundamental limitation due to the combinatorial complexity of language. Given the high degree of linguistic creativity and the vast number of possible word sequences, many contexts encountered in natural language occur rarely or not at all, even in large corpora. As a result, estimating probabilities directly from frequency counts becomes unreliable, particularly for longer contexts.

To address the complexity of natural language, language modeling applies probabilistic decomposition based on the chain rule of probability. The joint probability of a sequence of words $w_1, w_2, \ldots, w_n$ is expressed as the product of conditional probabilities of each word given its preceding context, as shown in Equation 30.

$$P(w_1, w_2, \ldots, w_n) = \prod_{k=1}^{n} P(w_k \mid w_1, \ldots, w_{k-1}) \tag{30}$$

Equation 30 illustrates that estimating the probability of an entire sequence requires estimating each conditional probability $P(w_k \mid w_1, \ldots, w_{k-1})$. In practice, directly computing these probabilities is intractable for long contexts, as many such sequences may never occur in the training data.

To make this estimation feasible, n-gram models introduce a simplifying assumption that reduces the dependency on the entire history to a fixed-size window of preceding words. This assumption is known as the **Markov assumption**, which posits that the probability of a word depends only on a limited number of preceding words rather than the full context (Jurafsky and Martin, 2023). For example, the bigram model approximates the conditional probability of a word given its full history by considering only the immediately preceding word, as shown in Equation 31.

$$P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-1}) \tag{31}$$

This concept generalizes to the n-gram model, where the probability of a word is conditioned on the previous $k-1$ words, with k indicating the order of the model and n denoting the position of the current word in the sequence, as shown in Equation 32.

$$P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-k+1:n-1}) \tag{32}$$

By applying this approximation to the chain rule decomposition of the joint probability of a sequence, the probability of an entire sequence $w_{1:n}$ is estimated as shown in Equation 33, where i denotes the position of the word in the sequence.

$$P(w_{1:n}) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-k+1:i-1}) \tag{33}$$

The probabilities are estimated using Maximum Likelihood Estimation (MLE), which computes conditional probabilities from observed frequency counts in a corpus. This estimation is given in Equation 34

$$P(w_n \mid w_{n-1}) = \frac{\text{count}(w_{n-1}w_n)}{\text{count}(w_{n-1})} \tag{34}$$

A major limitation of MLE is data sparsity, as many valid word sequences may not appear in the training corpus, resulting in zero probability estimates for unseen n-grams. This underestimate the likelihood of sequences that may occur in the test set, negatively affecting model performance. Moreover, if any word in a test sequence is assigned zero probability, the probability of the entire sequence also becomes zero, making it impossible to compute evaluation metrics, such as perplexity, which rely on the inverse probability of the test data (Jurafsky and Martin, 2023). To mitigate this, several smoothing techniques are employed, including Laplace smoothing, Good-Turing discounting, and Kneser-Ney smoothing, as well as interpolation and backoff.

Statistical language models have long provided the basis for probabilistic approaches to NLP. While n-gram models efficiently capture local dependencies, their reliance on fixed-length context and vulnerability to data sparsity limit their effectiveness. More advanced probabilistic methods, including Probabilistic Context-Free Grammars (PCFGs) (Manning and Schutze, 1999) and Maximum Entropy models (Berger et al., 1996), address some of these limitations by incorporating structural information and flexible feature-based conditioning. Nevertheless, the advent of neural language models has largely replaced statistical approaches, leveraging distributed representations and deep learning architectures to capture linguistic structures more effectively. The transition from statistical to neural models marks a significant shift in NLP, enabling more sophisticated and scalable language representations.

2.3.2 Neural Approaches to Language Modeling

Neural language modeling is an approach that forms the foundation of current NLP (Jurafsky and Martin, 2023). Compared to traditional statistical language models, Neural Language Models (NLMs) offer significant advantages in their ability to handle longer contextual histories, generalize across similar linguistic contexts, and achieve higher accuracy in predicting word sequences (Jurafsky and Martin, 2023).

Similar to statistical approaches, neural language models aim to approximate the probability distribution over the next word in a sequence, conditioned on the preceding context. However, while n-gram models approximate this probability by conditioning only on the $k-1$ most recent words, neural models replace discrete word identities with continuous embeddings that capture semantic and syntactic similarities between words (Jurafsky and Martin, 2023). The underlying approximation used by both approaches is shown in Equation 35.

$$P(w_t \mid w_1, \ldots, w_{t-1}) \approx P(w_t \mid w_{t-k+1}, \ldots, w_{t-1}) \tag{35}$$

However, the key difference lies in how the context $w_{t-k+1}, \ldots, w_{t-1}$ is represented. Statistical models rely on symbolic representations based on word identities and frequency counts, whereas neural models encode words as dense vectors in a continuous space. This allows them to capture similarities between words and generalize better to sequences not seen during training.

Neural language models are typically trained using a self-supervised learning approach, where no manual annotation is required. Generative models, such as GPT, are trained using an autoregressive objective. Given a sequence of tokens, the model learns to predict the next token using only the preceding tokens as context. In contrast, discriminative models, such as BERT, are trained using a masked language modeling objective. During training, a certain percentage of input tokens is randomly replaced with a special [MASK] token, and the model is trained to recover the original tokens based on both the left and right context. This bidirectional conditioning enables discriminative models to capture deeper contextual relationships. Additionally, some discriminative models include auxiliary tasks; for instance, BERT incorporates next sentence prediction to improve discourse-level understanding. These distinct training strategies are closely tied to the model architecture, as generative models typically use a decoder-only structure, whereas discriminative models rely only on the encoder. Sequence-to-sequence models, such as encoder-decoder architectures, can incorporate aspects of both objectives depending on their design. Both generative and discriminative models are optimized by minimizing the prediction error through backpropagation. This method makes it possible to train on large amounts of raw text, allowing the model to learn patterns and dependencies in language without explicit annotation.

One of the earliest architectures used for neural language modeling is the feed-forward neural language model (Bengio et al., 2003). This model uses a fixed-size context window to estimate the probability distribution over the next word, based on the embeddings of the preceding $k-1$ words. The modeling process follows a sequence of computational steps.

Each of the preceding words is represented as a one-hot vector $\mathbf{x}_{t-k+1}, \ldots, \mathbf{x}_{t-1}$. These one-hot vectors are mapped into a continuous, dense embedding space through multiplication with the embedding matrix $E \in \mathbb{R}^{d \times |V|}$, where d is the embedding dimension. Since each one-hot vector selects a single column from the embedding matrix, this operation retrieves the embedding vector for each context word. The resulting embeddings are concatenated into a single vector $\mathbf{e}$, as shown in Equation 36.

$$\mathbf{e} = [E\mathbf{x}_{t-k+1}; \ldots; E\mathbf{x}_{t-1}] \tag{36}$$

The concatenated embedding vector $\mathbf{e}$ serves as input to the hidden layer. A linear transformation, followed by a non-linear activation function computes the hidden representation $\mathbf{h}$, as defined in Equation 37.

$$\mathbf{h} = s(W\mathbf{e} + \mathbf{b}) \tag{37}$$

Here, W is the weight matrix of the hidden layer and $\mathbf{b}$ is the bias vector.

The hidden representation $\mathbf{h}$ is then transformed by another weight matrix U to produce the output logits $\mathbf{z}$, as shown in Equation 38.

$$\mathbf{z} = U\mathbf{h} \tag{38}$$

The model applies the softmax function to the logits $\mathbf{z}$ to obtain the probability distribution $\hat{\mathbf{y}}$ over the vocabulary for the next word prediction. This operation is defined in Equation 39.

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) \tag{39}$$

Each element $\hat{y}_i$ of the output vector represents the estimated probability of the i-th word in the vocabulary being the next word, given the previous $N-1$ words.

The model is trained using a self-supervised learning paradigm, where the objective is to minimize the prediction error between the predicted distribution $\hat{\mathbf{y}}$ and the true next word in the sequence. The parameters of the model, including the embedding matrix E, the hidden layer weights W, and the output layer weights U, are updated through backpropagation to optimize this objective. By learning dense word representations and non-linear transformations of context, the neural language model effectively captures semantic and syntactic regularities, offering substantial improvements over traditional statistical models.

Neural language models are trained using optimization algorithms, such as stochastic gradient descent, which iteratively adjust the model parameters to minimize the prediction error. The error between the predicted probability distribution $\hat{\mathbf{y}}$ and the true next word is quantified by a loss function, typically the cross-entropy loss.

The training process relies on error backpropagation, a method that enables the efficient computation of how changes in the model's internal parameters influence the value of the loss function. During this process, the error from the output layer is passed backward through the network. This allows the model to update the embedding matrix E, the hidden layer weights W, and the output layer weights U in a way that reduces the prediction error in future steps.

Despite these advances, feedforward neural language models are inherently limited in that they require fixed-length contexts and cannot naturally handle variable-length sequences, which are essential for modeling natural language (Goldberg, 2018). To address this constraint, Recurrent Neural Networks (RNNs) extend the neural language modeling framework by introducing a recurrent hidden state that evolves over time (Mikolov et al., 2010). RNNs process sequences word by word, updating the hidden state at each time step to summarize the preceding context. The hidden state h_t is updated according to the recurrence relation, as shown in Equation 40.

$$h_t = f(Wh_{t-1} + Ue_t + b) \tag{40}$$

Here, e_t denotes the embedding of the current word, h_{t-1} is the previous hidden state, and W, U, b are learnable parameters (Mikolov et al., 2010). This recursive mechanism allows RNN-based language models to capture sequential dependencies and represent contextual information dynamically, thereby modeling sentences and documents of arbitrary length. However, standard RNNs struggle with long-range dependencies, which led to further developments.

The Transformer architecture (Vaswani et al., 2017), introduced in Section 2.1.3, forms the foundation of modern **Transformer-based language models**. It replaces recurrence and convolution with self-attention mechanisms, enabling the model to directly capture dependencies between any pair of tokens in the input sequence.

As a first step, input sequences are tokenized using subword segmentation techniques, such as Byte-Pair Encoding (BPE) or WordPiece. These methods address the out-of-vocabulary problem by allowing the model to represent rare and morphologically rich words using subword units (Sennrich et al., 2016). Each token is then embedded and passed through multiple Transformer layers, where self-attention mechanisms enable the model to attend to all positions in the sequence. This allows the model to capture both local dependencies and long-range relationships

between tokens.

The output of the final Transformer layer consists of contextualized token representations. These hidden states are passed through a language modeling head—typically a linear layer followed by a softmax function—to produce a probability distribution over the vocabulary for next-token prediction. Transformer-based language models are usually pretrained on large-scale unlabeled corpora using self-supervised objectives. Common pretraining methods include autoregressive language modeling, as used in GPT (Radford et al., 2019), and masked language modeling, as introduced in BERT (Devlin et al., 2019).

Compared to earlier architectures, such as RNNs and CNNs, Transformer-based models provide significantly larger context windows. This extended receptive field allows the model to account for dependencies spanning hundreds or even thousands of tokens, which contributes to more coherent text generation and improved performance on discourse-level tasks (Brown et al., 2020a). For example, models like GPT-3 are capable of generating entire paragraphs of coherent text in response to a short prompt, while BERT-based models have achieved state-of-the-art results on tasks, such as question answering and named entity recognition.

In addition to their modeling capabilities, Transformer-based architectures offer practical advantages. Their non-recurrent design allows for efficient parallelization during training, and they scale effectively with increased data and model size (Kaplan et al., 2020). These properties have made them the architecture of choice for most state-of-the-art NLP systems across a wide range of applications, including machine translation, summarization, and dialogue systems. Thus, Transformer-based language models have profoundly influenced the field of speech and language processing (Jurafsky and Martin, 2023), and continue to serve as the core architecture in the development of increasingly powerful and general-purpose language technologies.

Training neural models from scratch for each downstream task is often computationally expensive and requires large amounts of labeled data. **Pre-trained Language Models (PLMs)** provide a more efficient solution by separating general representation learning from task-specific training. PLMs are first pre-trained on large-scale unlabeled corpora using self-supervised objectives, as explained in Section 2.1.1. After this initial stage, they are fine-tuned on labeled data tailored to specific tasks. This approach enables the model to learn general-purpose language representations that transfer well to new tasks. Pre-training also acts as a form of regularization, helping to reduce overfitting when fine-tuning on smaller datasets (Erhan et al., 2010).

Pre-training objectives differ across models and include Masked Language Modeling (MLM) in BERT (Devlin et al., 2019), causal language modeling in GPT (Radford et al., 2018), and span corruption in T5 (Raffel et al., 2020). These

objectives are designed to encourage the learning of rich, contextualized token representations that encode both linguistic features and factual knowledge (Petroni et al., 2019; Jiang et al., 2020b). Depending on the objective, pre-trained language models typically adopt encoder-only, decoder-only, or encoder-decoder architectures, with each being particularly well-suited to specific types of downstream tasks.

Once pre-trained, PLMs are fine-tuned on downstream tasks, such as sentiment analysis, machine translation, or question answering. Fine-tuning typically involves adding task-specific layers and adjusting the pre-trained weights, either fully or partially, on smaller, labeled datasets. This approach eliminates the need to retrain models from scratch and allows for adaptation to a new task or domain (Gururangan et al., 2020).

Prior research has shown that increasing both the amount of training data and the number of model parameters leads to significant improvements in the performance of PLMs (Radford et al., 2018; Brown et al., 2020a; Chowdhery et al., 2023). As a result, this scaling paradigm has resulted in the development of **Large Language Models (LLMs)**. These models typically rely on Transformer-based architectures and contain tens to hundreds of billions of parameters, trained on large and diverse text corpora. Empirical studies have shown that as model size increases, performance improves across a wide range of tasks and benchmarks (Kaplan et al., 2020; Brown et al., 2020a; Hoffmann et al., 2022). Moreover, once the number of parameters exceeds a certain threshold, LLMs begin to demonstrate the so-called emergent abilities (Wei et al., 2022a), such as in-context learning, instruction following, and step-by-step reasoning, which are not observed in smaller models (Zhao et al., 2023).

In-Context Learning (ICL) refers to the ability of LLMs to perform tasks by conditioning on a sequence of input-output examples provided in the prompt, without updating their parameters (Brown et al., 2020a; Jurafsky and Martin, 2023). Unlike traditional supervised learning, where model parameters are adjusted through gradient-based optimization, ICL relies entirely on forward computation during inference using pretrained weights. In this setting, the model infers task patterns by analogy, a process that resembles human reasoning (Winston, 1980; Dong et al., 2024). This behavior was first prominently observed in GPT-3 (Brown et al., 2020a), which demonstrated the ability to generalize from just a few examples — a phenomenon known as few-shot learning. However, ICL performance is highly sensitive to several factors. These include the format of the prompt, the selection and ordering of examples, and the nature of the task itself (Wang et al., 2023). Importantly, earlier models, such as GPT-2, did not exhibit in-context learning, which underscores the role of scale in enabling this capability. Both increased model capacity and large-scale pretraining data appear to be crucial for ICL to emerge (Brown et al., 2020a).

Instruction following refers to the ability of LLMs to perform novel tasks based solely on natural language instructions, without requiring task-specific examples at inference time. This ability is typically achieved through instruction tuning, a fine-tuning approach where the model is trained on a wide range of tasks presented as instructions (Ouyang et al., 2022; Sanh et al., 2022; Chung et al., 2024). Instead of learning to solve specific tasks, the goal of instruction tuning is to teach the model how to interpret and follow instructions across diverse domains (Jurafsky and Martin, 2023). As a result, instruction-tuned LLMs are able to generalize to new, unseen tasks when these are described in natural language, often showing strong performance in zero-shot settings. Unlike in-context learning, which depends on a few task examples provided at inference time, instruction following allows the model to infer the task directly from the prompt alone.

Finally, tasks requiring multi-step reasoning, such as mathematical word problems or logical inference, have traditionally posed significant challenges for language models (Rae et al., 2021; Nye et al., 2021). However, recent research has demonstrated that this limitation can be mitigated through Chain-of-Thought (CoT) prompting (Wei et al., 2022b). This strategy encourages language models to generate a sequence of intermediate reasoning steps before arriving at the final answer (Wei et al., 2022c; Suzgun et al., 2023). By guiding models through a step-by-step reasoning process, CoT prompting helps break down complex problems into smaller, logically coherent sub-tasks. As a result, language models perform more reliably on tasks that involve multiple stages of inference.

The emergence of these abilities represents a significant development in the capabilities of language models, enabling them to perform a wide range of tasks with minimal supervision. However, the mechanisms underlying the emergence of such capabilities remain only partially understood, and their stability across tasks, domains, and languages continues to be an active area of research (Wei et al., 2022a; Zhao et al., 2023).

2.3.3 Multilingual Language Models

Early PLMs were primarily developed for English, as large-scale, high-quality datasets were predominantly available in this language. However, their effectiveness decreased significantly when applied to other languages. To address this limitation, researchers introduced separate monolingual PLMs for individual languages. Although these models improved performance within their respective linguistic contexts, developing and maintaining a separate model for each language proved both computationally expensive and inefficient. Moreover, this approach limited opportunities for cross-lingual generalization and knowledge transfer.

Motivated by this limitation, the development of multilingual PLMs marked a major shift in language model design. Instead of building language-specific models,

researchers began to train unified models capable of processing text in multiple languages. These multilingual models are trained on corpora containing texts from a diverse set of languages and linguistic typologies. A key aspect of their design is the use of shared subword tokenization methods, such as SentencePiece or Byte Pair Encoding, which allow for a common representational space across languages. Combined with general training objectives, such as masked language modeling or sequence-to-sequence prediction, this design enables multilingual PLMs to learn robust and transferable representations.

As a result, multilingual PLMs demonstrate strong performance not only in high-resource contexts but also in low-resource scenarios. They are particularly effective in zero-shot and few-shot settings, where no or very little task-specific data is available for a target language. Models, such as XLM-RoBERTa (Conneau et al., 2020), mBART (Liu et al., 2020), and mBERT (Devlin et al., 2019) exemplify this multilingual approach. These models have achieved competitive results across a range of multilingual benchmarks, including tasks, such as cross-lingual natural language inference and machine translation, and methods, such as zero-shot transfer.

One of the main reasons for their success is their ability to map semantically similar inputs from different languages into a shared representational space. This capability supports cross-lingual alignment and facilitates knowledge transfer, even in the absence of direct supervision. As a result, a single multilingual PLM can serve multiple languages, enabling transfer from high-resource to low-resource languages, particularly when the languages are typologically related (Huang et al., 2023). Remarkably, some models have even shown the ability to generalize to languages that were not seen during training, demonstrating true zero-shot capabilities (Zhang et al., 2020). Consequently, multilingual PLMs have become foundational tools for a wide range of applications, including cross-lingual information retrieval, global-scale language technologies, and NLP systems for underrepresented languages.

Despite these advances, multilingual PLMs are not without limitations. One of the main challenges is representational bias caused by imbalanced training data. High-resource languages dominate most multilingual corpora and tend to influence the learned representations disproportionately. This often leads to degraded performance for low-resource languages, especially when they are typologically distant or poorly represented during pretraining (Lauscher et al., 2020; Blevins and Zettlemoyer, 2022). These biases affect both the quality of language understanding and the reliability of downstream task performance. Addressing such imbalances remains a key challenge for developing more equitable and linguistically inclusive NLP systems.

2.3.4 Explainability of Language Models

As language models have advanced in both scale and performance, they have become opaque (Belinkov and Glass, 2019; Zhao et al., 2024; Boselli et al., 2025). In particular, the shift from models with millions of parameters to those with tens or even hundreds of billions has added new layers of complexity, making it difficult to understand how specific outputs are produced (Frye et al., 2020). This lack of transparency raises important concerns related to trust, safety, and the potential for biased or harmful behavior (Zhao et al., 2024). As a result, explainability has become a central area of research, essential for interpreting and evaluating the behavior of language models.

Explainability refers to the ability to make a model's decisions comprehensible to humans by translating its internal operations into interpretable and meaningful concepts (Du et al., 2019). To support this goal, several post-hoc explanation methods have been proposed. Among the most widely used are Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016) and SHapley Additive exPlanations (SHAP) (Lundberg and Lee, 2017). These methods work by approximating a model's local behavior and identifying which input features most influence a specific prediction. In doing so, they help clarify why a model arrived at a particular output. Although originally developed for traditional machine learning models, such as decision trees and logistic regression, both LIME and SHAP have been adapted to provide local explanations for neural networks, including language models.

However, while methods, such as LIME and SHAP, are useful for interpreting classification decisions, they are less effective for large-scale language models trained with self-supervised objectives. These models transform textual input into dense vector representations through multiple layers of non-linear computation. As a result, their internal states are often inaccessible to simple interpretation techniques. Therefore, analyzing what kind of information is encoded in these hidden representations requires more specialized approaches. These go beyond feature attribution and aim to uncover how language models internally organize and represent knowledge within the embedding space.

Word Embedding Interpretability. Research on interpretable word embeddings can be broadly divided into two main approaches. The first is ante-hoc interpretability, which involves designing embedding models that are inherently interpretable. The second is post-hoc representation analysis, which seeks to uncover structure in pre-trained embeddings (Zini and Awad, 2022; Karwa and Singh, 2025).

Ante-hoc methods aim to impose constraints on the embedding space during training to enhance interpretability. One prominent line of work focuses on sparsi-

fication. For example, Non-Negative Sparse Embeddings (NNSE) by Murphy et al. (2012) use matrix factorization with non-negativity and sparsity constraints, resulting in embeddings where dimensions tend to align with interpretable semantic categories. Similarly, Sparse Overcomplete Word Vectors (SPOWV) (Faruqui et al., 2015) apply a dictionary learning framework to derive semantically meaningful vectors. In a different approach, Luo et al. (2015) propose Online Interpretable Word Embeddings (OIWE), which incrementally learn non-negative representations using projected gradient descent. Additionally, structured sparsity has been used to disentangle latent semantic features, making embeddings more interpretable (Subramanian et al., 2018; Guillot et al., 2023). However, widely used embedding models, such as Word2Vec, GloVe, and BERT, are not inherently interpretable (Incitti et al., 2023; Karwa and Singh, 2025). Moreover, ante-hoc approaches typically require access to the model's architecture and training data, which makes them unsuitable for analyzing pre-trained models without retraining.

In contrast, post-hoc methods aim to analyze and interpret existing embeddings without modifying the underlying model. One common strategy involves applying orthogonal transformations to project embeddings into lower-dimensional subspaces that concentrate task-relevant information while preserving geometric structure (Rothe et al., 2016). This method was extended by Rothe and Schütze (2016), who proposed learning an auxiliary transformation that aligns dimensions with interpretable linguistic attributes, such as polarity, concreteness, or part-of-speech, using labeled lexicons for supervision.

A distinct subclass of post-hoc interpretability methods focuses on embedding rotation. The goal of these methods is to reorder the embedding space so that individual dimensions align more closely with human-interpretable linguistic factors. This is typically achieved by applying mathematical transformations, such as orthogonal or factor-based rotations. For instance, Park et al. (2017) apply orthogonal rotation algorithms to GloVe and Word2Vec embeddings, showing that the rotated vectors maintain—or even improve—performance on tasks, such as word intrusion, similarity, and classification. Similarly, Ethayarajh (2019) demonstrate that word relationships, including analogies, can be better modeled through relation-specific orthogonal or linear transformations, rather than through uniform vector translations. These findings offer a more nuanced understanding of the geometric structure of embedding spaces. Importantly, such post-hoc techniques are particularly useful for analyzing pre-trained models that were not originally designed with interpretability in mind. They allow for the extraction of human-interpretable structure without modifying the model's architecture or requiring retraining.

Probing. Probing has become an established method for investigating what information is encoded in the internal representations of pre-trained language

models. These techniques are used to assess whether PLMs capture specific linguistic or factual properties. In this framework, a probe refers to a simple supervised model that is trained on fixed representations extracted from a PLM. The goal of the probe is to predict predefined properties, such as part-of-speech tags (Liu et al., 2019; Tenney et al., 2019), syntactic roles (Peters et al., 2018b; Hewitt and Manning, 2019; Clark et al., 2019), or factual triples (Petroni et al., 2019; Munda et al., 2025), using only these representations. In doing so, probing serves as an indirect measure of model interpretability (Belinkov and Glass, 2019; Liu et al., 2024).

Formally, given an input x, a language model $f(x;\theta)$ produces a hidden representation h. A probe $G(h;\phi)$ is then trained to predict a target label y, with learning guided by a loss function $L(y, G(h))$. During training, the parameters θ of the language model remain frozen, and only the probe parameters ϕ are updated. If the probe performs well on the task, this is taken as evidence that the relevant information is encoded in the representation h.

This interpretation, however, has been challenged on the grounds that probes may learn the task rather than encode linguistic structures (Hewitt and Liang, 2019). To address this concern, control tasks with randomly permuted labels are often used. These serve to evaluate selectivity, defined as the difference in performance between the original and control tasks. A large gap indicates that the probe is relying on genuine information in the representation, rather than learning the task from scratch (Hewitt and Liang, 2019).

Beyond linguistic properties, probing techniques have also been applied to assess the extent to which PLMs encode factual and domain-specific knowledge. Recent studies show that PLMs contain a wide range of information, including factual knowledge (Petroni et al., 2019; Jiang et al., 2020c), commonsense (Yin et al., 2022b), biomedical (Sung et al., 2021), and numerical knowledge (Lin et al., 2020). These studies typically construct datasets containing knowledge triples or question-answer pairs and evaluate whether the model, or a trained probe, can retrieve or predict the correct answers.

Prompting Techniques. While most probing methods rely on training task-specific classifiers to extract information from fixed representations, the emergence of large language models with instruction-following capabilities has enabled prompting as an additional approach for analysis. As model scale increases, language models exhibit emergent capabilities, such as the ability to follow natural language instructions without additional fine-tuning. This development has made it possible to probe model knowledge and reasoning through prompt-based interaction alone, treating models as zero- or few-shot learners rather than requiring additional classifiers.

This is particularly relevant when assessing the extent to which pretrained language models store factual knowledge without requiring additional supervision. Prompting allows querying factual knowledge by converting subject–relation–object triples into cloze-style sentences, such as *Dante was born in [MASK]*, and evaluating whether the model predicts the correct object. If the model ranks the correct token highly among a fixed vocabulary, it is taken as evidence that the fact (subject, relation, object) is encoded in its parameters (Petroni et al., 2019).

In addition to probing knowledge, recent research has explored ways to improve the interpretability of embeddings using natural language prompts. Rather than relying solely on vector space arithmetic or post-hoc visualization, these methods use pretrained language models to decode or verbalize the semantics encoded in embeddings. For instance, adapter-based techniques can project domain-specific embeddings into the token space of pretrained language models, allowing the model to function as a decoder that can generate natural language interpretations of these vectors (Tennenholtz et al., 2023a).

2.4 Knowledge Graphs

A Knowledge Graph (KG) is a directed labeled graph that encodes structured information by representing entities as nodes and their relations as labeled edges (Liu et al., 2022). Formally, a KG consists of a set of entities E and a set of relations R, where knowledge is expressed as a collection of triples (e_h, r, e_t); which are often referred to as facts. In this notation, e_h and e_t denote the head and tail entities, respectively, while r represents the relation connecting them. Mathematically, the set of all such triples, T, is defined as $T \subseteq E \times R \times E$, where each triple establishes a structured link between two entities (Liu et al., 2022).

Knowledge graphs have gained considerable traction in both academic and industrial contexts as a powerful mechanism for representing and organizing information derived from diverse sources. Although commonly described in mathematical terms as sets of subject–relation–object triples, there is no universal consensus on a strict, all-encompassing definition of a knowledge graph (Paulheim, 2017; Hogan et al., 2021). This ambiguity is addressed by Paulheim (2017), who proposes a set of minimal defining characteristics to distinguish knowledge graphs from other structured knowledge representations.

According to this view, a knowledge graph captures real-world entities and their interrelations in a graph structure. It typically includes a schema that defines the possible classes and relations among entities, yet it allows flexibility in how entities are connected. Unlike relational databases, which enforce strict domain and range constraints, knowledge graphs support more flexible interlinking of entities, even across domains. Moreover, they often span multiple topical areas, rather than

being limited to a single domain of knowledge.

2.4.1 Knowledge Representations

Humans naturally acquire, organize, and interpret knowledge to navigate the world. In contrast, computational systems do not possess such innate abilities. To perform tasks that involve reasoning and decision-making, they require structured access to external knowledge. Unlike human understanding, which is intuitive and context-driven, machines tend to rely on explicit mechanisms to represent and process factual information. One widely adopted approach is the use of knowledge graphs, which organize entities and their relationships in a structured and semantically meaningful way. These graphs serve as formalized representations of human knowledge and provide a foundation for tasks, such as inference, question answering, and commonsense reasoning. Without such structured representations, many of these tasks would remain inaccessible to purely statistical models.

2.4.2 Knowledge Graph Embeddings

A knowledge graph is a structured representation of information in the form of a multi-relational graph, where entities serve as nodes and relationships between them are represented as directed edges. Each edge is formally expressed as a triple (h, r, t), where h (head entity) and t (tail entity) are connected through a specific relation r. While knowledge graphs provide a structured and interpretable representation of data, their inherently symbolic nature often presents challenges for computational manipulation (Wang et al., 2017), as discrete symbolic structures do not easily integrate with continuous representations used in machine learning models. To address the challenges posed by the symbolic nature of knowledge graphs, knowledge graph embeddings have been introduced as a means of mapping entities and relations into a continuous vector space, enabling more efficient computational processing while retaining the structural and semantic properties of the KG (Wang et al., 2017).

Wang et al. (2017) define KG embedding as the process of representing entities and relations within a continuous, low-dimensional vector space that enables efficient reasoning and computation over structured knowledge. This process typically unfolds in three stages. First, entities and relations are mapped to real-valued representations in a continuous vector space. Entities are commonly modeled as dense vectors, while relations are encoded as transformations that operate on these vectors. Such transformations may take the form of vectors, matrices, tensors, or more complex probabilistic structures, including multivariate Gaussians. Second, a scoring function is defined to assess the plausibility of a given triple (h, r, t), assigning higher values to triples that are more likely to correspond to valid facts in

the knowledge graph. Third, the model parameters are learned by optimizing the scoring function over the observed data, thereby encouraging the model to assign higher plausibility scores to correct triples. Existing KG embedding approaches can be broadly classified into translational distance models, which employ distance-based scoring functions, and semantic matching models, which rely on similarity measures.

2.4.3 Translational Distance Models

TransE (Bordes et al., 2013) is one of the earliest knowledge graph embedding models that represents both entities and relations as vectors in the same vector space. For a given triple (h, r, t), the model assumes that the embedding of the tail entity $\mathbf{t}$ should be close to the vector sum of the head entity $\mathbf{h}$ and the relation $\mathbf{r}$, with the assumption that $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when the fact holds. The plausibility of a triple is determined by the negative distance between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$, as shown in Equation 41.

$$f_r(h, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{1/2} \tag{41}$$

In Equation 41, a higher score indicates a more plausible triple (Wang et al., 2017). The translation-based assumption transforms symbolic relational data into geometric relationships, as illustrated in Figure 4(a). TransE is simple and efficient, using a single vector per entity and relation and optimizing a margin-based ranking loss over valid and corrupted triples. However, TransE struggles with one-to-many, many-to-one, and many-to-many relations due to its strict translational assumption.

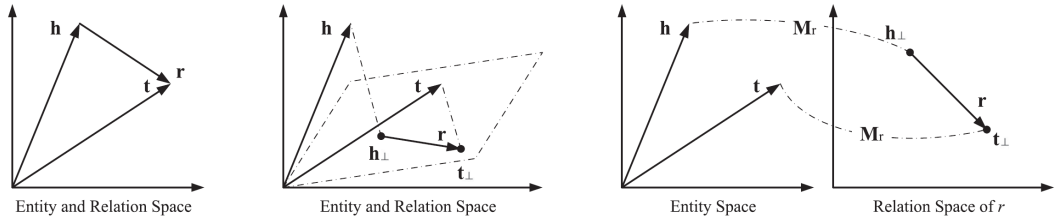
TransH (Wang et al., 2014) extends the limitations of TransE by introducing relation-specific hyperplanes to better model complex relational structures. Instead of embedding all entities and relations in the same space, TransH allows each entity to be represented differently depending on the relation in which it appears. For each relation $\mathbf{r}$, a hyperplane is defined by a normal vector $\mathbf{w}_r$, and entity embeddings $\mathbf{h}$ and $\mathbf{t}$ are first projected onto the hyperplane, as presented in Equation 42.

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r, \quad \mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r \tag{42}$$

The approach is based on the idea that the projected head entity plus the relation vector should approximate the projected tail entity, with the assumption that $\mathbf{h}_\perp + \mathbf{r} \approx \mathbf{t}_\perp$ when the triple holds. The plausibility of a triple is evaluated by the negative squared distance between these projected vectors.

$$f_r(h, t) = -\|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|_2^2 \tag{43}$$

As shown in Equation 43, a higher score indicates a more plausible triple. By allowing entities to have different embeddings depending on the relation context,



(a) TransE. (b) TransH. (c) TransR.

Figure 4: Visual representations of TransE, TransH, and TransR models for knowledge graph embedding. (Wang et al., 2017, 2726)

TransH provides a more expressive and adaptable framework for modeling relational structure. At the same time, it preserves the conceptual simplicity and efficiency of translation-based models, making it suitable for large-scale knowledge graphs. A visual representation of this projection mechanism is provided in Figure 4(b).

TransR (Lin et al., 2015) addresses the limitations of previous models by introducing relation-specific vector spaces. While entities are initially embedded in a common entity space, each relation is associated with its own distinct relation space. For a given triple (h, r, t), the entity embeddings are projected into the corresponding relation space using a relation-specific projection matrix $\mathbf{M}_r$, as defined in Equation 44.

$$\mathbf{h}_r = \mathbf{M}_r\mathbf{h}, \quad \mathbf{t}_r = \mathbf{M}_r\mathbf{t} \tag{44}$$

The scoring function used in TransR is the same as in TransH, but applied to the projected vectors. By allowing entities to have relation-specific representations, TransR provides enhanced flexibility for modeling complex relational structures that are difficult to capture in a single embedding space. Figure 4(c) provides a visual representation of the relation-specific projection used in TransR.

2.4.4 Semantic Matching Models

Semantic matching models assess the plausibility of a knowledge graph triple by computing similarity between entity and relation representations in a shared embedding space, typically using neural or similarity-based scoring functions (Wang et al., 2017).

RESCAL (Nickel et al., 2011) is one of the earliest semantic matching models for knowledge graph embedding, based on a bilinear formulation (Wang et al., 2017). It represents each entity as a dense vector capturing its latent semantics, and each relation as a full-rank matrix that models pairwise interactions between

the latent components of the head and tail entities. The plausibility of a triple (h, r, t) is scored using the bilinear function, as shown in Equation 45.

$$f_r(h, t) = \mathbf{h}^\top \mathbf{M}_r \mathbf{t} \tag{45}$$

Here, $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ are the vector embeddings of the head and tail entities, and $\mathbf{M}_r \in \mathbb{R}^{d \times d}$ is the relation-specific matrix. This formulation captures all pairwise interactions between the components of $\mathbf{h}$ and $\mathbf{t}$, offering high expressive power. However, due to the use of a full matrix per relation, RESCAL introduces $O(d^2)$ parameters per relation, which limits its scalability to large knowledge graphs.

DistMult (Yang et al., 2014) simplifies the RESCAL model by restricting the relation matrices to be diagonal, thereby reducing computational complexity. In this formulation, each relation r is associated with a diagonal matrix $\text{diag}(\mathbf{r})$, where $\mathbf{r} \in \mathbb{R}^d$ is the relation embedding. The plausibility of a triple (h, r, t) is computed via a bilinear scoring function as shown in Equation 46.

$$f_r(\mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \text{diag}(\mathbf{r}) \mathbf{t} \tag{46}$$

This function captures pairwise interactions between the corresponding dimensions of the head, relation, and tail embeddings. Because the scoring function is symmetric, meaning that $f_r(\mathbf{h}, \mathbf{t}) = f_r(\mathbf{t}, \mathbf{h})$, DistMult is inherently limited in representing asymmetric relations. Nonetheless, it offers an efficient and scalable solution for learning knowledge graph embeddings with significantly fewer parameters than full matrix-based models, such as RESCAL.

Holographic Embeddings (HolE) (Nickel et al., 2016) combine the expressive capacity of bilinear models, such as RESCAL, with the efficiency of simpler methods, such as DistMult. In HolE, both entities and relations are represented as vectors in a real-valued space $\mathbb{R}^d$. Given a triple (h, r, t), the model computes the compositional representation of the head and tail entities using the circular correlation operation, as represented in Equation 47.

$$[\mathbf{h} \star \mathbf{t}]_i = \sum_{k=0}^{d-1} [\mathbf{h}]_k \cdot [\mathbf{t}]_{(k+i) \bmod d} \tag{47}$$

This operation compresses pairwise interactions between the components of the entity embeddings into a single vector, which is then matched with the relation vector via a dot product to yield the plausibility score, as shown in Equation 48.

$$f_r(h, t) = \mathbf{r}^\top (\mathbf{h} \star \mathbf{t}) \tag{48}$$

The circular correlation in HolE captures rich interactions between entity embeddings while maintaining linear space complexity. Unlike DistMult, HolE can model asymmetric relations because circular correlation is not commutative; that is, $\mathbf{h} \star \mathbf{t}$

is not equal to $\mathbf{t} \star \mathbf{h}$. By applying the Fast Fourier Transform, HolE achieves a time complexity of $\mathcal{O}(d)$, making it more efficient than bilinear models that use matrix-based relation representations.

While RESCAL and its extensions rely on bilinear or factorized operations, their linear structure constrains the modeling of complex relational patterns. Neural architectures overcome this limitation by learning non-linear transformations, resulting in more expressive scoring functions and relational representations.

Semantic Matching Energy (SME) (Bordes et al., 2014) is an early neural network-based approach for knowledge graph embedding that computes plausibility scores through semantic matching. Given a triple (h, r, t), the head and tail entities are first combined with the relation in the hidden layer using learned transformation functions. The score is then defined as the dot product of the transformed vectors, as shown in Equation 49.

$$f_r(h, t) = g_u(\mathbf{h}, \mathbf{r})^\top g_v(\mathbf{t}, \mathbf{r}), \tag{49}$$

Here, g_u and g_v are non-linear mappings defined either linearly or bilinearly. For example, in the linear variant, $g_u(\mathbf{h}, \mathbf{r}) = \mathbf{M}_u^1\mathbf{h} + \mathbf{M}_u^2\mathbf{r} + \mathbf{b}_u$, and similarly for g_v (Wang et al., 2017). SME captures semantic similarity between entities and relations, but it is limited in expressiveness compared to later neural models, such as NTN.

Neural Tensor Network (NTN) (Socher et al., 2013) extends traditional neural architectures by introducing a bilinear tensor layer that models direct interactions between entity embeddings. Given a triple (h, r, t), where $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ are the head and tail entity vectors, NTN defines the scoring function as shown in Equation 50.

$$f_r(h, t) = \mathbf{r}^\top \tanh\left(\mathbf{h}^\top \underline{\mathbf{M}}_r \mathbf{t} + \mathbf{M}_r^1\mathbf{h} + \mathbf{M}_r^2\mathbf{t} + \mathbf{b}_r\right) \tag{50}$$

In this formulation, $\underline{\mathbf{M}}_r \in \mathbb{R}^{d\times d\times k}$ is a tensor that captures k different types of interactions between the head and tail entities. The matrices $\mathbf{M}_r^1$ and $\mathbf{M}_r^2 \in \mathbb{R}^{k\times d}$ are applied to the head and tail vectors. The vector $\mathbf{b}_r \in \mathbb{R}^k$ adds a bias, and $\mathbf{r} \in \mathbb{R}^k$ is used to combine the result into a final score.

The non-linear activation function tanh allows the model to learn more flexible relational patterns. By modeling direct interactions between entity embeddings, NTN can represent more complex relations than simpler models. However, it requires a large number of parameters, which makes it computationally expensive and less efficient for large knowledge graphs.

Multi-Layer Perceptron (MLP) (Dong et al., 2014) is a neural network-based model that represents each entity and relation as a vector. Given a triple (h, r, t), the embeddings are concatenated and passed through a non-linear hidden layer. The scoring function is defined as shown in Equation 51.

$$f_r(h,t) = \mathbf{w}^\top \tanh(\mathbf{M}^1\mathbf{h} + \mathbf{M}^2\mathbf{r} + \mathbf{M}^3\mathbf{t}) \tag{51}$$

Here, $\mathbf{M}^1$, $\mathbf{M}^2$, and $\mathbf{M}^3$ are weight matrices of the first layer, and $\mathbf{w}$ is the weight vector of the second layer (Wang et al., 2017). The MLP model captures non-linear interactions between entities and relations while maintaining moderate complexity, making it more expressive than linear models, such as TransE, yet more efficient than tensor-based approaches, such as NTN.

RDF2Vec (Ristoski and Paulheim, 2016) is an unsupervised method that learns latent vector representations of Resource Description Framework (RDF) graph entities by adapting neural language modeling techniques. Since most machine learning algorithms require input in the form of vectors, RDF2Vec provides a way to transform the graph-structured data of RDF into a sequential format that can be processed by such models. It draws inspiration from natural language processing by treating sequences of entities in the graph similarly to sequences of words in text, enabling the use of models, such as word2vec (Mikolov et al., 2013).

To convert the graph into sequences, RDF2Vec uses two strategies. The first is based on graph walks: for each entity in the RDF graph, multiple breadth-first walks of a fixed depth are generated to capture the surrounding structure. The second is based on Weisfeiler-Lehman subtree graph kernels, which iteratively relabel nodes based on their neighborhood structure, effectively encoding subtree information in the form of sequences. This approach is adapted to handle RDF-specific properties, such as directed edges and label consistency across iterations.

Once the sequences are created, a neural language model is trained on them using the skip-gram variant of word2vec. This model learns to predict surrounding entities within a fixed window, producing embeddings where entities that appear in similar structural and semantic contexts are placed close together in the vector space. These embeddings reflect both local neighborhood information and higher-order graph patterns.

Importantly, the embeddings learned by RDF2Vec are task-independent. This makes them applicable across a wide range of machine learning algorithms, such as support vector machines, random forests, k-nearest neighbors, and neural networks (Ristoski and Paulheim, 2016). Due to this flexibility, RDF2Vec supports various downstream tasks, including entity classification, link prediction, and recommender systems.

2.5 Vector Spaces

Vector spaces provide the mathematical foundation for representing and manipulating data in a structured and algebraically consistent manner. In NLP, vector spaces serve to represent text at varying levels of granularity, from individual tokens to

complete documents, within continuous, high-dimensional spaces. This section introduces the formal definition of vector spaces, describes linear transformations within these spaces, and outlines similarity measures for comparing vectors.

2.5.1 Linear Spaces

A linear space, also called a **vector space**, is one of the foundational structures in linear algebra and serves as the mathematical framework for a wide range of applications in theoretical and applied disciplines, including computational linguistics. Formally, a vector space V over a field $\mathbb{F}$ is a non-empty set equipped with two operations: vector addition, which combines any two vectors $\mathbf{u}, \mathbf{v} \in V$ to produce another vector $\mathbf{u} + \mathbf{v} \in V$, and scalar multiplication, which maps a scalar $a \in \mathbb{F}$ and a vector $\mathbf{v} \in V$ to another vector $a\mathbf{v} \in V$. These operations are required to satisfy a specific set of axioms that impose a linear structure on the space. The elements of V are referred to as vectors, and the elements of $\mathbb{F}$ as scalars.

These operations must satisfy the eight axioms listed below, which define the algebraic structure of a vector space, as defined by Friedberg et al. (2003), for all vectors $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and all scalars $a, b \in \mathbb{F}$:

1. For all vectors $\mathbf{u}$ and $\mathbf{v}$, $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$.
2. For all vectors $\mathbf{u}, \mathbf{v}$, and $\mathbf{w}$, $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$.
3. There exists a vector denoted $\mathbf{0}$ such that $\mathbf{u} + \mathbf{0} = \mathbf{u}$ for each vector $\mathbf{u}$.
4. For each vector $\mathbf{u}$, there is a vector $\mathbf{v}$ such that $\mathbf{u} + \mathbf{v} = \mathbf{0}$.
5. For each vector $\mathbf{u}$, $1\mathbf{u} = \mathbf{u}$.
6. For each pair of real numbers a and b, and each vector $\mathbf{u}$, $(ab)\mathbf{u} = a(b\mathbf{u})$.
7. For each real number a, and each pair of vectors $\mathbf{u}$ and $\mathbf{v}$, $a(\mathbf{u}+\mathbf{v}) = a\mathbf{u}+a\mathbf{v}$.
8. For each pair of real numbers a and b, and each vector $\mathbf{u}$, $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$.

Having established the foundational structure of vector spaces, it is natural to consider subsets of these spaces that themselves preserve the underlying linear structure. Such subsets are known as **subspaces**. Given a vector space V over a field $\mathbb{F}$, a subspace is a subset $U \subseteq V$ that is itself a vector space under the same operations of vector addition and scalar multiplication defined on V. In order to qualify as a subspace, the set U must be non-empty and closed under these two operations. This guarantees that all axioms of a vector space are inherited from V. In particular, the zero vector must be contained in U, since it results from scalar multiplication of any vector by zero.

Subspaces are fundamental in dimensionality reduction techniques, such as Principal Component Analysis (PCA) (Hotelling, 1933), where data points in a high-dimensional space are projected onto a lower-dimensional subspace that captures the directions of maximal variance. This projection reduces complexity while preserving the most informative components of the data, enabling both efficient computation and interpretation. Decomposition techniques, such as orthogonal projection, also rely on subspaces by decomposing vectors into components that lie within a subspace and those orthogonal to it.

In machine learning, subspace structures are frequently leveraged to reduce computational complexity or to interpret model behavior. Autoencoders, for example, learn to map high-dimensional input data to a latent subspace that captures the most salient features. Similarly, low-rank approximation techniques, such as those used in Low-Rank Adaptation (LoRA) (Hu et al., 2022), assume that model updates or representations lie in a subspace of significantly lower dimension than the original parameter space. This assumption enables efficient fine-tuning of large models with minimal resource overhead.

An important structural property of vector spaces is their **dimension**, which is a fundamental notion in linear algebra that quantifies the size or capacity of a vector space. Formally, the dimension of a vector space V over a field $\mathbb{F}$ is defined as the cardinality of a basis of V. A basis is a set of vectors in V that are linearly independent and span the entire space. In other words, every vector in V can be expressed in one and only one way as a finite linear combination of the basis vectors. The number of vectors in this basis remains constant for any basis of the space, a result guaranteed by the dimension theorem. This invariance is critical, as it ensures that the dimension is well-defined, independent of the choice of basis. In finite-dimensional spaces, the dimension gives the maximum number of linearly independent vectors that can exist in the space. For example, $\mathbb{R}^n$ has dimension n, and any set of more than n vectors in $\mathbb{R}^n$ must be linearly dependent. A subspace always has dimension less than or equal to that of the space it lies in; equality holds only when the subspace is the entire space.

2.5.2 Transformations

A **linear transformation** is a concept in linear algebra that captures the idea of structure-preserving mappings between vector spaces. Given two vector spaces V and W over the same field $\mathbb{F}$, a function $T : V \to W$ is considered to be a linear transformation if it satisfies the following condition shown in Equation (52) for all vectors $\mathbf{x}, \mathbf{y} \in V$ and all scalars $c, d \in \mathbb{F}$ (Strang, 2006).

$$T(c\mathbf{x} + d\mathbf{y}) = cT(\mathbf{x}) + dT(\mathbf{y}) \tag{52}$$

This condition ensures that linear transformations preserve the operations that define the vector space structure: vector addition and scalar multiplication. As a result, linear combinations are mapped to linear combinations, and the image of a subspace under a linear transformation is again a subspace. The zero vector is always preserved, since $T(\mathbf{0}) = \mathbf{0}$.

Linear transformations are useful because they allow for the analysis and manipulation of abstract vector spaces using algebraic tools. Every linear transformation between finite-dimensional vector spaces can be represented by a matrix, and this correspondence allows computations with transformations to be carried out using matrix multiplication. If $T : \mathbb{F}^n \to \mathbb{F}^m$ is a linear transformation, then there exists a unique $m \times n$ matrix A such that $T(\mathbf{x}) = A\mathbf{x}$ for all $\mathbf{x} \in \mathbb{F}^n$.

The set of all vectors in V that are mapped to the zero vector in W, meaning they are set to zero and lose all distinguishable information under the transformation, is the kernel of a linear transformation $T : V \to W$. While the set of all vectors in W that can be written as $T(\mathbf{x})$ for some $\mathbf{x} \in V$, and which describes all possible outputs of the transformation, is the image of T. The relationship between the kernel and image reflects how the transformation affects the dimensionality of the input space.

Geometrically, linear transformations correspond to operations, such as rotations, reflections, scalings, and projections. These transformations preserve the origin and the linear relationships between vectors. For example, a matrix can reflect a vector across a plane, rotate it by a fixed angle, or project it onto a lower-dimensional subspace. However, not every transformation is linear: translations or nonlinear distortions do not satisfy the linearity condition and cannot be represented by a matrix (Strang, 2009).

A special type of linear transformation that preserves the geometric structure of the vector space is called an orthogonal transformation. Formally, a linear transformation $Q : V \to V$ is called orthogonal if it is represented by an orthogonal matrix, meaning that $Q^\top Q = QQ^\top = I$, where $Q^\top$ denotes the transpose of Q, and I is the identity matrix. This condition implies that the inverse of Q is equal to its transpose, which has several important consequences. Most notably, orthogonal transformations preserve inner products, and hence also vector norms and angles between vectors. In other words, for any pair of vectors $\mathbf{h}, \mathbf{v} \in V$, it holds that $\langle Q\mathbf{h}, Q\mathbf{v}\rangle = \langle \mathbf{h}, \mathbf{v}\rangle$.

This invariance means that orthogonal transformations do not distort the underlying geometry of the space. They perform operations, such as rotations, reflections, or permutations of axes, without introducing scaling, shearing, or other forms of deformation. The transformation preserves distances between points and the overall shape of vector configurations, making it especially useful in applications where structural integrity of the data must be maintained. Importantly, orthogonal

transformations also maintain the dimensionality of the space, as they are invertible and norm-preserving mappings from a vector space onto itself. Because of their stability and interpretability, orthogonal transformations play a crucial role in numerous areas of linear algebra and its applications, including numerical analysis, computer graphics, and the analysis of embedding spaces in machine learning.

Linear transformations have become an essential tool for investigating the structure of hidden representations in PLMs. These transformations are used to assess whether specific linguistic or structural properties are encoded within the vector spaces that such models produce.

A foundational technique in this line of research involves training a transformation matrix that maps hidden representations to a new space where latent properties become explicitly measurable. For instance, one prominent approach focuses on determining whether syntactic structures, such as dependency parse trees, are implicitly encoded within contextual embeddings (Hewitt and Manning, 2019). The method constructs a linear transformation under which the squared Euclidean distances between transformed word embeddings approximate the distances between nodes in the syntactic parse tree, and the squared norms correspond to tree depths. This formulation enables direct probing of whether the geometry of syntactic trees can be reconstructed from learned representations through a linear mapping.

Extending this paradigm, attention-based methods (Coenen et al., 2019) evaluate whether attention patterns encode syntactic relations. In this case, attention vectors are constructed by concatenating attention weights across all layers and heads. Linear classifiers are then trained to determine whether these high-dimensional attention vectors can linearly separate syntactic roles. This approach operates under the assumption that if syntactic relations can be linearly classified from attention-derived features, then such information must be distributed in a structured and accessible way across the model's attention mechanism.

Beyond syntactic probing, linear transformations have also been applied to explore latent class structures in embedding spaces (Michael et al., 2020). In weakly supervised clustering approaches, transformation functions are learned that map contextual embeddings to subspaces in which semantically coherent classes, such as named entity categories or dependency labels, become separable. These methods rely on latent class induction mechanisms and treat the discovery of linguistic categories as an unsupervised or semi-supervised clustering problem in transformed vector spaces.

An alternative perspective on linear transformations considers the internal dynamics of the network architecture itself. Instead of probing for external linguistic properties, recent approaches aim to investigate the compositionality and informativeness of intermediate representations across different layers of transformer models (Yom Din et al., 2024). Linear maps are used to approximate the final model output

from intermediate layers. The observation that predictions derived from linearly transformed intermediate layers can match or exceed the performance of full model inference suggests that substantial predictive content is preserved linearly across layers.

Extending earlier work that applies linear transformations to uncover syntactic structures in language models, recent approaches have proposed analogous methods to investigate whether factual associations are implicitly encoded in the vector spaces of PLMs. The central hypothesis is that if such knowledge is present, it can be surfaced by learning a linear transformation that maps the original hidden representations into a space where factual relationships become geometrically explicit.

The method introduced by Munda et al. (2025) employs contextual embeddings derived from the final hidden layers of PLMs such as BERT and GPT-2. Since many entities in the knowledge graph correspond to multi-word expressions, their embeddings are computed by averaging token-level vectors and applying L2 normalization. In the case of BERT, both averaged embeddings and the [CLS] token are evaluated, as the latter is specifically optimized during pre-training to capture sequence-level semantics.

2.5.3 Vector-Based Similarity

In linear algebra and its applications, such as information retrieval, computational linguistics, and machine learning, the focus often extends beyond individual vectors to the mathematical relationships between them, such as relative direction, magnitude, and distance. These relationships are formalized through measures of vector similarity, which quantify how closely two vectors are related within a given vector space.

A central challenge arises from the fact that such vectors may differ in magnitude or orientation, making simple coordinate-wise comparisons insufficient. Simply comparing coordinate values is not meaningful when the vectors differ in scale, or when their orientation carries more semantic significance than their position. This necessitates the use of similarity functions that are sensitive or invariant to specific geometric properties, depending on the representational goals.

The most common similarity metric is the **cosine similarity**, which quantifies the angle between the vectors (Jurafsky and Martin, 2023). Given two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$, cosine similarity is defined by Equation 53.

$$\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{\|\mathbf{v}\| \, \|\mathbf{w}\|} = \frac{\sum_{i=1}^{n} v_i w_i}{\sqrt{\sum_{i=1}^{n} v_i^2} \sqrt{\sum_{i=1}^{n} w_i^2}} \tag{53}$$

Here, $\mathbf{v} \cdot \mathbf{w}$ denotes the dot product, and $\| \cdot \|$ the Euclidean norm. Since the cosine similarity depends only on the angle between the vectors and not on their magnitudes, it is invariant to scaling and is therefore well suited to contexts in which direction encodes the primary information.

The **Euclidean distance**, also referred to as L2 distance, in contrast, measures the geometric distance between two vectors by computing the length of the straight line connecting them in the underlying vector space, as defined in Equation 54.

$$\text{dist}(\mathbf{v}, \mathbf{w}) = \|\mathbf{v} - \mathbf{w}\|_2 = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2} \tag{54}$$

Unlike cosine similarity, Euclidean distance is sensitive to both direction and magnitude. It decreases as vectors approach each other and equals zero only when they are identical. In high-dimensional spaces, differences in vector length can dominate this metric, potentially reducing the sensitivity to directional alignment.

The **dot product**, also referred as the inner product in Euclidean space, is an operation that takes two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$, where $\mathbf{v} = (v_1, \ldots, v_n)$ and $\mathbf{w} = (w_1, \ldots, w_n)$, and returns a scalar defined by the expression in Equation 55.

$$\text{dot product}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{n} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n \tag{55}$$

Algebraically, it is the sum of the element-wise products of corresponding vector components. Geometrically, it encodes both the magnitudes of the vectors and the cosine of the angle θ between them, as shown in Equation 56:

$$\mathbf{v} \cdot \mathbf{w} = \|\mathbf{v}\| \, \|\mathbf{w}\| \cos\theta \tag{56}$$

The dot product is positive when the angle between vectors is acute, zero when they are orthogonal, and negative when the angle is obtuse. Therefore, it serves as the basis for defining orthogonality, projecting one vector onto another, and measuring directional alignment in vector spaces.

3 Related Work

Pre-trained Language Models (PLMs) have demonstrated strong performance across a variety of NLP tasks, largely attributed to their capacity to encode both linguistic patterns and factual knowledge. To investigate whether PLMs encode factual and relational knowledge, Petroni et al. (2019) introduced the LAnguage Model Analysis (LAMA) probe. This framework transforms structured factual data, such as subject–relation–object triples and question–answer pairs, into cloze-style prompts. These prompts serve as fill-in-the-blank queries designed to test the model's ability to recover missing information. By evaluating model predictions against a reference knowledge base constructed using a relation extraction pipeline, the LAMA probe provides a means of measuring how much factual content is stored in the model's internal representations. In particular, the study probes whether monolingual language models possess factual knowledge by assessing their ability to complete masked prompts with correct entities. The underlying assumption is that a stronger alignment between the model's predictions and the reference knowledge base indicates a greater capacity for representing factual information. In particular, the study probes whether monolingual PLMs are capable of recalling factual knowledge in this fill-in-the-blank format. Heinzerling and Inui (2021) and Wang et al. (2022) extended the evaluation of factual knowledge in English by applying a range of fill-in-the-blank datasets to PLMs.

Subsequent work has examined the limitations of factual knowledge retrieval under more complex linguistic conditions. Kassner and Schütze (2020) extended the probing paradigm to include negated statements, demonstrating that PLMs often struggle to correctly retrieve factual information when it is presented in a negated form. In parallel, efforts have been made to improve the formulation of prompts in order to elicit more reliable factual responses. For example, Shin et al. (2020), Reynolds and McDonell (2021), and Jiang et al. (2020b) proposed methods for optimizing prompt construction, thereby enhancing the models' ability to access and articulate factual knowledge. Beyond cloze-style probing, alternative evaluation strategies have also been explored. Bouraoui et al. (2020) and Heinzerling and Inui (2021) proposed alternative methodologies that move beyond the cloze tests.

Within the domain of Multilingual Pre-trained Language Models (ML-PLMs), a growing body of research has investigated the extent to which these models encode and retain factual knowledge across multiple languages. Empirical studies, including those by Jiang et al. (2020a), Kassner et al. (2021), Yin et al. (2022a),

Fierro and Søgaard (2022), and Keleg and Magdy (2023), provide evidence that ML-PLMs are capable of recalling factual information beyond English. However, the mechanisms through which these models acquire, store, and represent multilingual factual knowledge remain only partially understood. To enable more systematic evaluation, Kassner et al. (2021) introduced mLAMA, a multilingual adaptation of the original LAMA benchmark, which enables probing of factual knowledge in multilingual PLMs.

Concurrently, research has examined the robustness and consistency of factual knowledge retrieval in PLMs. Studies by Elazar et al. (2021) and Ravichander et al. (2020) reported notable inconsistencies in the factual responses generated by PLMs, particularly when the same queries are issued in different languages. Qi et al. (2023) further demonstrated that increasing model size tends to improve overall factual accuracy in probing tasks, yet this improvement does not reliably extend to cross-lingual consistency. Complementing these findings, Fierro and Søgaard (2022) observed that multilingual models, such as mBART and XLM-R, often produce inconsistent outputs when extracting factual knowledge, especially in languages other than English.

While supervised probing tasks and prompt-based approaches are effective for evaluating whether PLMs can output correct responses to specific queries, they assess the presence of information only indirectly and do not reveal the internal geometry of the embedding space where that information is encoded. Research that explicitly examines the representation of factual knowledge within the embedding spaces of multilingual PLMs remains limited. Existing work has predominantly focused on modeling structured ontological knowledge (Michael et al., 2020) or identifying the locations of specific facts within high-dimensional representations (Meng et al., 2022; Hase et al., 2024), rather than analyzing the structure of multilingual vector spaces themselves.

To address these limitations, several studies have explored methods that focus on enhancing the interpretability of language model embeddings through natural language interaction, as demonstrated by (Li and Liang, 2021; Tennenholtz et al., 2023b; Wu et al., 2024). In a separate but related approach, Hewitt and Manning (2019) introduced a linear transformation approach to evaluate whether syntactic structures, such as dependency trees, are preserved in the vector spaces of monolingual models, such as ELMo and BERT.

Building on the idea that linear transformations can reveal structured linguistic information, Rothe et al. (2016) proposed a model that learns orthogonal transformations to compress task-relevant information into a lower-dimensional subspace. Their approach, called Densifier, assumes that properties, such as sentiment or concreteness, can be isolated and encoded within ultradense regions of the embedding space. This aligns with the broader hypothesis that specific semantic features are

predominantly localized in interpretable subspaces. Extending this idea, Rothe and Schütze (2016) introduced a decomposition strategy in which embedding spaces are partitioned into orthogonal components, each capturing a distinct property, such as polarity or part-of-speech. These transformations improve interpretability while also supporting new embedding operations based on semantic dimensions. Although these studies focus on lexical-level semantics, the underlying principle of structured subspaces informs approaches aimed at uncovering factual knowledge in PLM vector spaces. However, these approaches focus exclusively on monolingual settings, leaving open the question of how effectively such methods generalize to multilingual language models.

Munda et al. (2025) proposed a novel approach to examine how factual knowledge is represented within the vector spaces of PLMs by making use of structured information from knowledge graphs. The study focused on evaluating the effectiveness of linear transformations applied to embeddings derived from English monolingual PLMs, specifically comparing BERT (Devlin et al., 2019), BART (Lewis et al., 2020), and GPT-2 (Cohen and Gokaslan, 2020). Experimental results demonstrated that contextual embeddings incorporating Wikipedia-based entity descriptions yielded better alignment with knowledge graph similarity structures than embeddings based solely on entity names. In addition, the investigation of reduced-rank transformation matrices showed that lower-dimensional subspaces can still preserve relevant factual properties, particularly in the case of BERT and BART. A qualitative examination of the models' outputs revealed that certain relation types, especially geographic and personal associations, remain challenging for all models examined. Overall, the findings emphasizes that PLMs are capable of encoding factual knowledge within their representations, although limitations persist in specific knowledge domains. However, the study was restricted to English monolingual PLMs, leaving open the question of how factual knowledge is encoded across other languages and within multilingual PLMs.

4 Methods

This chapter presents the methodological framework developed to investigate how pre-trained multilingual language models encode factual knowledge in alignment with multilingual knowledge graph embeddings. Section 4.1 outlines the training procedure to obtain multilingual KG embeddings, detailing the use of MTransE to obtain language-specific representations of entities and relations. Section 4.2 describes the dataset construction process, including the extraction and filtering of triples from a large-scale multilingual KG, and the computation of similarity scores that serve as supervision signals. Section 4.3 focuses on embedding KG entities using pre-trained language models, specifying the selection of PLMs and the extraction of contextual representations. Section 4.4 introduces the linear transformation framework designed to align PLM-derived embeddings with KG similarity scores, followed by Section 4.5, which details the training procedures and optimization settings for both the MTransE model and the transformation matrices. Finally, Section 4.6 describes the evaluation framework used to assess model performance, incorporating quantitative metrics, qualitative analysis, and visualizations of the transformed embedding spaces.

4.1 Multilingual KG embeddings

To train a Linear Transformation (LT) for evaluating whether multilingual PLMs encode factual knowledge, a dataset of entity pairs annotated with cosine similarity scores is required. In monolingual settings, prior work (Munda et al., 2025) used pre-trained KG embeddings obtained via the Wembedder API[1] (Nielsen, 2017) to compute the similarity scores. However, Wembedder relies on Wikidata entity IDs and therefore provides language-agnostic embeddings, returning identical cosine similarity values for a given entity pair regardless of the language. Due to its reliance on static entity IDs and shared embeddings, Wembedder remains unsuitable for multilingual setups, where meaningful differences across language-specific entity representations are expected.

To overcome this limitation, the approach adopted in this thesis aimed to satisfy two fundamental objectives. First, to obtain cosine similarity scores that capture

[1] `https://wembedder.toolforge.org`

multilingual distinctions across English, German, and Russian; and second, to construct a dataset suitable both for training language-specific knowledge graph embeddings and as supervision for linear transformation. As no publicly available multilingual knowledge graph embeddings were found that simultaneously offer language-specific vectors and include the underlying KG data used for training, it was necessary to train the knowledge graph embeddings from scratch.

4.1.1 Training Multilingual KG embeddings

A translation-based multilingual knowledge graph embedding model, MTransE (Chen et al., 2018), was selected as a suitable model due to its capacity to learn language-specific entity and relation embeddings. Although described as multilingual in the original paper, MTransE is designed for bilingual settings, where two knowledge graphs in different languages are embedded jointly. MTransE extends the TransE model (Bordes et al., 2013), described in Section 2.4.3, to bilingual settings by learning separate entity and relation embeddings for each language. Additionally, MTransE introduces a linear transformation that projects entity embeddings from one language into the embedding space of the other, enabling alignment between semantically corresponding entities.

Importantly, MTransE retains a relational structure $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, which is critical for preserving the semantics of the knowledge graph during embedding training. Unlike general-purpose multilingual embeddings, such as fastText, which do not model relational structure, MTransE is more appropriate in the context of LT-based knowledge probing.

The MTransE[2] model used in this study was further adapted to disable its cross-lingual alignment objective during the training. This modification was motivated by the observation that alignment links in Wikidata are frequently incomplete, noisy, or inconsistent (Haller et al., 2022; Shenoy et al., 2022). With the alignment loss removed, training depends solely on language-specific relational structure, resulting in two independent TransE embedding spaces. While this approach avoids distortion introduced by noisy or erroneous alignment links, it also removes any explicit mechanism for aligning the two embedding spaces. As a result, any cross-lingual similarity that appears is therefore incidental rather than learned.

For each training, English was paired with either German or Russian. Although the MTransE training script was executed with identical English datasets, model architecture, and hyperparameters across multiple runs, the resulting embeddings for English were not exactly identical. This behavior stems from multiple sources of nondeterminism inherent in the training process. First, by default, TensorFlow does not enforce fixed random seeds, and various components, such as weight

[2] `https://github.com/muhaochen/MTransE-tf/tree/master`

initialization, introduce stochasticity unless explicitly controlled. Second, the negative sampling procedure, used to generate corrupted triples during training, is influenced by runtime-dependent randomness, leading to different sampling sequences across runs. Third, the Adam optimizer, commonly used for training deep learning models, can exhibit nondeterministic behavior on GPU hardware due to parallel execution and non-associative floating-point arithmetic. As a result, unless randomness is explicitly managed, by fixing seeds across Python, NumPy, and TensorFlow, disabling time-dependent seeding, and enforcing deterministic computation on GPU, the model learns embeddings that are similar in distribution but not numerically identical across runs.

The objective of this thesis was to obtain multilingual knowledge graph embeddings for a variety of language pairs. Since the subsequent evaluation relies on computing cosine similarity between subjects and objects in sampled triples and minor variations in embedding initialization and negative sampling do not significantly affect overall similarity patterns at scale, strict determinism was not enforced. The random seed settings were therefore left unchanged from the original implementation in order to preserve the intended training behavior.

4.2 Datasets

To train MTransE and construct the supervision dataset for the linear transformation training, a collection of language-specific subsets was created from the Wikidata5M (Wang et al., 2021) inductive split. Wikidata5M is a large-scale dataset derived from the Wikidata knowledge graph (Vrandečić and Krötzsch, 2014), consisting of 4,579,609 entities, 822 relation types, and 20,496,514 triples. The dataset is designed to support entity-centric tasks, including link prediction and entity classification (Daza et al., 2021), as well as knowledge graph completion (Jiang et al., 2023). Each entity and relation is identified by a unique ID, which serves as a language-independent reference across the knowledge graph. Each triple is formatted as a tab-separated line of the form `[subject, relation, object]`, where both entity and relation identifiers follow the Wikidata naming convention: entity identifiers are prefixed with 'Q' and relation identifiers with 'P'. For example, a triple, such as `[Q174 P36 Q40]`, represents the factual statement `[Vienna, is the capital of, Austria]`, where Q174 corresponds to Vienna, P36 denotes the relation capital of, and Q40 refers to Austria.

Importantly, Wikidata5M follows the identifier system used in Wikidata, allowing direct access to external resources, such as textual labels and Wikipedia summaries. This consistency in entity and relation identifiers enables the retrieval of multilingual labels and summaries via the Wikidata and Wikipedia Application Programming Interfaces (APIs), making the dataset particularly suitable for multilingual setup.

4.2.1 Preparing triples for MTransE and LT training

To ensure a balanced and multilingual selection of triples across languages for a custom dataset a frequency-based filtering and quota allocation algorithm was applied. The 1,000 most frequent relations were identified from the original dataset and filtered using the Wikidata API[3] endpoint to retain only those with textual labels available in English, German, and Russian, ensuring that the final selection could support consistent comparison across languages. From the filtered candidates, the top 200 relations were selected by frequency. A per-relation sampling quota was then computed proportionally to frequency, with a minimum of 50 triples per relation. The quotas were then rebalanced to sum to exactly 60,000 while preserving the original distribution as closely as possible. The resulting statistics, including frequency, proportion, final quota, and multilingual labels, were written to a Tab-Separated Values (TSV) file. An overview of the final datasets for English, German, and Russian is provided in Table 1.

Table 1: Overview of the final multilingual datasets constructed for English, German, and Russian. Each dataset contains approximately 60,000 triples sampled from the top 200 most frequent relations with complete textual labels

Type	**en**	**de**	**ru**
Triples	60,000	59,977	59,945
Entities	79,337	73,826	71,386
Relations	200	200	200

To better understand the frequency distribution of relations and assess the impact of filtering, Figure 5 presents a comparative analysis of the original and reduced datasets. The Cumulative Distribution Function (CDF) reveals a strong skew in the original set, with a small number of relations accounting for a large proportion of the triples. In contrast, the filtered subset exhibits a more even distribution, as confirmed by the Kernel Density Estimation (KDE) on the right. These plots highlight the importance of quota-based sampling to mitigate frequency imbalance and ensure a more uniform representation across relation types.

For training the linear transformation, in addition to labels, both the subject and object entities had to be associated with Wikipedia summaries, retrieved using the Wikipedia REST API[4], specifically the `/api/rest_v1/page/summary/` endpoint. The API returns the introductory section of the article corresponding to a given entity. As factual content in Wikipedia is typically concentrated in the

[3] `https://www.wikidata.org/w/api.php`
[4] `https://{lang}.wikipedia.org/api/rest_v1/`

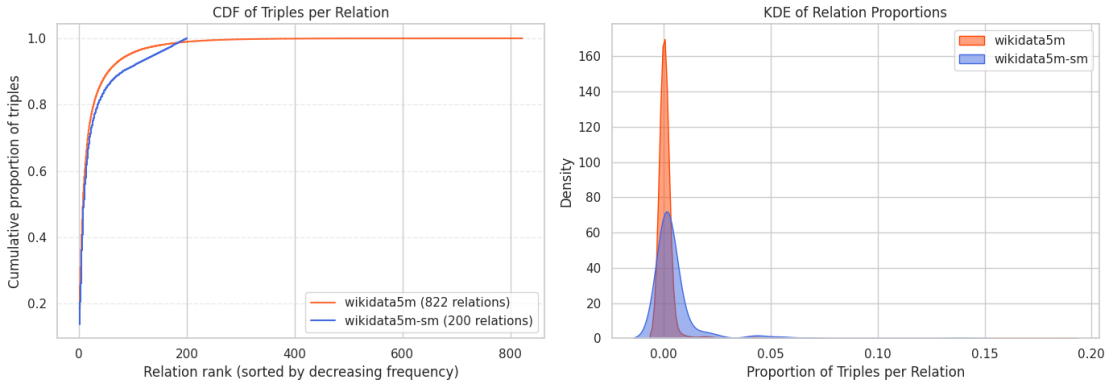


Figure 5: Distributional characteristics of relation frequencies in the original and reduced datasets. The left panel shows the CDF of triples per relation, sorted by decreasing frequency, for the full Wikidata5M set (822 relations) and the filtered small Wikidata5M-sm subset (200 relations). The right panel displays the KDE of relation-wise triple proportions, illustrating the long-tailed nature of the original dataset and the more balanced distribution achieved through filtering

opening sentences, these summaries serve as a valuable source of semantically rich descriptions. In the context of this research, such descriptions are particularly valuable, as they provide semantic context that can improve the quality of the embeddings extracted from PLMs.

Nevertheless, Wikipedia summaries, while informative, often include redundant or peripheral content beyond the initial sentences. To limit noise and standardize input length, all descriptions were segmented using the sentence tokenizer provided by the Natural Language Toolkit (NLTK) library (Bird and Loper, 2004), with each summary truncated to its first two sentences.

To ensure consistent formatting, tab characters were used as column separators in the dataset, separating fields, such as subject, relation, object, and their corresponding descriptions. However, since Wikipedia summaries are extracted from natural text, they may occasionally contain tab characters, which can interfere with the expected column structure. To address this, a validation step was implemented to check the number of columns per row. Rows that did not conform to the expected format were discarded to maintain structural integrity and preserve consistency for further processing.

A parallelized sampling pipeline was implemented to extract triples in accordance with the computed quotas, separately for English, German, and Russian. The process was distributed across multiple worker processes using chunk-based partitioning for scalability. Each worker was assigned a disjoint subset of relations based on

modulo partitioning. Candidate triples were first filtered to ensure the availability of labels for all entities, using cached API responses to minimize redundant queries. Triples lacking labels were excluded and logged. Moreover, duplicate triples were removed across all processing runs and data chunks to ensure that each triple appeared only once in the final dataset. To enable training with MTransE, a separate dataset was constructed from the filtered triples described above. Given the specific input format required by MTransE, all triple files were converted into the input format expected by the model, where each triple is serialized as a single line containing the subject, relation, and object labels, separated by the delimiter `@@@`. For example, the triple consisting of the subject `Vienna`, the relation `capital of`, and the object `Austria` is represented as `Vienna@@@capital of@@@Austria`. The resulting files were saved as Comma-Separated Value (CSV) files, with one triple per line.

4.2.2 Computing Similarity Scores between Entities

In order to extract cosine similarity scores between subject and object entities using the multilingual embeddings trained with MTransE, it was essential that all entities in the dataset are present in the original training vocabulary of the embedding model. Since MTransE was trained on a filtered subset of Wikidata5M, attempts to retrieve embeddings for unseen entities would result in missing vectors, compromising the reliability of the similarity computation. To ensure full coverage and avoid incomplete supervision, the original 60,000-triple dataset was reduced to 42,000 triples. The set of triples was re-sampled using the same frequency-based method as described in Section 4.2.1, adjusted to meet the dataset size. By sampling only from entities present in the original training corpus, the reduced 42K set avoided the need for additional validation or entity mapping, thereby preserving consistency across all downstream evaluations and simplifying the linear transformation training pipeline.

For each triple in the dataset, the textual labels of the subject and object were used to retrieve their corresponding embeddings from the trained MTransE model. To ensure consistency in the retrieval process, all labels were first canonicalized by lowercasing, normalizing whitespace, and stripping quotation marks. The resulting strings were then matched against the model's vocabulary, which maps labels to embedding indices. If both entities were found in the vocabulary, their embeddings were extracted from the multilingual model and their cosine similarity was computed. In cases where one or both entities were not present in the embedding vocabulary, due to their absence during MTransE training, the cosine similarity was marked as missing.

For each language, the final curated dataset was stored in CSV format and includes, for every triple, the subject and object entities, their corresponding

textual descriptions, and the precomputed similarity scores between them. Although the relation IDs and labels are not directly required during training, they were deliberately retained in the final dataset to support the computation of relation-specific statistics and to provide essential contextual information for the qualitative evaluation.

For model training and evaluation, the dataset is divided into training (80%), validation (10%), and test (10%) sets using the `train_test_split` function from Scikit-Learn.[5] The training set is used to optimize the model parameters by minimizing the loss function. During training, the validation set serves to monitor model performance and supports hyperparameter tuning through iterative evaluation. Finally, the test set is used exclusively for the final assessment, providing an unbiased estimate of the model's generalization performance on unseen data.

Table 2: Dataset statistics after splitting into training, validation, and test sets for English (en), German (de), and Russian (ru). Each split preserves the original distribution over the top 200 relations, with minor variation in the number of distinct relations per split due to quota rounding. The number of entities varies across splits and languages, reflecting language-specific availability of labels and descriptions

Language	Type	Training	Validation	Test	Total
en	Triples	33,593	4,199	4,200	41,992
	Entities	46,833	6,709	6,689	57,594
	Relations	200	200	200	200
de	Triples	33,580	4,197	4,198	41,975
	Entities	44,598	6,669	6,550	54,306
	Relations	200	200	196	200
ru	Triples	33,552	4,194	4,194	41,940
	Entities	43,557	6,500	6,536	52,949
	Relations	200	198	199	200

4.3 Embedding Entities with PLMs

The following section describes the process of embedding knowledge graph entities using multilingual pre-trained language models. It outlines the selection of appropriate models, the strategies used to obtain contextual embeddings for both single-word and multi-word entities, and the normalization steps applied to ensure comparability across representations.

[5] `https://scikit-learn.org/`

4.3.1 Pre-trained Language Models

This thesis focuses on the representation of factual knowledge in multilingual PLMs. While previous studies have primarily examined monolingual models, such as BERT and GPT-2, the present work extends the analysis to multilingual settings by selecting models that are explicitly trained to handle input from a wide range of languages. The primary models used in this study are multilingual BERT cased base model (Devlin et al., 2019), referred to as mBERT, and mBART-50 (Tang et al., 2020), referred to as mBART. These models differ in both architecture and training objectives. mBERT adopts a Transformer encoder-only architecture and is trained using the Masked Language Modeling (MLM) objective. In contrast, mBART is built upon the encoder–decoder Transformer architecture and is trained using a denoising objective, making it suitable for conditional generation tasks where an output sequence must be generated conditioned on an input. Both models are publicly available through the Hugging Face platform (Wolf et al., 2020), which provides access to pretrained weights and hidden states necessary for extracting contextual embeddings.

An overview of the technical specifications of the selected models, including their architecture, dimensionality, and language coverage, is presented in Table 3.

Table 3: Architectural specifications of the selected multilingual PLMs, including the number of parameters, hidden dimension size (H-Dim), count of encoder layers (E-Layers), and count of decoder layers (D-Layers)

Model	**Params**	**H-Dim**	**E-Layers**	**D-Layers**	**Languages**
mBERT	110M	768	12	/	104
mBART	610M	1024	12	12	50

4.3.2 Extracting Contextual Embeddings

Once the PLMs have been selected, the next step involves extracting contextual embeddings for the entities present in the dataset. In cases where entities consist of multiple tokens, it becomes necessary to derive a single vector representation that captures the semantics of the entire span. A commonly adopted approach for this is mean pooling, which involves computing the average of the token-level embeddings produced by the model.

Formally, given a tokenized entity sequence $S = (w_1, w_2, \ldots, w_n)$, the PLM encodes each token w_i into a contextualized embedding $\mathbf{h}_i \in \mathbb{R}^d$, where d denotes the hidden size of the model. The aggregated embedding $\mathbf{h}_{\text{mean}} \in \mathbb{R}^d$ for the entire

sequence is then obtained by taking the mean over all token embeddings in the last hidden layer, as shown in Equation 57.

$$\mathbf{h}_{\text{mean}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{h}_i \tag{57}$$

This method provides a fixed-size vector representation for each entity, regardless of its length, and serves as the basis for the evaluation presented in later sections.

An alternative approach to obtaining a single vector representation from a sequence of tokens relies on the special [CLS] token, which is included by design in several Transformer-based encoder models, such as BERT, RoBERTa, and ALBERT. The acronym [CLS] stands for "classification," indicating its original purpose during pretraining, where it was used to support sentence-level classification tasks. This token is inserted at the beginning of the input sequence and serves as a dedicated position for capturing information about the entire input. During pretraining, the model is optimized so that the embedding of the [CLS] token reflects a global representation of the sequence, typically in the context of classification objectives. Consequently, the final hidden state of the [CLS] token encodes a high-level, context-aware summary of the input and can be retrieved from the first position in the model's last hidden layer. This approach offers a practical and computationally efficient means of generating fixed-size vector representations for entities or phrases, particularly when alternative aggregation methods are not required

For mBERT, contextual embeddings are extracted using both the mean pooling strategy and the [CLS] token representation, allowing for an analysis of how these two approaches compare when applied to multilingual data. In contrast, mBART does not include a [CLS] token due to its encoder-decoder architecture; therefore, only the mean pooling method is applied to obtain a single vector representation for each input.

As discussed in Section 4.2, the dataset used in this study contains entities consisting of either single words or multi-word expressions. When such entities are passed into PLMs in isolation, the limited contextual input may restrict the model's ability to produce semantically rich embeddings, particularly in cases where the entity is ambiguous or lacks sufficient syntactic structure. To address this limitation, an alternative approach is introduced that incorporates additional descriptive information. Specifically, for each entity, the first two sentences of its corresponding Wikipedia description are retrieved and appended to the input in the following format: entity + [SEP] + description. This method aims to enhance the semantic representation of the entity by supplying the PLM with auxiliary factual content that can inform its encoding process.

Descriptions are particularly advantageous in that they provide disambiguating context, offer structured factual background, and reflect how an entity is typically

introduced in natural language. These aspects can help the model better interpret the entity, especially when presented with limited standalone information. Due to the multilingual nature of the PLMs used in this study, it was necessary to construct language-specific datasets in which descriptions were retrieved directly from Wikipedia in the corresponding language. This step ensures that the descriptive context aligns with the linguistic characteristics of the input. The decision to retain only the first two sentences of each description is motivated by two considerations: first, key factual details are typically concentrated at the beginning of Wikipedia entries, and second, restricting the length helps to remain within the token limits imposed by the models. By contrasting entity-only inputs with description-enhanced inputs, the study investigates whether and how additional linguistic context contributes to more informative and accurate entity representations.

Normalization. When working with contextual embeddings derived from PLMs, a key challenge lies in ensuring that the resulting representations are directly comparable across different inputs. This becomes particularly important when the inputs, such as knowledge graph entities, vary in length and structure. In this study, embeddings are extracted using two strategies: mean pooling across token representations and selecting the [CLS] token embedding. Both methods produce a single vector per entity, but the magnitude of these vectors can differ depending on the number of tokens, the composition of the input, or the distribution of attention within the model.

If differences in vector magnitude are not addressed, they can distort similarity measurements and affect the behavior of models that rely on distance-based computations. For example, longer sequences or certain token patterns may produce embeddings with disproportionately large norms, introducing bias that is unrelated to the actual semantic content of the input. This concern is especially relevant in the context of this study, where entity representations are obtained using both mean pooling over token embeddings and the [CLS] token. While mean pooling averages across tokens, the final vector can still be influenced by input length or structure. Similarly, the [CLS] token embedding may vary depending on the extent and distribution of contextual information. Without any adjustment, such variability can lead to inconsistencies when comparing embeddings across different entities.

To address this, all contextual embeddings are normalized using L2 normalization, which scales each vector to unit length by dividing it by its L2 norm. This ensures that every embedding lies on the surface of the unit hypersphere in the vector space, regardless of the original sequence characteristics. Normalization is particularly important when using cosine similarity, as this metric depends on the angle between vectors and assumes a consistent scale. Without normalization, similarity scores

would be affected by both magnitude and direction, making them less interpretable.

Following the extraction and normalization of contextual embeddings, the resulting data were stored to disk to ensure reproducibility and reduce computational overhead in later stages of the study. The embeddings, together with their corresponding similarity scores, were serialized using PyTorch's .pt format, which preserves the tensor structure and allows for direct reuse during training and evaluation. Each file consists of a dictionary containing two elements: one for the normalized embeddings and one for the associated similarity labels. The files were systematically organized by model type, embedding strategy, and data split, allowing for consistent access and clear separation across experimental conditions.

4.4 Transformations

Inspired by the approach proposed by Munda et al. (2025), this work extends the originally monolingual factual probing methodology to a multilingual setting by introducing a factual probe for three languages and applying it to both encoder-only and encoder–decoder multilingual PLMs, which were not considered in the original approach.

The factual probe is defined as a linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$, realized through a learnable matrix $B \in \mathbb{R}^{n \times n}$, which is applied to the input embedding $h \in \mathbb{R}^n$ to produce a transformed embedding $t \in \mathbb{R}^n$:

$$t = Bh$$

The parameters of the matrix B are set by a uniform distribution and optimized via gradient descent. The training objective is to approximate the cosine similarity between transformed embeddings of entity pairs in the PLM space to the similarity values computed from reference knowledge graph embeddings. Formally, the loss function is given by Equation 58 in Munda et al. (2025).

$$L = \min_B \frac{1}{N} \sum_{\ell=1}^{N} \left| s_T(e_s^{(\ell)}, e_o^{(\ell)}) - s_B(t_s^{(\ell)}, t_o^{(\ell)}) \right|^2 \tag{58}$$

Here, $s_T(e_s, e_o)$ denotes the gold-standard similarity between the subject and object in the knowledge graph embedding space, and $s_B(t_s, t_o)$ is the predicted similarity based on the transformed contextual embeddings. The dimensionality of the embeddings is preserved throughout to ensure that no information is artificially injected or lost during transformation.

Empirical evaluation reveals that the transformed vector spaces yield significantly higher correlation with the reference similarities, particularly when using BERT's

[CLS] token representations. Both encoder-based (BERT) and decoder-based (GPT-2) models exhibit improvements, though BERT consistently achieves stronger alignment with factual associations, suggesting that bidirectional context and masking objectives may better facilitate the encoding of factual information.

4.5 Training and Optimization

The following section outlines the procedures and configurations used for training the models applied in this study. It first details the training of MTransE for obtaining language-specific knowledge graph embeddings, followed by the optimization of the linear transformation matrices used in the probing experiments. Particular attention is given to hyperparameter selection, optimization strategies, and configuration choices to ensure reproducibility and comparability with prior work.

4.5.1 Training MTransE

To obtain language-specific knowledge graph embeddings, MTransE (Chen et al., 2018) was trained on each language pair separately. Two bilingual configurations were used: English-German (en-de) and English-Russian (en-ru). In both cases, the model was configured to preserve the relational structure of the underlying knowledge graph while disabling cross-lingual alignment, ensuring that embeddings for each language were learned independently.

The training was conducted using the TensorFlow implementation of MTransE with the hyperparameter settings shown in Table 4. These values correspond to the original configurations reported by Chen et al. (2018), which were obtained through systematic evaluation across several language pairs and datasets. Adopting these settings ensures methodological consistency with prior work and is supported by evidence of their effectiveness in multilingual scenarios. As a result, two separate monolingual knowledge graphs, one per language, were used to independently learn 50-dimensional entity and relation embeddings. The model was trained using the standard TransE loss, which preserves relational semantics by minimizing the distance between head, relation, and tail embeddings, as shown in Equation 59.

$$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|^2 \tag{59}$$

A margin of $m_1 = 0.5$ was applied to enforce separation between positive and negative triples, while the Bernoulli negative sampling was used to dynamically corrupt either the head or tail entity during training.

The models were trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. The batch size for the structural loss was set to 128. The resulted embeddings were later used to compute cosine similarities between subject

and object entity pairs, forming the basis for constructing supervision signals in downstream probing tasks. The alignment loss component was disabled by setting the alignment weight $a_1 = 0.0$, and the alignment fold parameter (`AM_fold`) was set to zero. The structural loss weight a_2 was fixed at 0.5. An L_2 distance function (`L1=False`) was used for computing the energy score of each triple. Periodic loss reporting was enabled every 150 mini-batches. Training was executed on a single GPU.

At the end of training, each bilingual configuration produced two checkpoint files, one containing the English embeddings and the other containing embeddings for the target language, such as German or Russian. These trained embeddings were subsequently used to compute cosine similarities between subject and object entity pairs. The resulting similarity scores served as supervision signals for downstream probing via linear transformation.

Table 4: Hyper-parameter configuration for MTransE training with cross-lingual alignment disabled

Parameter	**Value**
Embedding dimension (d)	50
Batch size — KG triples (B_{K})	128
Batch size — alignment pairs (B_{A})	64
Learning rate	0.001
Optimizer	Adam
Margin for KG loss (m_1)	0.5
Alignment weight (a_1)	0.0
Alignment margin (a_2)	0.5
Alignment mini-batch fold (AM_fold)	0
Distance metric	L_2 (`L1 = False`)
Epochs	100
Save frequency	every 100 epochs
Half-loss samples / epoch	150

4.5.2 Training LT

Model implementation and optimization were carried out using the deep learning framework PyTorch (Paszke et al., 2019). To refine the training process, a manual hyperparameter tuning procedure was employed for each linear transformation matrix. This process focused on three key parameters known to influence the effectiveness of the learned transformations: the initial learning rate, the weight decay coefficient, and the number of training epochs. This involved empirical testing

and iterative adjustments across different PLM–language configurations, guided by observed performance on validation data. The initial learning rate was selected from a range between 5e-4 and 5e-3, depending on the model–language configuration. To further improve convergence and enhance the likelihood of reaching a local minimum, learning rate scheduling was applied during training. Specifically, the ReduceLROnPlateau strategy was used to adaptively lower the learning rate when validation performance plateaued. The scheduler monitored a selected metric and reduced the learning rate by a factor of 0.1 if no improvement was observed for 5 consecutive epochs. This approach allowed training to begin with a relatively high learning rate, promoting exploration, and then gradually refine the parameter space as optimization progressed.

Table 5 presents the experimentally identified hyperparameter settings applied during training, including the number of epochs, weight decay, and learning rate scheduler configuration. In addition, Table 6 details the specific learning rate values and resulting performance metrics, such as validation loss and Pearson correlation coefficient, across different combinations of model architectures and languages.

Table 5: Overview of training hyperparameters and their corresponding values

Hyperparameter	**Value**
Nr. of training epochs	100
Learning rate	5e-4 $\leq x \leq$ 5e-3
Learning rate scheduler factor	1e-1
Learning rate patience	5
Weight decay	5e-3
Early stopping patience	10

4.6 Evaluation methods

This section outlines the evaluation strategies used to assess the effectiveness of the learned transformations. It includes quantitative metrics for measuring how well the predicted similarity scores align with reference values, visualization techniques to explore the structure of the embedding spaces, and qualitative analysis to examine the properties and limitations of the transformed representations

4.6.1 Quantitative Evaluation

The transformed output vectors produced by the models are compared using cosine similarity to generate a scalar value that reflects the semantic closeness between

Table 6: Performance comparison of different model–language configurations based on validation loss, Pearson correlation coefficient, and selected learning rates for training. The scores are computed for embeddings containing only entity names (Ent.) and for embeddings augmented with corresponding Wikipedia descriptions (Wiki.)

Emb.	**Lang.**	**↓Loss**	**↑PCC**	**LR**
mBERT-cls (Wiki.)	en	0.0693	0.7540	5e-3
	de	0.0551	0.7799	1e-3
	ru	0.0581	0.8360	1e-3
	en2	0.0397	0.8182	5e-4
mBERT-cls (Ent.)	en	0.0430	0.8121	1e-3
	de	0.0556	0.7702	1e-3
	ru	0.0594	0.8282	1e-3
	en2	0.0377	0.8228	1e-3
mBERT-avg (Wiki.)	en	0.0439	0.8118	1e-3
	de	0.0546	0.7794	1e-3
	ru	0.0586	0.8327	1e-3
	en2	0.0392	0.8192	1e-3
mBERT-avg (Ent.)	en	0.0425	0.8135	1e-3
	de	0.0546	0.7736	1e-3
	ru	0.0576	0.8332	1e-3
	en2	0.0376	0.8227	1e-3
mBART-avg (Wiki.)	en	0.0528	0.7814	7e-4
	de	0.0610	0.7559	6e-4
	ru	0.0628	0.8211	7e-4
	en2	0.0505	0.7868	9e-4
mBART-avg (Ent.)	en	0.0559	0.7708	1e-3
	de	0.0672	0.7226	1e-3
	ru	0.0700	0.7979	1e-3
	en2	0.0504	0.7785	9e-4

pairs of entities. These predicted similarity scores are then evaluated against gold standard similarity values to assess how accurately the model captures relational information. Since the task involves predicting continuous values, standard regression evaluation metrics are applied. Specifically, the Pearson correlation coefficient is used to measure linear correlation between predicted and reference scores, while Spearman's rank correlation assesses the consistency of their relative orderings. Additionally, the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are computed to quantify the average and scaled magnitude of prediction errors, respectively.

Pearson Correlation Coefficient. The Pearson Correlation Coefficient (PCC), sometimes referred to as Pearson's r, measures the linear correlation between two sets of values. In the analyses that follow, the first variable comprises the similarity scores predicted from the transformed contextual embeddings ($\hat{Y}$), while the second consists of the reference similarity scores derived from the labelled knowledge-graph dataset (Y).

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{60}$$

In Equation 60, x_i represents the predicted similarity score for the i-th entity pair, and y_i is the corresponding true similarity score. The values $\bar{x}$ and $\bar{y}$ denote the mean (average) of the predicted and true scores, respectively. The numerator measures how much the predicted and true scores vary together, while the denominator scales this value by the overall variability in each set of scores. The coefficient takes values in the range $[-1, 1]$, where a value close to 1 indicates a strong positive linear relationship, meaning that as one variable increases, the other also increases. A value close to -1 indicates a strong negative linear relationship, where an increase in one variable corresponds to a decrease in the other. A value near 0 suggests that there is no linear relationship between the two variables. The computation of the Pearson Correlation Coefficient is performed using the pearsonr function from the open-source scientific computing library SciPy (Virtanen et al., 2020). During training and validation, the Pearson correlation coefficient is computed alongside the loss to provide an additional quantitative indicator of model performance.

Spearman's Rank Correlation Coefficient. In addition to Pearson's r, this thesis employs Spearman's Rank Correlation Coefficient (SRCC), denoted as ρ, as a complementary evaluation metric. Unlike PCC, which assesses linear relationships based on actual values, SRCC evaluates the strength and direction of the relationship between two variables by comparing the ranks of their values. This allows it to capture monotonic associations, even when the underlying relationship is non-linear.

In this study, SRCC is computed between the predicted similarity scores $\hat{Y}$, derived from the transformed contextual embeddings, and the true similarity scores Y, obtained from the labelled knowledge-graph dataset. Prior to correlation, both sets of scores are converted into ranks, as shown in Equation 61.

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{61}$$

Here, d_i denotes the difference between the ranks of the predicted and true scores for the i-th entity pair, and n is the total number of entity pairs. The coefficient takes values in the range $[-1, 1]$, where $\rho = 1$ indicates perfect agreement in rankings, $\rho = -1$ indicates completely reversed rankings, and $\rho = 0$ suggests no association between the rankings. The inclusion of SRCC is particularly relevant in this context, as the computation of cosine similarity between vector representations is not a linear operation in the algebraic sense. Therefore, relying solely on PCC may overlook cases where the model correctly ranks the similarity between entity pairs without reproducing the precise numerical values. SRCC addresses this limitation by focusing on ordinal agreement, offering a more flexible measure of predictive accuracy. As with Pearson's r, the metric is computed using the `spearmanr` function from the SciPy library (Virtanen et al., 2020).

Mean-Squared Error. Mean-Squared Error (MSE) is used to evaluate how far the predicted similarity scores deviate from the actual reference scores. Unlike Pearson's and Spearman's correlation measures, which focus on the trend or rank consistency between predicted and true values, MSE quantifies the magnitude of prediction errors directly. It computes the average of the squared differences between each predicted and actual score, giving more weight to larger errors.

A lower MSE indicates that the predicted similarity scores are numerically closer to the true scores, reflecting better model performance in terms of precision. In this study, MSE is used both as an evaluation metric and as the loss function during model training. The values are computed using the `mean_squared_error` function from the Scikit-learn library (Pedregosa et al., 2011).

Root Mean-Squared Error. Root Mean-Squared Error (RMSE) provides an interpretable way to assess the prediction error by expressing it in the same unit as the predicted and true similarity scores. It is obtained by taking the square root of the Mean-Squared Error (MSE), which helps to bring the metric back to the original scale of the data. Similarly to MSE, RMSE reflects the average magnitude of the differences between predicted and actual values, but without squaring those differences in the final output. The key advantage of RMSE is that it gives a more intuitive sense of the typical prediction error. Because it penalizes larger

errors more than smaller ones, due to the squaring step before taking the root, it is particularly useful when larger deviations are more undesirable. In this study, RMSE is used alongside MSE, Pearson's r, and Spearman's ρ to provide a more complete view of model accuracy. It is computed using the `mean_squared_error` function from Scikit-learn with the `squared=False` option, which directly returns the root of the average squared error.

4.6.2 Visualizations

t-distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton, 2008) is a non-linear dimensionality reduction technique that extends the original Stochastic Neighbor Embedding (SNE) algorithm proposed by Hinton and Roweis (2002). t-SNE enables the visualization of high-dimensional data in two or three dimensions by preserving local structure in the low-dimensional embedding. For each pair of high-dimensional points $\mathbf{x}_i$ and $\mathbf{x}_j$ it defines a conditional similarity, as shown in Equation 62.

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2\right)} \tag{62}$$

Here, the bandwidth σ_i is chosen so that the entropy of each distribution $P_i = (p_{j|i})_j$ matches a user-specified perplexity, typically between 5 and 50. To obtain symmetric similarities in the high-dimensional space, the conditional probabilities are averaged and normalised, as depicted in Equation 63.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{63}$$

In Equation 63, N denotes the total number of data points. The resulting values p_{ij} form a joint probability distribution over all distinct pairs and therefore sum to one.

Each point $\mathbf{x}_i$ is mapped to a low-dimensional vector $\mathbf{y}_i$. Similarities in the map are computed using a heavy-tailed Student's t distribution, as in Equation 64.

$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}} \tag{64}$$

The heavy tails allocate extra area to moderately separated points, mitigating the crowding that arises when many high-dimensional neighbours must share a small two-dimensional region.

The mapping $\{\mathbf{y}_i\}$ is obtained by minimizing the Kullback–Leibler divergence (van der Maaten and Hinton, 2008), as shown in Equation 65.

$$KL(P \,\|\, Q) \;=\; \sum_i \sum_j p_{ij} \, \log \frac{p_{ij}}{q_{ij}}, \tag{65}$$

Here, $KL(P \,\|\, Q)$ quantifies the information lost when the map-space distribution Q, whose elements are q_{ij}, is used to approximate the original similarity distribution P, defined by the values p_{ij}.

Gradient descent with momentum is typically used to minimize the KL divergence. The gradient with respect to point $\mathbf{y}_i$ is given in Equation 66.

$$\frac{\partial \, KL}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij}) \, \frac{\mathbf{y}_i - \mathbf{y}_j}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2} \tag{66}$$

Optimization begins by randomly initializing the low-dimensional vectors $\mathbf{y}_i$ with small Gaussian noise. During the initial phase of training, the high-dimensional similarities p_{ij} are intentionally magnified, for example, by a factor of four. This technique helps to spread out clusters in the map and prevents them from collapsing into a single region. Once the global layout stabilizes, the exaggeration factor is removed, allowing the model to fine-tune the positions of the points. The resulting low-dimensional representation retains the local structure of the original data while separating dissimilar groups, thereby enabling clear visual identification of clusters and patterns.

The `TSNE` class in the Scikit-learn library provides an efficient implementation of the t-SNE algorithm, incorporating the optimization and similarity computation procedures described in Equations 62-66. It enables the transformation of high-dimensional vectors into a low-dimensional representation, which can then be visualized using standard plotting libraries, such as Matplotlib (Hunter, 2007).

4.6.3 Qualitative Evaluation

While the primary evaluation relies on quantitative metrics to assess how well the linear transformation approximates reference similarity scores, these figures alone do not provide a full account of the underlying model behavior. To complement the numerical analysis, a targeted qualitative evaluation was conducted. This step is essential, particularly in a multilingual setup, where the ability to preserve factual associations across language variants depends not only on numerical similarity but also on the semantic coherence of the transformed embeddings.

The qualitative analysis focuses on subject–object pairs that show the largest discrepancy between predicted and reference similarity values. By examining the 20 lowest-scoring examples for each model variant, it becomes possible to investigate which types of relations or entity combinations consistently resist alignment through the linear transformation. The goal is not to evaluate the transformation in isolation,

but rather to probe whether the representations produced by the underlying PLM encode sufficient factual information for the transformation to approximate semantic similarity at all.

This analysis is particularly relevant in the multilingual context of this study. Since English embeddings are transformed under different training conditions, reviewing the worst-performing cases helps determine whether the transformed English representations behave consistently across settings. Identifying such patterns enables a more fine-grained understanding of where factual associations remain stable and where they begin to diverge.

5 Results

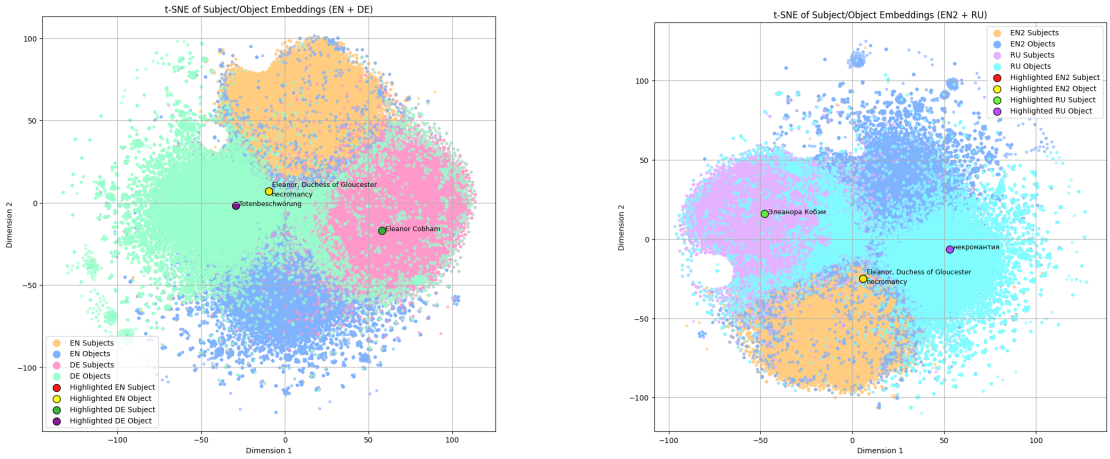
This chapter presents the results obtained using the methodology outlined in Chapter 4. Section 5.1 summarizes the training performance of both the knowledge graph and transformation models. Section 5.2 presents the quantitative evaluation results, focusing on how well the transformed PLM embeddings align with KG-derived similarity scores. Section 5.3 provides a visual analysis of the learned vector spaces and transformation matrices. Lastly, Section 5.4 offers a qualitative assessment based on selected entity pairs and relation types.

5.1 Training Results

Figure 6 presents two t-SNE projections of subject and object embeddings trained with MTransE on 60,000 triples for English-German (left) and English-Russian (right). Each model was trained separately, reflecting the bilingual nature of MTransE, which processes only two languages at a time. The visualizations depict subject and object embeddings color-coded by language and entity role. The highlighted annotations indicate selected entity pairs that appear in identical triples in all three languages.

As discussed in Section 4.1.1, removing the alignment loss from MTransE results in fully decoupled embedding spaces, where each language learns its own relational structure without any direct cross-lingual alignment constraints. This pattern is evident in both plots, where entities from different languages, although projected into the same vector space, form clusters that exhibit a strong language-wise separation. The absence of an explicit alignment signal prevents the model from learning consistent cross-lingual representations, leading to largely disjoint semantic regions per language.

To further explore this phenomenon, selected subject–object pairs that appear in identical triples across all three language subsets are highlighted in the figure to illustrate how their multilingual representations are distributed in the embedding space. One such example is the triple `[Eleanor, Duchess of Gloucester, convicted of, necromancy]`. In this case, both the subject *Eleanor, Duchess of Gloucester* and the object *necromancy* appear in the visualizations for the English–German and English–Russian training pairs. The position of *Eleanor, Duchess of Gloucester* and *Necromancy* varies across language pairs: in the English–German



(a) en-de (b) en-ru

Figure 6: A two-dimensional (t-SNE) projection of subject and object embeddings trained with MTransE for two language pairs

space, the two embeddings remain relatively close, with their German counterparts *Eleanor Cobham* and *Totenbeschwörung* appearing nearby. In contrast, in the English–Russian space, the embeddings of *Eleanor, Duchess of Gloucester* and its Russian counterpart *Элеонора Кобэм* are more widely separated, as are *necromancy* and *некромантия*.

Furthermore, the t-SNE projections reveal a notable structural regularity in how subject and object embeddings are distributed. Although the model is not explicitly trained to separate subjects from objects, their spatial distribution reveals emergent regularities: subject embeddings tend to coalesce into contiguous areas that are partially distinct from those formed by objects.

5.2 Quantitative Results

This section presents a quantitative evaluation of multilingual knowledge graph embeddings, focusing on the capacity of linear transformation to approximate semantic similarity across languages. To contextualize the inclusion of both `en` and `en2` in Table 8, it is important to recall the experimental setup described in Section 4.1.1. As outlined there, the bilingual design of MTransE required separate training runs for English–German and English–Russian language pairs. Although the English dataset, model architecture, and hyperparameters were kept constant across these runs, the inherent nondeterminism of the training process, which originates from stochastic weight initialization, random negative sampling, and GPU-based optimization, led to small but non-negligible differences in the

resulting English embeddings. The embedding set denoted `en` originates from the English–German training, while `en2` was produced during the English–Russian training. The availability of two independently trained English embedding sets also allows for an internal consistency check of the learned representations. By comparing `en` and `en2`, it becomes possible to evaluate the stability of the model's outputs across bilingual training runs involving different language pairs.

Table 7: Evaluation metrics for English (en, en2) embeddings and other languages (de, ru) without applying the linear transformation. Δ columns show differences computed as $score_{lang} - score_{en}$, where positive values indicate higher scores than English and negative values indicate lower scores. Results are reported for name-only embeddings (Ent.) and embeddings augmented with Wikipedia descriptions (Wiki)

Model	**Metric**	en	en2	Δen2	de	Δde	ru	Δru
mBERT-cls	PCC (Wiki)	0.4717	0.4551	-0.0166	0.3271	-0.1446	0.3168	-0.1549
	PCC (Ent.)	0.3152	0.3200	+0.0048	0.1806	-0.1346	0.3388	+0.0236
	SRCC (Wiki)	0.4686	0.4530	-0.0156	0.3497	-0.1189	0.3458	-0.1228
	SRCC (Ent.)	0.3280	0.3257	-0.0023	0.2240	-0.1040	0.4134	+0.0854
	MSE (Wiki)	0.4413	0.3711	-0.0702	0.6318	+0.1905	0.7377	+0.2964
	MSE (Ent.)	0.3493	0.2891	-0.0602	0.4907	+0.1414	0.6157	+0.2664
	RMSE (Wiki)	0.6643	0.6092	-0.0551	0.7948	+0.1305	0.8589	+0.1946
	RMSE (Ent.)	0.5910	0.5377	-0.0533	0.7005	+0.1095	0.7847	+0.1937
mBERT-avg	PCC (Wiki)	**0.4937**	**0.4786**	-0.0151	**0.4390**	-0.0547	**0.4782**	-0.0155
	PCC (Ent.)	0.3845	0.3835	-0.0010	0.3605	-0.0240	0.4753	+0.0908
	SRCC (Wiki)	**0.4946**	**0.4781**	-0.0165	**0.4205**	-0.0741	0.4637	-0.0309
	SRCC (Ent.)	0.3858	0.3833	-0.0025	0.3448	-0.0410	**0.4828**	+0.0970
	MSE (Wiki)	0.3058	0.2505	-0.0553	0.3683	+0.0625	0.4884	+0.1826
	MSE (Ent.)	**0.2015**	**0.1618**	-0.0397	**0.2360**	+0.0345	**0.2858**	+0.0843
	RMSE (Wiki)	0.5530	0.5005	-0.0525	0.6068	+0.0538	0.6988	+0.1458
	RMSE (Ent.)	**0.4489**	**0.4023**	-0.0466	**0.4858**	+0.0369	**0.5346**	+0.0857
mBART-avg	PCC (Wiki)	0.4027	0.3902	-0.0125	0.3712	-0.0315	0.4377	+0.0350
	PCC (Ent.)	0.1541	0.1524	-0.0017	0.1539	-0.0002	0.1406	-0.0135
	SRCC (Wiki)	0.4181	0.4094	-0.0087	0.3682	-0.0499	0.4362	+0.0181
	SRCC (Ent.)	0.1650	0.1582	-0.0068	0.1463	-0.0187	0.1305	-0.0345
	MSE (Wiki)	0.6683	0.5780	-0.0903	0.6874	+0.0191	0.7031	+0.0348
	MSE (Ent.)	0.6606	0.5711	-0.0895	0.7033	+0.0427	0.7120	+0.0514
	RMSE (Wiki)	0.8175	0.7603	-0.0572	0.8291	+0.0116	0.8385	+0.0210
	RMSE (Ent.)	0.8128	0.7557	-0.0571	0.8387	+0.0259	0.8438	+0.0310

Table 7 presents the difference in performance between `en` and `en2`, and between `en` and other languages (`de`, `ru`), across four evaluation metrics: Pearson Correlation Coefficient (PCC), Spearman's Rank Correlation Coefficient (SRCC), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Lower differences indicate higher similarity between the language-specific embeddings, as measured by their behavior on similarity prediction task.

To compute these differences, the metric value obtained for English, denoted x_{en}, is compared to the corresponding value for a target language, denoted x_{lang}. The difference is then calculated as shown in Equation 67.

$$\Delta(x) = x_{\text{lang}} - x_{\text{en}} \tag{67}$$

The results show that the differences between `en` and `en2` are consistently smaller than those observed for `de` or `ru`, especially for the `mBERT-cls` and `mBERT-avg` models. This suggests that the English embeddings generated in two independent training runs (`en` and `en2`) are highly consistent, even in the absence of strict determinism in training. The low deltas in PCC, SRCC, and error-based metrics indicate that the overall similarity structure within the embedding space is largely preserved across runs.

By contrast, the larger differences between `en` and non-English languages reflect the variability introduced by different input triples and language-specific entity distributions. Notably, in the `mBART-avg` setting, certain metrics for German approach the consistency observed for `en2`, suggesting model-specific differences in how multilingual information is encoded.

Table 8 presents the evaluation scores for three multilingual models (mBERT-cls, mBERT-avg, and mBART-avg) across English (`en`, `en2`), German (`de`), and Russian (`ru`), using both the original and the transformed embedding spaces. For each configuration, two input variants were considered: entity names only (Ent.) and entities augmented with their corresponding Wikipedia descriptions (Wiki.). Results are reported using Pearson and Spearman Correlation Coefficients (PCC, SRCC), as well as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), allowing a comprehensive comparison of prediction accuracy and ranking consistency.

In the original space, all models exhibit modest performance, with relatively low correlation scores and high error values. Among the original embeddings, mBERT-avg consistently outperforms mBERT-cls and mBART-avg across all languages and input types. Notably, the inclusion of Wikipedia descriptions generally improves performance over name-only representations, particularly for German and Russian. In the transformed space, however, the relative advantage of Wikipedia descriptions over name-only representations diminishes, with several configurations showing comparable or even slightly higher scores for name-only embeddings. However, mBART-avg performs the weakest overall in the untransformed setting, with negligible correlation scores and high error rates across languages.

After applying the transformation, a substantial improvement is observed across all models and languages, reflected consistently in both correlation and error-based metrics. In the case of mBERT-cls, the Pearson correlation coefficient (PCC) for English (`en`) increases from 0.4717 (Wiki) and 0.3152 (Ent.) to 0.7511 and 0.8099,

Table 8: Evaluation results for the original (top) and transformed (bottom) embedding spaces. Scores are reported for entity representations based on names only (Ent.) and for representations augmented with Wikipedia descriptions (Wiki.)

		↑PCC		↑SRCC		↓MSE		↓RMSE	
Orig.	Lang.	Wiki.	Ent.	Wiki.	Ent.	Wiki.	Ent.	Wiki.	Ent.
mBERT-cls	en	0.4717	0.3152	0.4686	0.3280	0.4413	0.3493	0.6643	0.5910
	de	0.3271	0.1806	0.3497	0.2240	0.6318	0.4907	0.7948	0.7005
	ru	0.3168	0.3388	0.3458	0.4134	0.7377	0.6157	0.8589	0.7847
	en2	0.4551	0.3200	0.4530	0.3257	0.3711	0.2891	0.6092	0.5377
mBERT-avg	en	**0.4937**	0.3845	**0.4946**	0.3858	0.3058	**0.2015**	0.5530	**0.4489**
	de	**0.4390**	0.3605	**0.4205**	0.3448	0.3683	**0.2360**	0.6068	**0.4858**
	ru	**0.4782**	0.4753	0.4637	**0.4828**	0.4884	**0.2858**	0.6988	**0.5346**
	en2	**0.4786**	0.3835	**0.4781**	0.3833	0.2505	**0.1618**	0.5005	**0.4023**
mBART-avg	en	0.4027	0.1541	0.4181	0.1650	0.6683	0.6606	0.8175	0.8128
	de	0.3712	0.1539	0.3682	0.1463	0.6874	0.7033	0.8291	0.8387
	ru	0.4377	0.1406	0.4362	0.1305	0.7031	0.7120	0.8385	0.8438
	en2	0.3902	0.1524	0.4094	0.1582	0.5780	0.5711	0.7603	0.7557
		↑PCC		↑SRCC		↓MSE		↓RMSE	
Transf.	Lang.	Wiki.	Ent.	Wiki.	Ent.	Wiki.	Ent.	Wiki.	Ent.
mBERT-cls	en	0.7511	0.8099	0.7508	0.7878	0.0695	0.0424	0.2636	0.2060
	de	0.7746	0.7762	0.7760	0.7703	0.0564	0.0544	0.2374	0.2333
	ru	0.8301	0.8294	0.8229	0.8157	0.0580	0.0573	0.2407	0.2394
	en2	0.8082	0.8141	0.7868	**0.7869**	0.0410	0.0387	0.2024	0.1968
mBERT-avg	en	0.8049	**0.8132**	0.7888	**0.7895**	0.0443	**0.0414**	0.2104	**0.2036**
	de	**0.7788**	**0.7788**	**0.7784**	0.7732	0.0547	**0.0535**	0.2338	**0.2314**
	ru	0.8287	**0.8344**	**0.8197**	**0.8197**	0.0579	**0.0554**	0.2405	**0.2355**
	en2	0.8040	**0.8143**	0.7813	0.7854	0.0415	**0.0385**	0.2036	**0.1962**
mBART-avg	en	0.7738	0.7653	0.7666	0.7487	0.0531	0.0543	0.2304	0.2330
	de	0.7538	0.7235	0.7549	0.7180	0.0616	0.0669	0.2482	0.2586
	ru	0.8124	0.8041	0.8012	0.7888	0.0638	0.0661	0.2525	0.2571
	en2	0.7743	0.7759	0.7634	0.7575	0.0502	0.0480	0.2240	0.2192

respectively, while the RMSE is reduced from 0.6643 and 0.5910 to 0.2636 and 0.2060. Comparable trends are evident for the German (`de`) and Russian (`ru`) subsets, with German entity-only embeddings improving from a PCC of 0.1806 to 0.7762 and Russian embeddings rising from 0.3388 to 0.8294. These results demonstrate that the transformation consistently enhances the model's capacity to capture semantic similarity, even in settings where the original vector spaces perform poorly. Notably, the German entity-only RMSE drops from 0.7005 to 0.2333, while the Russian variant drops from 0.7847 to 0.2394.

Improvements are also prominent for the average-based mBERT-avg variant. In the Russian subset, for instance, PCC rises from 0.4782 (Wiki) and 0.4753 (Ent.) to 0.8287 and 0.8344, respectively, with MSE decreasing from 0.4884 and 0.2858 to 0.0579 and 0.0554. The secondary English set (`en2`) shows similar trends: the mBERT-avg model improves from 0.4786 to 0.8040 (Wiki) and from 0.3835 to 0.8143 (Ent.), reflecting strong generalization of the transformation. Finally, even the mBART-avg model, which underperforms in the original configuration, demonstrates substantial gains. For example, in the Russian subset, PCC rises from 0.4377 to 0.8124 (Wiki), and RMSE falls from 0.8385 to 0.2525. Although the improvement is slightly more modest for German, with entity-only PCC increasing from 0.1539 to 0.7235, the general trend remains consistent across all evaluated conditions.

The application of the linear transformation substantially improves the alignment between model-derived similarity scores and the ground-truth cosine similarities computed from knowledge graph embeddings. This improvement holds across all evaluated languages and models, but the degree of improvement is not uniform. Interestingly, Russian consistently achieves the highest post-transformation scores, with Pearson correlation coefficients (PCC) exceeding 0.83 for both mBERT-cls (0.8301) and mBERT-avg (0.8287) when using Wikipedia-enhanced embeddings. This suggests that, for Russian, the latent vector space encoded by the PLMs already contains rich factual associations that the transformation is able to uncover and align effectively with the reference signal.

English and the secondary English set (`en2`) also benefit considerably from the transformation, particularly in the mBERT variants. For instance, mBERT-avg reaches PCC values above 0.80 in both settings (0.8049 for `en`, 0.8040 for `en2` with Wiki inputs), while the RMSE drops below 0.21 in all cases. These results confirm that the PLMs encode substantial factual knowledge for English, and that the transformation successfully renders this knowledge more linearly accessible. By contrast, German, although improved, shows slightly lower performance across all metrics. For example, the PCC for mBERT-avg (Wiki) reaches 0.7788 in German, compared to 0.8049 in English and 0.8287 in Russian, and the corresponding RMSE remains marginally higher (0.2338 vs. 0.2104 and 0.2405).

Taken together, these findings indicate that factual knowledge is implicitly embedded in the vector spaces of multilingual PLMs, and that it can be made linearly accessible through transformation. The strength of this alignment varies by language, likely reflecting the underlying language-specific training data and model capacity. Russian exhibits the clearest signal, followed by English (`en`, `en2`), while German lags slightly behind. These differences suggest a language-dependent variance in how factual associations are encoded, and provide further evidence for the potential of probing methods to uncover structured knowledge in PLMs.

5.3 Visual Results

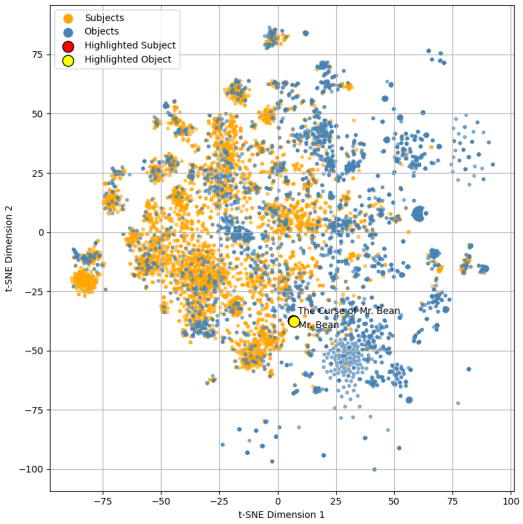
To address the challenge of visual clutter in large-scale t-SNE plots, two representative entities are randomly selected and individually traced before and after the transformation. By monitoring their positional changes in the embedding space, it becomes possible to examine how the transformation affects their angular relationship. Since cosine similarity is based on the angle between vectors rather than their magnitude, the primary interest lies in whether the entities move closer in terms of angular proximity.

The four t-SNE visualizations in Figure 7 illustrate how the linear transformation reshapes the entity-only mBERT-avg embedding space for English (top row) and Russian (bottom row). In each plot, subjects are colored orange, objects blue, and one subject–object pair is highlighted in red and yellow, respectively. The purpose of these visualizations is twofold: to assess how the transformation affects the global organization of the embedding space, and to examine how it brings semantically related entities closer in angular proximity, consistent with the cosine similarity objective.

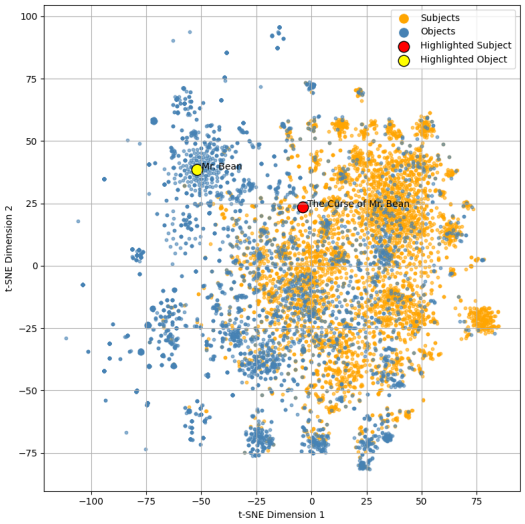
The English embedding space (Figure 7a) is relatively well-structured even before the transformation. Subject and object embeddings are densely distributed and occupy largely overlapping regions, particularly in the central part of the plot. While a mild directional separation is observable—with subjects slightly more concentrated in the upper left and objects drifting toward the lower right—this distinction is not strongly pronounced. The highlighted subject and object nearly coincide.

After transformation (Figure 7b) the embedding space becomes more compact and symmetrically distributed. The transformation realigns subject and object embeddings to reduce their angular divergence, producing a denser central cluster with improved intermixing of roles. The highlighted subject and object are now positioned closer together and exhibit reduced angular displacement.

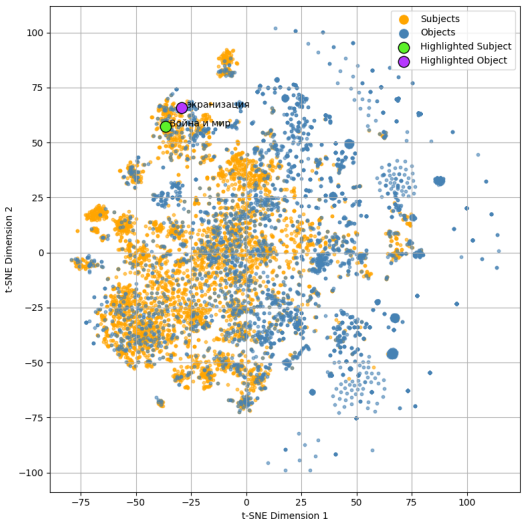
Before the transformation (Figure 7c), the embeddings are widely distributed across the space, with both subjects (orange) and objects (blue) appearing in



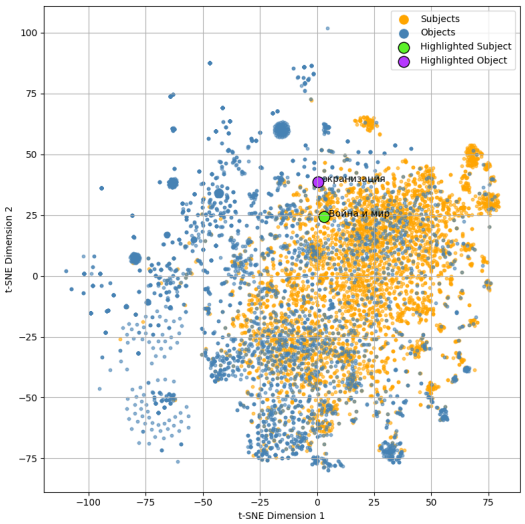
(a) Before transformation – en



(b) After transformation – en



(c) Before transformation – ru



(d) After transformation – ru

Figure 7: t-SNE projections of mBERT (avg) embeddings based on entity names only, before and after applying the linear transformation, for English (top row) and Russian (bottom row), illustrated with the English entities *The Curse of Mr. Bean* and *Mr. Bean*, and the Russian entities *Война и мир* (War and Peace) and *экранизация* (film adaptation)

overlapping regions. A denser concentration of embeddings is visible in the upper-left portion of the plot. The highlighted subject is located in the upper-left area, while the highlighted object appears closer to the plot center, positioned further to the right.

In the visualization for the Russian embeddings after transformation (Figure 7d) the distribution becomes more compact, with subject and object embeddings forming a tighter grouping around the center of the space. The highlighted subject and object move closer together and are now located near the central region, indicating a reduced distance between the two in the transformed embedding space.

5.4 Qualitative Results

The goal of the qualitative analysis is to identify consistent weaknesses at the level of both relation types and entities, and to highlight patterns that are specific to particular languages and model configurations. To this end, for each multilingual PLM and language, the 30 worst-performing data points were manually analyzed. Across six model configurations, such as mBART with entity labels only, mBERT with averaged entities, mBERT with [CLS] token entities, mBART with descriptions, mBERT with averaged descriptions, and mBERT with [CLS] token descriptions, a consistent subset of relation types emerges as particularly challenging.

The relation types such as `instance of`, `occupation`, and `member of sports team` are among the ten most represented in the dataset; nevertheless, they consistently rank among those with the lowest similarity predictions after the transformation. Their prominence in the training set does not appear to enhance the ability of the transformation to preserve their semantic structure. A closer analysis shows that many of these difficult cases involve biographical or geographic information, specifically:

- Biographical relations include `occupation`, `place of birth`, `place of death`, `given name`, `country of citizenship`, and `educated at`.
- Geographic relations include `country`, and `located in the administrative territorial entity`.

Relations such as `taxon rank`, `given name`, and `place of birth` also occur frequently among the misjudged triples, particularly in configurations that use additional textual descriptions. Notably, `given name` and `place of birth` are frequently misjudged primarily in models that use descriptions, whereas they are largely absent from the set of most frequently misjudged relations in entity-only setups.

While prior work has reported that English BART performs poorly on geographical relations, particularly `shares border with`, mBART shows only a small number of low-performing triples involving this relation. Examples include in English the triples `[Andorra, shares border with, Spain]` and `[Varallo Pombia, shares border with, Castelletto sopra Ticino]`, and in German the triples `[Cuisia, gemeinsame Grenze mit, Gizia]` and `[Étalante, gemeinsame Grenze mit, Aignay-le-Duc]`. Across configurations and languages, mBART shows recurring difficulties with a relatively small set of relations that can be grouped into three broad categories. The first category includes type and classification relations, such as `instance of`, `taxon rank`, and `parent taxon`. While the `twinned administrative body` relation poses difficulties for the monolingual BERT-avg configuration, it does not present challenges for any of the multilingual model variants in this study.

The second category covers personal attributes and affiliations, such as `occupation`, `given name`, `educated at`, `member of sports team`, and `cast member`. The third category relates to geopolitical or locational attributes, such as `country`, `country of citizenship`, `place of birth`, and `located in the administrative territorial entity`. Among these, `occupation` and `instance of` are the most consistently error-prone across all languages and configurations. Some variation is language-specific, for example, `member of sports team` is particularly problematic in Russian, `taxon rank` and `place of birth` are more prominent in English, and `country` and `place of birth` occur frequently in German. Description-based inputs tend to produce more errors for personal attributes, such as `given name`, whereas entity-only inputs lead to increased confusion over geopolitical relations, such as `country` and `country of citizenship`. These findings suggest that certain relation categories, in particular type and classification, as well as personal affiliation, present persistent challenges regardless of the input representation, while others are more dependent on the language or configuration used.

A closer examination of the triple `[Эбенезер Хаган, член спортивной команды, Ашанти Голд]` (Ebenezer Hagan, member of sports team, Ashanti Gold) reveals that the object is historically referred to as *Обуаси Голдфилдс* (Obuasi Goldfields), the club's former name prior to its official renaming to *Ашанти Голд* (Ashanti Gold) in 2004. In this example, the mBART may fail to resolve that both names denote the same entity, suggesting incomplete robustness to entity name changes or historical renamings.

For the triple `[Дайва Тушлайте, член спортивной команды, Forno d'Asolo Colavita]`, the object in English is *Forno d'Asolo Colavita*. Similar to the earlier example, there is potential for confusion arising from changes in the team's registered name over time; historical records indicate that the team competed under

the name *Forno d'Asolo Colavita* in 2009, whereas later records list it as *Forno d'Asolo - Astute*, reflecting shifts in co-sponsorship. In addition, the subject's name introduces further ambiguity, as it appears as *Дайва Рагажинскене* in Russian sources while in English-language records it may be given as *Daiva Tušlaitė-Ragažinskienė* or *Daiva Ragažinskienė*. Such discrepancies in both entity and team names may hinder the model's ability to consistently identify and link them, revealing incomplete robustness to variations due to transliteration, marriage name changes, co-sponsorship modifications, and historical renamings.

Similar to the triples discussed above, `[Nicolò Barella, member of sports team, Italy national under-16 football team]`, `[Jon McKain, Mitglied von Sportmannschaft oder -verein, Australische Fußballnationalmannschaft (U-23-Männer)]`, and `[Лукас Фернандес, член спортивной команды, Сборная Дании по футболу (до 17 лет)]` (Lukas Fernandes, member of sports team, Denmark national under-17 football team), and share a common characteristic, as each denotes membership in a national youth team, a relation that is inherently temporal and restricted to a specific stage of the athlete's career. Such affiliations are frequently replaced in later records by senior team associations, reducing their visibility in contemporary corpora. Consequently, models, such as mBART and mBERT, may struggle to retrieve them accurately, as this requires capturing the temporal validity of the relation and linking entities to historically accurate but no longer current team memberships.

In contrast to previous work that identified many entities labeled as English but presented in non-English forms, which posed challenges for an English-pretrained model, multilingual PLMs used in this study demonstrate improved handling of such cases. Nevertheless, entity names in foreign languages still occur in the triples, potentially complicating accurate linking across languages.

Furthermore, as previously observed, certain relations are grounded primarily in factual rather than linguistic similarity, providing few lexical cues for the model to exploit. In a multilingual setting, this challenge extends to relations, such as `instance of`, `taxon rank`, or historical sports team memberships, where the connection between subject and object relies on factual knowledge that is not readily inferable from surface forms. For example, in the English triple `[Andorra, shares border with, Spain]`, accurate resolution requires geographical knowledge of national borders rather than lexical similarity between the entity names. Such relations therefore demand explicit factual encoding in the model's representations, which may be insufficiently captured during pre-training.

Both mBART and mBERT, in their variants incorporating descriptions, encountered difficulty with the triple `[Louis Necker, sibling, Jacques Necker]` in English, as well as with the triple `[Takelot II., Geschwister, Namilt]` in German, in both entity-only and description embeddings. These cases suggest that

the `sibling` relation, particularly when expressed through less frequently occurring historical or non-Western entities, remains challenging for the models, even when enriched with descriptive information.

The relation `mother` is inaccurately approximated for only one triple, `[Eleonora Gonzaga, Mutter, Eleonora de' Medici]`, in German in the mBERT model, for both the average and [CLS] variants using entities only. Interestingly, although present in the dataset, relations such as `spouse`,`father` and `child`, did not present any difficulty for either mBERT or mBART.

A further examination of the entities involved in the most challenging triples reveals several recurring characteristics that likely contribute to model errors. Many correspond to historical figures, such as *Takelot II*, *Eleonora Gonzaga*, *Judah Dana* and *Charles Calvert, 3rd Baron Baltimore*, whose representation in contemporary multilingual corpora is limited. A number of these entities show orthographic and transliteration variability across languages, including Cyrillic to Latin alternations, such as *Сисоѣат Коссамак* versus *Sisowath Kossamak*, the presence or absence of diacritics, such as *Tušlaitė* and *Ragažinskienė*, or language specific name components such as *von* and *de*. Others pertain to non-Western or regionally localized contexts, such as Cambodian royalty, African or South American athletes, or rulers of smaller historical polities where coverage is sparse and naming conventions are inconsistent. Several cases involve temporally bound affiliations, most notably historical sports team memberships under former names, which require disambiguation across time, a capability not explicitly modelled by mBART or mBERT. Finally, relations such as `mother` and `sibling` in these contexts rely on genealogical or factual knowledge rather than on lexical co-occurrence, making them especially prone to errors arising from inconsistencies in name representation.

6 Discussion

This research aimed to investigate whether multilingual PLMs encode factual knowledge in a way that can be made explicitly accessible through a linear transformation. Specifically, the research investigated whether applying a linear transformation to contextual embeddings enables PLMs to approximate semantic similarity between knowledge graph entities, across multiple languages.

The results demonstrate that PLMs do indeed contain latent factual knowledge that can be rendered more accessible through a supervised linear mapping. Quantitative evaluations reveal consistent performance gains after transformation, as evidenced by improvements in Pearson correlation, Spearman correlation, and reductions in error metrics, such as MSE and RMSE, across English, German, and Russian. The findings hold across multiple model architectures and input strategies, e.g. using entity names versus Wikipedia descriptions. Notably, the transformed embeddings show the strongest alignment with KG-derived similarity scores in Russian and English, with German performing slightly less robustly. Qualitative analysis further highlights systematic challenges associated with specific relation types, such as `instance of`, `occupation`, and `member of sports team`, that persist across languages and model variants.

The observed quantitative improvements after applying the linear transformation confirm prior findings by Munda et al. (2025) for monolingual models, and extend them to a multilingual context. While the earlier research demonstrated that factual knowledge in English PLMs can be made more accessible via linear transformation, this study shows that similar techniques are effective for German and Russian, albeit with varying degrees of improvement. Consistent with the results observed for BERT in previous work, mBERT achieved the highest alignment scores; however, in contrast to the monolingual results reported by Munda et al. (2025), the average variant of mBERT outperformed its [CLS]-based counterpart, while mBART was consistently outperformed by mBERT across all evaluated settings.

Unexpectedly, high-frequency relations, such as `instance of` and `occupation`, which are well represented in the training dataset, remain among the most error-prone. This indicates that frequency alone does not ensure semantic consistency after transformation. These relation types often involve abstract or temporally dynamic attributes, such as professional roles and taxonomic categories, which may not be adequately captured by a simple linear mapping. Their complexity suggests that more expressive models or additional context-aware features may be needed

to resolve such semantic ambiguity.

It is importand to acknowledge that this study has several limitations. A first constraint stems from the absence of publicly available multilingual knowledge graph embeddings. As a result, obtaining the embeddings necessary for training the linear transformation required the prior training of MTransE models. These models were independently trained for each language pair (English–German and English–Russian) using a curated multilingual dataset. While this approach ensured alignment with the multilingual focus of the study, it introduced a dependency on custom-trained embeddings, which may limit the comparability of results across different setups. Furthermore, the method for training knowledge graph embeddings might have a considerable impact on the results, even if the overall trend of improving factual representations may still be observed.

Another important limitation relates to data availability. The requirement that each triple include textual labels and Wikipedia summaries in the target language led to a substantial reduction in the usable portion of the dataset. This filtering process, although necessary for ensuring semantic consistency in the probing task, significantly restricted the size and diversity of the final training set. As a consequence, the study concentrated on high-resource languages, such as English, German, and Russian, for which sufficient label coverage and textual descriptions could be obtained. The exclusion of lower-resourced languages limits the generalizability of the findings and prevents broader conclusions about the universality of the transformation technique across the full spectrum of linguistic diversity. Despite these constraints, the results remain valid. The consistent improvements observed across languages and PLM architectures provide robust evidence that factual knowledge is embedded in multilingual PLM representations and can be linearly extracted.

7 Conclusions

This thesis aimed to explore whether multilingual Pre-trained Language Models (PLMs), such as mBERT and mBART, encode factual knowledge in their hidden representations in a way that can be explicitly uncovered through a linear transformation. To address limitations of prior work and address the research questions, multilingual knowledge graph embeddings were trained on a multilingual dataset to obtain similarity scores between the entities. Subsequently, these scores served as a reference for training separate linear transformations for English, German, and Russian. Within this approach, the structural probing task involves taking the contextual embeddings of multilingual knowledge graph entities as input and applying a transformation to the vector space such that the resulting representations approximate the reference cosine similarity scores provided in a curated dataset consisting of 42,000 triples per language. The underlying assumption was that if the transformation demonstrates accurate similarity predictions, this would indicate that the latent spaces of multilingual PLMs already encode factual knowledge implicitly, acquired through exposure to large-scale data during pre-training.

The empirical analysis shows that, while multilingual PLMs demonstrate limited alignment for factual similarity in their original spaces, applying a learned linear transformation substantially improves alignment, the transformation consistently increases correlation with the reference similarity scores derived from multilingual knowledge graph embeddings and reduces prediction error across the evaluated settings. Among the examined configurations, mBERT-avg achieved the highest correlations and lowest errors, outperforming both the mBERT [CLS] variant and mBART in the transformed space. The qualitative analysis reveals that model errors frequently arise in cases involving historical or less widely represented entities, particularly when orthographic, transliteration, or naming variations occur across languages. Additional challenges are observed for entities associated with temporally bound affiliations or those situated in regionally localized contexts with sparse multilingual coverage. These factors indicate that variability in naming conventions, limited corpus representation, and temporal specificity substantially hinder the models' ability to produce consistent and accurate entity linking.

The primary contributions of this study include extending the investigation of factual knowledge in PLMs beyond English by constructing parallel datasets and evaluating models in English, German, and Russian, thereby enabling analysis of how factual knowledge is represented in a multilingual setting. It demonstrates that

a learned linear transformation can align multilingual PLM representations with the relational geometry of knowledge graphs, effectively decoding factual similarity across languages without the need to modify model parameters. In addition, it provides trained multilingual knowledge graph embeddings, previously unavailable as a public resource, along with the corresponding linear transformation, both of which can serve as reusable assets for further research in multilingual factual knowledge probing and neuro-symbolic integration.

Given the promising results obtained for high-resource languages, future research could explore whether similar patterns hold in lower-resource settings. Although low-resource languages typically suffer from limited coverage in external knowledge sources, such as Wikidata or Wikipedia, multilingual PLMs are pre-trained on vast and diverse corpora that often include at least partial representations of such languages. It would therefore be of significant interest to investigate whether factual knowledge is similarly embedded in PLM parameters for low-resource languages and whether it can be extracted using the same linear transformation techniques. Such investigations could offer new insights into the degree to which multilingual PLMs generalize across the resource spectrum and help identify strategies for enhancing factual representation in underrepresented languages.

Another promising direction for future work involves examining the cross-lingual generalization capabilities of multilingual PLMs. Instead of training linear transformations separately for each language, a single transformation could be learned over a merged multilingual dataset, constructed by combining similarity-annotated triples from multiple languages while controlling for language-specific biases through balanced sampling. Embeddings for all languages would be extracted jointly, and the transformation would be optimized to generalize across the multilingual input. If successful, this approach could provide evidence that multilingual PLMs encode factual knowledge in a way that supports shared semantic representations across languages. Such findings would not only advance the understanding of cross-lingual generalization in PLMs but could also suggest that this type of multilingual training has the potential to improve factual representation for low-resource languages, where labeled knowledge graph data is scarce.

Bibliography

Israa Alghanmi, Luis Espinosa Anke, and Steven Schockaert (2021). Probing pre-trained language models for disease knowledge. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3023–3033, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Yonatan Belinkov and James Glass (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Steven Bird and Edward Loper (2004). NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022.

Terra Blevins and Luke Zettlemoyer (2022). Language contamination helps explains the cross-lingual capabilities of English pretrained models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages

3563–3574, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio (2014). A semantic matching energy function for learning with multi-relational data. *Mach. Learn.*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko (2013). Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Roberto Boselli, Simone D'Amico, and Navid Nobani (2025). explainable ai for word embeddings: A survey. *Cognitive computation*, 17(1):19.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert (2020). Inducing relational knowledge from bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7456–7463.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020a). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020b). Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo (2018). Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 3998–4004. AAAI Press.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Noam Chomsky (1957). *Syntactic Structures*. Mouton and Co., The Hague.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel (2023). Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tai, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei (2024). Scaling instruction-finetuned language models. *J. Mach. Learn. Res.*, 25(1).

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning (2019). What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural*

Networks for NLP, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg (2019). *Visualizing and measuring the geometry of BERT*. Curran Associates Inc., Red Hook, NY, USA.

Vanya Cohen and Aaron Gokaslan (2020). Opengpt-2: open language models and implications of generated text. *XRDS*, 27(1):26–30.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv'e J'egou (2018). Word translation without parallel data. In *Proceedings of ICLR 2018*.

Joe Davison, Joshua Feldman, and Alexander Rush (2019). Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

Daniel Daza, Michael Cochez, and Paul Groth (2021). Inductive entity representations from text via link prediction. In *Proceedings of the Web Conference 2021*, WWW '21, page 798–808, New York, NY, USA. Association for Computing Machinery.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui (2024). A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610.

Mengnan Du, Ninghao Liu, and Xia Hu (2019). Techniques for interpretable machine learning. *Commun. ACM*, 63(1):68–77.

Jacob Eisenstein (2018). Natural language processing. *Jacob Eisenstein*, 507.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg (2021). Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio (2010). Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660.

Kawin Ethayarajh (2019). Rotate king to get queen: Word relationships as orthogonal transformations in embedding space. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3503–3508, Hong Kong, China. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.

Constanza Fierro and Anders Søgaard (2022). Factual consistency of multilingual pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3046–3052, Dublin, Ireland. Association for Computational Linguistics.

John Rupert Firth (1957). A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis, Special Volume/Blackwell*.

Stephen H. Friedberg, Arnold J. Insel, and Lawrence E. Spence (2003). *Linear Algebra*, 4 edition. Prentice Hall.

Christopher Frye, Colin Rowat, and Ilya Feige (2020). Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić (2019). How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 710–721, Florence, Italy. Association for Computational Linguistics.

Yoav Goldberg (2018). Neural network methods for natural language processing. *Computational Linguistics*, 44(1):194–195.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Simon Guillot, Thibault Prouteau, and Nicolas Dugue (2023). Sparser is better: one step closer to word embedding interpretability. In *Proceedings of the 15th International Conference on Computational Semantics*, pages 106–115, Nancy, France. Association for Computational Linguistics.

Nuno Guimarães, Ricardo Campos, and Alípio Jorge (2024). Pre-trained language models: What do they know? *Wiley interdisciplinary reviews. Data mining and knowledge discovery*, 14(1):e1518.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith (2020). Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Armin Haller, Axel Polleres, Daniil Dobriy, Nicolas Ferranti, and Sergio J. Rodríguez Méndez (2022). An analysis of links in wikidata. In *The Semantic Web: 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29 – June 2, 2022, Proceedings*, page 21–38, Berlin, Heidelberg. Springer-Verlag.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun (2024). Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Benjamin Heinzerling and Kentaro Inui (2021). Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1772–1791, Online. Association for Computational Linguistics.

John Hewitt and Percy Liang (2019). Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning (2019). A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Geoffrey E Hinton and Sam Roweis (2002). Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

Sepp Hochreiter and Jürgen Schmidhuber (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre (2022). Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann (2021). Knowledge graphs. *ACM computing surveys*, 54(4):1–37.

Harold Hotelling (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. (2022). Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Kaiyu Huang, Peng Li, Jin Ma, Ting Yao, and Yang Liu (2023). Knowledge transfer in incremental learning for multilingual neural machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15286–15304, Toronto, Canada. Association for Computational Linguistics.

John D. Hunter (2007). Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95.

Francesca Incitti, Federico Urli, and Lauro Snidaro (2023). Beyond word embeddings: A survey. *Inf. Fusion*, 89(C):418–436.

Ray Jackendoff (2002). *Foundations of Language: Brain, Meaning, Grammar, Evolution*, 1 edition. Oxford University Press, Oxford.

Fred Jelinek (1990). Self-organized language modeling for speech recognition. *Readings in speech recognition*, pages 450–506.

Fan Jiang, Tom Drummond, and Trevor Cohn (2023). Don't mess with mister-in-between: Improved negative search for knowledge graph completion. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1818–1832, Dubrovnik, Croatia. Association for Computational Linguistics.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig (2020a). X-FACTR: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959, Online. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig (2020b). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig (2020c). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Daniel Jurafsky and James H Martin (2023). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei (2020). Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Saniya Karwa and Navpreet Singh (2025). Disentangling linguistic features with dimension-wise analysis of vector embeddings. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 461–488, Albuquerque, New Mexico. Association for Computational Linguistics.

Nora Kassner, Philipp Dufter, and Hinrich Schütze (2021). Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3250–3258, Online. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze (2020). Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Amr Keleg and Walid Magdy (2023). DLAMA: A framework for curating culturally diverse facts for probing the knowledge of pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6245–6266, Toronto, Canada. Association for Computational Linguistics.

Yoon Kim (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

E. Kokiopoulou, J. Chen, and Y. Saad (2011). Trace optimization and eigenproblems in dimension reduction methods. *Numerical linear algebra with applications*, 18(3):565–602.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš (2020). From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao (2016). Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang (2021). Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren (2020). Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6862–6868, Online. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2181–2187. AAAI Press.

Lixian Liu, Amin Omidvar, Zongyang Ma, Ameeta Agrawal, and Aijun An (2022). Unsupervised knowledge graph generation using semantic similarity matching. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 169–179, Hybrid. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith (2019). Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang (2021a). Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Zhenhua Liu, Tong Zhu, Chuanyuan Tan, Bing Liu, Haonan Lu, and Wenliang Chen (2024). Probing language models for pre-training data detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1576–1587, Bangkok, Thailand. Association for Computational Linguistics.

Zhuang Liu, Wayne Lin, Ya Shi, and Jun Zhao (2021b). A robustly optimized bert pre-training approach with post-training. In *Chinese Computational Linguistics: 20th China National Conference, CCL 2021, Hohhot, China, August 13–15, 2021, Proceedings*, page 471–484, Berlin, Heidelberg. Springer-Verlag.

Scott M. Lundberg and Su-In Lee (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun (2015). Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1687–1692, Lisbon, Portugal. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning (2015). Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Denver, Colorado. Association for Computational Linguistics.

Christopher Manning and Hinrich Schutze (1999). *Foundations of statistical natural language processing*. MIT press.

Warren S McCulloch and Walter Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

Oren Melamud, Jacob Goldberger, and Ido Dagan (2016). context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov (2022). Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Julian Michael, Jan A. Botha, and Ian Tenney (2020). Asking without telling: Exploring latent ontologies in contextual representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6792–6812, Online. Association for Computational Linguistics.

Tomas Mikolov, Chen Kai, Corrado Greg, and Dean Jeffrey (2013). Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at ICLR*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari.

Marvin Minsky and Seymour Papert (1969). An introduction to computational geometry. *Cambridge tiass., HIT*, 479(480):104.

Giacomo Munda, Dagmar Gromann, and Tobias Heinzle (2025). Vector space transformations to uncover knowledge graphs in neural language models. In

Handbook on Neurosymbolic AI and Knowledge Graphs, pages 121–145. IOS Press.

Brian Murphy, Partha Pratim Talukdar, and Tom Mitchell (2012). Learning effective and interpretable semantic models using non-negative sparse embedding. In *International Conference on Computational Linguistics (COLING 2012), Mumbai, India*, pages 1933–1949. Association for Computational Linguistics.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio (2016). Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1955–1961. AAAI Press.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA. Omnipress.

Finn Årup Nielsen (2017). Wembedder: Wikidata entity embedding web service. *CoRR*, abs/1710.04099.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena (2021). Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.

Kriti Ohri and Mukesh Kumar (2021). Review on self-supervised image recognition using deep neural networks. *Knowledge-based systems*, 224:107090.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe (2022). Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Sungjoon Park, JinYeong Bak, and Alice Oh (2017). Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 401–411, Copenhagen, Denmark. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan

Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Heiko Paulheim (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830.

Hao Peng, Xiaozhi Wang, Shengding Hu, Hailong Jin, Lei Hou, Juanzi Li, Zhiyuan Liu, and Qun Liu (2022). COPEN: Probing conceptual knowledge in pre-trained language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5015–5035, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018a). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih (2018b). Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller (2019). Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Jirui Qi, Raquel Fernández, and Arianna Bisazza (2023). Cross-lingual consistency of factual knowledge in multilingual language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10650–10666, Singapore. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. (2018). Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving (2021). Scaling language models: Methods, analysis insights from training gopher. *CoRR*, abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Veenu Rani, Syed Tufael Nabi, Munish Kumar, Ajay Mittal, and Krishan Kumar (2023). Self-supervised learning: A succinct review. *Archives of computational methods in engineering*, 30(4):2761–2775.

Abhilasha Ravichander, Eduard Hovy, Kaheer Suleman, Adam Trischler, and Jackie Chi Kit Cheung (2020). On the systematicity of probing contextualized

word representations: The case of hypernymy in BERT. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 88–102, Barcelona, Spain (Online). Association for Computational Linguistics.

Laria Reynolds and Kyle McDonell (2021). Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California. Association for Computational Linguistics.

Petar Ristoski and Heiko Paulheim (2016). Rdf2vec: Rdf graph embeddings for data mining. In *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I*, page 498–514, Berlin, Heidelberg. Springer-Verlag.

Adam Roberts, Colin Raffel, and Noam Shazeer (2020). How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Frank Rosenblatt (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze (2016). Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777, San Diego, California. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze (2016). Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–517, Berlin, Germany. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard (2019). A survey of cross-lingual word embedding models. *J. Artif. Int. Res.*, 65(1):569–630.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja,

Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng-Xin Yong, Harshit Pandey, Michael Mckenna, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush (2022). Multitask prompted training enables zero-shot task generalization.

Rico Sennrich, Barry Haddow, and Alexandra Birch (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro Szekely (2022). A study of the quality of wikidata. *Journal of Web Semantics*, 72:100679.

Naichen Shi, Dawei Li, Mingyi Hong, and Ruoyu Sun (2021). Rmsprop converges with proper hyper-parameter. In *9th International Conference on Learning Representations (ICLR)*. Spotlight paper.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh (2020). AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng (2013). Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 926–934, Red Hook, NY, USA. Curran Associates Inc.

Felix Stahlberg (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418.

Gilbert Strang (2006). *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA.

Gilbert Strang (2009). *Linear Algebra and Its Applications*. Wellesley-Cambridge Press, Wellesley, MA.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy (2018). Spine: sparse interpretable neural embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Mujeen Sung, Jinhyuk Lee, Sean Yi, Minji Jeon, Sungdong Kim, and Jaewoo Kang (2021). Can language models be biomedical knowledge bases? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4723–4734, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Richard S. Sutton and Andrew G. Barto (1998). *Reinforcement learning: An introduction*, volume 1. MIT Press.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei (2023). Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan (2020). Multilingual translation with extensible multilingual pretraining and finetuning. *CoRR*, abs/2008.00401.

Guy Tennenholtz, Yinlam Chow, Chih-Wei Hsu, Jihwan Jeong, Lior Shani, Azamat Tulepbergenov, Deepak Ramachandran, Martin Mladenov, and Craig Boutilier (2023a). Demystifying embedding spaces using large language models. *CoRR*, abs/2310.04475.

Guy Tennenholtz, Yinlam Chow, Chih-Wei Hsu, Jihwan Jeong, Lior Shani, Azamat Tulepbergenov, Deepak Ramachandran, Martin Mladenov, and Craig Boutilier (2023b). Demystifying embedding spaces using large language models. *CoRR*, abs/2310.04475.

Ian Tenney, Dipanjan Das, and Ellie Pavlick (2019). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. (2023). Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Laurens van der Maaten and Geoffrey Hinton (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Denny Vrandečić and Markus Krötzsch (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang (2021). KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li (2022). Finding skill neurons in pre-trained transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11132–11152, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang (2023). Large language models are latent variable models: explaining and finding good demonstrations for in-context learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen (2014). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 1112–1119. AAAI Press.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. (2022a). Emergent abilities of large language models. *CoRR*, abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou (2022b). Chain-of-thought prompting elicits reasoning in large language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou (2022c). Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Paul J Werbos (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Patrick H Winston (1980). Learning and reasoning by analogy. *Communications of the ACM*, 23(12):689–703.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Junda Wu, Tong Yu, Rui Wang, Zhao Song, Ruiyi Zhang, Handong Zhao, Chaochao Lu, Shuai Li, and Ricardo Henao (2024). Infoprompt: Information-theoretic soft prompt tuning for natural language understanding. *Advances in Neural Information Processing Systems*, 36.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. (2016). Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation. *CoRR*, abs/1609.08144.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng (2014). Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Da Yin, Hritik Bansal, Masoud Monajatipoor, Liunian Harold Li, and Kai-Wei Chang (2022a). GeoMLAMA: Geo-diverse commonsense probing on multilingual pre-trained language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2039–2055, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Da Yin, Hritik Bansal, Masoud Monajatipoor, Liunian Harold Li, and Kai-Wei Chang (2022b). Geomlama: Geo-diverse commonsense probing on multilingual pre-trained language models.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva (2024). Jump to conclusions: Short-cutting transformers with linear transformations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9615–9625, Torino, Italia. ELRA and ICCL.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich (2020). Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du (2024). Explainability for large language models: A survey. *ACM Trans. Intell. Syst. Technol.*, 15(2).

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. (2023). A survey of large language models. *CoRR*, abs/2303.18223(2).

Julia El Zini and Mariette Awad (2022). On the explainability of natural language processing deep models. *ACM Computing Surveys*, 55(5):1–31.