Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Faculty of Mathematics
and Computer Science
Institute of Computer Science
Human-Computer Interaction

Master's Thesis

Submitted in conformity with the requirements
for the degree Master of Science (M.Sc.)
in the program of Human-Computer Interaction
at the University of Würzburg

Adaptive Distributional Word Models for Robust Semantic Information Systems

Submitted by
Jonas Müller
Matriculation Number: 1966248

June 6, 2019

Supervisors:
PD Dr. Joachim Baumeister, Computer Science VI,
University of Würzburg
Chris Zimmerer, M.Sc., Computer Science IX,
University of Würzburg

Acknowledgments

I would like to express my sincere gratitude to Dr. Joachim Baumeister and Chris Zimmerer who have both been great supervisors. Thank you for supporting me with thoughtful feedback during the whole course of working on this thesis, and providing me with the freedom to research into the fields of my interest. Furthermore, I want to thank Veronika and Jonathan for our group meetings which were great for exchanging ideas and forming inspiring new impulses. I would also like to thank Albi and Stefan for helping me with all kinds of hardware and GPU-related problems, as well as Alex for setting up the online questionnaire system. Last but not least, I want express special thanks to my family for their unconditional support and encouragement throughout my years of study.

Abstract

Voice assistants in the Technical Service domain can support technicians with predominantly manual maintenance or repair tasks via an audio-verbal channel. Traditional assistants often parse user input by matching it against a set of pre-defined patterns. However, this technique is not capable of dealing with syntactic variation in the input request. This thesis evaluates the applicability of state-of-the-art language representation models in the Technical Service domain. Such models map words or word sequences onto multi-dimensional vector embeddings enabling an automatic analysis of syntactic and semantic similarity relationships. Five different models, Word2Vec, GloVe, FastText, ELMo, and BERT, are evaluated on three different evaluation tasks: an intrinsic evaluation task measures the correlation between model-generated and human-annotated word-word similarity scores. Two extrinsic evaluations assess how well the models can classify natural user input questions. The results indicate that self-trained and fine-tuned model instances, in general, perform better than pre-trained ones. ELMo achieved the best performance among all fine-tuned models, FastText was the best self-trained instance, and GloVe represented the best pre-trained model. Furthermore, a prototypical implementation incorporating high-performing models is described and its advantages are outlined based on a case study. The prototype also includes a self-learning mechanism capable of dynamically adapting to user feedback. Further work is required and discussed to productively use the models in a real-world context.

Zusammenfassung

Sprachassistenten können Beschäftigte im Technischen Service bei ihren hauptsächlich manuellen Wartungs- und Reparaturaufgaben über einen auditiv-verbalen Informationskanal unterstützen. Herkömmlicherweise müssen solche Nutzereingaben mit vordefinierten Satzmustern übereinstimmen, um sie verarbeiten zu können. Diese Herangehensweise lässt jedoch keine syntaktische Variation in der Formulierung der Nutzeranfrage zu. Deshalb wird in dieser Arbeit untersucht, inwiefern sich Modelle aktueller Forschungsarbeiten im Bereich der natürlichen Sprachrepräsentation auf den Technischen Service anwenden lassen. Solche Modelle bilden Wörter oder Wortsequenzen auf multi-dimensionalen Vektoren ab. Dadurch können semantische und syntaktische Ähnlichkeitsbeziehungen automatisch analysiert werden. Fünf Modelle (Word2Vec, GloVe, FastText, ELMo und BERT) werden in drei unterschiedlichen Evaluationen untersucht: In einer intrinsischen Evaluation wird die Korrelation zwischen menschlich annotierter und durch das Modell errechneter Ähnlichkeit zwischen Wortpaaren gemessen. In zwei extrinsischen Evaluationen wird ermittelt, wie gut die Modelle natürlichsprachliche Nutzerfragen klassifizieren können. Die Ergebnisse deuten darauf hin, dass Selbstlernen und Anpassungslernen (Lerntransfer) von Modellen allgemein besser funktionieren als vortrainierte Modelle. ELMo erreichte die beste Leistung im Anpassungslernen, FastText stellte die beste selbstgelernte Instanz, und GloVe das beste vortrainierte Modell. Außerdem wird eine prototypische Implementierung beschrieben, die performante Modelle beinhaltet. Die Vorteile dieser Implementierung werden anhand einer Fallstudie aufgezeigt. Der Prototyp umfasst außerdem ein selbstlernendes System, das sich auf Basis von Nutzerfeedback dynamisch anpasst. Weitere Forschungsarbeit ist nötig, um die Modelle im Produktiveinsatz verwenden zu können.

Contents

List of Figures ix

List of Tables xi

Abbreviations xiii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Use Case . . . 2
- 1.3 Objectives and Approach . . . 3
- 1.4 Setting . . . 4
- 1.5 Thesis Overview . . . 4

2 Background and Related Work 7
- 2.1 Semantic Information Systems . . . 7
- 2.2 Language Modeling . . . 8
  - 2.2.1 Simple Language Models . . . 8
  - 2.2.2 Bidirectional Language Models . . . 9
  - 2.2.3 Neural Language Model Architectures . . . 9
- 2.3 Semantic Word Representations . . . 12
  - 2.3.1 Word2Vec . . . 13
  - 2.3.2 FastText . . . 14
  - 2.3.3 GloVe . . . 15
- 2.4 Beyond Words: Sentence and Contextual Embeddings . . . 15
  - 2.4.1 Embeddings for Sentences . . . 16
  - 2.4.2 Contextual Word Embeddings from Pre-trained Architectures . . . 17
- 2.5 Assessing Semantic Similarity . . . 21
  - 2.5.1 Assessing Word Similarity . . . 22
  - 2.5.2 Beyond Words . . . 24
- 2.6 Summary . . . 26

3 Methodology 29
- 3.1 Service Mate Information System . . . 29
  - 3.1.1 Service Mate Voice Assistant . . . 30
- 3.2 Data . . . 30
  - 3.2.1 Overview . . . 31
  - 3.2.2 Data Preprocessing . . . 31
  - 3.2.3 Data Corpus Analysis . . . 32
- 3.3 Models . . . 33
  - 3.3.1 Model Selection . . . 33
  - 3.3.2 Training . . . 33

3.4 Intrinsic Evaluation . . . 35
3.4.1 Evaluation Task . . . 35
3.4.2 Contextual Words Dataset . . . 36
3.4.3 Evaluation Setup and Procedure . . . 42
3.5 Extrinsic Evaluation . . . 44
3.5.1 Question Pair Classification Task . . . 44
3.5.2 Question Similarity Classification Task . . . 50
3.6 Summary . . . 53

4 Results **55**
4.1 Sentence Similarity Plots . . . 55
4.2 Intrinsic Evaluation . . . 57
4.2.1 Model Performance with respect to Part-Of-Speech (POS) Class . . . 57
4.2.2 Benefits of Self-training and Fine-tuning . . . 58
4.2.3 Effects of Lemmatization and Stopwords . . . 59
4.3 Extrinsic Evaluation . . . 59
4.3.1 Question Pair Classification . . . 59
4.3.2 Question Similarity Classification . . . 61
4.3.3 Effects of Pooling Strategy . . . 63
4.3.4 Benefits of Self-training and Fine-tuning . . . 63
4.3.5 Effects of Lemmatization and Stopwords . . . 64
4.4 Summary . . . 65

5 Implementation **67**
5.1 Model Selection . . . 67
5.2 System Architecture . . . 68
5.2.1 Service Mate . . . 69
5.2.2 Python Server . . . 71
5.3 System Sequence . . . 71
5.3.1 Pattern Initialization . . . 72
5.3.2 Obtaining an Answer for a Natural Language User Request . . . 72
5.3.3 Adapting from User Feedback . . . 74
5.4 Case Study . . . 75
5.4.1 User Study . . . 76
5.4.2 Resource Evaluation . . . 80
5.5 Summary . . . 81

6 Discussion **83**
6.1 Language Representation Models . . . 83
6.2 Evaluation and Results . . . 84
6.3 Implementation and Self-Learning Mechanism . . . 85

7 Conclusion **89**

References **91**

Appendix **100**

List of Figures

1.1 Steps of speech processing. . . . 1
1.2 Two conversation snippets between a user and a voice assistant. . . . 2

2.1 Architecture of the first neural language model (Bengio, Ducharme, Vincent, & Jauvin, 2003). . . . 10
2.2 Basic visualizations of Recurrent Neural Network (RNN) architectures. . . . 10
2.3 A typical Convolutional Neural Network (CNN) for text classification. . . . 11
2.4 Illustration of the two Word2Vec algorithms Continuous Bag of Words (CBOW) and skip-gram. . . . 14
2.5 Architecture of ELMo. . . . 18
2.6 OpenAI GPT and BERT architectures. . . . 21

3.1 Service Mate information system interface. . . . 29
3.2 Typical source documents. . . . 31
3.3 POS distribution of the text corpus. . . . 32
3.4 Overview of the intrinsic evaluation. . . . 36
3.5 Distribution of the first words. . . . 37
3.6 A word-in-context pair to be rated. . . . 40
3.7 Distribution of human similarity ratings by POS. . . . 41
3.8 Inter-rater agreement with respect to POS class. . . . 42
3.9 Language representation model architectures. . . . 43
3.10 Question pair classification task. . . . 44
3.11 Average pooling. . . . 47
3.12 Max pooling. . . . 47
3.13 Question similarity classification task. . . . 50
3.14 Pattern centroid generation. . . . 52
3.15 Classification of question embeddings. . . . 53

4.1 Two-dimensional projections of questions embedded with FastText. . . . 56
4.2 Two-dimensional projections of questions embedded with GloVe. . . . 56
4.3 Spearman correlation results for the intrinsic evaluation. . . . 58
4.4 Spearman correlation results by POS class. . . . 58
4.5 The effect of self-training and fine-tuning on intrinsic task performance. . . . 59
4.6 Three-dimensional t-distributed Stochastic Neighbor Embedding (t-SNE) projections of high performing models. . . . 62
4.7 Three-dimensional t-SNE projections of low performing models. . . . 62
4.8 The effect of self-training and fine-tuning on the performance of the first extrinsic task. . . . 63
4.9 The effect of self-training and fine-tuning on the performance of the second extrinsic task. . . . 64

5.1 Overview of the system architecture. . . . 69
5.2 User Interface (UI) of the voice assistant. . . . 70
5.3 Generation of pattern centroid embeddings. . . . 72
5.4 Sequence diagram for obtaining an answer. . . . 73
5.5 Sequence diagram of the feedback mechanism. . . . 74
5.6 UI of the feedback mechanism. . . . 75
5.7 Dynamic adaptation of the voice assistant. . . . 76
5.8 Procedure of the user study. . . . 78
5.9 Classification result with respect to distance. . . . 80

A.1 Distribution of the frequency of question prefix trigrams. . . . 102

B.1 Pre-trained Word2Vec sentence similarity plot. . . . 104
B.2 Self-trained Word2Vec sentence similarity plot. . . . 104
B.3 Pre-trained GloVe Sentence Similarity Plot. . . . 105
B.4 Self-trained GloVe sentence similarity plot. . . . 105
B.5 Pre-trained FastText sentence similarity plot. . . . 106
B.6 Self-trained FastText sentence similarity plot. . . . 106
B.7 Pre-trained ELMo sentence similarity plot. . . . 107
B.8 Fine-tuned ELMo sentence similarity plot. . . . 108
B.9 Pre-trained BERT sentence similarity plot. . . . 109
B.10 Comparison between the pre-trained and self-trained Word2Vec model. . . . 111
B.11 Comparison between the pre-trained and self-trained GloVe model. . . . 111
B.12 Comparison between the pre-trained and self-trained FastText model. . . . 112
B.13 Comparison between the pre-trained and fine-tuned ELMo model. . . . 112

List of Tables

2.1 Example word pairs from the WS-353 dataset. 23
2.2 Example word-in-context pairs from the Stanford's Contextual Word Similarities (SCWS) dataset. 25
2.3 Excerpt from the Quora Question Pairs (QQP) dataset. 26

3.1 Overview of model training. 35
3.2 Example word pairs. 38
3.3 Example word-in-context pairs. 39
3.4 Checkpoint questions. 40
3.5 Ratings obtained by the participants. 41
3.6 Word vector dimensions. 43
3.7 User-generated questions for physical quantity values. 45
3.8 Question pair dataset. 46
3.9 Input dimensions of the question pair embeddings. 49
3.10 User-generated questions for physical quantity values. 51

4.1 Spearman's ρ correlation values between model variants and human ratings. . . 57
4.2 Model performance (F1) on the first extrinsic evaluation task. 60
4.3 Model performance (macro-averaged F1) on the second extrinsic evaluation task. 61

5.1 Evaluation ranking of the pre-trained model instances. 68
5.2 Evaluation ranking of the self-trained/fine-tuned model instances. 68
5.3 Questions read by the participants. 77
5.4 Classification accuracy with respect to the models. 79
5.5 Example question transcripts. 79
5.6 Time resource consumption. 80
5.7 Space resource consumption. 80

A.1 Distribution of question categories of the human-generated questions 101
A.2 Wh-Question distribution by interrogative word 102

Abbreviations

AUC Area Under the Curve. 63, 110

BOW Bag Of Words. 12, 16, 33, 46

CBOW Continuous Bag of Words. ix, 13
CNN Convolutional Neural Network. ix, 9, 11, 12, 26

GWT Google Web Toolkit. 68, 69, 70, 71

IDF Inverse Document Frequency. 47, 48, 59, 60, 61, 63, 71

LSTM Long Short-Term Memory Network. 10, 18, 19, 26

NLP Natural Language Processing. 3, 7, 8, 11, 13, 15, 16, 17, 19, 20, 26, 27, 30, 44, 83, 84, 89
NLTK Natural Language Toolkit. 31

OOV Out-Of-Vocabulary. 8, 14, 46

PCA Principal Component Analysis. 55, 103
POS Part-Of-Speech. viii, ix, 32, 36, 37, 40, 41, 42, 57, 58

QQP Quora Question Pairs. xi, 26, 45

RNN Recurrent Neural Network. ix, 9, 10, 11, 12, 26
ROC Receiver Operator Characteristic. 63, 110
RPC Remote Procedure Call. 70

SCWS Stanford's Contextual Word Similarities. xi, 24, 34, 35, 36, 37, 39, 40, 42

t-SNE t-distributed Stochastic Neighbor Embedding. ix, 61

UI User Interface. x, 69, 70, 72, 75

XHTML Extensible Hypertext Markup Language. 30, 31, 53

1 Introduction

1.1 Motivation

Machinery and plant engineering represent a substantial contribution to German economy: the sector constitutes the largest job provider, and the Federal Ministry for Economic Affairs and Energy defines it being the central point of the German capital goods industry (Bundesministerium für Wirtschaft und Energie, 2018). As machinery becomes more sophisticated, it is also getting more challenging to maintain and repair due to increasing machine complexity. Therefore, innovative solutions are needed, particularly when it comes to maintenance and repair tasks in order to allow for unimpeded and efficient operation.

In this context, digital information systems can provide more effective and efficient access to machine documentation, thereby facilitating and accelerating the accomplishment of tasks of mechanics working in the field of Technical Service. For example, semantic search and interactive diagnostic systems can support a technician during maintenance and repair tasks. Voice assistants are a promising form of human-computer interaction in this regard. They appear to be eminently suitable as they can provide information to predominantly manual and visual tasks via an audio-verbal interaction channel (Wickens, Gordon, Liu, et al., 1998).

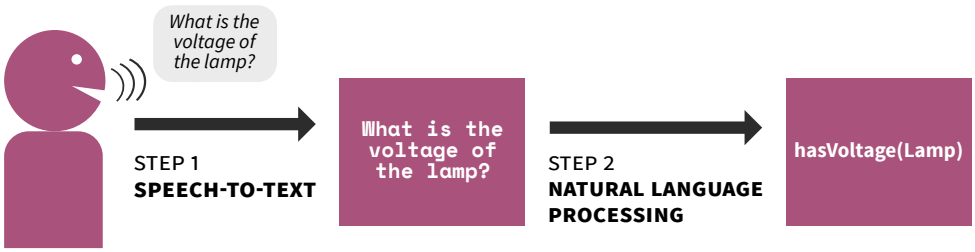
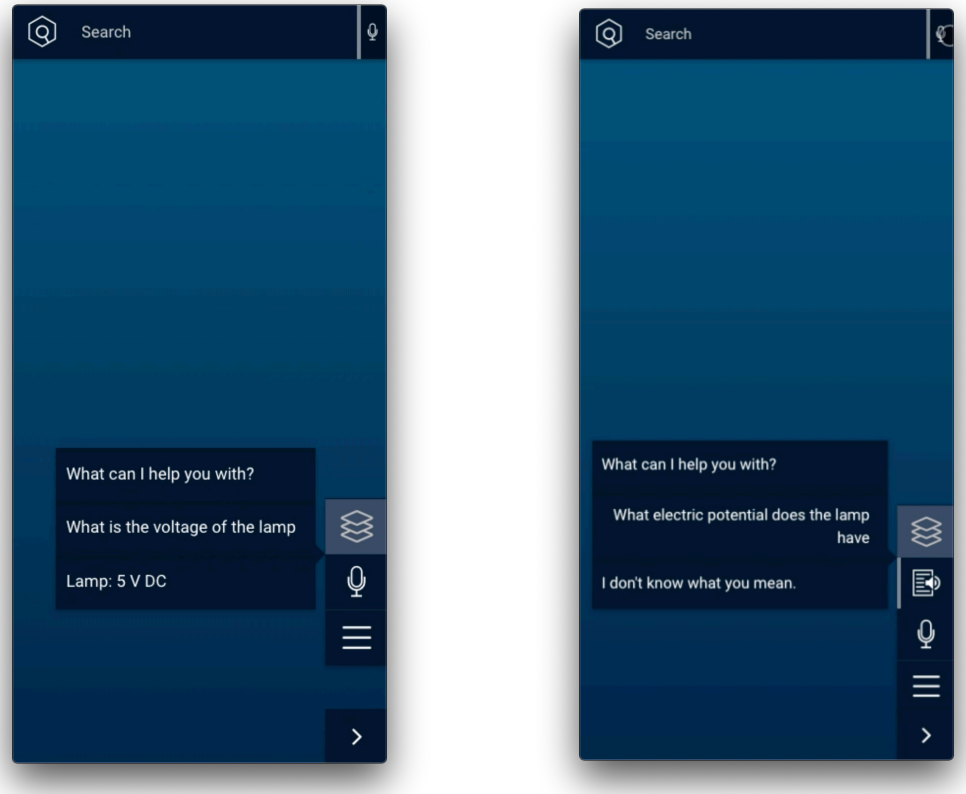


Figure 1.1. Two steps of speech processing: (1) Speech (sound waves) uttered by a user is transcribed to text by a speech-to-text engine. (2) The resulting text is further processed to analyze the semantic content, e.g. by extracting concepts and relationships from it.

However, there are problems utilizing speech-enabled assistants. Typically, these problems can occur at two different stages when formulating a request to a voice assistant: (1) Natural language

input is transformed into text by a speech-to-text engine before (2) the transcribed text is further processed to analyze and interpret its semantic content (Figure 1.1). This work focuses on the second step, that is, semantically analyzing the content of a natural language request given in text form. This processing step usually is not very robust: requests often cannot be processed adequately as the user is required to express pre-defined phrase structures in order to parse the relevant pieces of input information. This problem is further described with the help of an example use case in the following section.

1.2 Use Case



(a) System knows phrasing pattern. **(b)** System does not know phrasing pattern.

Figure 1.2. Two conversation snippets between a user and a voice assistant. In both cases the user asks for an electric potential value of a machine component (lamp). In case 1.2a, the speech-enabled information system is aware of the formulation and outputs the adequate answer. Different phrasing of the otherwise semantically identical request prevents the system from outputting the correct answer in case 1.2b.

Imagine a service technician tasked with repairing a malfunctioning machine component such as a lamp. In order to detect the cause of the malfunction, the technician wants to measure

the electric potential at the socket of the lamp and compare it to the normal value. Taking the measurement is a predominantly manual and visual task. Thus, asking a voice assistant for the electric potential value (auditory-verbal task) is less interfering with the current task than for example looking it up in a paper documentation (Wickens et al., 1998). However, the technician can paraphrase their request in different ways. Figure 1.2 exemplifies two semantically equal yet syntactically different verbalizations of a request asking for the potential value. An assistant that is not robust with respect to different formulations might not be able to determine an adequate answer to a user's request as in Figure 1.2b.

This specific example can be generalized to any form of user input to a voice assistant system where a user might come up with differently phrased yet semantically equal formulations. As a result, the human-machine interaction is not natural as the user first has to learn how to paraphrase their request to a voice assistant in order to get a satisfactory answer. On the other hand, enabling a voice assistant to understand a request independent of its phrasing leads to increased usability of the system (Zadrozny et al., 2000; Allen et al., 2001; Gnewuch, Morana, & Mädche, 2017).

1.3 Objectives and Approach

This master's thesis addresses the problems outlined in the previous two sections. It aims for an increased robustness of voice assistants with respect to natural language requests in the domain of Technical Service. Concretely, the overall objective is to enhance robustness concerning user input that is syntactically varying but can be attributed to a known semantic category. The objective is supposed to be achieved by accurately classifying arbitrarily formulated natural user input to a pre-defined set of categories. For example, in the previous use case, this would be a classification of the unknown phrasing to the category electric potential. In order to systematically achieve this objective, the approach consists of the following milestones.

M1 Based on current work in the field of Natural Language Processing (NLP), different state-of-the-art approaches to semantic language representation are described and analyzed. Models that are appear suitable for usage in the Technical Service domain are selected.

M2 The selected models are adapted to the domain context. That means, they are adjusted in such a way, that they can represent semantic properties of the Technical Service domain. In order to get an understanding of the benefit of adjusting models to domain-specific data, pre-trained models, that have been adjusted to general-domain data corpora, are considered as well.

M3 In the context of the Technical Service domain, these language representation models are evaluated to assess how well they would perform in a technical voice assistant. Evalu-

ations are conducted both intrinsically and extrinsically. Intrinsic tasks measure model performance on immediate tasks. In this thesis, models are evaluated intrinsically by measuring how aptly they can capture semantic similarity between word pairs given a human-annotated gold standard of similarity ratings. Extrinsic tasks assess model performance on a more realistic downstream task. The extrinsic evaluation consists of two classification tasks, measuring how well the models can classify natural user input questions.

M4 Based on the evaluation results, high-performing models are selected for a prototypical implementation into an existing voice assistant of a semantic information system.

M5 For adaptivity, an online learning procedure is developed and integrated into the prototype. This mechanism enables the voice assistant system to continually learn and update itself from user feedback.

M6 A first case study is conducted to compare the prototypically implemented models (Milestone M4) and test the feedback mechanism (Milestone M5).

1.4 Setting

This work constitutes a subpart of an industry research project conducted at denkbares GmbH[1] in Würzburg. The more comprehensive goal of this research project is to implement a robust and multimodal information system capable of processing both visual (images) and auditory (speech) input. As described above, this work focuses on the latter part considering the robustness of voice assistants.

For example cases and implementation purposes, the semantic information system *Service Mate*[2] is used throughout this thesis. The system incorporates a rudimentary voice assistant which is based on parsing pre-defined patterns. Further information on the Service Mate system as well as its voice assistant is provided in Section 3.1.

A German manufacturer of farming machinery provides the relevant data used within the scope of this work. The documentation of harvesting machines, consisting of nearly 50,000 text files, serves as source data for the methods applied in the course of this work.

1.5 Thesis Overview

The remainder of the thesis is structured as follows: Background knowledge and related work are summarized in Chapter 2. Subsequently, an elaborate description of experimentation and

[1] https://www.denkbares.com
[2] https://www.servicemate.de

evaluation methods is presented in Chapter 3, whose results are stated in Chapter 4. The prototypical implementation is outlined in Chapter 5. In Chapter 6, the findings and limitations of the methodology and the implementation are discussed before the thesis finishes with a conclusion and outlook on future work in Chapter 7.

2 Background and Related Work

Language is one of the key characteristics that set the human species apart from other living beings. Particularly, the ability to use syntactical-grammatical languages is considered inherently human, constituting a primary factor of intelligent inter-human communication (Roth & Dicke, 2005). Transferring the naturalness of inter-human communication to the interaction between humans and computers has long been a research subject towards Natural User Interfaces (Manaris, 1998). The domain of NLP, a subfield of Artificial Intelligence, is placed at the intersection between human language and machines as it is dealing with the automatic processing, understanding, and generation of natural human language (Chaubard, Fang, Genthial, Mundra, & Socher, 2017). Although early attempts to NLP date back to the late 1940s (Liddy, 2001), the recent trend towards architectures based on machine learning helped performance leap on a wide variety of NLP tasks (Young, Hazarika, Poria, & Cambria, 2018). Consequently, NLP techniques have increasingly been getting applied in human-computer interfaces, the success of which for example becomes apparent through the rising popularity of consumer voice assistants like *Amazon's Alexa* or *Google Assistant* (Canalys, 2018).

This work aims for enhancing the robustness of an information system voice assistant with respect to semantic similarity of a user's input. Therefore, this chapter outlines some of the recent research activities and developments in the field of NLP that are relevant for the improvement of intelligent voice assistants (Milestone M1). First, some general characteristics of semantic information systems are presented (Section 2.1). Next, the focus is put on techniques that are capable of modeling semantic properties of natural language. In this context, first language modeling is described in general (Section 2.2) before a closer look is taken on models capable of capturing the semantic content of words (Section 2.3) as well as contextualized words and sentences (Section 2.4). The chapter finishes with an overview of methods to assess the performance of such models (Section 2.5).

2.1 Semantic Information Systems

Semantic information systems provide users with task-specific access to required information. In contrast to traditional information systems, semantic information systems usually rely on a knowledge layer where the relevant information is stored in a structured way (Abecker & van Elst, 2004). Knowledge is typically represented in the form of ontologies, which, for a

given knowledge domain, model relevant components and the relations between them. These component-relation systems form directed graph structures known as semantic networks. In order to obtain knowledge stored in an ontology, a user has to formulate a semantic query (Guha, McCool, & Miller, 2003). However, the formulation of such queries requires a standardized format which is in stark contrast with user-friendly, natural human-machine interaction (Baumeister, 2016). Hence, in the following sections related work on semantic language representation constructs is summarized. The idea is to use such a construct to extract the semantic meaning of an arbitrarily formulated user request so that it can be plugged into a semantic query.

2.2 Language Modeling

A common approach attempting to capture the rich context and semantics of syntactical-grammatical languages is generating artificial constructs that perform well on a language modeling objective. In the domain of NLP, language modeling describes the process of assigning probabilities to arbitrary word sequences estimating the likelihood of their syntactic and semantic correctness. Formally speaking, language models are described as models that compute the probability of a sentence or a word sequence W of n words $w_1, w_2, ..., w_n$ (Jurafsky & Martin, 2014). They do so by calculating the probability of an upcoming word given a set of previous words (Equation 2.1).

$$P(W) = P(w_1, w_2, ..., w_n) = \prod_i P(w_i \mid w_1, ..., w_{i-1}) \tag{2.1}$$

2.2.1 Simple Language Models

The simplest language modeling form is based on *n-grams*, which are fixed-length sequences of words (cf. Markov, 1913). n-gram language models are based on the Markov assumption stated in Equation 2.2: they assume that the probability of an upcoming word can be approximated by taking only k previous words into account.

$$P(W) \approx \prod_i P(w_i \mid w_{i-k}, ..., w_{i-1}) \tag{2.2}$$

The individual conditional probabilities can be obtained for example by maximum likelihood estimation counting n-gram occurrences over a text corpus as stated in Equation 2.3.

$$P(w_i \mid w_{i-k}, ..., w_{i-1}) = \frac{count(w_{i-k}, ..., w_i)}{count(w_{i-k}, ..., w_{i-1})} \tag{2.3}$$

However, as these n-gram models are too simplistic to model an entire human language grammar (Chomsky, 1957), researchers have developed more sophisticated forms of language models. Popular approaches include, for example, the introduction of smoothing for unknown Out-Of-Vocabulary (OOV) words (Kneser & Ney, 1995). As of the time this thesis is written, *neural language models* mark the current state-of-the-art. Therefore, they are further described in Section 2.2.3.

2.2.2 Bidirectional Language Models

Bidirectional language models assume that language should not only be modeled from start to end of a word sequence, as described in the previous section, but also in the opposite direction. The intuition behind this is that the likelihood of a word occurring can also be influenced by other words following later in the sequence (Goodfellow, Bengio, & Courville, 2016). A backward language model can be described as follows in Equation 2.4:

$$\overleftarrow{P}(W) = \prod_i P(w_i \mid w_{i+1}, ..., w_{i+k}) \tag{2.4}$$

Joining a forward and a backward language model yields the bidirectional model (Equation 2.5).

$$P(W) = \vec{P}(W) \times \overleftarrow{P}(W) \tag{2.5}$$

2.2.3 Neural Language Model Architectures

Neural language models employ artificial neural networks for learning language representations. In 2003, Bengio, Ducharme, Vincent, and Jauvin pioneered by suggesting a basic two-layer feed-forward neural network. It executes the language modeling task with a softmax classifier predicting the next word given a set of previous words (Figure 2.1): at every time step, the model takes a fixed number of previous words as input. Each of the input words is represented by a randomly initialized vector that gets updated as the model is trained on the language modeling objective. These word vector representations can be understood as predecessors to the nowadays popular word embeddings (Section 2.3). Collobert and Weston showed that suchlike representations contain both syntactic and semantic information about the corresponding words (Collobert & Weston, 2008; Collobert et al., 2011).

Currently, most neural language modeling approaches make use of one of the following three model architectures: (1) Recurrent Neural Networks (RNNs), (2) Convolutional Neural Networks (CNNs) and (3) Transformer Networks. In the following, each of these architectures will be described briefly.

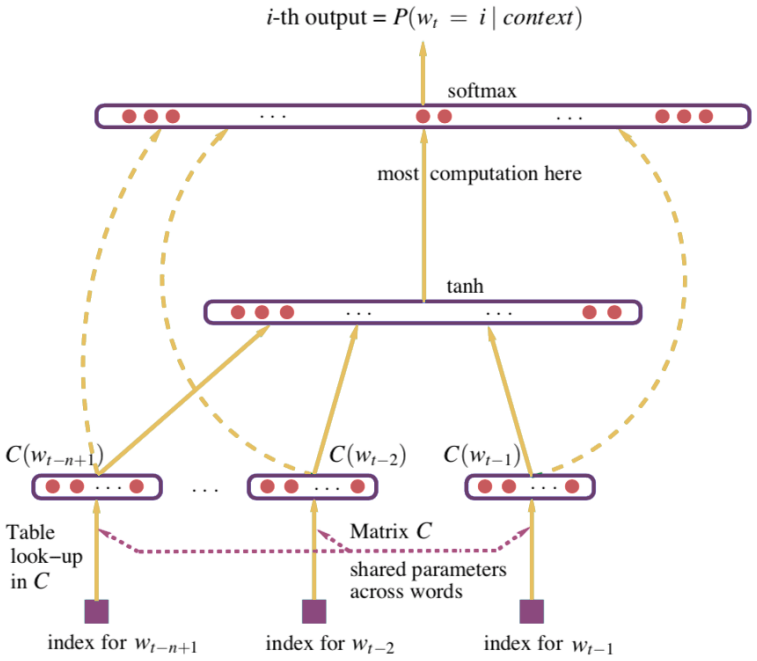
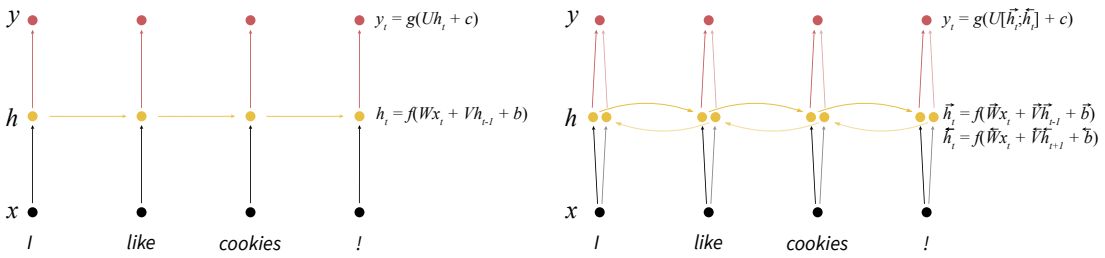


Figure 2.1. Architecture of the first neural language model (Bengio, Ducharme, Vincent, & Jauvin, 2003). Distributed word representations $C(\cdot)$ are fed into a tanh activation layer. Its output is in turn put through a softmax layer generating a probability distribution over the vocabulary indicating the most likely next word.

Recurrent Neural Networks



(a) A basic RNN. **(b)** A bidirectional RNN.

Figure 2.2. Basic visualizations of RNN architectures. The input x is a sequence of word vector representations. For each word vector x_t, a hidden state representation h_t is generated by an activation function f using trainable weight and bias parameters W, V and b weighting the current word vector as well as the previous hidden state h_{t-1}. Finally, an output representation y is generated by output function g and trainable weight and bias parameters U and c. The bidirectional RNN depicted on the right concatenates forward and backward hidden states for output generation. Graphics replicated from Irsoy and Cardie, 2014.

The most popular neural language modeling approaches are based on RNNs (Mikolov, Karafiát, Burget, Černocky, & Khudanpur, 2010), an architecture consisting of consecutive neural network cells, which treat the given text as a temporal sequence of words (Figure 2.2a). Mostly, RNN adaptations capable of modeling long-term dependencies between more distant words, such as Long Short-Term Memory Networks (LSTMs) (Hochreiter & Schmidhuber, 1997) or Gated Recurrent Units (Cho, van Merrienboer, Gulcehre, et al., 2014), are used. Bidirectional RNNs (Schuster & Paliwal, 1997), combining two reverse RNNs to model dependencies from start to end of a text and in the opposite direction as well (Figure 2.2b), are an enhancement to these architectures. They represent a bidirectional language model, as described in Section 2.2.2.

Convolutional Neural Networks

Apart from RNNs, CNN architectures (LeCun, Bengio, et al., 1995) have started being used for language modeling tasks (Dauphin, Fan, Auli, & Grangier, 2017). Moreover, researchers have applied them for variety of other NLP classification tasks, both syntactic (Collobert et al., 2011) and semantic (Kalchbrenner, Grefenstette, & Blunsom, 2014; Kim, 2014). Initially, they were predominantly applied in the computer vision community after in 2012 a CNN by Krizhevsky, Sutskever, and Hinton had been able to gain vast improvements on the ImageNet task (Russakovsky et al., 2015). Due to their convolutional operations, CNNs are capable of taking advantage of neighboring input data features such as pixels in images. Transferring these properties to NLP, they can be trained on language modeling objectives by encoding neighboring text parts such as adjacent words or characters (Figure 2.3).

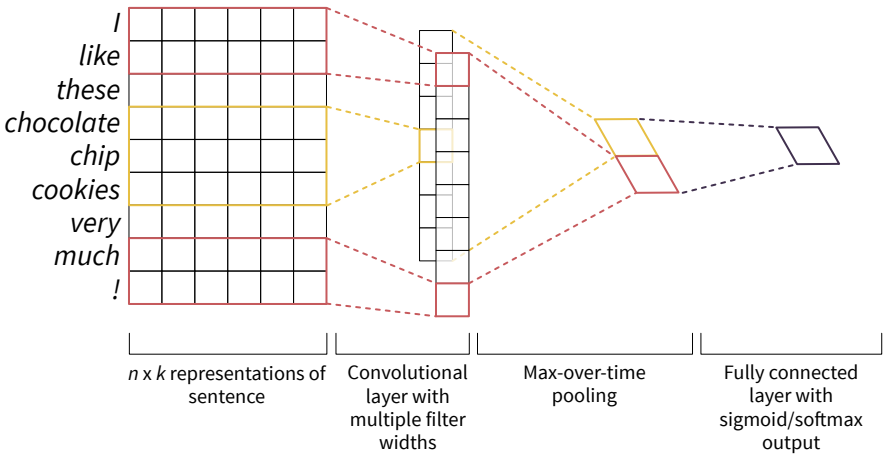


Figure 2.3. A typical CNN for text classification. Each of the n words of the input sentence is embedded into a k-dimensional vector. These embeddings are convoluted by differently sized filters and repeatedly pooled afterwards. A final sigmoid/softmax layer yields soft label outputs. In the case of language modeling, these outputs predict the upcoming word. Graphic simplified from Kim, 2014.

Transformer Networks

Finally, in 2017, Vaswani et al. introduced the Transformer Network[1]. It employs an encoder-decoder structure (Cho, van Merrienboer, Bahdanau, & Bengio, 2014) and a novel form of attention (Bahdanau, Cho, & Bengio, 2014), the so-called *Multi-Head Attention*, using scaled dot-products on several parallel layers to calculate self-referential attention (Cheng, Dong, & Lapata, 2016; Paulus, Xiong, & Socher, 2017). The Transformer disposes of CNNs and RNNs entirely and was originally designed for sequence-to-sequence tasks (Sutskever, Vinyals, & Le, 2014) such as *neural machine translation.* To account for the sequential character of text, the model generates *positional encodings* based on trigonometric functions.

In contrast to Bengio et al.'s 2003 model, nowadays RNNs, CNNs, and Transformers are generally not preceded by a layer that continually learns and updates word vectors. However, most of the published models take a very similar form of vectorized distributed word representations as input, which are described in the following section.

2.3 Semantic Word Representations

The concepts of language models and semantic word representations, also known as word embeddings or word vectors, are not clearly separable from each other, since most word embedding approaches aim to predict probabilities over sequences of words to a certain extent. Therefore, word embeddings can be described as a technique of language modeling. The objective is to capture the semantic properties of words in a numerical way (Turian, Ratinov, & Bengio, 2010). To achieve this, words are mapped onto a multi-dimensional vector space. The position of such an embedding in the vector space represents semantic properties of the corresponding word.

An early-stage example of actually quantifying words and putting them in a relative context with each another are Osgood, Suci, and Tenenbaum's *semantic differentials* proposed in 1957. They obtained word vectors by having humans rate words numerically on the dimensions valence, arousal, and dominance. Methods for automatically generating distributional representations were first introduced in the 1990s with Latent Semantic Analysis (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990). It is a global matrix factorization method: given a set of text documents, it builds a word-document matrix counting the occurrence frequency of words in the documents. Subsequently, based on mutual occurrence, it clusters terms (i.e. words) and documents in a dense vector space by applying *singular value decomposition.* Latent Dirichlet Allocation (Blei, Ng, & Jordan, 2003) constitutes a refinement to this approach, capable of annotating documents with topics. These two methods use documents and treat them as a *Bag*

[1] A straightforward animated description of the Transformer can be found here: http://jalammar.github.io/illustrated-transformer/

Of Words (BOW): models are computed solely based on frequency counts, yet without regard to word order – as if one would put all document words in a bag.

Another approach is to map semantic relationships between words by taking advantage of the context they appear in. This idea originates from structural linguistics, for example, the works of the British linguist John R. Firth who coined the *distributionalist intuition* "You shall know a word by the company it keeps" (Firth, 1957). The Hyperspace Analogue to Language model (Lund & Burgess, 1996) does so by taking adjacent words as context: it generates a word-to-word co-occurrence matrix over a text corpus and takes its rows (or columns) as word vector representations. Hence, it directly refers to Firth's notion, that is, words obtain their inherent meaning from other words in relation to the frequency they appear adjacent to them.

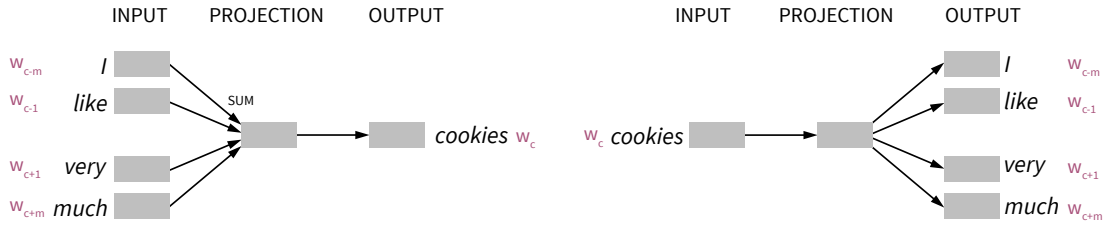
Recently, the popularity of neural approaches has grown. Many of them build upon the results of Bengio et al.'s approach described in Section 2.2.3: the authors obtain word representations as a byproduct when training the neural language model. Collobert and Weston demonstrated the benefits of using suchlike semantic word embeddings in a neural architecture for downstream tasks (Collobert & Weston, 2008). This paved the way for many present-day approaches, some of which will be summarized in the following as they will be used in the methodology described in Chapter 3.

2.3.1 Word2Vec

The publication of Word2Vec (Mikolov, Chen, Corrado, & Dean, 2013) marked a seminal point for the application of word embedding models. The authors showed that the generation of such distributed semantic models is possible in a computationally efficient manner, i.e., without having to propagate every word through a deep neural network as in Bengio et al.'s 2003 model. Another advantage is, that embeddings, which have been generated on large general-domain corpora, have shown universal usefulness among many different downstream NLP tasks: utilizing such embeddings as fixed input feature layer to a downstream model has shown beneficial for both model performance and training time (Kim, 2014), while also providing higher robustness against overfitting (Mou et al., 2016).

Following Firth's distributionalist intuition, Word2Vec represents a sliding-window approach and comes with two algorithms: Continuous Bag of Words (CBOW) and skip-gram. CBOW tries to predict a center word given a window of surrounding words by maximizing the probability $P(w_c \mid w_{c-m}, ..., w_{c-1}, w_{c+1}, ..., w_{c+m})$ with w_c being the center word vector and $2m$ being the window size (Figure 2.4a). The skip-gram algorithm does the exact opposite of the CBOW approach: the surrounding words are predicted from a center word: given a center word w_c, the algorithm tries to learn embeddings such that the probability $P(w_{c-m}, ..., w_{c-1}, w_{c+1}, ..., w_{c+m} \mid w_c)$ is maximized (Figure 2.4b). Both CBOW and skip-gram utilize a shallow neural network for training

the embedding vectors. After training, the weight matrices of this network yield the word embeddings.



(a) CBOW. **(b)** Skip-gram.

Figure 2.4. Illustration of the two Word2Vec algorithms. CBOW tries to predict a center word from its surrounding context words (left). The skip-gram algorithm does the opposite by predicting a certain number of context words from a center word (right).

2.3.2 FastText

FastText (Bojanowski, Grave, Joulin, & Mikolov, 2017) is a refinement to Word2Vec that attempts to tackle the problem of different inflection forms of words. The authors argue this to be especially important for morphologically rich languages like German or Turkish, where many word forms exist. Many of these inflected forms might only occur on very rare occasions, thus making them hard to learn. The majority of previously proposed models would learn a completely exclusive representation for each word form though.

FastText builds upon the skip-gram model of Word2Vec, with the addition of enriching the word vectors with subword information, more precisely, with character n-grams. For example, consider the word *cookie*. First, the word gets marked with word boundary symbols < and > yielding `<cookie>`. Its character trigrams are:

`<co, coo, ook, oki, kie, ie>`

Now, instead of only training an embedding for `<cookie>`, FastText also creates embeddings for each character n-gram. The authors state that this approach enriches the word embedding with the internal structure of the word. Additionally, this method is also capable of dealing with OOV words as character n-grams can always be generated regardless of whether the word in question has been seen at training time or not.

2.3.3 GloVe

Whereas Word2Vec and FastText constitute predictive and local window-based approaches, the count-based GloVe model, short for *Global Vectors*, attempts to take advantage of global co-occurrence statistics (Pennington, Socher, & Manning, 2014) .

Similarly to the previously mentioned count-based method Hyperspace Analogue to Language (see Section 2.3), it starts by generating a global co-occurrence matrix $\mathbf{X}$ over the whole text corpus with each cell $\mathbf{X}_{ij}$ denoting the co-occurrence count of two words i and j. In contrast to prior methods though, GloVe does not apply computationally expensive matrix factorization methods like singular value decomposition for dimensionality reduction. Instead, the authors propose a weighted least squares approach. Let d be the embedding vector dimension, $w_i \in \mathbb{R}^d$ a randomly initialized word vector of word i, and $\tilde{w}_j \in \mathbb{R}^d$ a separate randomly initialized context vector of word j. Then, the objective function stated in Equation 2.6 is optimized.

$$J(\theta) = \sum_{i,j=1}^{|V|} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}) \tag{2.6}$$

where θ represents the trainable parameters w_i and w_j as well as their respective bias terms b_i and $\tilde{b}_j$. f is a weighting function in order to neither overweight very rare nor very common co-occurrences, and $|V|$ is the size of the vocabulary. The word vectors w and context word vectors $\tilde{w}$ are continually updated during optimization of J. Finally, summing w_i and $\tilde{w}_i$ yields the GloVe embedding vector of word i.

In their publication, the authors of GloVe point out the close relationship between their model and the Word2Vec approach. This is in line with findings indicating that the two models perform very similarly on word similarity and analogy tasks (O. Levy & Goldberg, 2014).

2.4 Beyond Words: Sentence and Contextual Embeddings

Using pre-trained word embeddings such as Word2Vec, FastText, or GloVe as first layer features for downstream neural NLP architectures has become the de facto standard. However, the remaining layers of such deep architectures must be trained from the ground up for each new task, which can be seen as a fundamental limitation of word vector models. On the other hand, in the field of computer vision, *transfer learning* has been applied for several years and become the de facto standard for a majority of computer vision tasks (Donahue et al., 2014; Oquab, Bottou, Laptev, & Sivic, 2014; Razavian, Azizpour, Sullivan, & Carlsson, 2014): neural architectures that have been pre-trained on ImageNet (Russakovsky et al., 2015) or similar datasets, are transferred to and fine-tuned on a different task (Pan & Yang, 2010). This strategy has shown to be beneficial

on a wide variety of tasks such as image segmentation (Long, Shelhamer, & Darrell, 2015), medical object detection (Shin et al., 2016) or image captioning (Socher, Karpathy, Le, Manning, & Ng, 2014). Note, that apart from the term transfer learning, this strategy is also known as *fine-tuning*. Both terms will be used interchangeably throughout this work.

Very recently, the NLP community has begun to adopt fine-tuning pre-trained deep neural architectures. Dai and Le, 2015 showed that pre-trained architectures could be applied to a wide range of NLP tasks. Important applications include, for example, the embedding of sentences and the generation of contextualized word embeddings.

2.4.1 Embeddings for Sentences

One of the first pre-trained deep neural architectures for semantic representation is the Skip-Thought model (Kiros et al., 2015). An encoder-decoder architecture (Sutskever et al., 2014) is employed to generate sentence embeddings on a text corpus: during training, for each sentence in the corpus, the encoder creates a sentence embedding, and the decoder tries to predict both the preceding and succeeding sentence. After training, the encoder part can be used to generate universal sentence encodings capturing semantic aspects of an arbitrary input sentence. Based on this, Hill, Cho, and Korhonen, 2016 proposed two adaptations. The first is encoding sentences with a special variant of a denoising autoencoder (Vincent, Larochelle, Bengio, & Manzagol, 2008), which they call sequential denoising autoencoder. The second, called FastSent, is a log-bilinear model that optimizes on the Skip-Thought objective but represents each sentence as the sum of its word vectors. Subramanian, Trischler, Bengio, and Pal, 2018 proposed extending the Skip-Thought architecture to a multi-task objective: apart from the Skip-Thought objective, they furthermore train their model on supervised tasks like *neural machine translation* and *natural language inference*. In a similar vein, Cer et al., 2018 generate sentence embeddings using Transformer encoders trained on a multi-task objective. Another supervised approach trained on a natural language inference dataset has been suggested by Conneau, Kiela, Schwenk, Barrault, and Bordes, 2017. Recently, this approach has been extended to the multilingual model LASER that covers 93 languages (Artetxe & Schwenk, 2018).

Surprisingly, much simpler BOW approaches achieve competitive results outperforming some of the deep architectures mentioned above. The simplest, yet effective, way is to embed sentences by calculating the average vector of the respective word embeddings (Wieting, Bansal, Gimpel, & Livescu, 2015). Kenter, Borisov, and de Rijke, 2016 proposed Siamese CBOW, an adaptation of Word2Vec that generates word vectors optimized for the purpose of being averaged to a sentence representation. Finally, Arora, Liang, and Ma, 2017 achieved strong results calculating the weighted average of word embeddings in a sentence with a weighting function utilizing (estimated) word frequency.

Other approaches include, for example, Doc2Vec (Le & Mikolov, 2014). It is a variant of Word2Vec capable of embedding documents or sentences by including a separate document vector to the Word2Vec training objective.

2.4.2 Contextual Word Embeddings from Pre-trained Architectures

Pre-training deep NLP encoders does not only help with generating sentence embeddings. A recent trend manifested in generating contextual word embeddings from pre-trained models. The objective is to enrich word representations with context. This approach is capable of dealing with polysemous words: for example, one wants to obtain different word vectors for the word "*mouse*" in a sentence about the animal in contrast to the word appearing a sentence about a computer mouse. In general, there are currently two approaches to generating contextualized word embeddings: feature-based and fine-tuning approaches.

Feature-based Approaches

Feature-based approaches are predominantly used as additional input features to task-specific models. We have seen feature-based models before, namely in the form of word embeddings. Word embeddings are very shallow features though, as they only attribute for the first layer of a deeper architecture. In this section, on the other hand, the focus is put on deep pre-trained neural encoders that are capable of generating contextual word embeddings, which can be applied as features for different NLP tasks.

CoVe. The introduction of the CoVe model, short for Contextualized Vectors, (McCann, Bradbury, Xiong, & Socher, 2017) marked a crucial step in this direction. Its objective is to enrich GloVe vectors with context. CoVe embeddings are created by a pre-trained sequence-to-sequence (Sutskever et al., 2014) architecture with an attention mechanism (Bahdanau et al., 2014). That means, it incorporates an encoder and a decoder which are pre-trained on a neural machine translation task. In analogy to the Skip-Thought model, after pre-training, the decoder part of the model is discarded, and the encoder part is used to obtain the contextualized embeddings: feeding the encoder with an input sentence represented as GloVe vectors generates the contextualized CoVe vectors. Thus, CoVe embeddings consist of higher level representations than plain word vectors, since GloVe embeddings are augmented with the hidden states of the encoder.

ELMo. ELMo, short for Embeddings from Language Models, (Peters et al., 2018) follows a similar direction as CoVe, as it enriches word representations with context information in order to model how words differ across contexts. However, in contrast to CoVe, which optimizes on a supervised machine translation task, ELMo is pre-trained on an unsupervised language modeling task.

ELMo utilizes a bidirectional language model (cf. Section 2.2.2) consisting of L layers: it employs a bidirectional LSTM architecture (Figure 2.5), which is pre-trained on a language modeling task. Due to the bidirectionality of the model, each layer generates, in fact, two hidden representations for a given word, yielding a total of $2L + 1$ representations for each word.

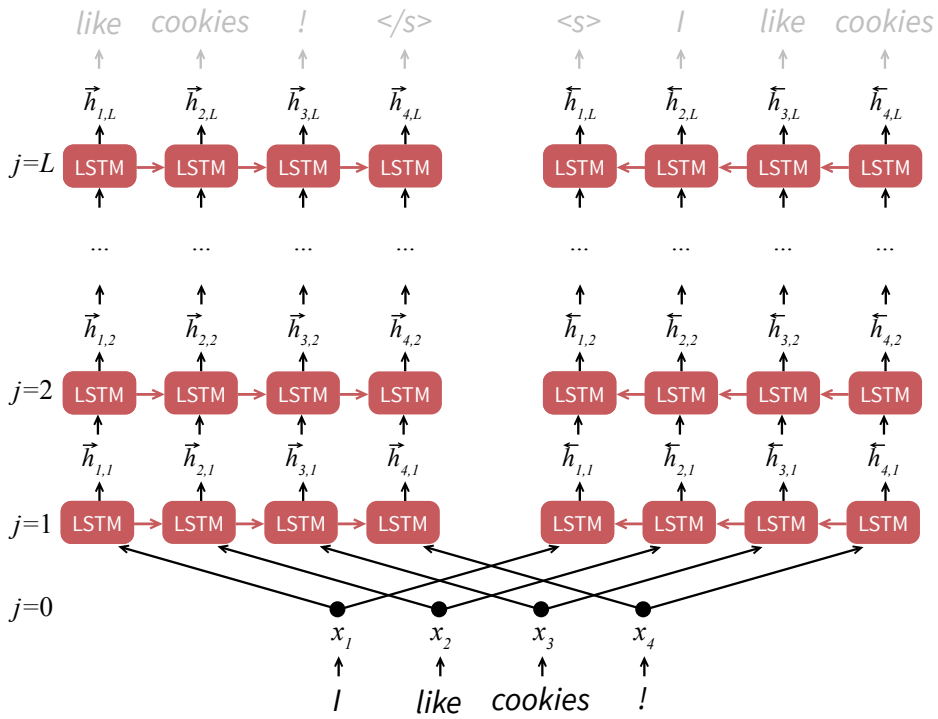


Figure 2.5. Architecture of ELMo. Each word w_k is embedded to x_k and then propagated through a stacked bidirectional LSTM architecture yielding a hidden state representation for each word and layer $h_{k,j}$. The training objective is a bidirectional language modeling task as indicated in grey at the top.

Formally speaking, let w_k be a word, j a layer in the language model, $\vec{h}_{k,j}$ the forward representation of w_k in layer j and $\overleftarrow{h}_{k,j}$ the respective backward representation. Then, the numerical representation R_k of w_k can be stated as follows in Equation 2.7.

$$\begin{aligned} R_k &= \left\{ x_k, \vec{h}_{k,j}, \overleftarrow{h}_{k,j} \;\middle|\; j \in [1, L] \right\} \\ &= \left\{ h_{k,j} \;\middle|\; j \in [0, L] \right\} \end{aligned} \tag{2.7}$$

where $h_{k,0} = x_k$ is the context-independent embedding of w_k (i.e. a plain word vector) and $h_{k,j} = \left[\vec{h}_{k,j}; \overleftarrow{h}_{k,j}\right]$. To get the final ELMo vector representation of w_k, the authors propose either taking the top layer $E(R_k) = h_{k,L}$ or integrating the representations R_k into a single vector $\mathbf{ELMo}_k$. This vector is dependent on the specific downstream task and calculated as stated in Equation 2.8.

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j} \tag{2.8}$$

where s^{task} are task-specific layer weights, and γ^{task} is a task-specific parameter weighting the entire embedding of w_k. The respective parameters for s_j^{task} and γ^{task} are learned when ELMo is applied in a downstream task.

Fine-tuning-based Approaches

Whereas the two previously described methods represent feature-based approaches, meaning they are predominantly used as additional input features to task-specific models, fine-tuning-based approaches are entitled to be applied to multiple tasks out of the box. That is, the model is pre-trained and then used as is for another downstream NLP task. Hence, the goal of fine-tuning-based approaches is to create universally applicable architectures that can be fine-tuned to a specific supervised learning objective applying transfer learning and hyperparameter tuning. Exemplifying approaches have been taken by Dai and Le, 2015, as well as Howard and Ruder, 2018.

OpenAI GPT. Another prominent approach in this direction is the OpenAI GPT architecture, short for generative pre-training, (Radford, Narasimhan, Salimans, & Sutskever, 2018). A key difference to previous publications is that this model employs a Transformer (cf. section 2.2.3) in order to be able to model long-term relationships better than with an LSTM-based architecture (Figure 2.6a).

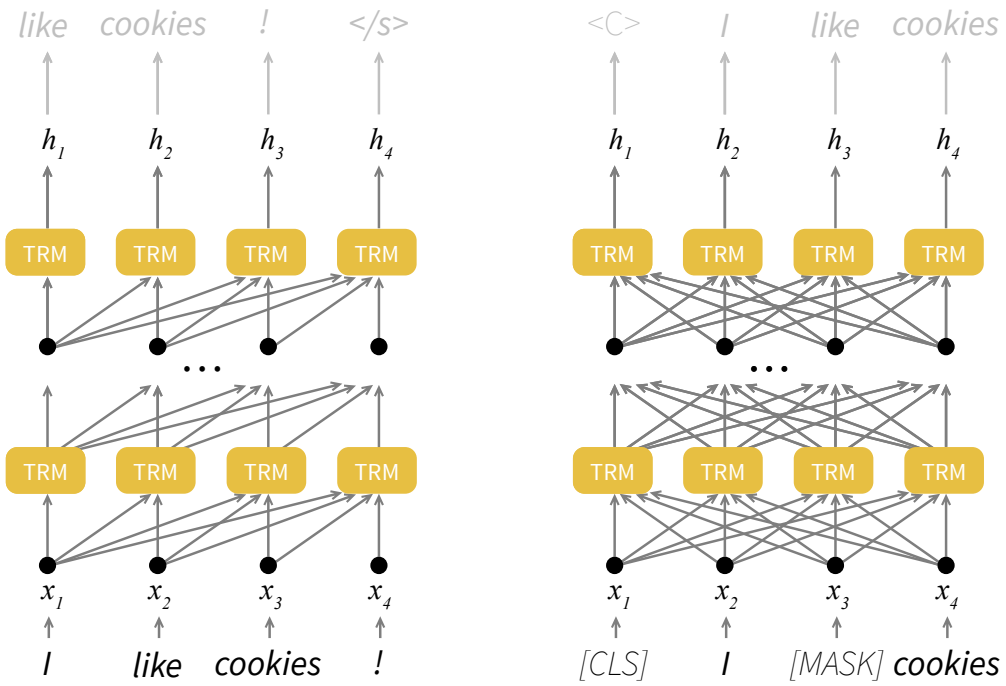
Pre-training is conducted on a large text corpus optimizing on a unidirectional unsupervised language modeling objective as described in section 2.2. Following Liu et al., 2018, only the decoder portion of a Transformer is utilized for probability maximization. After pre-training, the model can be applied to a multitude of supervised downstream tasks. Recently, Radford et al., 2019 published a second larger version of this model, which became popular in the media for its high-quality natural language generation capabilities.

BERT. The Bidirectional Encoder Representations from Transformers (Devlin, Chang, Lee, & Toutanova, 2018), or short BERT, marked another milestone on the way towards a model capturing the richness of human language. The architecture of BERT is quite similar to that of OpenAI GPT, yet it comes with some modifications. First, instead of taking the decoder of a Transformer, BERT uses the encoder and stacks a multitude of them together. Second, at pre-training time, OpenAI GPT only considers left-to-right context. The authors of BERT argue, though, that the integration of deep bidirectionality might yield better performance as for many downstream tasks the right-to-left context is important as well. Therefore, BERT aims for embedding bidirectionality on every level of the Transformer. On a language modeling task, however, deep bidirectionality is problematic as consecutive words would be exposed to the model due to the bidirectional character. The model would be able to "see" adjacent words, rendering the prediction task a trivial one. Consequently, this would not equip the model with an understanding of the semantic and syntactic features of the language at hand.

Therefore, the authors propose a new unsupervised task for the pre-training stage, which they call *Masked Language Modeling* (Figure 2.6b). Inspired by the *Cloze* task (Taylor, 1953), during pre-training, the model randomly selects 15 % of words from the training corpus set. The training objective is to predict these selected words correctly. 80 % of these words are masked, meaning they are replaced by a [MASK] token. Another 10 % are replaced with a random word, and 10 % are not replaced to give the model a bias towards the correct word.

Furthermore, apart from predicting masked tokens, a second task, called *Next Sentence Prediction*, is incorporated in pre-training: the intuition behind this is that many downstream tasks take advantage of a representation of the relationship between two sentences, like for example *question answering* where a question and an answer are evaluated. That is why, when conducting pre-training, the input to the model actually consists of two text sequences A and B where B is the successor of A in 50 % of the time, and a text sequence taken randomly from the training corpus otherwise. Thus, next to learning to predict the masked words correctly, the model is furthermore pre-trained on a classification task predicting whether A and B are belonging together or not.

BERT can be pre-trained on an arbitrary text corpus as long as it is sufficiently large. The inputs to the model are as follows: As described above, at each pre-training step two text sequences A and B are sampled from the text corpus and bundled into a single sequence, separated by a special separator token. Three embeddings represent each word of each sentence: (1) a WordPiece embedding (Wu et al., 2016) , (2) a sentence embedding to indicate which sentence (A or B) the word belongs to, as well as (3) a positional embedding to give the Transformer a sense of order. Furthermore, the first sentence is preceded by a special [CLS] token whose output embedding h_1 is trained on the classification task indicating whether sentences A and B belong together.



(a) OpenAI GPT architecture. **(b)** BERT architecture.

Figure 2.6. Comparison of OpenAI GPT and BERT architectures. Both make use of a stacked Transformer Model (TRM). During pre-training, OpenAI GPT optimizes on a left-to-right language modeling task (left) whereas BERT optimizes bidirectionally on a Masked Language Modeling task (right). Note, that for the sake of simplicity the second sentence of the Next Sentence Prediction objective is omitted here.

BERT has become popular for breaking performance records on a wide variety of NLP tasks such as question answering, sentiment analysis, or natural language inference. However, apart from using it for such NLP tasks, the model is also capable of outputting contextualized word embeddings like ELMo. This is the use case BERT will be applied for in the course of this work. The contextualized word embeddings can be obtained from the hidden states $h_2 ... h_n$ of each layer.

2.5 Assessing Semantic Similarity

The previous two sections summarized models concerned with the semantic representation of words, words in context, and sentences. Towards more robust voice assistants for the domain of Technical Service, an underlying system should utilize such models to recognize semantically similar forms of user input. This section gives an overview of methods to evaluate whether one language representation model is better capable of capturing semantic similarity than another.

Such evaluations can be either *extrinsic* or *intrinsic* (Mundra, Peng, Socher, & Sohmshetty, 2017). In extrinsic evaluations, semantic representations are assessed indirectly by plugging them into an architecture performing a downstream task and measuring performance changes. For example, Schnabel, Labutov, Mimno, and Joachims, 2015 test the performance of different word embedding models on a noun chunking (syntactic task) and a sentiment classification task (semantic task). Intrinsic evaluations, on the other hand, measure the performance of language representation models on a specific, immediate task. An extensive intrinsic analysis of various models has been conducted by Baroni, Dinu, and Kruszewski, 2014. In the following Sections 2.5.1 and 2.5.2 related datasets and tasks to evaluate language representation models are presented.

2.5.1 Assessing Word Similarity

The most basic intrinsic evaluation is to assess how well models can capture semantic similarity on word level. In the case of word vector embeddings, for a pair of words, semantic similarity can be expressed with the cosine similarity measure between the two associated vectors: let x_u and x_v be word vectors of the words u and v, then the cosine similarity between the two vectors is calculated as follows in Equation 2.9.

$$\text{cosine}(x_u, x_v) = \frac{x_u \cdot x_v}{\|x_u\| \, \|x_v\|} \tag{2.9}$$

with $\|x\|$ being the ℓ_2-norm of a word vector and $x_u \cdot x_v$ being the dot product between two word vectors. Given the cosine similarity ratings of word vector pairs, different techniques exist to assess the quality of these ratings.

Human Annotated Datasets. The prevalent intrinsic approach to evaluate the quality of word embedding similarity is to measure the correlation between human-rated similarity of words and the cosine similarity indicated by the respective model. There are many datasets containing human intuition similarity ratings. Traditionally, such datasets consist of a set of word pairs, each of which is annotated with a similarity or relatedness score. The first dataset was published in 1965 by Rubenstein and Goodenough. The authors had 36 students rate 65 word pairs in terms of semantic similarity on a scale from 0 to 4. Finkelstein et al., 2002 published a very similar, yet larger, dataset: The WS-353 dataset consists of 353 word pairs with averaged human similarity ratings between 0 and 10. Table 2.1 contains a short excerpt from the dataset.

Later, WS-353 was split into two parts (Agirre et al., 2009), one subset focusing on synonym relations (e.g. *football/soccer*) and the other on relatedness relations (e.g. *money/bank*). Building on this, Hill, Reichart, and Korhonen, 2014 proposed the SimLex-999 dataset featuring 999 word

Table 2.1. Example word pairs with human annotated similarity ratings from the WS-353 dataset. The Sim column depicts the average human similarity rating.

Word 1	Word 2	Sim
money	bank	8.50
oil	stock	6.34
lad	wizard	0.92
smart	student	4.62
smart	stupid	5.81

pairs (*noun-noun*, *verb-verb*, *adjective-adjective*). The authors explicitly advised human annotators to rate the pairs in terms of synonym-like similarity, rather than in terms of relatedness or association.

Luong, Socher, and Manning, 2013 published the Stanford Rare Word Similarity Dataset containing 2034 pairs, each of which consists of at least one rare word, to address the problem that most previous datasets contain predominantly frequent words. Finally, the MEN dataset (Bruni, Tran, & Baroni, 2014) obtained its 3000 word pair similarity ratings in a slightly different way: human raters were presented with two word pairs at a time. They had to select the one they considered more related. Each pair was rated 50 times resulting in a score between 0 and 50, according to how often the respective pair was rated to be more related than another one.

Multiple Choice Datasets. Multiple Choice Datasets incorporate a synonym detection task. Given a target word, a language representation model is tasked to select the synonym word from a set of possible alternatives. For example, the TOEFL dataset (Landauer & Dumais, 1997) contains 80 target words, each associated with four answer alternatives. The aim is to select one answer alternative that is a synonym of the respective target. An updated version of the task has been introduced by J. Levy, Bullinaria, and McCormick, 2017.

Analogy Tasks. Yet another intrinsic evaluation is to measure how well a model is at finding word analogies in the word representation space. For a word relation a is to b as c is to d, given a, b, and c, the task is to correctly find target word d. Consider the popular example *man* is to *king* as *woman* is to *queen*. Solving $b - a + c$ should yield a word vector representation close to vector d, that is, for the example case, $king - man + woman \approx queen$. The task was first suggested by Turney, 2012. Later, in the course following the Word2Vec publication (see Section 2.3.1), it was Mikolov, Yih, and Zweig, 2013, who released a word analogy dataset containing both syntactic and semantic word relations.

Other Intrinsic Tasks. Finally, there are some lesser-known intrinsic evaluation methods. For example, *concept categorization* tasks a model with clustering words into a pre-defined number of categories and compares it with a gold standard categorization (Almuhareb & Poesio, 2006; Baroni, Evert, & Lenci, 2008). In the *selectional preference* task (Padó & Lapata, 2007; McRae, Spivey-Knowlton, & Tanenhaus, 1998), the goal is to rate the typicality of a noun occurring together with a verb either as subject or as object (e.g., considering the verb *eat*, the noun *cookies* is much more typical to be the object than the subject of a sentence – we often eat cookies, but cookies are not capable of eating something).

2.5.2 Beyond Words

A major shortcoming of the previously described datasets is that the included words are isolated, i.e., they do not incorporate any context information. However, as touched on in Section 2.4.2, words can have different meanings in different contexts. For example, in the context of withdrawing cash, the word *bank* belongs to a different semantic field than when it is expressed in the context of a river bank. In the following, this section outlines evaluation methods that take context information into consideration, either in the form of contextual word datasets or as methods for assessing sentence similarity.

Contextual Similarity Datasets. The problem described above has been addressed by Stanford's Contextual Word Similarities (SCWS) dataset (Huang, Socher, Manning, & Ng, 2012). It contains 2003 word pairs where each word is augmented with sentential context. To this end, homonymous and polysemous words (nouns, verbs, and adjectives) together with their context were sampled from Wikipedia and assembled in pairs. Each pair is associated with an averaged human similarity rating ranging from 0 to 10. Table 2.2 depicts some example word-in-context pairs from the dataset.

Sentence Similarity Datasets. An elaborate advance towards measuring sentence representation quality is the *SentEval* (Conneau & Kiela, 2018) toolbox. It consists of different intrinsic and extrinsic evaluation methods, as well as a standardized evaluation pipeline. The intrinsic part of the task set is based on the Semantic Textual Similarity benchmark tasks 2012-2016 (Agirre, Cer, Diab, & Gonzalez-Agirre, 2012; Agirre, Cer, Diab, Gonzalez-Agirre, & Guo, 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016). The associated datasets consist of sentence pairs annotated with human similarity judgments between 0 and 5. By analogy with word-level assessment, the quality of sentence embeddings is measured by calculating the correlation between the cosine similarity of coupled sentence vector embeddings and human annotated similarity. The

Table 2.2. Example word-in-context pairs with human annotated similarity ratings from the SCWS dataset. The respective words are highlighted in bold type face. The Sim column depicts the average human similarity rating.

Word 1	Word 2	Sim
… The value formerly used in the United States was 6 080.20 feet = 1 nautical (geographical or sea) **mile** . …	… Running is often measured in terms of pace in minutes per mile or **kilometer** . …	8.9
… Some earlier spreadsheets required a manual request to recalculate , since recalculation of large or complex spreadsheets often **reduced** data entry speed . …	… The combination CMA IAC , which the assembler let you **abbreviate** as CIA , produced the arithmetic inverse of AC : the twos-complement negation . …	4.5
… the Treasury was given authority to block mergers of bank holding companies and banks with other banks and bank holding companies that had a bad history of preventing money **laundering** . …	… When the band went to New York to perform the numbers on " The Ed Sullivan Show " , they were ordered to **change** the lyrics of the refrain to " let 's spend some time together " . …	2.1
… National Geographic Magazine introduced the American cultured pearl as a commercial product in their August 1985 **issue** . The Tennessee pearl farm has emerged as a tourist destination in recent years , but commercial production of freshwater pearls has ceased . …	… He noted that he would continue to use his veto power against ill-drafted bills and his power to **issue** decrees on issues he deemed important , and that such decrees would remain in force until suitable laws were passed . …	3.65

extrinsic part of the task set incorporates various classification tasks such as sentiment analysis (Pang & Lee, 2005; Socher et al., 2013), question type classification (Voorhees & Tice, 2000), product review classification (Hu & Liu, 2004), subjectivity/objectivity identification (Pang & Lee, 2004), as well as opinion polarity classification (Wiebe, Wilson, & Cardie, 2005). Natural language inference (Bowman, Angeli, Potts, & Manning, 2015; Cer, Diab, Agirre, Lopez-Gazpio, & Specia, 2017), paraphrase detection (Dolan, Quirk, & Brockett, 2004) and image caption retrieval (Lin et al., 2014) complete the extrinsic task ensemble.

Another dataset targeted towards the extrinsic evaluation of sentence embedding models is the Quora Question Pairs (QQP) dataset (Iyer, Dandekar, & Kornél, 2017). The dataset features a paraphrase detection task. It consists of more than 400,000 question pairs sampled from Quora[2]. Each question pair is associated with a binary label indicating whether the two questions are semantically equal or not. A downstream classifier fed with question embeddings should be able to disambiguate semantically equal from unequal pairs correctly if the embeddings are of high quality. Table 2.3 showcases two examples from the dataset.

Table 2.3. Excerpt from the QQP dataset. The third column indicates whether the two questions are semantically equal (1) or not (0).

Question 1	Question 2	Duplicate
Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1
Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0

2.6 Summary

This chapter outlined important work related to semantic language representation. Section 2.1, started by describing semantic information systems which often store knowledge in a structured way in the form of ontologies.

Next, Section 2.2 covered the domain of language modeling and its objective to make assumptions about word sequence probabilities. Relevant language modeling architectures, such as RNNs, CNNs, or Transformers, were introduced. Based on this, Section 2.3 gave an overview of embedding approaches, many of which optimize on language modeling tasks. The focus was put on

[2]https://www.quora.com

the three seminal models Word2Vec, FastText, and GloVe, which are capable of mapping individual words onto vector embeddings. Word2Vec and FastText generate such word embeddings by training a shallow neural network on predicting context words. In the case of FastText, prediction is additionally conducted on context character sequences in order to also encode subword information. GloVe, on the other hand, obtains embeddings by optimizing on a global objective taking advantage of word co-occurrence counts. Due to their favorable results on NLP benchmarks and popularity in current research, these models will be assessed in terms of applicability for the purposes of this thesis. After that, cutting-edge architectures for representing words in context as well as whole sentences were described in Section 2.4. In addition to the three plain word embedding approaches, ELMo and BERT will be picked for further evaluation as they achieved promising results on a wide variety of NLP tasks. Both of these represent deep neural architectures which can be used to obtain contextualized word vectors. ELMo constitutes a deep bidirectional LSTM architecture that is trained on a language modeling objective, whereas BERT utilizes stacked, bidirectional Transformers optimized on a Masked Language Modeling task.

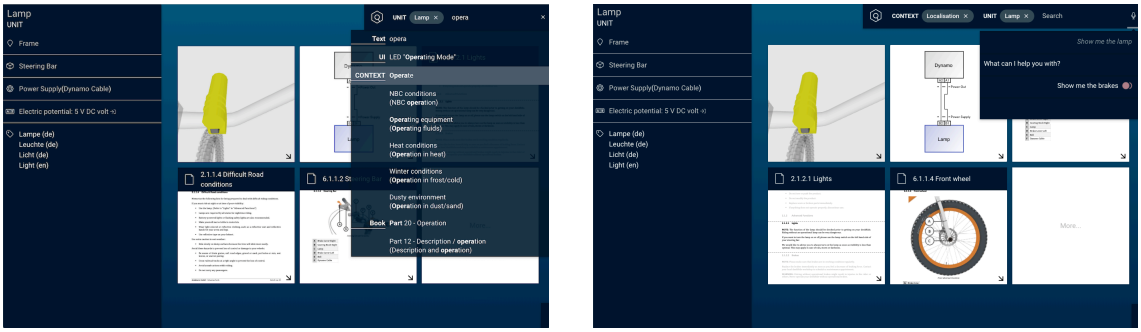
Finally, methods to intrinsically and extrinsically evaluate models for semantic language representation were presented in Section 2.5. Intrinsic methods assess performance on immediate tasks like measuring the ability to capture semantic similarity by comparing it to human similarity judgment ratings. Extrinsic methods, on the other hand, test models indirectly by measuring performance on different downstream NLP tasks. Faruqui, Tsvetkov, Rastogi, and Dyer, 2016 emphasize the importance of extrinsic evaluations of word embedding models. They argue that intrinsic evaluations should not be used as the sole means of assessment. Therefore, in the methodology of this thesis, the performance of language representation models will be measured both intrinsically and extrinsically.

3 Methodology

This chapter reports on the approach to evaluating which semantic language representation model might be suitable to enhance the robustness of speech-enabled information systems. It starts by describing the Service Mate Information System and its voice assistant, which serves as a use case example throughout the remainder of this thesis (Section 3.1). Subsequently, proceeding from a description of the data corpus (Section 3.2), insight into the model selection and training process (Milestone M2) is given in Section 3.3. The remaining sections are concerned with the methods used for intrinsic (Section 3.4) and extrinsic evaluation (Section 3.5) of the models (Milestone M3).

3.1 Service Mate Information System

Service Mate[1] is an information system for the Technical Service domain. It supports technicians with diagnosis, repair, and maintenance tasks at complex machines. For example, it provides interactive tools for fault diagnosis and recovery. Furthermore, it enables the user to semantically search the technical documentation of the machine at hand. Apart from classic semantic typing, i.e., semantically autocompleted text search (Figure 3.1a), a variety of other multimedia instances are provided, such as 3D models, Virtual and Augmented Reality interfaces, and a video conferencing system to contact a remote service desk expert.



(a) Semantic Typing. **(b)** Voice Assistant.

Figure 3.1. Service Mate information system interface. Information can for example be searched by semantic typing (left) or a voice assistant (right).

[1] https://www.servicemate.de

3.1.1 Service Mate Voice Assistant

Service Mate also incorporates a voice assistant that enables hands-free interaction with the system (Figure 3.1b). However, the current voice assistant is based on a rigid pattern matching approach. That means each user input is processed by matching it against a set of predefined patterns. Listing 3.1 depicts an example of three patterns used for parsing a request for an electric potential value. Revisiting the use case example outlined in Section 1.2, note that the patterns would be able to parse an input of the form *"What is the voltage of the lamp"*, whereas they would fail to parse *"What electric potential does the lamp have"*. Though being semantically equal, the latter phrasing is not matched by any one of the three patterns leading to an undesired output of the system.

A robust voice assistant, on the other hand, should be able to identify semantically equivalent input formulations. Therefore, it is the primary goal of this work to apply techniques from NLP for being able to capture the semantic content of user input. Hence, the following sections describe how some of the methods introduced in the previous chapter are utilized to achieve this goal.

Listing 3.1. Patterns for parsing input requesting an electric potential value.
Explanation of pattern variables and operators:
`$PPART` denotes a pronoun or article.
`$` followed by a verb denotes an arbitrary inflection form of the given verb.
`$1` denotes a variable concept word referring to a machine component like *lamp*.
`?` denotes an optional word.
`|` denotes an OR operator.
`*` is the Kleene star indicating that the given word or variable may occur zero or more times.

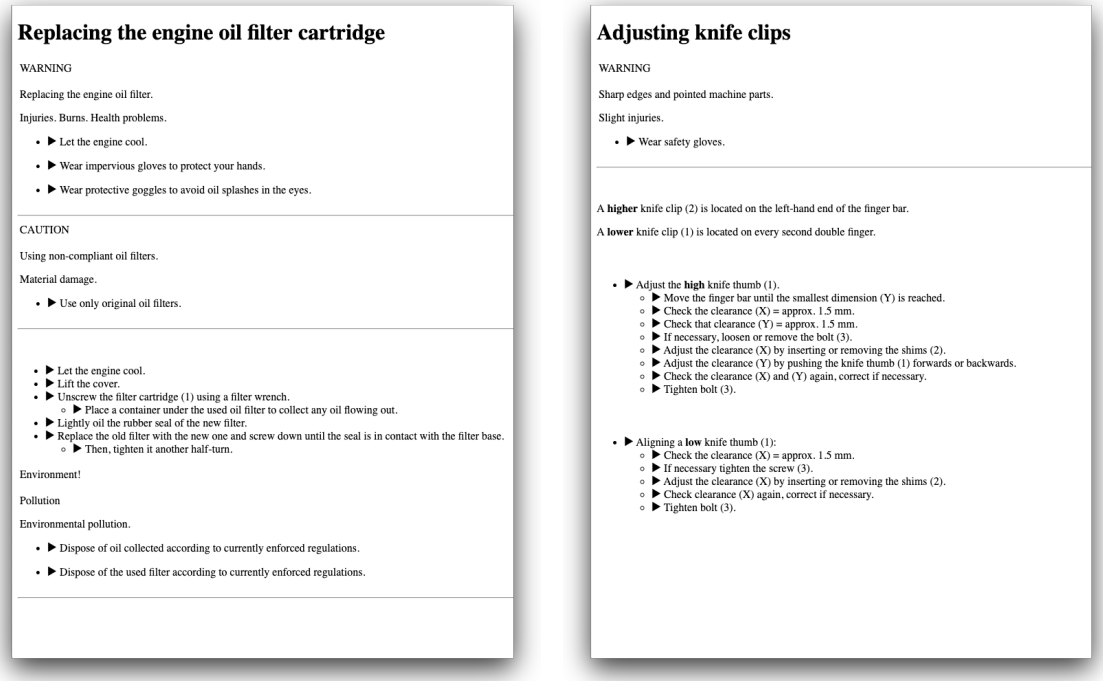
```
(How|What) high? $be $PPART* electric? (potential|voltage) $PPART* $1

($Show|$Say|$Tell) $PPART* electric? (potential|voltage) $PPART* $1

How (much|many) electric? (potential|volt|volts|voltage) $do $PPART* $1
    ↪ $have
```

3.2 Data

As this project is realized with support of a German manufacturing company for farming and harvesting technology, the technical documentation of harvesting machines is used as source data.

3.2.1 Overview

The data corpus consists of 49,661 files written in the Extensible Hypertext Markup Language (XHTML). These files include machine component lists, manuals, repair guides, maintenance handbooks, technical blueprints and other, mostly text-based, content. The language of the textual data is English. On average, each individual document consists of 10, 229.76 (SD = 30, 655.63) characters. In contrast to most text corpora language representation models are usually trained on, this text corpus is mainly not written in a prose-like form. Even though there are occasional prose paragraphs, major text parts are structured in the form of tables or bullet lists. Figure 3.2 depicts an example of typical corpus documents.



Replacing the engine oil filter cartridge

WARNING

Replacing the engine oil filter.

Injuries. Burns. Health problems.

- ▶ Let the engine cool.
- ▶ Wear impervious gloves to protect your hands.
- ▶ Wear protective goggles to avoid oil splashes in the eyes.

CAUTION

Using non-compliant oil filters.

Material damage.

- ▶ Use only original oil filters.

- ▶ Let the engine cool.
- ▶ Lift the cover.
- ▶ Unscrew the filter cartridge (1) using a filter wrench.
  - ▶ Place a container under the used oil filter to collect any oil flowing out.
- ▶ Lightly oil the rubber seal of the new filter.
- ▶ Replace the old filter with the new one and screw down until the seal is in contact with the filter base.
  - ▶ Then, tighten it another half-turn.

Environment!

Pollution

Environmental pollution.

- ▶ Dispose of oil collected according to currently enforced regulations.
- ▶ Dispose of the used filter according to currently enforced regulations.

Adjusting knife clips

WARNING

Sharp edges and pointed machine parts.

Slight injuries.

- ▶ Wear safety gloves.

A **higher** knife clip (2) is located on the left-hand end of the finger bar.

A **lower** knife clip (1) is located on every second double finger.

- ▶ Adjust the **high** knife thumb (1).
  - ▶ Move the finger bar until the smallest dimension (Y) is reached.
  - ▶ Check the clearance (X) = approx. 1.5 mm.
  - ▶ Check that clearance (Y) = approx. 1.5 mm.
  - ▶ If necessary, loosen or remove the bolt (3).
  - ▶ Adjust the clearance (X) by inserting or removing the shims (2).
  - ▶ Adjust the clearance (Y) by pushing the knife thumb (1) forwards or backwards.
  - ▶ Check the clearance (X) and (Y) again, correct if necessary.
  - ▶ Tighten bolt (3).

- ▶ Aligning a **low** knife thumb (1):
  - ▶ Check the clearance (X) = approx. 1.5 mm.
  - ▶ If necessary tighten the screw (3).
  - ▶ Adjust the clearance (X) by inserting or removing the shims (2).
  - ▶ Check clearance (X) again, correct if necessary.
  - ▶ Tighten bolt (3).

Figure 3.2. Two typical source documents from the document corpus. Text is mostly not written in prose, but rather in a structured form.

3.2.2 Data Preprocessing

Before training language representation models on the text, word tokens of the source data are generated. First, XHTML tags are removed. After that, tokens are lowercased, and numbers, single character words, and stopwords are removed using Natural Language Toolkit (NLTK)[2]. Two versions of tokenized text are generated: one with unaltered words and one with lemmatized word forms. Henceforth, the author will refer to the first version as *standard version* and

[2]https://www.nltk.org

the second one as *lemmatized version.* spaCy[3] is employed for lemmatization. Listing 3.2 gives an example of preprocessed standard tokens, Listing 3.3 depicts the corresponding lemmatized version.

Listing 3.2. Excerpt from preprocessed text tokens of the first example document.

```
replacing engine oil filter cartridge warning replacing engine oil filter
    ↪ injuries burns health problems let engine cool wear impervious gloves
    ↪  protect hands wear protective goggles avoid oil splashes eyes [...]
```

Listing 3.3. Excerpt from preprocessed, lemmatized text tokens of the first example document.

```
replace engine oil filter cartridge warn replace engine oil filter injury
    ↪ burn health problem let engine cool wear impervious glove protect
    ↪ hand wear protective goggle avoid oil splash eye [...]
```

3.2.3 Data Corpus Analysis

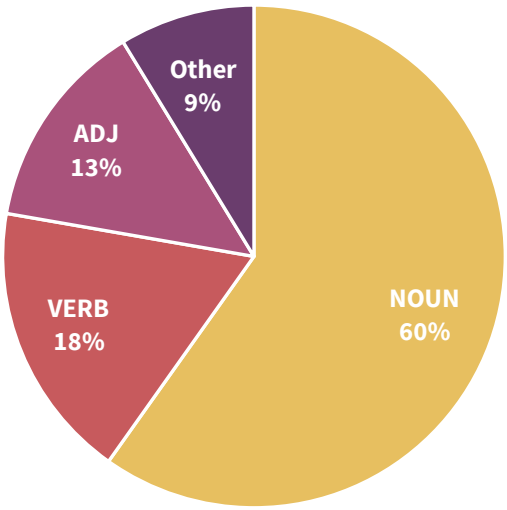


Figure 3.3. Part-Of-Speech (POS) distribution of the text corpus. Nouns are represented most, followed by verbs, adjectives and other word classes.

After preprocessing, the data corpus consists of 8,892,797 word tokens for the standard version and 8,893,843 for the lemmatized version. Furthermore, the standard tokens comprise a vocabulary size of 47,984 whereas the lemmatized tokens hold a vocabulary size of 44,898. Note that the different vocabulary sizes result from the lemmatization process. POS tags are determined

[3]https://spacy.io

using spaCy. The majority of words are nouns, followed by verbs and adjectives, as depicted in Figure 3.3.

3.3 Models

Proceeding from the tokenized text corpus, different word representation approaches are employed to generate embeddings of the word tokens.

3.3.1 Model Selection

Five different models are selected for evaluation. Among these are the three prominent word embedding architectures Word2Vec, GloVe, and FastText as well, as the two cutting-edge architectures ELMo and BERT, which are capable of generating contextualized word embeddings. The selection process excludes approaches that encode entire sentence embeddings for several reasons: first and foremost, the intrinsic evaluation (see Section 3.4) will be conducted on a word-level task. Second, as described in Section 2.4.1, simple BOW-averaging of classic word embeddings yields competitive results compared with the more complex sentence embedding models. Third, the architectures of ELMo and BERT closely resemble pre-trained sentence embedding architectures. BERT can, for instance, be described as an improved version of the sentence embedding architecture proposed by Cer et al., 2018.

3.3.2 Training

In the following, the training procedure is described for each of the five models. In addition to training on our text corpus, for each model, a pre-trained variant is taken into consideration as well. Table 3.1 gives an overview of the training methods for all models.

Word2Vec. 300-dimensional Word2Vec embeddings are trained using Gensim library[4]. Embeddings are generated with the skip-gram variant of Word2Vec, predicting content words with a window size of 5. Two sets of embeddings are produced, one on the standard word tokens and one on the lemmatized tokens, each training for five epochs. Furthermore, evaluations will be conducted on pre-trained Word2Vec embeddings[5] trained on a part of Google News Dataset, which consists of about 100 billion words sampled from Google News. The pre-trained model incorporates about 3 million 300-dimensional word vectors.

[4]https://radimrehurek.com/gensim/models/word2vec.html
[5]https://code.google.com/archive/p/word2vec/

GloVe. GloVe embeddings (300-dimensional) are generated using the training script provided by the authors[6]. Again, two sets of embeddings are produced, one on the standard word tokens and one on the lemmatized tokens, each training for 15 iterations on the word-word co-occurrence matrix. Besides, a 300-dimensional pre-trained model will be evaluated. It contains 2.2 million vectors trained on the 840 billion token Common Crawl dataset[7].

FastText. FastText embeddings (300-dimensional) are trained using Gensim library[8]. Embeddings are generated with a window size of 5 and character n-gram sizes ranging from 3 to 6. As before, two sets of embeddings are produced, one on the standard word tokens and one on the lemmatized tokens, each training for five epochs. Furthermore, the pre-trained English FastText embeddings[9] will be evaluated. They have been trained on Common Crawl and Wikipedia (Grave, Bojanowski, Gupta, Joulin, & Mikolov, 2018). The pre-trained model incorporates 2 million 300-dimensional word vectors.

ELMo. For ELMo, the pre-trained "Original" model is obtained from the authors' website[10]. This model was trained on the 1 Billion Word Benchmark (Chelba et al., 2013) that contains about 800 million tokens. The architecture consists of three layers, each with an embedding dimension of 1024. Apart from only evaluating the pre-trained model, transfer learning (i.e. fine-tuning) is employed to further train the model on our tokens. As suggested by the authors, transfer learning is conducted on cased tokens, i.e., for ELMo, lowercasing is omitted at the preprocessing stage. Furthermore, an additional model is fine-tuned on a cased dataset with stopwords included. Each of these two models is fine-tuned for two epochs, taking about two hours on an NVIDIA GeForce GTX 1070 GPU. Transfer learning is conducted using the AllenNLP framework (Gardner et al., 2018) following the authors' instructions[11].

BERT. Finally, for BERT the pre-trained BERT-Base, Multilingual Cased model[12] is used. It was trained on Wikipedia dumps for 100 different languages. Due to computational limitations, the model will not be fine-tuned, but only the pre-trained version will be evaluated. The authors argue, however, that the pre-trained version already incorporates a sufficiently rich language representation that renders it usable without fine-tuning. bert-as-service (Xiao, 2018) is used for obtaining 768-dimensional contextualized word embeddings from the pre-trained model.

[6] https://nlp.stanford.edu/projects/glove/
[7] http://commoncrawl.org
[8] https://radimrehurek.com/gensim/models/fasttext.html
[9] https://fasttext.cc/docs/en/crawl-vectors.html
[10] https://allennlp.org/elmo
[11] https://github.com/allenai/bilm-tf
[12] https://github.com/google-research/bert

Table 3.1. Overview of model training. Text in brackets denotes the source data the respective model was trained/fine-tuned on.

Pre-trained	Trained from Scratch	Fine-tuned
Word2Vec (Google News Dataset)	**Word2Vec** (Standard Tokens) **Word2Vec** (Lemma Tokens)	–
GloVe (Common Crawl)	**GloVe** (Standard Tokens) **GloVe** (Lemma Tokens)	–
FastText (Common Crawl & Wikipedia)	**FastText** (Standard Tokens) **FastText** (Lemma Tokens)	–
ELMo (1 Billion Word Benchmark)	–	**ELMo** (Cased Tokens, with Stopwords) **ELMo** (Cased Tokens, without Stopwords)
BERT (Multilingual Wikipedia)	–	–

3.4 Intrinsic Evaluation

Summing up all trained, fine-tuned, and pre-trained models results in a total of 13 different model manifestations to be evaluated. As outlined in Section 2.5, intrinsic evaluations are popularly performed on human intuition datasets. In order to be able to leverage and assess the ability of ELMo and BERT – that is, enriching word embeddings with context – a dataset is created similar to the SCWS dataset described in Section 2.5.2.The following Section 3.4.1 briefly points out the task conducted on the models. After that, a detailed description of the contextual dataset (Section 3.4.2) and the experimental procedure (Section 3.4.3) is given.

3.4.1 Evaluation Task

Based on the source data documents, a contextual dataset resembling the SCWS dataset is created. Each item of this dataset consists of two words depicted in their respective sentential context. Domain experts rate the similarity of each word-in-context pair. After that, for each of the 13 model variants, the cosine similarity value of every word-in-context pair is calculated. The final evaluation metric for each model is the correlation between the human annotated similarity ratings and the cosine similarities derived from the respective model. This work assumes that

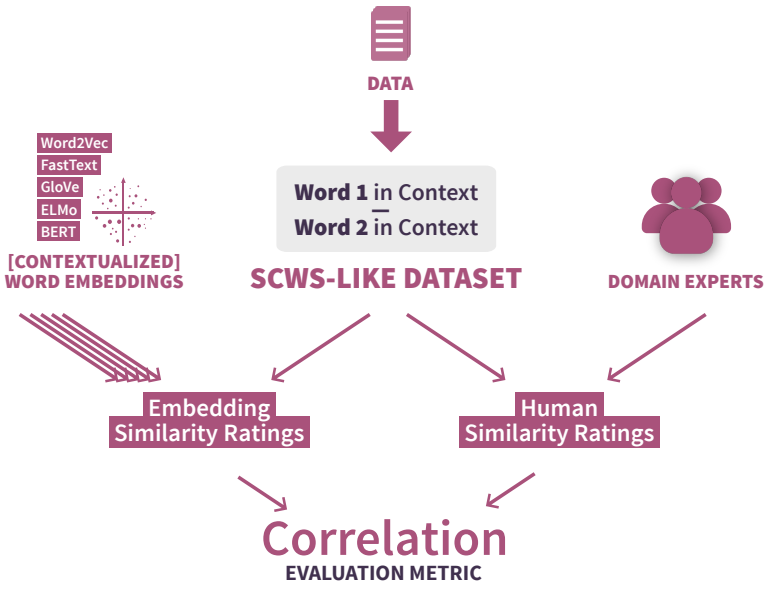


Figure 3.4. Overview of the intrinsic evaluation. Word-in-context similarity ratings of the model variants are compared to human annotated similarity ratings.

models with a higher correlation are better at capturing semantic word-in-context similarity with respect to our data. An overview of the intrinsic evaluation process is depicted in Figure 3.4.

3.4.2 Contextual Words Dataset

Construction

The construction of the contextual words dataset partly follows the construction of the SCWS dataset (Huang et al., 2012). In total, 60 word-in-context pairs are generated by applying the following three steps: (1) sample a list of words from the source data, (2) for each of these words, pair another word which has an interesting relation to the first word and (3) enrich each paired word with context. 70 % (≙ 42) of the word pairs are sampled by automated means, whereas 30 % (≙ 18) are collected manually. The individual steps are described in the following.

Step 1: Sampling First Words. In step 1, a set of first words is sampled. 42 of the 60 first words are collected automatically from the source data considering two word characteristics in order to get a diverse and yet structured set of first words. First, words get categorized by their occurrence frequency and sorted in three buckets: the first bucket contains the 2,000 most frequent words, the second bucket words ranked between 2,000 and 5,000 and the third bucket words ranked between 5,000 and 10,000 in terms of occurrence frequency. Second, for each of these buckets,

words are categorized depending on their POS tag. Nouns, verbs, and adjectives are sampled in a relation of 4 : 2 : 1 as the author deems nouns to be most important for the technical domain and nouns represent the largest POS class in the dataset followed by verbs and adjectives (cf. Section 3.2.3).

In addition, 18 of the 60 first words are collected manually in order to represent the technical domain well. Eight of these constitute keywords from user-generated questions asking for physical quantity values. These questions have been collected in a preliminary project and are further described in Section 3.5.1. For example, from the question *"What should be the electrical potential of the part?"*, the keyword *potential* is taken as a first word. In the following, this word category will be referred to as QUESTION. The next step samples six keywords from the input parsing patterns of Service Mate, as presented in Section 3.1.1. For example, the keyword *resistance* is sampled from the pattern *"(How|What) (high)? $be $PPART* (electric)? resistance $PPART* $1"*. This category will be referred to as PATTERN. Finally, the last four first words are collected from a machine component list. An example of such a component term is the word *engine*. This category will be referred to as COMP. Figure 3.5 graphically illustrates the distribution of the first words.

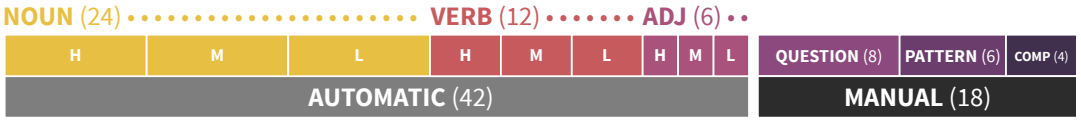


Figure 3.5. Distribution of the first words. 42 words are sampled automatically. 24 of these are nouns, 12 verbs and 6 adjectives, each of which are further categorized due to their occurrence frequency (high (H), medium (M) or low (L)). Furthermore, 18 words are sampled manually, of which 8 are keywords from user-generated questions, 6 are keywords from patterns and 4 are machine-specific component terms.

Step 2: Pairing Words. In this step, the 60 first words collected in step 1 are paired with a second word each. For the 42 automatically sampled words, as well as the words belonging to the PATTERN and COMP categories, WordNet (Miller, 1995) is employed to find second words with an interesting relationship. Huang et al., 2012 argue that such an interesting relationship can be obtained by utilizing WordNet synset relationships. Thus, following Huang et al., 2012, and Manandhar, Klapaftis, Dligach, and Pradhan, 2010, for each first word, this step first chooses a random WordNet synset. Then, a set of words is generated containing hypernyms, hyponyms, holonyms, meronyms, and antonyms of the selected synset. From this set, a second word is sampled, which has to fulfill two requirements. First, it must be contained in our text corpus in order to later augment it with context. Second, it must predominantly occur with the same POS tag as the first word, that is, this step omits second word candidates with a tendency of less

than 90 % towards the POS tag of the first word. Thus, in contrast to SCWS, only noun-noun, verb-verb, and adjective-adjective pairs are sampled.

For the eight words of the QUESTION category, second words are collected so that four of the pairs originate from the same physical category and the other four originate from a different category. For example, for the first word *potential*, a second keyword of the same category would be *volt* since both belong to the electric potential category. On the other hand, the word *ampere* would belong to a different category (electric current). Table 3.2 presents several examples of sampled word pairs.

Table 3.2. Example word pairs, the category of the respective first word as well as the relationship between the two words.

Word 1	Word 2	Category of Word 1	Relationship
station	position	High Frequency Noun	Hypernym Synset
destroy	kill	Medium Frequency Verb	Hyponym Synset
steep	gradual	Low Frequency Adjective	Antonym Synset
potential	current	QUESTION	Different Category
bars	pressure	QUESTION	Same Category
circuit	path	PATTERN	Hypernym Synset
engine	motor	COMP	Hypernym Synset

Step 3: Adding Context. Finally, each word of each pair is enriched with sentential context. In order to do that, for the WordNet generated pairs, this step samples a random sentence from the source data which contains the respective word. Text in the form of tables is skipped so that the human annotators understand the sentential context more easily. Furthermore, additional sentences are manually added to a few items where the author assumes that single sentence context is too vague. For the QUESTION category, random user-generated questions are taken from the question collection that contain the respective word. Table 3.3 depicts the contexts for the word pairs shown above.

Human Similarity Ratings

Before being able to evaluate the word representation models intrinsically, human similarity ratings are needed. The following paragraphs describe the process of obtaining these scores.

Participants. Following the recommendation of Snow, O'Connor, Jurafsky, and Ng, 2008, this work aims to collect ten ratings for each word-in-context pair. In total, 16 (1 female, 15 male) subjects participated. The subjects were on average 34.9 (SD = 9.67) years old. All raters were

Table 3.3. Example word-in-context pairs sampled from the source data corpus.

Word 1	Word 2
The transmission path has been interrupted by an external radio **station** .	Activate the tractor 's control valve to circulate oil in the machine and leave it in this **position** .
Excessive operation of the emergency steering pump will **destroy** it due to overheating .	Touching damaged live parts may cause serious electrocution and injure or even **kill** persons .
In the case of **steep** slopes , muddy ground , and when driving into a clamp silo , the rear wheels take the path of least resistance and controlled tracking is no longer possible .	The **gradual** increase in pressure allows the densification force in the machine to be kept constant .
What should be the electrical **potential** of the part ?	What electric **current** can be measured on the part ?
How many **bars** does the component have ?	What is the **pressure** of this item ?
Fill the engine cooling **circuit** .	Never place yourself in the **path** of the bolt when disassembling it .
Before going under the tractor to carry out this operation , stop the **engine** , apply the handbrake and remove the ignition key .	To stop the **motor** gently , put the control valve in float position by pushing the control fully forward until it engages .

experts working as software engineers, scientific engineers or consultants with an average experience of 5.0 (SD = 5.54) years in the Technical Service domain.

Questionnaire Structure. The 60 word-in-context pairs are divided into four equally large blocks, each containing an equal amount (wherever possible) of the different word pair groups (noun, verb, adjective, QUESTION, PATTERN, and COMP). Each of the four question blocks furthermore contains one checkpoint question to ensure that participants have understood the task and are still paying attention. Checkpoint questions are taken from the SCWS dataset and selected the following way: they are such word pairs where the two words are orthographically equal. For the checkpoint questions, two word pairs are sampled from SCWS that have been rated rather similar and two that have been rated rather dissimilar. Table 3.4 depicts two of the four checkpoint questions.

Table 3.4. Two of the four checkpoint questions to ensure that participants have understood the task and are paying attention. The last column indicates the SCWS dataset rating of the word-in-context pair. This table showcases one checkpoint question with a high SCWS rating and one with a low rating.

Word 1	Word 2	Sim
The longest **suspension** bridge in the world is the Akashi Kaikyo Bridge in Japan.	There are six main types of bridges: beam bridges, cantilever bridges, arch bridges, **suspension** bridges, cable-stayed bridges and truss bridges.	10.0
A cocktail **party** was held at the Gardens Restaurant at 4311 Magnolia Avenue in Burbank, California.	The chairman of the RNC is chosen by the President when the Republicans have the White House or otherwise by the **Party**'s state committees.	1.05

Procedure. To avoid order and fatigue effects, the procedure randomizes the order of the blocks as well as the order of questions inside of each block. After a preliminary pilot study, one subject rating all four blocks turned out to be too cognitively demanding. Therefore, subjects are divided into four groups, where each group rates a subset combination of only three blocks. In an online questionnaire, subjects are asked to rate the similarity of the word-in-context pairs on a scale between 1 and 10 with 1 being *Very Dissimilar* and 10 being *Very Similar*. Figure 3.6 shows an example question.

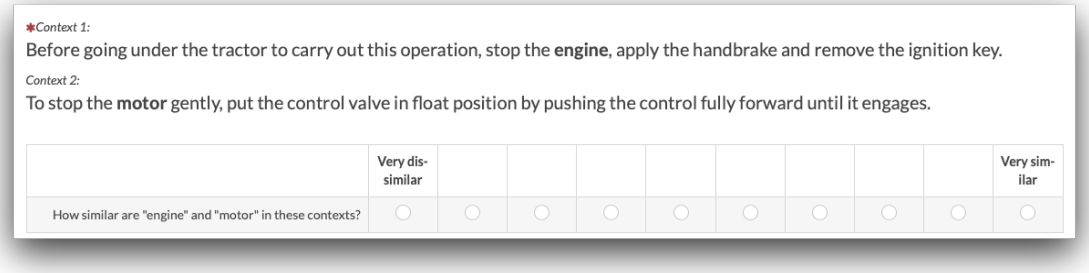


Figure 3.6. A word-in-context pair to be rated by participants in terms of similarity.

Finally, participants are asked to personally estimate their English skill level on a scale between 1 and 7 with 1 being *No Skills* and 7 being *Native*. The author excludes ratings of subjects who either rate their English level below 4 or whose rating of at least one of the checkpoint questions deviates more than two points from the SCWS rating. This has lead to the exclusion of 2 subjects. For each word pair, 10-11 ratings have been received, which averaged yield the final human intuition similarity ratings.

Dataset Analysis

On average, participants rate word pair similarity with a score of 4.87 (SD = 2.42). Table 3.5 gives an overview of three differently rated pairs. The distribution of similarity ratings among word pairs with respect to POS class is shown in Figure 3.7. Examining the distribution over all words indicates that ratings are slightly skewed to the left, i.e., they incorporate a bias towards lower scores. Taking a closer look at ratings with respect to POS class, it becomes apparent that the skew intensifies proceeding from noun pairs via verb pairs to adjective pairs.

Table 3.5. Examples of ratings obtained by the participants. Word-in-context pairs with high, low and mediocre similarity scores are shown.

Word 1	Word 2	Sim
Before going under the tractor to carry out this operation , stop the **engine** , apply the handbrake and remove the ignition key .	To stop the **motor** gently , put the control valve in float position by pushing the control fully forward until it engages .	9.72
Control device has not been notified of control terminal properties (**font** style , font size , etc.) .	The oil flows from the oil pump through holes in the cylinder block to a plate **type** oil cooler .	1.1
Use a **thermometer** to ensure that the temperature does not exceed 180 ° c .	Remove compression spring , piston and the **thermocouple** element .	4.82

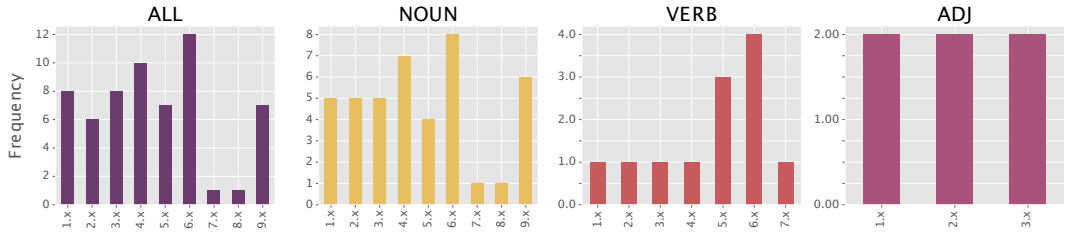


Figure 3.7. Distribution of human similarity ratings by POS. From left to right: distribution over all word pairs, noun word pairs, verb word pairs and adjective word pairs. The bar over the 1.x label indicates the frequency of ratings with a score between [1,2) and so on.

Inter-rater Agreement. Following related work (Padó, Padó, & Erk, 2007; Reisinger & Mooney, 2010; Silberer & Lapata, 2014; Hill et al., 2014), inter-rater agreement is calculated by averaging pairwise computed Spearman's ρ correlation values among all participants. Spearman's ρ is a rank-based correlation coefficient which, given a pair of data arrays X and Y, is computed as

follows:

$$\rho = \frac{\text{cov}(\text{rg}_X, \text{rg}_Y)}{\sigma_{\text{rg}_X} \sigma_{\text{rg}_Y}} \tag{3.1}$$

where $\text{cov}(\text{rg}_X, \text{rg}_Y)$ is the covariance over the rank values of X and Y and σ_{rg_X} and σ_{rg_X} are the respective standard deviations.

The dataset obtains an inter-rater agreement of $\rho = 0.56$. Comparing with other datasets, applying the same method, this score resides markedly above the agreement on SCWS ($\rho = 0.35$), but below WS-353 ($\rho = 0.61$) and SimLex-999 ($\rho = 0.67$). Computing inter-rater agreement with respect to POS class (Figure 3.8) indicates that the raters' consistency is highest on noun pairs ($\rho = 0.60$), followed by verb ($\rho = 0.50$) and adjective pairs ($\rho = 0.40$).

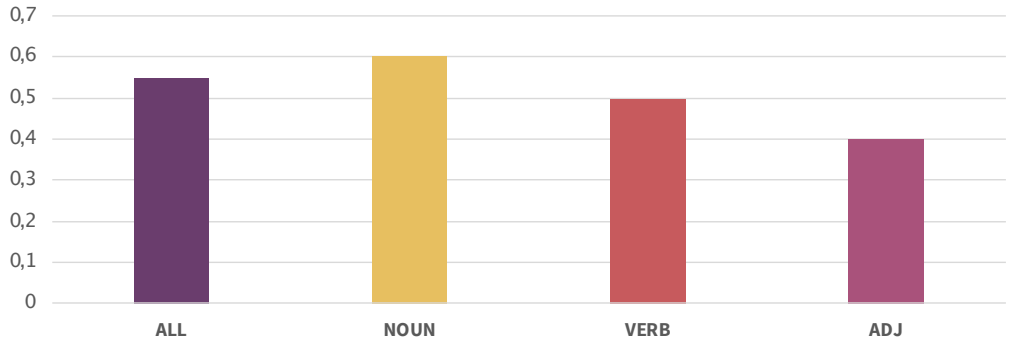


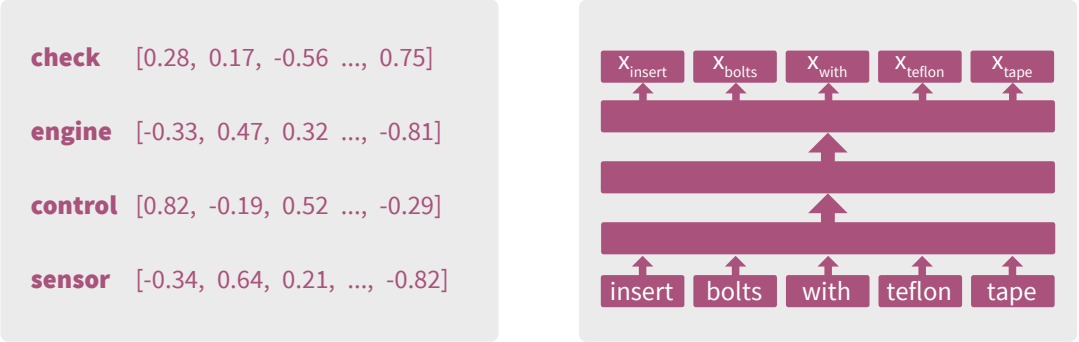
Figure 3.8. Inter-rater agreement (Spearman's ρ) with respect to POS class of word pairs.

3.4.3 Evaluation Setup and Procedure

As introduced in Section 3.4.1, the intrinsic evaluation metric is the correlation between the human annotated similarity ratings and the model cosine similarity ratings. In order to be able to compute cosine similarity values for each model, we first need to obtain word embeddings of the word-in-context pairs. For the instances of Word2Vec, GloVe and FastText, model training (cf. Section 3.3.2) yields a dictionary file where each vocabulary word is associated with an embedding vector (Figure 3.9a). Thus, from the respective dictionary file, the two embedding vectors for each word pair can simply be queried. Note that these three models ignore the sentential context of the words.

On the other hand, the contextual embedding architectures ELMo and BERT constitute pre-trained/fine-tuned deep learning models. Word embeddings are obtained by individually propagating the sentential contexts through the respective deep architecture (Figure 3.9b). This process yields a context embedding tensor of shape $\left[L, len(context), d_h\right]$[13] with L being the number of hidden layers (layer dimension), $len(context)$ the length of the word context (word dimension)

[13]For BERT, the shape is actually $\left[L, len(context) + 1, d_h\right]$ accounting for the additional initial `[CLS]` token.



(a) Word Vector Dictionary File. **(b)** Pre-trained/Fine-tuned Language Model.

Figure 3.9. Word2Vec, GloVe and FastText create dictionary files during training mapping from words to vectors (left). ELMo and BERT on the other hand constitute deep neural architectures. Propagating a word sequence through the model yields contextual embeddings (right).

and d_h the embedding size of the hidden layers (embedding dimension). For ELMo, contextual word embeddings are obtained by averaging along the layer dimension and picking the vector at the index of the target word at the word dimension, resulting in an embedding size of d_h. For BERT, as proposed by Xiao, 2018, the second-to-last layer at the layer dimension and the vector at the index of the target word at the word dimension is picked, also resulting in an embedding size of d_h.

Table 3.6. Word vector dimensions of the baseline and the 13 embedding model variants.

Model	Embedding Dimension
Random Baseline	300
Word2Vec Pre-trained	300
Word2Vec Self-trained (Standard)	300
Word2Vec Self-trained (Lemmatized)	300
GloVe Pre-trained	300
GloVe Self-trained (Standard)	300
GloVe Self-trained (Lemmatized)	300
FastText Pre-trained	300
FastText Self-trained (Standard)	300
FastText Self-trained (Lemmatized)	300
ELMo Pre-trained	1024
ELMo Fine-tuned (Cased, with Stopwords)	1024
ELMo Fine-tuned (Cased, without Stopwords)	1024
BERT Pre-trained	768

Finally, 300-dimensional random vectors are generated for each word in the vocabulary and used as a naïve baseline. Table 3.6 displays the embedding dimensions of all evaluated model variants.

Successively for each model variant, the evaluation is conducted in the following way: for each word pair in the contextual dataset, the cosine similarity of the associated embedding vectors is generated. After that, Spearman's ρ correlation coefficient (cf. Equation 3.1) is computed over all cosine measures and human similarity ratings constituting the intrinsic evaluation metric.

3.5 Extrinsic Evaluation

As pointed out in Section 2.5, extrinsic evaluations are assessments of model performance on downstream NLP tasks. This work conducts two downstream tasks where the experimental setup is kept constant, and only the language representation model (independent variable) is varied. In doing so, one can attribute changes in task performance (dependent variable) to the evaluated model and thus measure its influence on task outcome. The extrinsic evaluation tasks are designed towards the final use case: naturally phrased user input is supposed to be classified to a known category by analyzing its semantic content. The following sections present the two extrinsic evaluation tasks: (1) a question pair classification task (Section 3.5.1) and (2) a question similarity classification task (Section 3.5.2).

3.5.1 Question Pair Classification Task

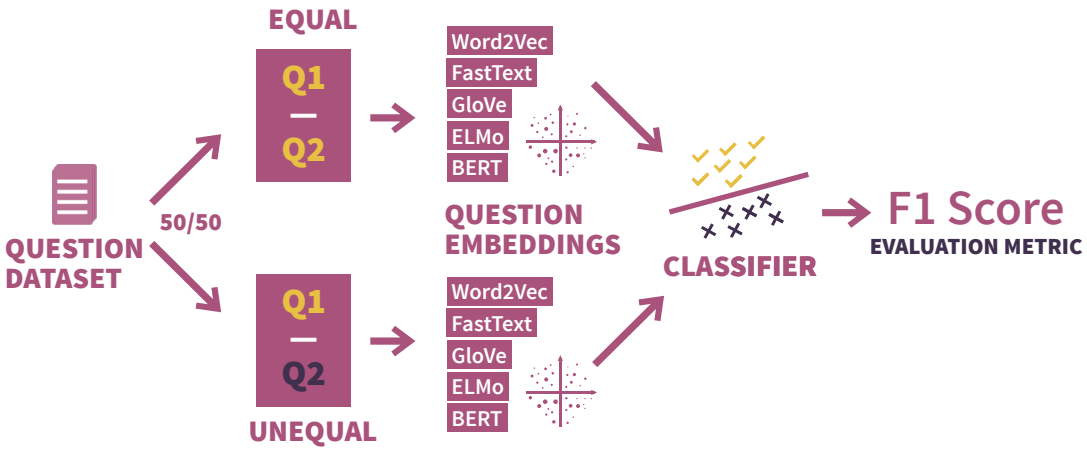


Figure 3.10. Overview of the extrinsic question pair classification task.

In the first extrinsic evaluation task, the language representation models are assessed on question pair classification. The objective is to disambiguate question pairs depending on whether they belong to the same category or not (binary classification task). To this end, user-generated questions from different physical categories are paired and labeled accordingly. After embedding the question pairs, the resulting question embedding vectors are fed into a classifier which is trained to discriminate equal category question pairs from unequal ones. Finally, it is evaluated which embedding technique achieves the best classification score. Figure 3.10 depicts an overview of the task sequence. In the following, each step is described in further detail.

Question Dataset

In the course of a preliminary project, users were asked to articulate questions to a voice assistant asking for different physical quantity values of a machine component. 127 questions have been collected comprising the four categories of electric current, electric potential, barometric pressure, and electric resistance. The questions are phrased in natural language and formulated in a varying form. Table 3.7 presents example questions for each of the four categories. A more elaborate analysis of the user-generated questions is presented in Appendix Section A.1.

Table 3.7. User-generated questions asking for physical quantity values. <x> represents a variable for an arbitrary machine component.

Question	Category
What is the electric current of <x>?	Electric Current
How much voltage does <x> have?	Electric Potential
How many bars does component <x> have?	Barometric Pressure
How resistant is <x>?	Electric Resistance

Originating from the user-generated questions, a question pair dataset is constructed similar to the QQP dataset described in Section 2.5.2. The dataset consists of a training, validation, and test set fraction of paired questions. To do that, for each category, 30 % of the user-generated questions are sampled as holdout set. The remaining 70 % of questions of each category are used for training set construction which is done with the following three steps: first, for each category individually, pairwise combinations are generated, resulting in four sets of equal category question pairs. Next, pairwise combinations of questions coming from differing categories are sampled. Finally, the resulting question pair sets are shuffled together and labeled, indicating whether both questions originate from the same category or not. The validation and test set are

constructed in the same way with the modification that, for each question pair, one question is taken from the holdout set.

The resulting training set consists of 1948 question pairs. The validation and test set incorporate 880 questions each. Each of the three sets contains roughly 50 % equal and 50 % unequal category pairs. Example question pairs are depicted in Table 3.8.

Table 3.8. Example questions from the question pair dataset. The third column indicates whether the two questions are originating from the same category (1) or not (0).

Question 1	Question 2	Same Category
Tell me the ampere value of <x>.	Which current can you measure on <x>?	1
What is the resistance value of <x>?	What is the pressure measurement of <x>?	0

Question Embeddings

In order to assess the influence of the language model variants on the classification task performance, we need to generate embeddings of the questions. To this end, the individual question words are taken, and, for each word, a word embedding is obtained by following the same approach as described in Section 3.4.3. OOV words are embedded with zero vectors. Consequently, for the random baseline model, as well as the variants of Word2Vec, GloVe, and FastText, this process obtains two embedding sets with shapes $\left[len(question\_1), d_h\right]$ and $\left[len(question\_2), d_h\right]$ with $len(question\_\{i\})$ being the word count of the respective question (word dimension) and d_h the word embedding size (embedding dimension). For ELMo and BERT[14], the embeddings are furthermore supplemented with a third dimension of size L, denoting the number of hidden layers. However, these differently shaped question embeddings cannot be fed into a classifier. Thus, the embeddings must be squashed into fixed-sized vectors. In order to do that four different BOW pooling strategies are employed for each question embedding: (1) average pooling, (2) IDF average pooling, (3) max pooling, and (4) average + max pooling.

Average Pooling. As pointed out in Section 2.4.1, a simple yet effective way to generate embeddings of word sequences takes the average along the word dimension (Figure 3.11). For the contextual model BERT, which incorporates an additional layer dimension, the second-to-last

[14] Again, for BERT, the word dimension has a length of $len(question\_\{i\}) + 1$ accounting for the initial [CLS] token.

layer at the layer dimension is picked before averaging along the word dimension as has already been done for the intrinsic evaluation. Thus, for a given question, average pooling yields an embedding size of d_h for Word2Vec, GloVe, FastText, and BERT. For ELMo, all hidden layer embeddings are concatenated before averaging. Consequently, averaging along the word dimension yields a question embedding size of $L \cdot d_h$ for the ELMo variants.

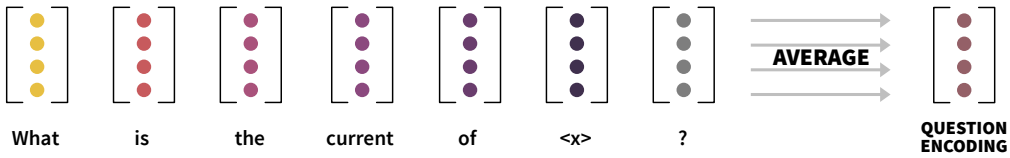


Figure 3.11. Average pooling along the word dimension.

IDF Average Pooling. This strategy is identical to average pooling described above, with the modification, that each word vector is weighted with the corresponding word's Inverse Document Frequency (IDF). The IDF is typically used to leverage words that occur only in a few documents of a document corpus (Jurafsky & Martin, 2014). The intuition is that these words carry a lot of semantic information for the given document.

For a given word t, this strategy first calculates the document frequency $\texttt{df}_t$, which is the number of documents in our source corpus in which t appears. With N being the total number of documents, the IDF is calculated following Equation 3.2.

$$\texttt{idf}_t = \log_{10}\left(\frac{N}{\texttt{df}_t}\right) \tag{3.2}$$

Consequently, performing the average pooling operation weighted with the IDF of each word yields the same embedding dimension as plain average pooling.

Max Pooling. Following Conneau et al., 2017, a max pooling strategy is also employed to obtain fixed-size question encodings. In contrast to average pooling, max pooling picks the maximum value along the word dimension for each index of the embedding vectors (Figure 3.12). Hence, the embedding sizes are identical to those of the average pooling operation.

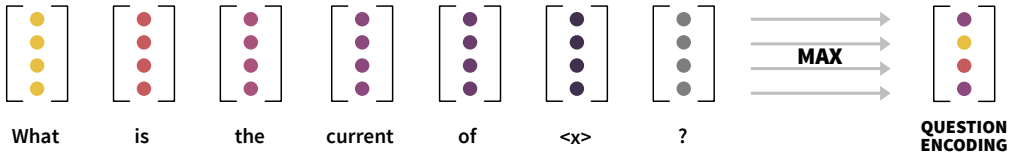


Figure 3.12. Max pooling along the word dimension.

Average + Max Pooling. Finally, average and max pooling are combined. To do so, the average pooled and the max pooled question encodings are concatenated. As a result, this strategy outputs question embeddings of twice the size of the previously described approaches.

Classification

Proceeding from the pooled question encodings, the two associated encoding vectors of each question pair are concatenated. The resulting vector is used as input to a classifier. The specific input dimensions for the individual language representation model variants are depicted in Table 3.9.

A multi-layer perceptron (Rosenblatt, 1958) with a single hidden layer of size 50 and ReLU activation is used for classifying whether question pairs originate from the same category or not. The perceptron is trained on the training partition of the question pair dataset described above. The classifier is trained with a batch size of 4. Adam (Kingma & Ba, 2014), a technique dynamically adapting the learning rate, is used as optimization function. Early stopping is employed with patience of 10, monitoring accuracy on the validation set. After the first training has completed, additional perceptrons are trained with patience of 5 until no more improvement on validation accuracy has shown. The perceptron with the best validation accuracy is selected for final evaluation.

Evaluation Metric

Model performance is evaluated with the F1 score. For this purpose, for each language representation model instance, the respective question pair encodings of the test set are propagated through the trained classifier perceptron. For a binary classification task like the one at hand, F1 measures the harmonic mean between precision and recall of the labels predicted by the classifier and the ground truth labels:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.3}$$

where precision and recall are defined as relations between the number of question pairs that have been predicted either correctly as equal (true positives – TP), incorrectly as equal (false positives – FP), or incorrectly as unequal (false negatives – FN):

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.4}$$

Table 3.9. Input dimensions of the question pair embeddings of the different language representation models. Due to concatenating the question pair encodings, the dimensions are $2 \cdot L \cdot d_h$ for ELMo (with $L = 3$) and $2 \cdot d_h$ for the remaining models. Combining average and max pooling doubles the input dimension.

Model	Input Dimension	
	Avg / IDF Avg / Max Pooling	Avg + Max Pooling
Random Baseline	600	1,200
Word2Vec Pre-trained	600	1,200
Word2Vec Self-trained (Standard)	600	1,200
Word2Vec Self-trained (Lemmatized)	600	1,200
GloVe Pre-trained	600	1,200
GloVe Self-trained (Standard)	600	1,200
GloVe Self-trained (Lemmatized)	600	1,200
FastText Pre-trained	600	1,200
FastText Self-trained (Standard)	600	1,200
FastText Self-trained (Lemmatized)	600	1,200
ELMo Pre-trained	6,144	12,288
ELMo Fine-tuned (Cased, with Stopwords)	6,144	12,288
ELMo Fine-tuned (Cased, without Stopwords)	6,144	12,288
BERT Pre-trained	1,536	3,072

3.5.2 Question Similarity Classification Task

As a second extrinsic evaluation, a question similarity classification task is conducted. The goal is to classify user questions correctly to one out of four different classes (multiclass classification task). The same user-generated questions as in the previous extrinsic evaluation are used for classification. These questions are classified by calculating pairwise cosine similarity scores with category centroids obtained from the Service Mate voice assistant patterns. Again, it is evaluated for which embedding technique classification works best. Figure 3.13 depicts an overview of the task sequence. Each task step is described more thoroughly in the following.

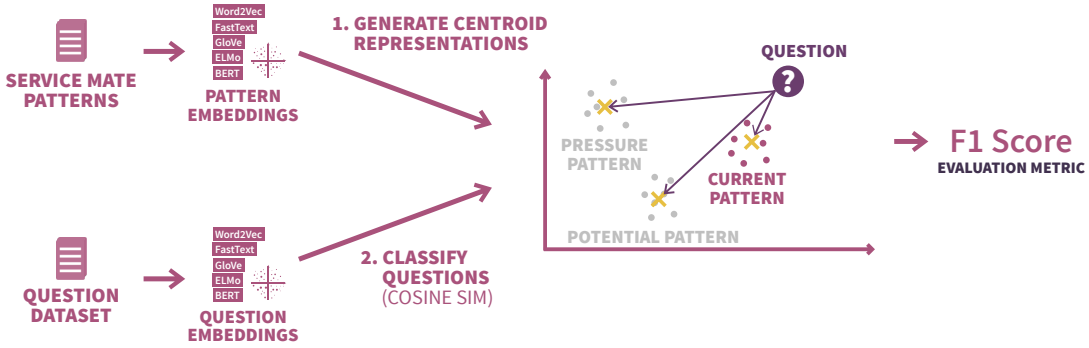


Figure 3.13. Overview of the extrinsic question similarity classification task.

Service Mate Patterns

As described in Section 3.1.1, the current voice assistant of Service Mate uses a pattern matching approach for parsing user input. As part of this evaluation, these patterns are taken advantage of since embeddings are generated from them.

At first, pattern-specific operators are removed by unfolding the patterns to its terminals. Listing 3.4 presents an example pattern of the electric potential category as well as a part of the unfolded word sequences derived from it. This unfolding process is performed on all patterns which currently exist for the four categories of electric current, electric potential, barometric pressure, and electric resistance. Note, that the automatic unfolding operation does not necessarily yield grammatically correct sentences. However, the author assumes the unfolded word sequences to be sufficient to generate semantically meaningful pattern embeddings.

Listing 3.4. Example of an electric potential pattern (top) and a part of its unfolded version (bottom). <s> and </s> denote special start and end of sequence markers.

```
(How|What) high? $be $PPART* electric? (potential|voltage) $PPART* $1
---
<s> How high be PPART* electric potential PPART* $1 </s>
<s> How high be PPART* electric voltage PPART* $1 </s>
<s> How high be PPART* potential PPART* $1 </s>
<s> How high be PPART* voltage PPART* $1 </s>
<s> How be PPART* electric potential PPART* $1 </s>
[...]
<s> What high be PPART* voltage PPART* $1 </s>
<s> What be PPART* electric potential PPART* $1 </s>
<s> What be PPART* electric voltage PPART* $1 </s>
<s> What be PPART* potential PPART* $1 </s>
<s> What be PPART* voltage PPART* $1 </s>
```

Question Dataset

The same user-generated questions as described in Section 3.5.1 are used for classification. In contrast to the previous evaluation, questions are not paired for this evaluation, but simply labeled with their respective physical category, as shown in Table 3.10.

Table 3.10. User-generated questions labeled with physical categories.

Question	Category
What is the electric current of <x>?	Electric Current
How much voltage does <x> have?	Electric Potential
How many bars does component <x> have?	Barometric Pressure
How resistant is <x>?	Electric Resistance

Pattern and Question Embeddings

Fixed-size embeddings of the unfolded patterns, as well as the user-generated questions, have to be generated in order to perform the classification task. Embeddings for both patterns and questions are generated following the same approach described in the previous extrinsic evaluation

(Section 3.5.1). The same pooling strategies are applied in order to obtain fixed-size encodings. The output of this step is four sets of pattern embeddings as well as four sets of user question embeddings, one for each physical category respectively.

Classification

Proceeding from the pattern and question embeddings, this extrinsic evaluation task aims to classify the question embeddings to their respective categories correctly. Classification is performed by the following two steps. First, centroid vectors of the pattern embeddings are generated. Second, the question embeddings are classified based on cosine similarity with the pattern centroids. The two steps are pointed out in further detail below.

Step 1: Generate Centroid Pattern Representations. Centroid embeddings for each of the four physical categories are generated by pooling together the associated embeddings of the unfolded patterns. Pooling is conducted by taking the average of the corresponding pattern embeddings, which yields a centroid vector. This centroid constitutes a fixed-size representation of the respective category. Centroid generation is visualized in Figure 3.14.

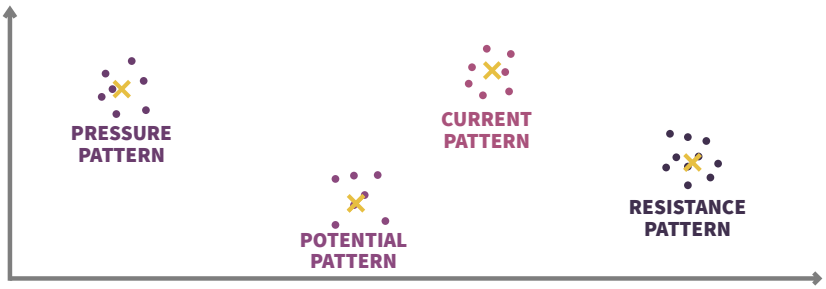


Figure 3.14. Pattern centroid generation. Each dot depicts an embedding of an unfolded pattern. Yellow Xs represent the centroids of the respective patterns.

Step 2: Classify Question Embeddings. Next, the question encoding vectors are classified by evaluating the similarity with the pattern centroids. To this end, for each question encoding and pattern centroid, pairwise cosine similarity values are calculated. Question encodings are classified to the category of the pattern centroid which emerges most similar. This classification process is visualized in Figure 3.15.

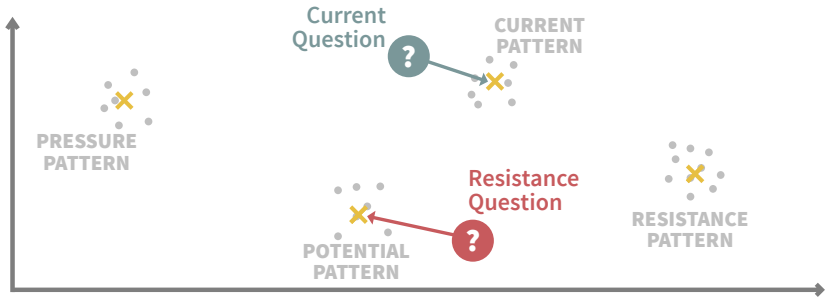


Figure 3.15. Classification of question embeddings. Questions are categorized by their cosine similarity with the pattern centroids. The green symbol (current question) depicts a correctly and the red symbol (resistance question) an incorrectly categorized question.

Evaluation Metric

The performance of the different language representation model variants is evaluated using macro-averaged F1 score: each of the four classes is treated as an isolated binary classification problem, and the individual F1 scores are calculated by comparing predicted and ground truth labels as described in Section 3.5.1. Subsequently, the four individual F1 scores are averaged yielding the final evaluation metric:

$$F_1^{\texttt{macro}} = \texttt{avg}(F_1^{\texttt{electric\_current}}, F_1^{\texttt{electric\_potential}}, F_1^{\texttt{pressure}}, F_1^{\texttt{electric\_resistance}}) \quad (3.5)$$

3.6 Summary

This chapter was concerned with the methodology applied for evaluating the performance of different language representation models for the use case of improving a voice assistant in the Technical Service domain.

First, the semantic information system Service Mate was described in Section 3.1. Its voice assistant, which is currently based on naïve pattern matching, serves as a use case for the methods outlined in this chapter.

Section 3.2 gave an overview of the data corpus and preprocessing steps used in the course of this work: proceeding from XHTML documentation files of harvesting machines, two text token corpora are generated – one version with inflected and one with lemmatized word tokens.

Next, in Section 3.3, the language representation model selection and training processes were described. Five different models have been chosen for evaluation, three of which are plain word

vector approaches (Word2Vec, GloVe, and FastText) and two are approaches capable of generating contextualized word embeddings (ELMo and BERT). In total, 13 model instances were obtained for evaluation.

Section 3.4 gave detailed insight into the intrinsic evaluation procedure. A contextualized word similarity dataset was produced containing human expert annotations of word-in-context similarity. These annotations are compared to cosine similarity scores calculated by the different language representation model instances. A high correlation between human-annotated scores and model scores indicates that the respective model rates word pair similarity in a similar way as a human expert rater does.

Finally, Section 3.5 was concerned with two extrinsic evaluation tasks. The first extrinsic task aimed for predicting whether pairs of questions belonged to the same category or not (binary classification). The objective of the second extrinsic task was to correctly assign questions to one out of four categories (multiclass classification).

Combining the results of the intrinsic and extrinsic evaluations gives a holistic and nuanced insight into model performance with respect to the use case. The evaluation results are stated in the following chapter.

4 Results

This chapter presents the outcomes of the intrinsic and extrinsic evaluations of the language representation models. First, some visual sentence similarity plots for two different models are showcased in Section 4.1. After that, the results of the intrinsic evaluation are outlined in Section 4.2, followed by the results of the two extrinsic evaluation tasks in Section 4.3.

4.1 Sentence Similarity Plots

In order to gain some intuition into how the different models encode words and sentences, this section showcases Principal Component Analysis (PCA) projections of the multi-dimensional embeddings. For this purpose, the human-generated questions described in Section 3.5.2 are utilized: five example items are sampled from each question category. These questions are embedded using average pooling over their corresponding word embeddings as described in Section 3.5.1. For visualization, dimensionality reduction is conducted, and two principal components are extracted using PCA.

Figure 4.1 depicts the two-dimensional projections of the question encodings embedded with the FastText instance that was trained on the standard dataset. The figure illustrates that this model is especially strong at mapping syntactic relationships: "How"-questions are located in the upper half of the plot whereas "What"-questions cluster around the left center. Questions starting with "Tell" are shifted to the right. On the semantic end, the figure illustrates that pressure questions are positioned on the right-hand side. The model furthermore clusters questions from the electric categories together. Inside these clusters, there is another slight trend visible: questions from the current category are positioned more to the bottom whereas potential questions reside at the top, and resistance questions are located somewhere in the middle.

Figure 4.2 depicts the two-dimensional projections of the question encodings embedded with the GloVe instance that was trained on the standard dataset. In this plot, there are no clear trends visible in terms of syntactic representation. Furthermore, apparent semantic clusters are not forming either.

Sentence similarity plots for the remaining pre-trained, self-trained, and fine-tuned models are depicted in Appendix Section B.1.

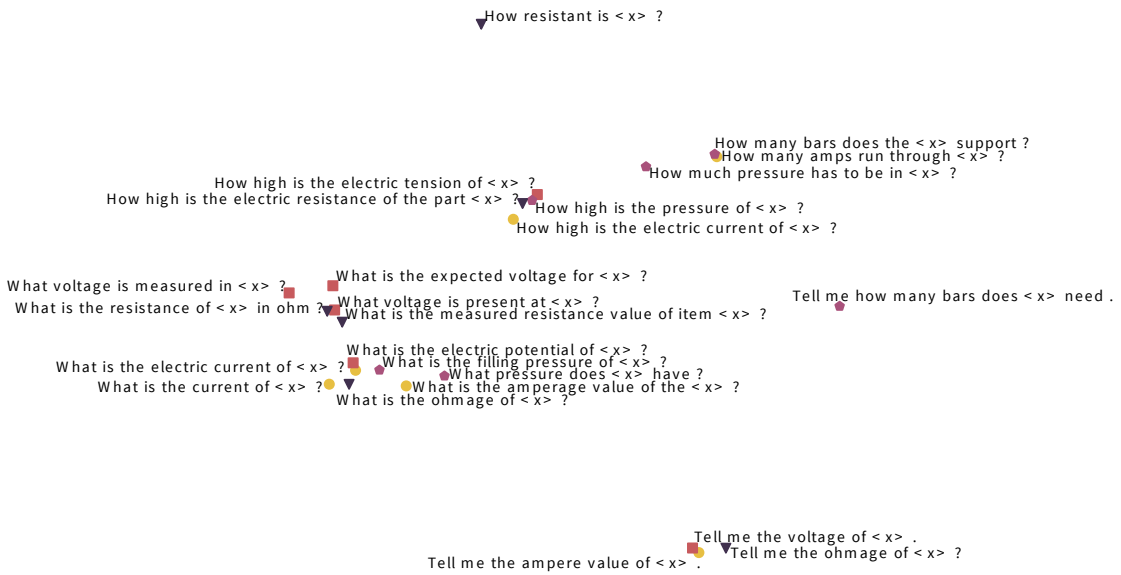


Figure 4.1. Two-dimensional projections of questions embedded with the FastText instance trained on the standard dataset. The embeddings cluster both in terms of syntactic as well as semantic relations.

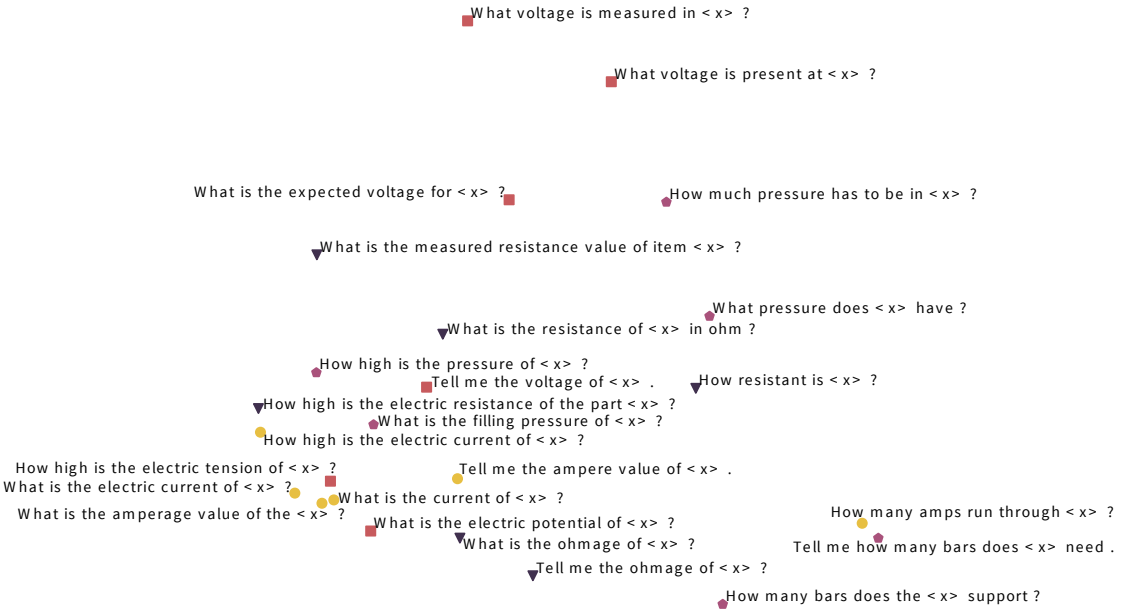


Figure 4.2. Two-dimensional projections of questions embedded with the GloVe instance trained on the standard dataset. Neither syntactic nor semantic obvious clusters are visible.

Table 4.1. Spearman's ρ correlation values between model variants and human ratings.

Model	Spearman's ρ
Random Baseline	0.0382
Word2Vec Pre-trained	0.1762
Word2Vec Self-trained (Standard)	0.2835
Word2Vec Self-trained (Lemmatized)	0.2917
GloVe Pre-trained	0.1850
GloVe Self-trained (Standard)	0.1547
GloVe Self-trained (Lemmatized)	0.0910
FastText Pre-trained	0.2663
FastText Self-trained (Standard)	0.3552
FastText Self-trained (Lemmatized)	0.4439
ELMo Pre-trained	0.4111
ELMo Fine-tuned (Cased, with Stopwords)	**0.4641**
ELMo Fine-tuned (Cased, without Stopwords)	0.4500
BERT Pre-trained	0.4463

4.2 Intrinsic Evaluation

Table 4.1 shows the results of the intrinsic evaluation; a graphical representation is depicted in Figure 4.3. The scores denote the correlation between human-rated and model-rated similarity of word-in-context pairs. All model variants obtain distinctly higher Spearman values than the Random Baseline model that barely shows any correlation. The highest performance is achieved by the ELMo model fine-tuned on the cased dataset with stopwords included. The instance fine-tuned on the dataset without stopwords ranges close behind it. BERT achieves the highest correlation score among all pre-trained model instances followed by the pre-trained ELMo instance. FastText is the best performing model of the three uncontextualized approaches with the variant trained on the lemmatized dataset version achieving results in the range of the contextualized approaches. At the other end of the spectrum, Word2Vec constitutes the worst pre-trained instance whereas GloVe represents the poorest performing self-trained model, with the variant trained on the lemmatized word tokens ranking last.

4.2.1 Model Performance with respect to POS Class

With respect to POS class, averaging Spearman values over all models (without the baseline model) yields a correlation score of $\rho = 0.31$ ($SD = 0.15$) for noun pairs, $\rho = 0.34$ ($SD = 0.24$) for verb pairs and $\rho = 0.35$ ($SD = 0.25$) for adjective pairs. The higher standard deviation values for

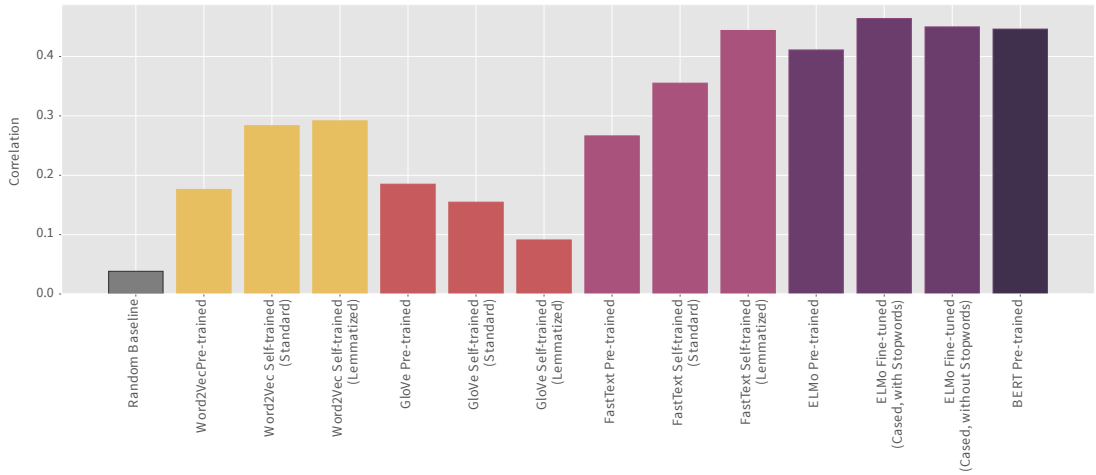


Figure 4.3. Spearman correlation of the evaluated models for the intrinsic evaluation.

verb and adjective correlation indicate higher inter-model variation among these POS classes. This is in line with the lower human inter-rater agreement scores on these classes. Therefore, scores on these classes should be interpreted more cautiously. Figure 4.4 depicts POS scores for the best performing variant of each model, visually indicating higher inter-model variation among verb and adjective pairs.

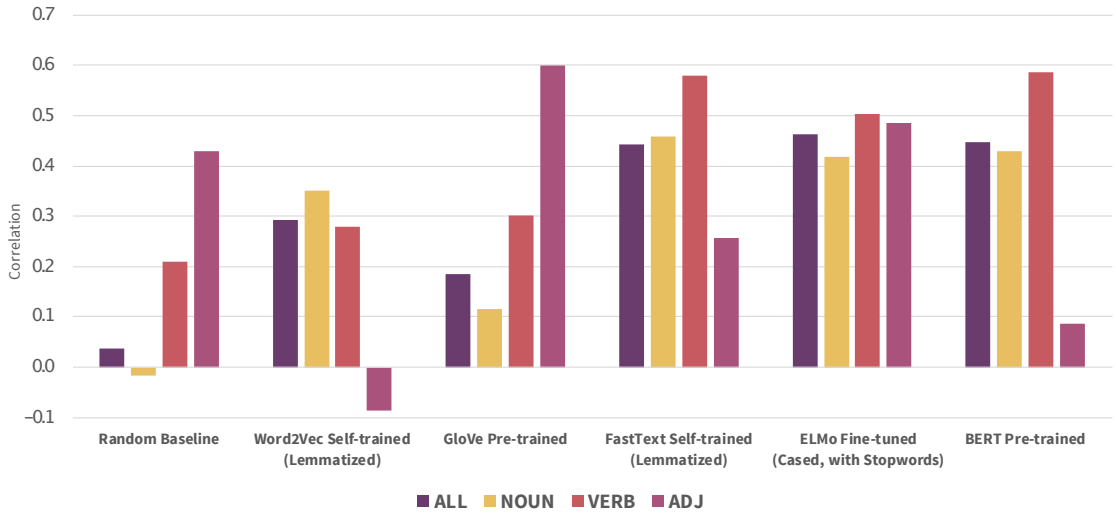


Figure 4.4. Spearman correlation results by POS class. For each model, the overall best performing variant is showcased.

4.2.2 Benefits of Self-training and Fine-tuning

Considering pre-trained versions is especially important when it comes to integrating word vector models into voice assistants for domains with sparse or no training/fine-tuning data. Figure

4.5 plots the correlation scores of the pre-trained variants against the best self-trained/fine-tuned model variants of Word2Vec, GloVe, FastText, and ELMo.

It becomes apparent that self-training and fine-tuning models show beneficial over using pre-trained variants for all models but GloVe. In this context, it is worth noting, that there are no large margins between the pre-trained variant of ELMo and its fine-tuned counterparts. This indicates that, for this task, the effect of self-training for Word2Vec and FastText is stronger than the effect of fine-tuning for ELMo. Furthermore, the pre-trained version of BERT also achieves high Spearman values outperforming all other models except for the fine-tuned ELMo variants.

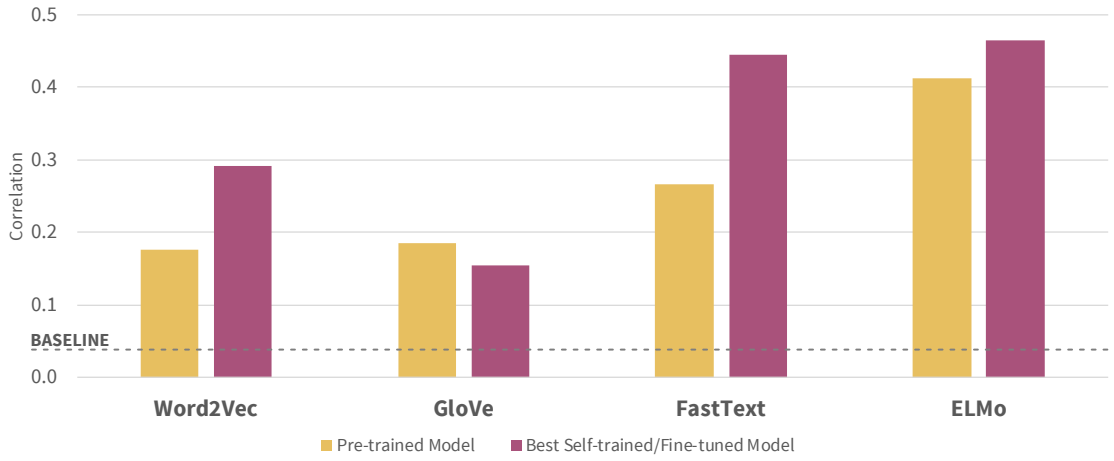


Figure 4.5. The effect of self-training and fine-tuning on intrinsic task performance. For each model, the overall better performing variant is depicted for the self-trained/fine-tuned case.

4.2.3 Effects of Lemmatization and Stopwords

A similar trend arises for lemmatizing word tokens before model training: it is beneficial for the task performances of Word2Vec and FastText, but not for GloVe. For ELMo, fine-tuning on a dataset including stopwords marginally boosts performance over fine-tuning on a dataset without stopwords.

4.3 Extrinsic Evaluation

4.3.1 Question Pair Classification

Table 4.2 presents the results of the first extrinsic evaluation. Utilizing a multi-layer perceptron, the objective was to classify question pairs in terms of whether the two questions belonged to the same category or not. The F1 score denotes the performance on this binary classification task.

Table 4.2. Model performance (F1) on the first extrinsic evaluation task. The scores are separated into columns depending on the respective pooling strategy. For each model, the highest score is printed in bold type face. The overall best score is underlined.

Model	F1 Score			
	Avg	IDF Avg	Max	Avg + Max
Random Baseline	0.0	**0.7936**	0.0	0.1061
Word2Vec Pre-trained	0.9689	0.9149	**0.9737**	0.9571
Word2Vec Self-trained (Standard)	0.9661	0.9420	0.9471	0.9600
Word2Vec Self-trained (Lemmatized)	0.9459	0.9524	0.9218	0.9006
GloVe Pre-trained	**0.9886**	0.9463	0.9748	0.9376
GloVe Self-trained (Standard)	0.9033	0.9059	0.9034	0.9163
GloVe Self-trained (Lemmatized)	0.8945	0.9379	0.8800	0.9049
FastText Pre-trained	0.7513	0.8844	0.6249	0.5868
FastText Self-trained (Standard)	0.9793	0.9527	0.9467	0.9747
FastText Self-trained (Lemmatized)	**0.9898**	0.9400	0.9680	0.9710
ELMo Pre-trained	0.9613	0.9498	0.9119	0.9236
ELMo Fine-tuned (Cased, with Stopwords)	0.9344	0.9614	0.5545	0.9095
ELMo Fine-tuned (Cased, without Stopwords)	**0.9909**	0.9638	0.8794	0.9399
BERT Pre-trained	0.7963	**0.8033**	0.7277	0.7100

A strong baseline is constituted by the IDF-average-pooled random embeddings that outperform several model variants.

Furthermore, similarly to the intrinsic evaluation, the highest performance is achieved by fine-tuned ELMo. However, for this task the average-pooled variant fine-tuned on the dataset without stopwords performs better than the one trained on the version including stopwords.

Average-pooled GloVe achieves the highest F1 score among all pre-trained model instances, followed by strong results of the pre-trained Word2Vec instance. In line with the intrinsic results, self-trained FastText again is the best performing model of the three uncontextualized approaches: the average-pooled variant trained on the lemmatized dataset achieves a score close to the leading ELMo variant. However, the pre-trained version of FastText performs distinctly worse than the two self-trained variants.

The worst performance is represented by the max-pooled ELMo model fine-tuned on the dataset with stopwords. Pre-trained BERT achieves scores marginally revolving above the random baseline for the variants using average and IDF average pooling, and such below the baseline for the other two pooling strategies.

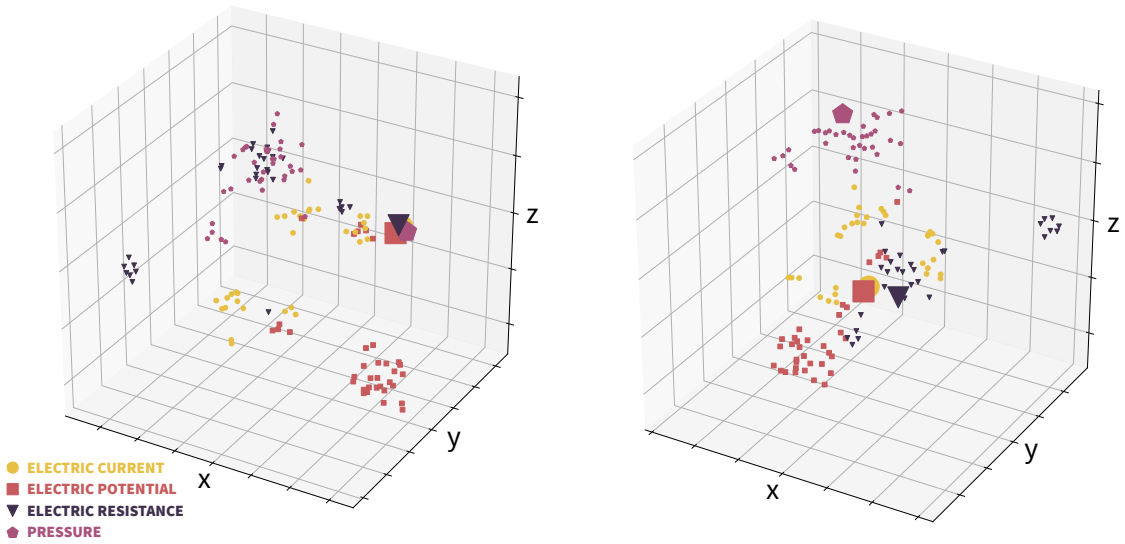
4.3.2 Question Similarity Classification

Table 4.3. Model performance (macro-averaged F1) on the second extrinsic evaluation task. The scores are separated into columns depending on the respective pooling strategy. For each model, the highest score is printed in bold type face. The overall best score is underlined.

Model	Macro-averaged F1 Score			
	Avg	IDF Avg	Max	Avg + Max
Random Baseline	0.7927	**0.8001**	0.7448	0.7724
Word2Vec Pre-trained	0.8481	0.8475	0.8326	0.8326
Word2Vec Self-trained (Standard)	0.8389	0.8730	0.8515	0.8800
Word2Vec Self-trained (Lemmatized)	0.8558	**0.8878**	0.8085	0.8447
GloVe Pre-trained	0.7744	0.7293	**0.8624**	0.8424
GloVe Self-trained (Standard)	0.7771	0.8107	0.8108	0.8251
GloVe Self-trained (Lemmatized)	0.8086	0.8250	0.8147	0.8011
FastText Pre-trained	0.2347	0.6862	0.1890	0.1980
FastText Self-trained (Standard)	0.8806	0.8731	0.7514	0.7028
FastText Self-trained (Lemmatized)	**0.9465**	0.8802	0.7837	0.8225
ELMo Pre-trained	0.6546	0.7692	0.6184	0.5881
ELMo Fine-tuned (Cased, with Stopwords)	0.7728	0.8383	0.6745	0.5934
ELMo Fine-tuned (Cased, without Stopwords)	0.8341	**0.9058**	0.8154	0.8086
BERT Pre-trained	0.4180	**0.4907**	0.2497	0.2883

Table 4.3 shows the results of the second extrinsic evaluation. The objective was to classify questions correctly to one out of four categories. The macro-averaged F1 score denotes the performance of the model instances on this multiclass classification task. Like with the first extrinsic evaluation, the IDF-average-pooled random embeddings set up a strong baseline again, outperforming even more model variants than in the previous task.

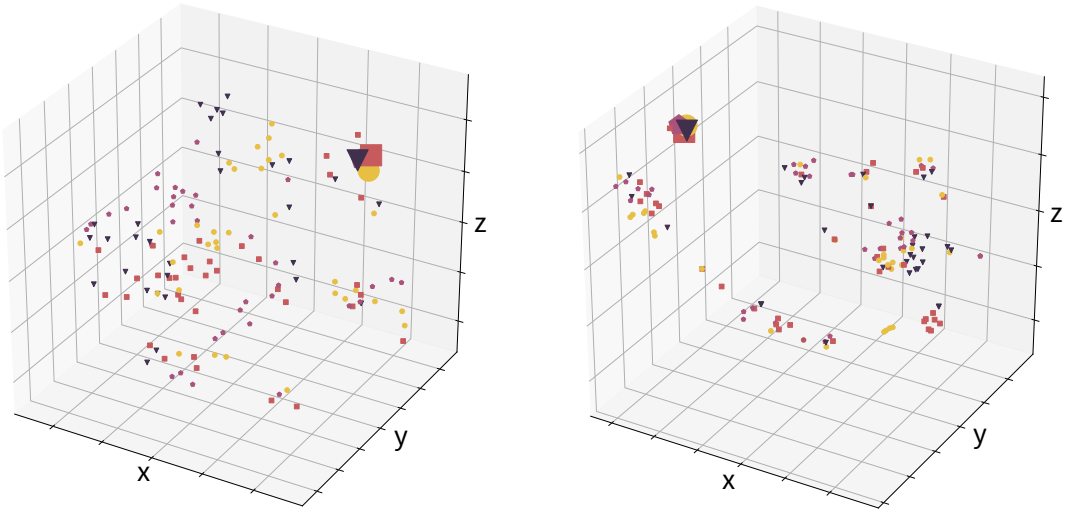
In line with the previous findings, the average-pooled FastText instance trained on the lemmatized dataset version and the fine-tuned ELMo instance (without stopwords, IDF average pooling) represent the strongest competitors. GloVe again performs best among all pre-trained model instances: this time, the version using max pooling attains the highest F1 score. It is again followed by the pre-trained Word2Vec instance. Figure 4.6 illustrates three-dimensional t-distributed Stochastic Neighbor Embedding (t-SNE) projections (Maaten & Hinton, 2008) of the individual question embeddings as well as the category pattern centroids. The plots depict the best performing FastText model (Figure 4.6a) as well as the best performing GloVe model (Figure 4.6b).



(a) FastText Self-trained (Lemmatized, Avg Pooling).

(b) GloVe Pre-trained (Max Pooling).

Figure 4.6. Three-dimensional t-SNE projections of high performing model instances. Pattern centroids are illustrated by large symbols, question embeddings by small symbols. Category clusters are forming for both model instances. On the three dimensional projections, for FastText (left) the pattern centroid embeddings are less distinct than for GloVe (right).



(a) BERT Pre-trained (Max Pooling).

(b) FastText Pre-trained (Max Pooling).

Figure 4.7. Three-dimensional t-SNE projections of low performing model instances. Both BERT (left) and FastText (right) fail to generate distinct category clusters.

Like previously, the pre-trained version of FastText performs distinctly worse than the two self-trained variants. Accordingly, it is the max-pooled version of this model ranking last. Finally, the pre-trained BERT achieves rather low scores as well, performing substantially worse than

the baseline model. Three-dimensional question and pattern centroid t-SNE projections of low performing models are shown in Figure 4.7. The projections represent the worst BERT as well as the worst FastText instance.

4.3.3 Effects of Pooling Strategy

For both extrinsic evaluation tasks, no trend showed towards a specific pooling strategy. Among the five models, the best results are achieved with either average, IDF average, or max pooling. Only the combination of average and max pooling did not achieve a top score when considering overall model performance. These findings are in line with research as there has not yet been reported a single best pooling strategy.

4.3.4 Benefits of Self-training and Fine-tuning

In order to make assertions about the benefits of self-training and fine-tuning with respect to the extrinsic evaluation tasks, one must differentiate between the different models. Figure 4.8 plots the scores of the best pre-trained variants against the respective best self-trained/fine-tuned variant for the first extrinsic task, Figure 4.8 does the same for the second extrinsic task. For closer examination, Receiver Operator Characteristic (ROC) curves and associated Area Under the Curve (AUC) scores comparing pre-trained and self-trained/fine-tuned model performance on the first extrinsic task are depicted in Appendix Section B.2.

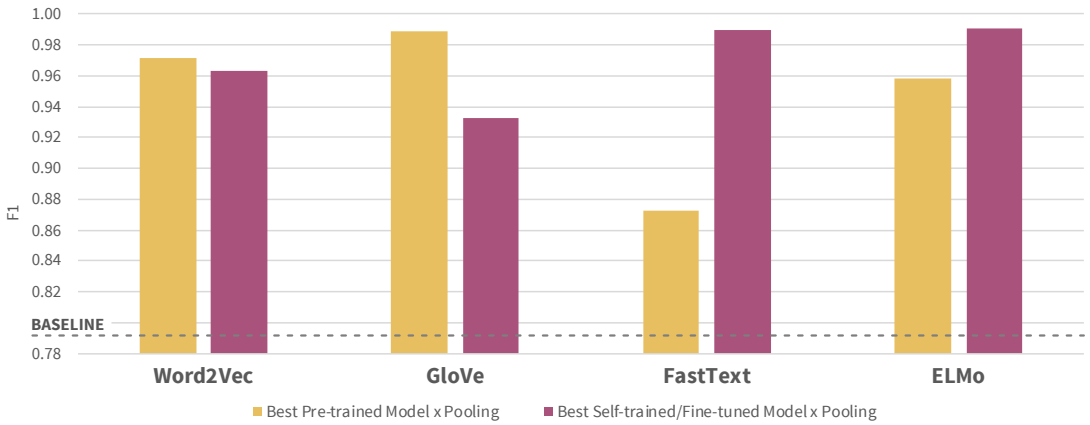


Figure 4.8. The effect of self-training and fine-tuning on the performance of the first extrinsic task. For each model, the best performing combination of model training and pooling variant is depicted.

Looking at both tasks, Word2Vec shows rather even results between the pre-trained and self-trained instances: Whereas in the first task, an instance of the pre-trained model (max-pooling strategy) performs slightly better than the self-trained variants, the opposite is true for the second

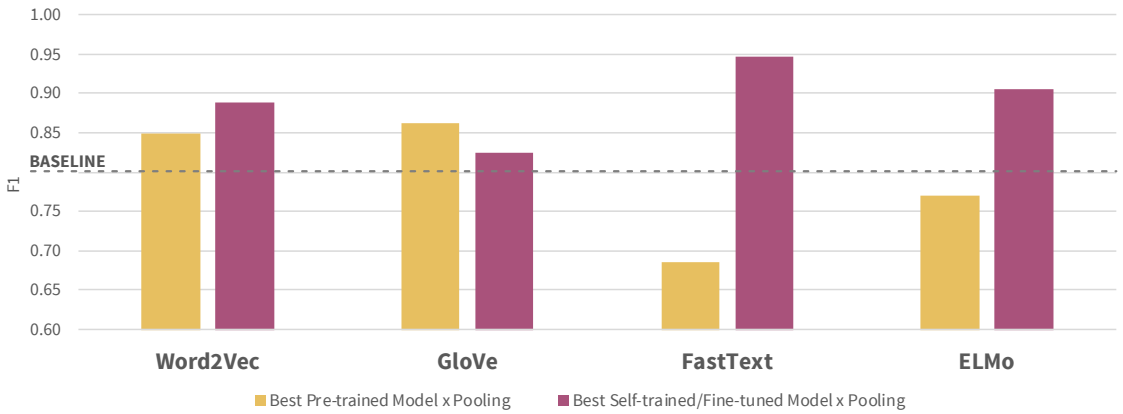


Figure 4.9. The effect of self-training and fine-tuning on the performance of the second extrinsic task. For each model, the best performing combination of model training and pooling variant is depicted.

task. However, in total, there is no apparent difference between the pre-training and self-training noticeable.

GloVe, on the other hand, paints a different picture: similarly to the results of the intrinsic evaluation, for the first task, the pre-trained model outperforms every single self-trained instance. In fact, the worst pre-trained pooling strategy performs better than the best self-trained model-pooling combination. The difference is not as strong for the second task, but overall, the pre-trained model is able to attain better results on this objective as well.

In contrast to GloVe, for FastText, the self-trained instances perform significantly better than the pre-trained model. Whereas the self-trained version trained on the lemmatized dataset constitutes top ranking scores for both tasks, the pre-trained model performs rather poor, especially when it comes to classifying questions to their respective category in the second task.

Finally, the task performance of ELMo benefits from fine-tuning. However, the differences are not as strong as the differences between the pre-trained and self-trained FastText instances. Especially for the second task, fine-tuning on the cased dataset version without stopwords boosts the performance of ELMo to reach scores above the Baseline model.

4.3.5 Effects of Lemmatization and Stopwords

The effect of lemmatization also has to be examined for each model individually. For the performance of Word2Vec, lemmatizing training tokens does not influence model performance heavily. Whereas three of four model instances trained on the lemmatized dataset perform slightly worse on the first task, there are no overall differences visible on the second task. Similar conclusions

can be drawn for GloVe: even though, to a small degree, three out of four variants perform better in the second task, GloVe does not seem to show substantial systematic differences between the model trained on the lemmatized dataset and the one trained on standard tokens either. On the other hand, in line with the findings of the intrinsic evaluation, for each pooling strategy of FastText, the respective model instance trained on the lemmatized tokens performs better than its standard counterpart.

Finally, considering ELMo, there are also differences noticeable between the model version fine-tuned on the dataset with stopwords and the one fine-tuned without stopwords. In contrast to the intrinsic evaluation, however, the model fine-tuned without stopwords obtains considerably better scores for each pooling variant.

4.4 Summary

This chapter presented the results of the intrinsic and extrinsic evaluations. In summary, two models emerged showing strong performance on all evaluated tasks: FastText trained on the lemmatized dataset, and the fine-tuned ELMo model, which was fine-tuned on the dataset version with cased tokens and no stopwords.

Taking a look at the pre-trained instances, BERT was the most performant model on the intrinsic objective, but its scores were distinctly worse on the two extrinsic objectives. GloVe, on the other hand, showed the highest scores of all pre-trained models on these tasks. It was also the only model whose pre-trained version worked consistently better than the self-trained instances, whereas for the other models, in general, the opposite was true.

Finally, considering encoding word sequences by pooling over the individual word embeddings, no systematic trend towards a certain strategy could be determined: there are strong results for different pooling and model combinations without any recognizable pattern.

5 Implementation

Based on the results of the intrinsic and extrinsic evaluation tasks, well-performing models are prototypically integrated into the Service Mate voice assistant. This chapter describes the implementation process. Section 5.1 explains how models are selected based on the evaluation results. The following Section 5.2 is concerned with the integration of the selected models into the Service Mate architecture (Milestone M4). Section 5.3 gives account of the system sequence steps that take place between receiving a natural language user request to outputting an answer. Furthermore, this section describes a mechanism to adaptively learn from user feedback to the system (Milestone M5). Finally, the results of a first small-scale user evaluation are reported in Section 5.4 (Milestone M6).

5.1 Model Selection

In this section, the model selection process for the prototypical implementation is described. A prerequisite is that at least one pre-trained and one self-trained/fine-tuned model are to be selected: pre-trained models are especially important for domains with insufficient data for model training. Self-trained/fine-tuned models, on the other hand, might take advantage of domain-specific semantic properties when there is enough training data.

Models are selected for implementation by a weighted ranking approach: models are ranked in terms of their performance on the three evaluation tasks. In the case of the extrinsic tasks, the selection process considers the strongest pooling strategy for each model instance. It weights the ranks in terms of resemblance with the final use case, that is the categorization of user input to a known class. Of the three conducted evaluations, the intrinsic one is least similar to the final use case. Therefore it is weighted by 1. The extrinsic question pair classification task shows more resemblance with the final task, which is why it is weighted by 2. Finally, the extrinsic question similarity classification task is very close to the final use case. Therefore, performance on this task is weighted by 3.

As the aim is to select a pre-trained and a self-trained/fine-tuned model, these instances are rated separately from each other. Table 5.1 shows the ranked distribution over the different pre-trained model instances and evaluation tasks. Table 5.2 does the same for the self-trained/fine-tuned instances.

Table 5.1. Evaluation ranking of the pre-trained model instances. The last column depicts the overall weighted rank. The best overall rank is underlined.

Model	Evaluation Rank			
	Intrinsic Task	Question Pair Task	Question Similarity Task	**Total**
Word2Vec	5	2	2	**2.50**
GloVe	4	1	1	**1.50**
FastText	3	4	4	**3.83**
ELMo	2	3	3	**2.83**
BERT	1	5	5	**4.33**

Table 5.2. Evaluation ranking of the self-trained/fine-tuned model instances. The last column depicts the overall weighted rank. The best overall rank is underlined.

Model	Evaluation Rank			
	Intrinsic Task	Question Pair Task	Question Similarity Task	**Total**
Word2Vec (Standard)	6	4	5	**4.83**
Word2Vec (Lemmatized)	5	6	3	**4.33**
GloVe (Standard)	7	7	7	**7.00**
GloVe (Lemmatized)	8	8	8	**8.00**
FastText (Standard)	4	3	4	**3.67**
FastText (Lemmatized)	3	2	1	**1.67**
ELMo (Cased, with Stopwords)	1	5	6	**4.83**
ELMo (Cased, without Stopwords)	2	1	2	**1.67**

Based on the weighted ranks, one pre-trained and two self-trained/fine-tuned model instances are selected for implementation: (1) GloVe pre-trained, (2) FastText self-trained on the lemmatized dataset, and (3) ELMo fine-tuned on the cased dataset without stopwords.

5.2 System Architecture

This section describes the architecture of the prototype and its integration into the Service Mate architecture. Figure 5.1 illustrates a high-level overview of the overall architecture: Service Mate is a client-server web application written in Java, whereas the language representation model processes take place on a Python server. The following sections successively describe each module individually. The described modules have either been newly created (Python server) or extended (Service Mate modules) for the sake of the prototypical evaluation.

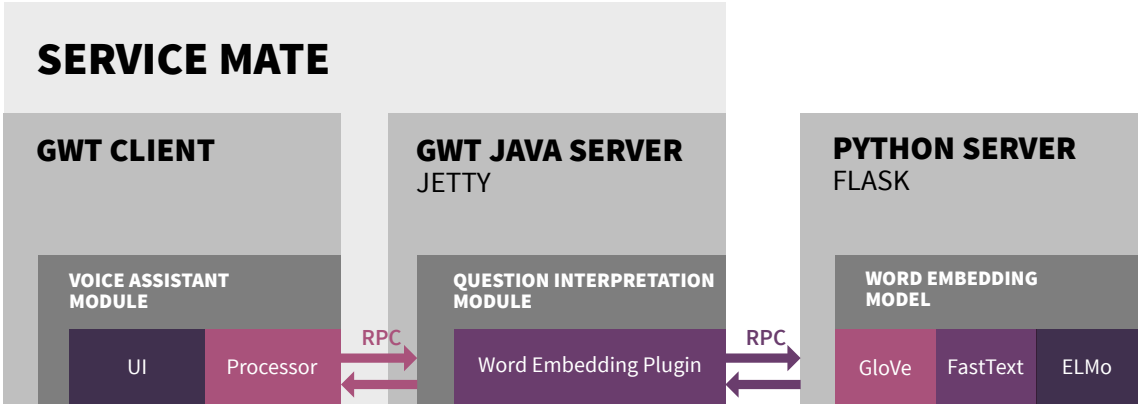


Figure 5.1. Overview of the system architecture.

5.2.1 Service Mate

Service Mate is a Google Web Toolkit (GWT)[1] application. GWT is a toolset for the development of web applications where both client and server code are written in Java. Upon deployment, client-side code is compiled to JavaScript, so that the application is, for example, accessible by a browser. This allows Service Mate to run platform-independently on a multitude of devices. In the following, client and server modules are described which are relevant for the processing of natural language input.

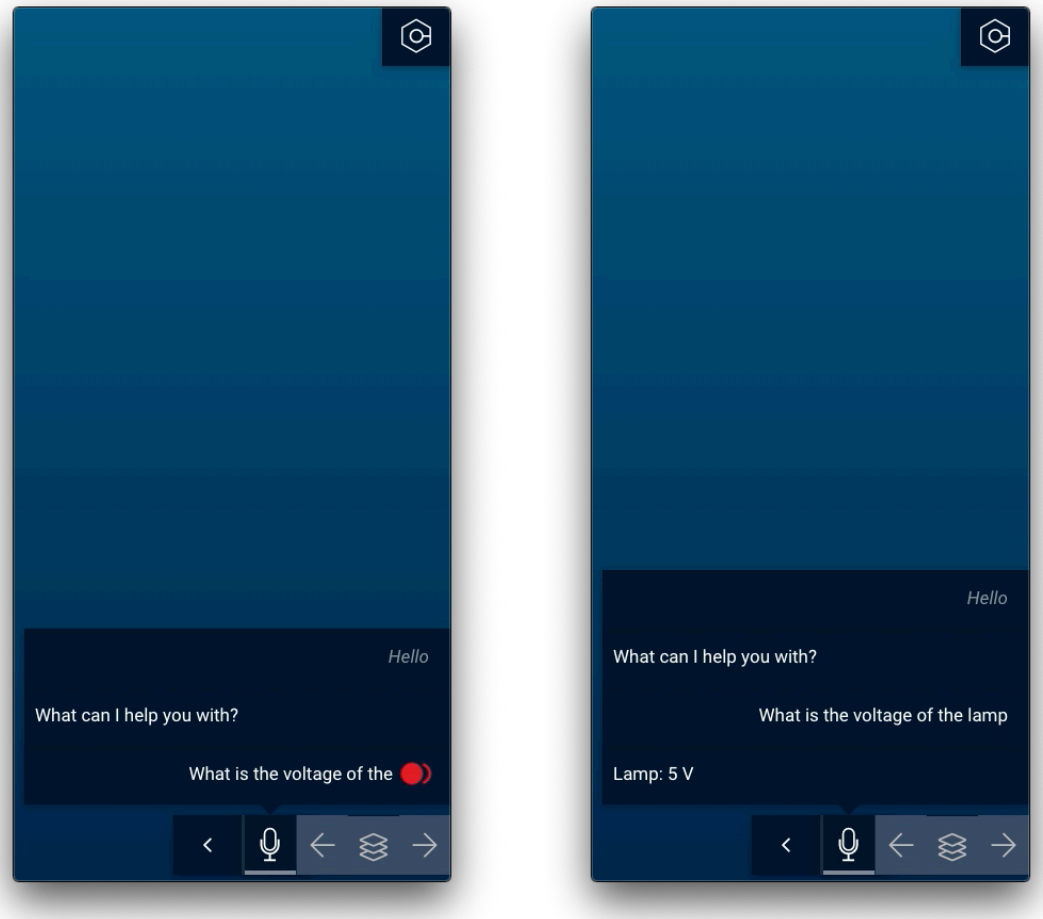
GWT Client

The client module of Service Mate consists of multiple modules handling the interaction with the end user. Here, the focus is put on the *voice assistant module*, which consists of a *User Interface (UI)* and a *processor*.

UI. The UI of the voice assistant has two primary responsibilities: (1) recording user input, and (2) presenting voice recognition results to the user. For recording, the voice assistant is capable of dynamically integrating a speech-to-text engine of the platform Service Mate is currently running on. For example, if the system is running in a browser, Speech Recognition[2] of the Web Speech API is integrated. While recording user input, the interim text transcript is visually presented to the user (Figure 5.2a). After the user has finished their input, the final transcript is forwarded to a processor (see next paragraph). Upon receiving a result to the request, the voice assistant UI is responsible for presenting it to the user (Figure 5.2b). Results can be presented both

[1] http://www.gwtproject.org
[2] https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition

visually as well as auditory. By analogy with speech recording, for auditory result presentation, the voice assistant dynamically integrates the text-to-speech engine of the current platform. For example, in a browser Speech Synthesis[3] of the Web Speech API is utilized.



(a) Input recording. **(b)** Result presentation.

Figure 5.2. UI of the voice assistant. It is responsible for recording and presenting user input (left) as well as the visual and auditory presentation of results (right).

Processor. The processor acts as an intermediary between the UI and the GWT Java server. As described in the previous paragraph, the final transcript of a recorded user request is forwarded to the processor. Subsequently, the processor emits a Remote Procedure Call (RPC) to the server (see next section) containing the user request transcript. The answer of the server consists of a set of actions which are handled by the processor. For example, a *FactAction* contains a fact associated with a machine component such as an electric potential value. The processor handles a FactAction by telling the UI to present the fact to the user as in Figure 5.2b. FactActions are

[3]https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis

a simple example, but the answer of the server might contain more sophisticated actions. For instance, given a *SearchAction*, the processor might initialize a text search by telling the UI to trigger a search event based on the user request.

GWT Java Server

GWT utilizes Eclipse Jetty[4] as a web server. On the server, the *question interpretation module* is responsible for generating actions from a user request text transcript. As described in the previous section, the GWT server receives a text transcript of the user request from the client. The question interpretation module matches the user request with a set of pre-defined patterns as described in Section 3.1.1. Given the category of a matched pattern, this module generates a set of actions. For example, if the text transcript is matched with an electric potential pattern, the module generates a semantic query obtaining the relevant electric potential value from the underlying ontology, and returns it encapsulated in a FactAction.

For the sake of the prototypical implementation, the question interpretation module is extended with a word embedding plugin. This plugin handles the unfolding process of existing patterns by analogy with the approach of the question similarity task described in Section 3.5.2. Furthermore, it is responsible for calling the Python server (see next section) for pattern centroid as well as user request embeddings.

5.2.2 Python Server

A Python Flask server[5] is used for all processes concerned with the language representation models. Python is used for efficiency and compatibility reasons. The main responsibility of the Python server is the generation of word sequence embeddings. Based on the model selection process (Section 5.1), currently three approaches are supported for generating embeddings: GloVe, FastText, and ELMo. Word sequences are embedded using the same approach, as described in Section 3.5.1. However, based on the results of the evaluation tasks, different pooling strategies are applied: GloVe word vectors are max pooled whereas FastText and ELMo word vectors are average pooled and IDF average pooled respectively.

5.3 System Sequence

This section describes the process sequence of the implemented prototype employing the use case example introduced in Section 1.2. The implementation features a three-step process: (1) at

[4] https://www.eclipse.org/jetty/
[5] http://flask.pocoo.org

program start, the pattern centroids are initialized once. Subsequently, whenever a user starts a request, (2) an answer is obtained, and (3) the system adaptively learns from user feedback. The following sections individually describe the sequences of each step.

5.3.1 Pattern Initialization

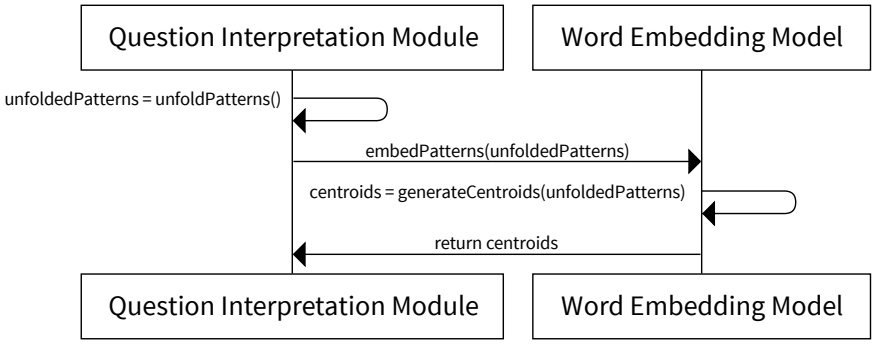


Figure 5.3. Sequence diagram of the generation of pattern centroid embeddings (simplified).

Upon program start, the question interpretation module unfolds the pre-defined patterns as described in Section 3.5.2. As of the time this thesis is written, Service Mate incorporates 58 different pattern categories. For the prototypical implementation, a subset of 12 categories is picked. These categories originate from the fact domain, thus incorporating patterns about physical fact values such as electric potential, pressure, length or weight values. This step results in a list of unfolded patterns for each category. After that, the word embedding model is called to generate centroid embeddings for each pattern category. Pattern centroid embeddings are generated as described in Section 3.5.2 by averaging the embeddings of each individual pattern for a given category. The initialization sequence is illustrated in Figure 5.3.

5.3.2 Obtaining an Answer for a Natural Language User Request

In the example use case described in Section 1.2, a user requests the electric potential value of a lamp by asking: *"What electric potential does the lamp have?"*. Figure 5.4 illustrates the sequence of processes for obtaining an answer to this request in the prototypical implementation. In the beginning, the recorded voice transcript is forwarded via the processor to the question interpretation module. As described above, the question interpretation module tries to match a pattern to the given request. If the matching process is successful, a semantic query obtains the electric potential value for the given component (lamp) from the underlying ontology. The inferred answer (5 Volt) is encapsulated in a FactAction object which in turn is put into a singleton list and returned to the client-side processor. The processor extracts the answer from the FactAction and

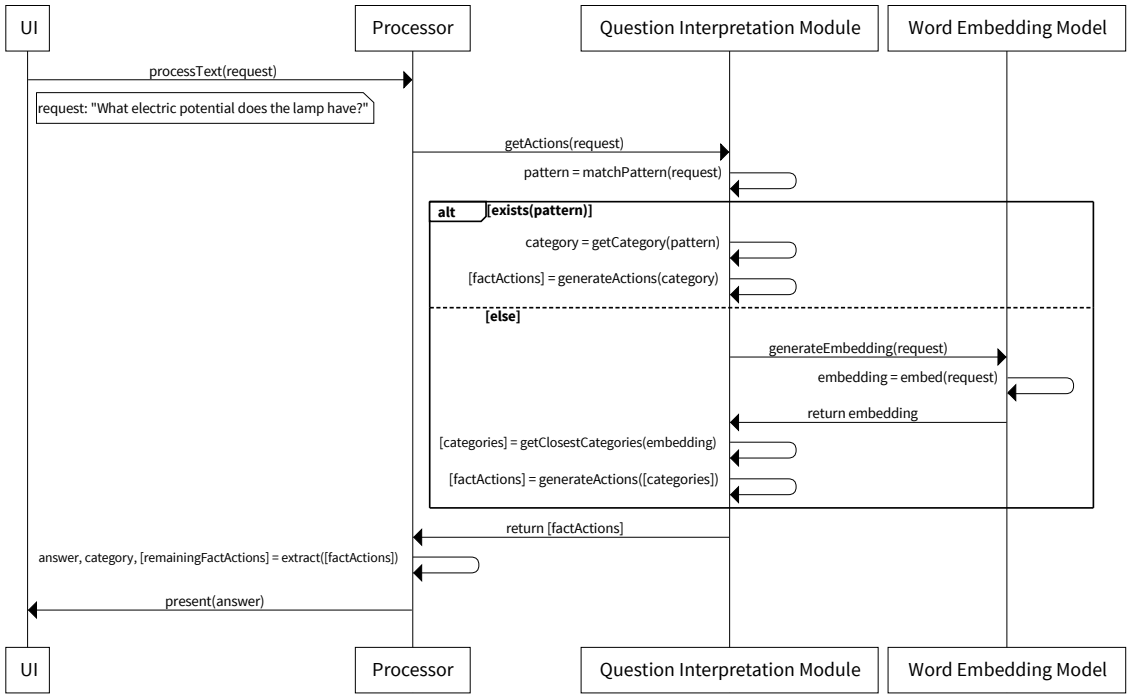


Figure 5.4. Sequence diagram for obtaining an answer (simplified).

calls the UI to present the answer to the user. So far, the process sequence does not diverge from the traditional voice assistant.

However, if no matching pattern can be found, that is the case when the word embeddings become relevant: in this case, the quint interpretation module first tries to extract a component – in our case "lamp" – from the user request by matching word-level n-grams of the request against a component list. On a side note, if no component can be extracted, the module returns an empty result to the client. Otherwise, subsequently, the word embedding model is called to generate an embedding for the user request. The returned embedding is compared to the previously initialized pattern centroid embeddings. Based on the categories which are defined for the extracted component, a ranked category list is generated with respect to the cosine similarity between the categories' pattern centroid embeddings and the request embedding. For instance, the component lamp has definitions for the categories of electric potential, current, and resistance. Thus, the resulting category ranking contains these three categories. For each ranked category, the corresponding fact value is queried from the underlying ontology and encapsulated in a FactAction. The FactActions are put into a list and returned to the client-side processor which takes the best-ranked answer and returns it to the user.

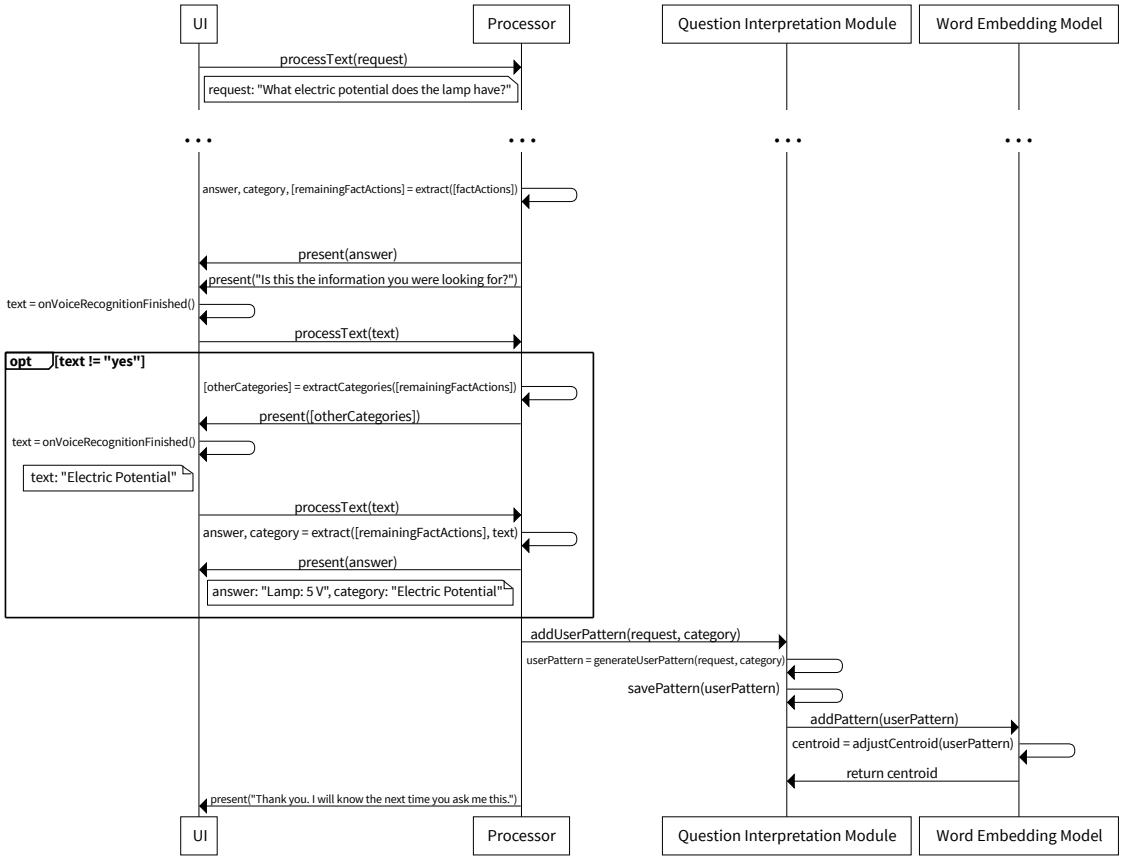


Figure 5.5. Sequence diagram of the feedback mechanism (simplified).

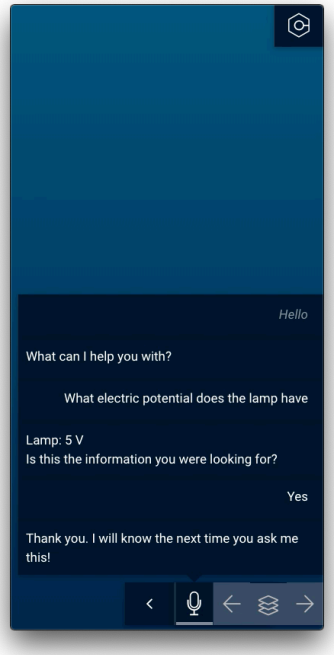
5.3.3 Adapting from User Feedback

It might well be that the best ranked answer has been misclassified. For instance, the previous step might have ranked an electric current value best even though the user was asking for an electric potential value. For this purpose, a user feedback mechanism is integrated to adapt the model to user requests dynamically. The sequence of the feedback mechanism is illustrated in Figure 5.5.

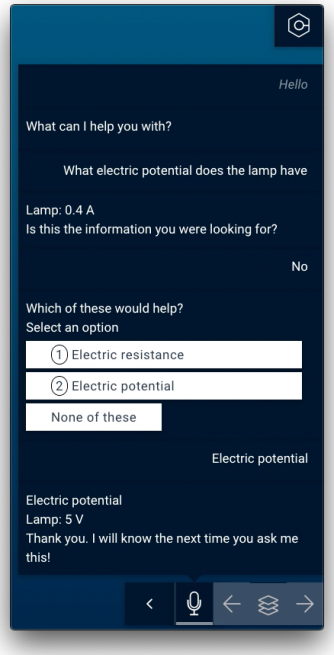
If the list of returned FactActions is larger than one, the user is asked if the presented answer contains the information they were looking for. First, let us assume this is the case, i.e., the answer was "5 Volt". The user may confirm (Figure 5.6a), and the question interpretation module is called to add the request as a new pattern for the given category: first, the user-generated pattern is modified by replacing the component expression ("lamp") with a wildcard symbol so that future requests with arbitrary components will be matched: *"What electric potential does the $1 have?"*. Subsequently, it is persisted to the user's folder so that the next time this user expresses the same request, the pattern will be matched (Figure 5.7). Finally, the word embedding model is called to integrate the user's pattern into the electric potential centroid embedding. This step

shifts the centroid embedding towards the embedding of the request, dynamically adapting the category embedding towards the way the user phrases their requests.

On the other hand, let us now assume the user was not presented the information they were looking for. For instance, the user was presented with an answer saying "0.4 Ampere". The user may deny that this is the correct information. In this case, they are presented with the next best-ranked categories and asked to select the correct one. Hence, the user might select the category electric potential. Then, the correct answer ("5 Volt") is presented (Figure 5.6b), and the request is added as a new pattern the same way as described above.



(a) Confirming an answer.



(b) Correcting an answer.

Figure 5.6. UI of the feedback mechanism. The user may either confirm an answer (left) or correct a misclassification (right).

5.4 Case Study

This section reports the results of a first small-scale user study as well as measurements of time and resource consumption of the three different models.

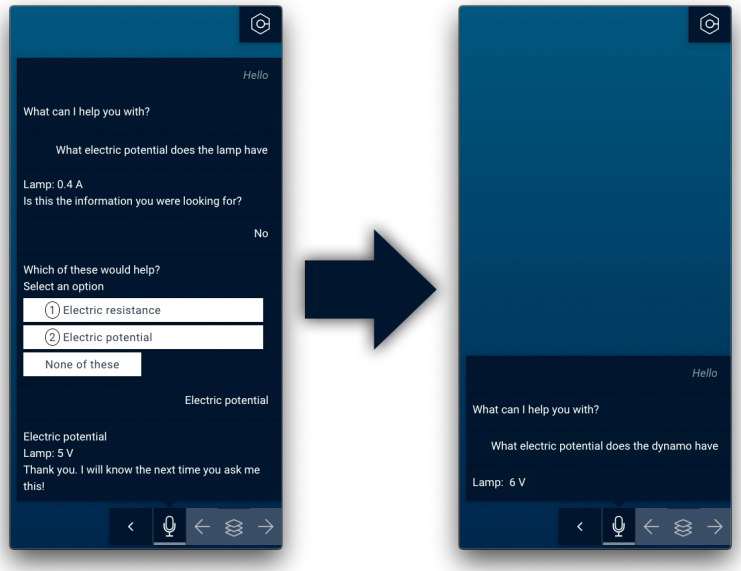


Figure 5.7. The voice assistant dynamically adapts to the way the user expresses their input: on the left the user corrects a misclassified request. If the same request is phrased again, the assistant outputs the correct answer (right).

5.4.1 User Study

As introduced in Section 1.1, voice assistants typically consist of a two-stage structure: (1) natural language input sound is transformed into text by a speech-to-text engine before (2) the transcribed text is further processed to analyze and interpret its semantic content. Even though this thesis is concerned with the improvement of the second stage, a robust language representation model might compensate for errors occurring at the first stage as well. In other words, utilizing a robust model might provide a user with the intended information even though the text has not been transcribed correctly. In order to assess the robustness of the three implemented models, a first user study is conducted.

Participants

A total of six persons (3 female, 2 male, 1 did not wish to disclose) with an average age of 23.5 years (SD = 2.26 years) participated. On average participants assess their English skill level with 4.83 (SD = 0.41) on a scale from 1 (= no skills) to 7 (= native). Furthermore, they rate their experience with voice assistants on average with 4.33 (SD = 1.75) on a scale from 1 (= no experience) to 7 (= a lot of experience). At the time of the study, five participants were students and one was a software quality specialist.

Table 5.3. Questions read by the participants. Each question was read three times by each participant.

Category	Question
Electric Current	What is the rated amperage of the lamp?
	What is the strength of electric current of the lamp?
	What current runs through the dynamo?
	What should be the measured amperage of the dynamo?
Electric Potential	What voltage is measured in the lamp?
	How many volts does the lamp need?
	What should be the electric potential of the dynamo?
	What voltage is present at the dynamo?
Electric Resistance	What is the ohmage of the lamp?
	What is the resistance of the lamp in ohm?
	What should be the electrical resistance for the dynamo?
	What is the value of ohmage of the dynamo?
Pressure	How much pressure has to be in the front wheel?
	Tell me how many bars does the front wheel need?
	How many bar pressure are in the rear wheel?
	How much pressure does the rear wheel hold?

Study Design

Participants are asked to read a set of pre-defined questions to a speech-to-text engine. The questions are taken from the human-generated question dataset introduced in Section 3.5.1. For each question category, four items are sampled to be read to the engine. In the study, a bike is used as machine representative. A reason for this is not to disclose any real customer data. Thus, the participants are supposed to ask for different fact values concerning components from the bike domain, such as lamp, dynamo or wheel. The questions are depicted in Table 5.3. Participants are asked to read each sentence three times. During reading the live text transcript is presented to the participants. Speech Recognition[6] of the Google Chrome[7] Web Speech API is utilized as speech-to-text engine.

After question transcriptions have been collected from each participant, they are fed into the prototypical Service Mate implementation. For each model (GloVe, FastText, and ELMo) the accuracy of the question transcript classifications is measured. Two different classification methods are conducted for each model: (1) the first method keeps pattern centroid embeddings fixed to the initial state, and (2) the second one adapts pattern centroid embeddings by giving feedback with the correct category after each classification as described in Section 5.3.3.

Procedure

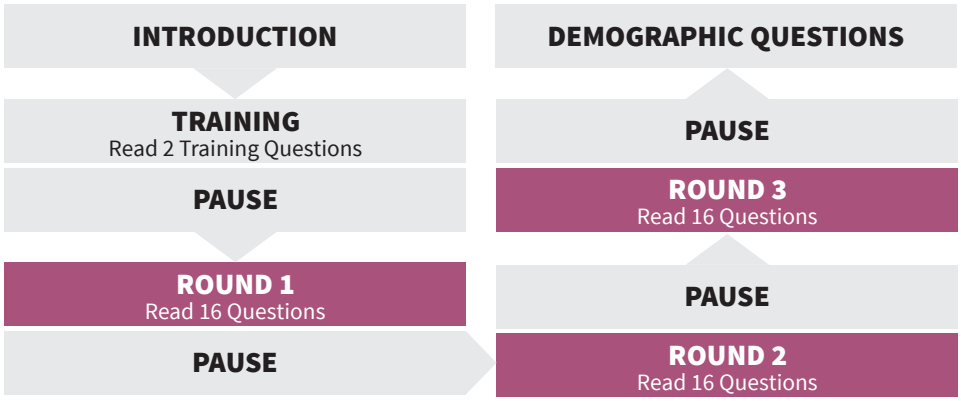


Figure 5.8. Procedure of the user study.

Figure 5.8 illustrates the procedure of the user study. After an introduction, participants are asked to read two training questions. Then, they are asked to read the sampled questions three times. The study concludes with demographic questions.

As described above, after question transcripts have been collected from all participants, they are shuffled and classified by the prototypical Service Mate voice assistant implementation.

[6] https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition
[7] https://www.google.de/chrome

Results

Table 5.4. Classification accuracy with respect to the implemented models. Bold values indicate the best result for the respective condition.

Condition	GloVe	FastText	ELMo
Fixed Centroid Embeddings	0.6465	0.7071	**0.7273**
Adaptive Centroid Embeddings	0.7980	0.7778	**0.8586**

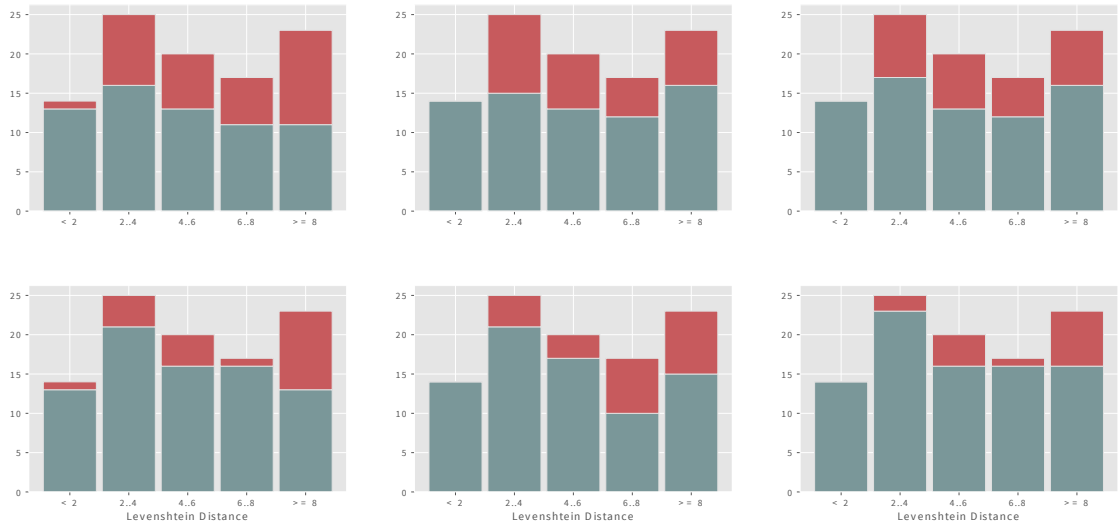
In total, 99 different user transcripts have been collected for the 16 questions – identical transcripts have been discarded. Table 5.5 depicts four example transcripts, each paired with the associated original question and the Levenshtein distance (Levenshtein, 1966) between the two.

The classification accuracy with respect to the three models is presented in Table 5.4. ELMo achieves the highest accuracy scores for both fixed and adaptive pattern centroid embeddings. In the fixed condition FastText closely follows ELMo on the second rank and GloVe ranks third by a more considerable margin. ELMo clearly outperforms the other models in the adaptive condition where GloVe ranks second closely followed by FastText. All models achieve distinctly higher scores in the second condition when centroid embeddings are adapted to the user's input questions.

Table 5.5. Example question transcripts generated by the speech-to-text engine. The last column indicates the case-independent Levenshtein distance between the original sentence and the transcript. Note that the question mark is excluded from distance calculation.

User Transcript	Original Question	Distance
How many volts does the lamp need	How many volts does the lamp need?	0
What is the value of omega of the dynamo	What is the value of ohmage of the dynamo?	3
What does the strength of electric urine off the lamp	What is the strength of electric current of the lamp?	8
What's a beaver macedon parrot of the dynamo	What should be the measured amperage of the dynamo?	21

Figure 5.9 illustrates the performance of the models with respect to the Levenshtein distances between the classified questions and the original ones. Note that the models classify almost all questions correctly when the distance is low (< 2). For higher distances, classification accu-



(a) GloVe. **(b)** FastText. **(c)** ELMo.

Figure 5.9. Classification result with respect to the Levenshtein distance for fixed (top) and adaptive (bottom) pattern centroid embeddings. Green bars indicate correctly classified and red bars incorrectly classified items.

racy becomes worse. Comparing the two conditions, it becomes apparent that adapting centroid embeddings can compensate especially for low/medium distances, but compensation becomes worse for large distances. In other words, if the discrepancy between the transcript and the original question is too large, classification cannot be improved by adapting centroid embeddings.

5.4.2 Resource Evaluation

Table 5.6. Time resource consumption. As the embedding of a sentence is dependent on the length, this table reports the average time the model takes to embed a word in the sentence.

Process	GloVe	FastText	ELMo
Model Initialization	173.72 s	22.37 s	**10.13 s**
Embedding All Pattern Centroids	70.38 s	**3.94 s**	72.02 s
Embedding Sentence (Time per Word)	0.026 s	**0.002 s**	0.027 s

Table 5.7. Space resource consumption.

GloVe	FastText	ELMo
2178 MB	707 MB	**374 MB**

Another critical factor towards the real-world application of such models is the consumption of resources. The measurements reported here have been conducted on a 2017 MacBook Pro with a 3.3 GHz Intel Core i5 processor, 16 GB RAM, and no dedicated GPU.

Table 5.6 depicts the consumption of time resources, whereas Table 5.7 indicates the space needed by the models. The pre-trained GloVe model needs the most resources both in terms of time and space. The self-trained FastText model is the one that generates embeddings fastest, on average more than ten times faster than the other models. ELMo, on the other hand, consumes the least space, which is why it takes the least amount of time to initialize.

5.5 Summary

This section described the prototypical integration of three language representation models into the Service Mate architecture. Section 5.1 was concerned with the selection of models for integration. The selection process ranked models with respect to their performance on the evaluation tasks. Ranking was weighted depending on the tasks' resemblance with the final use case. Three models emerged for integration: (1) GloVe pre-trained, (2) FastText self-trained on the lemmatized dataset, and (3) ELMo fine-tuned on the cased dataset without stopwords.

Next, Section 5.2 described the architecture of the prototype, and Section 5.3 showed the process sequence of obtaining an answer to a user request. Service Mate features a client-server architecture. The client is concerned with handling user interaction, whereas the server contains a question interpretation module. This module is responsible for inferring actions from user requests. A newly created word embedding plugin to the question interpretation module handles the communication with a Python server that generates word sequence embeddings. Finally, a new interaction mechanism allows the system to learn and adapt from user feedback dynamically.

Finally, Section 5.4 presented a first small-scale user study. Participants were asked to read predefined questions into a speech-to-text engine. The prototype classified the resulting transcripts. ELMo achieved the highest accuracy scores among the three model instances. In terms of resource requirements, FastText embeds sentences fastest, whereas ELMo consumes the fewest space.

6 Discussion

The overall objective of this work was to enhance the robustness of voice assistants in the Technical Service domain. Robustness should be improved in terms of semantically classifiable yet syntactically varying user input. The use case presented in Section 1.2 introduced a concrete example of how syntactically different yet semantically equal user input might look like. Different milestones have been identified to approach and achieve this objective (see Section 1.3). The approach, results, and limitations will be discussed in the following.

6.1 Language Representation Models

Approaching Milestone M1, research into related work revealed that most state-of-the-art approaches to semantic language representation map words or word sequences onto multi-dimensional vector spaces (Turian et al., 2010). Such vector representations are typically generated by optimizing on predicting the likelihood of characters, words, or word sequences occurring in a given context. This technique is also known as language modeling (Goodman, 2001). Language models currently top benchmarks on a multitude of NLP tasks such as question answering (Devlin et al., 2018) or sentiment analysis (Peters et al., 2018). However, these tasks are very general and not tied to a specific domain.

Thus, this thesis examined the performance of five techniques currently dominating the field of NLP on a narrower task domain. To this end, five models, Word2Vec, GloVe, FastText, ELMo, and BERT, have been selected. In order to get comparable results, the scope of the intrinsic evaluation limited the model selection process to models capable of generating embeddings for individual words or words in context. However, it would be interesting to also gain insights into the performances of sentence embedding models as introduced in Section 2.4.1. As semantic information systems often operate in multiple languages, approaches, which are designed for multilingual representation, like LASER (Artetxe & Schwenk, 2018), are particularly interesting in that context.

The selected models have been adjusted to the technical documentation of harvesting machines as described in Milestone M2. Furthermore, pre-trained model instances have been obtained in order to get an intuition to what extent self-training and fine-tuning benefit model performance. Due to computational constraints, however, no fine-tuned instance of the BERT model could be

created. The current advances in hardware technology might render this operation feasible in the near future though (Dean, Patterson, & Young, 2018).

6.2 Evaluation and Results

As described in Milestone M3, the selected models have been evaluated on an intrinsic and two extrinsic tasks. The evaluation results stated throughout Chapter 4 indicate that the selected models can indeed perform reasonably well in the rather narrow Technical Service domain. Summarizing the results over all three evaluation tasks shows a trend towards models self-trained/fine-tuned on a domain-specific dataset yielding better results than utilizing instances that have been pre-trained on general-domain datasets. For the most part, self-trained and fine-tuned model instances outperformed their pre-trained instances both in the intrinsic as well as the two extrinsic evaluation tasks. However, GloVe constitutes a notable exception to this trend: for each evaluation task, the pre-trained GloVe model surpassed the self-trained instances. Simultaneously, it is the only one of the five evaluated techniques which does not generate word embeddings by optimizing a neural architecture on a language modeling task. Instead, GloVe is a count-based model taking advantage of global co-occurrence statistics. It is imaginable that the dataset used for self-training, consisting of about nine million tokens, is too small for this global technique to generate high-quality embeddings. The pre-trained GloVe embeddings have been trained on datasets containing at least six billion tokens; the pre-trained version utilized in this work has been trained on 840 billion tokens – orders of magnitudes more than the dataset applied for self-training. In contrast to the global GloVe model, Word2Vec and FastText are local predictive models. Looking at the results of the experiments, the self-trained instances of these models were capable of achieving stronger results than their pre-trained counterparts. Therefore, one can conclude that such local predictive models are capable of generating useful domain-specific embeddings given a comparatively small dataset.

In this context, another surprising result is worth noting: even though ranking best in the intrinsic evaluation, the pre-trained BERT model showed weak performance in the two extrinsic tasks. This result contradicts the results on general-domain NLP benchmarks, where BERT advanced the current state-of-the-art for eleven tasks (Devlin et al., 2018). Drawing a parallel to the conceptually similar ELMo model, one can assume that fine-tuning BERT might help with achieving better results in the extrinsic tasks: fine-tuning improved ELMo's performance particularly on the second extrinsic task where both pre-trained ELMo and BERT instances operated rather poorly. However, further experiments are needed to confirm this. Furthermore, as described above, fine-tuning BERT is computationally expensive, but it is likely to become more affordable in the future.

Certain limitations have to be addressed considering the evaluation procedure. First, the extrinsic evaluation has only been carried out on four different fact categories. A real-world application, however, might be concerned with a lot more categories. Constraining the findings of this evaluation to the fact domain, this should not be a problem though: single machine components are rarely ever associated with more than four different fact values. Thus, the difficulty of a real-world task constrained to the categorization of only facts remains reasonably the same. However, one might want to extend the categorization to classes other than the fact domain. For example, as mentioned in Section 5.3.1, Service Mate currently features 58 different pattern categories. These include, for instance, patterns that match requests asking for the location of a component in a 3D model, or the working principle of a particular item. It has yet to be discovered how well a model will work if more and more of these categories are included.

Another limitation is that one might question the generalizability of the results of the self-trained and fine-tuned models. Within the scope of this study, they have been trained on the technical documentation of harvesting machines. However, the results of the extrinsic evaluation tasks are actually generalizable to other machine domains as well: specific terms or component names relating to harvesting machines have not been used in the course of the extrinsic experiments. On the contrary, the examples that have been used are transferable to the vast majority of machine types: fact categories like electric potential or barometric pressure apply to almost any machine domain. Therefore, due to the machine-agnostic character of the extrinsic evaluations, the model instances will show equal performance with other machine types as well. Thus, the results of the models trained on the harvesting machine dataset are transferable beyond this specific machine type. However, it remains to be examined if models trained or fine-tuned on different data of other machines achieve similar results on these evaluation tasks. Further studies and data from different machine types and domains are needed to assess this aspect.

6.3 Implementation and Self-Learning Mechanism

Milestone M4 determined the integration of well-performing models into the Service Mate voice assistant. In total, three models were selected for implementation: a fine-tuned ELMo instance, a self-trained FastText model, and a pre-trained GloVe model. The implementation serves as a research prototype demonstrating the strengths of utilizing language representation models for the Technical Service domain. By extending the traditional pattern-based voice assistant, a hybrid platform was created that has not lost any of its former functionality. On the contrary, the prototype is now capable of classifying natural user input requests that did not return satisfactory results before. The ability to give feedback to the system (Milestone M5), thereby triggering a self-learning process is another strength featured in the research prototype. As the results reported in Table 5.4 indicate, this mechanism seems to improve the recognition quality distinctively. The

improvement showed for each of the three implemented models. However, a larger-scale study with more transcripts is endorsed to further support this trend. Since the self-learning system is tied to the individual user, it allows the system to adapt to the very way the user paraphrases their requests. Thus, a user may personalize the system to their own preferences.

As outlined in Milestone M6, a first case study was conducted. It showed that the implemented FastText and ELMo models were capable of classifying all requests correctly which had been correctly transcribed by the speech-to-text engine. Pre-trained GloVe classified 13 of 14 transcriptions correctly. To a certain extent, the models were even able to compensate for failures occurring at speech-to-text transcription time (Figure 5.9). Even though the self-trained FastText and fine-tuned ELMo instances achieved better results in the small-scale case study (Section 5.4.1), pre-trained GloVe performed decently as well. This is a crucial factor to consider in the light of limited data domains, where training a model from scratch is not feasible. On the other hand, as described above, the extrinsic evaluations were conducted without respect to a specific machine. Therefore, it is even imaginable to transfer the self-trained FastText or fine-tuned ELMo instances to other machine domains as well.

Apart from the strengths, the implementation also comes with some limitations, which will be discussed in the following. First, the current implementation is not immune against adversarial examples: if the system cannot match a given user request to a pattern, but at the same time can extract a component name, the question interpretation module will always return an answer. This answer might not necessarily make sense though: the system will classify a category whose centroid embedding emerges most similar. For example, imagine a nonsense user input containing the word “dynamo” like the incorrectly transcribed request “What’s a beaver macedon parrot of the dynamo” depicted in Table 5.5: even though this request does not make sense, the system would return the value of a category associated with the component “dynamo” whichever computes the highest cosine similarity with the input embedding. Thus, further work is needed to mitigate the impact of adversarial examples. Different approaches are imaginable. First, one could integrate thresholds concerning the cosine similarity scores between centroid and input embeddings and reject classifications scoring below. Another idea would be to generate a centroid embedding based on adversarial examples: the system would not return an answer if the adversarial centroid emerged most similar with the input embedding.

Moreover, in the current prototype, component names are extracted from user requests by naïve word-level n-gram matching. However, this approach is constrained to the exact wording of components. For example, imagine a user asking for a fact value of the component “light” instead of “lamp”: the current prototype would not be able to parse the component correctly, and, therefore, could not give an appropriate answer. Thus, a valuable extension would be the integration of a named entity recognizer classifying which word, or word sequence, contains a

component. Current state-of-the-art named entity recognition is conducted on neural architectures (Lample, Ballesteros, Subramanian, Kawakami, & Dyer, 2016), which in turn would require a sufficiently large labeled dataset.

Finally, as described in Section 5.4.1, the comparatively high resource demands limit the practical applicability of the language representation models. Particularly the practical feasibility of the pre-trained GloVe model is questionable as it occupies more than 2 GB of space, and, in the experiments of the prototypical implementation, needed approximately four minutes for the model to initialize and embed pattern centroids. Yet, the embedding of a single user request transcript executes reasonably fast (Table 5.6) so that after the initial loading process the system is efficiently usable. In this context, Joulin et al., 2016 suggested techniques to lower the memory demands of word embeddings using the example of FastText. Alvarez, Prabhavalkar, and Bakhtin, 2016 presented a technique to represent and execute deep acoustic models efficiently. This approach might also be applicable to deep language models, such as ELMo, thereby reducing time and space constraints of the model.

7 Conclusion

This thesis showed the usefulness of state-of-the-art language representation models in the Technical Service context. Such models enable the mapping of words or word sequences onto multi-dimensional vector spaces. Towards the goal of this thesis, namely improving the robustness of a voice assistant in the Technical Service domain, the performances of five different models, Word2Vec, GloVe, FastText, ELMo, and BERT, have been evaluated.

Three different evaluations were conducted, one of which was an intrinsic task, and two were extrinsic tasks. The intrinsic task measured the correlation between human-annotated and model generated similarity ratings of words in the context of technical documentation. The extrinsic tasks measured model performance on downstream NLP tasks resembling the operating principle of a voice assistant in the Technical Service domain: one task evaluated how well a neural classifier was able to correctly label whether question pairs originated from the same physical domain or not given a specific model (binary classification). The other task assessed the classification accuracy of the questions to a physical category given a model (multiclass classification).

The evaluation results indicate that, in general, self-trained and fine-tuned model instances perform better than pre-trained ones. The relatively small training dataset, the technical documentation of harvesting machines, was sufficient for generating semantically meaningful word and sentence vector embeddings performing well on the evaluation tasks. However, this trend did not show for the GloVe model whose pre-trained model outperformed the self-trained instances.

Based on the evaluation results, three models were selected for a prototypical implementation into a voice assistant of a semantic information system for the Technical Service domain: a pre-trained GloVe model, a self-trained FastText model, and a fine-tuned ELMo model. The implemented prototype classified user input to the voice assistant by comparing a model generated input embedding to pre-calculated centroid embeddings representing potential user intents. A first case study indicated the advantages of the prototypical implementation over the traditional voice assistant, which classified user input based on pattern matching. Furthermore, a self-learning mechanism was integrated to further enhance the robustness of the system and compensate for shortcomings in the embeddings: this mechanism allowed the system to adaptively learn from user feedback. It furthermore granted users the ability to adjust the voice assistant to their personal interaction style.

There are some promising directions to further develop this work, and especially the implementation, in the future. First, the quality of pattern centroid embeddings could be improved by setting up a more extensive and diverse set of pre-defined patterns. Furthermore, classification could be conducted using a neural classifier instead of a naïve comparison of cosine similarity scores. Similarly to the question pair classification task described in Section 3.5.1, such a neural classifier could take a sentence embedding (or individual word embeddings) as input and output a category class. However, a large, labeled dataset mapping from user input examples to categories would be required to train an accurate neural classifier.

Finally, another possibility to improve the approach is enriching the word embeddings with structural knowledge. As described in Section 2.1, semantic information systems typically take advantage of structured knowledge stored in ontologies. A technique called *retrofitting* could be applied to combine word embeddings with knowledge stored in the associated ontology. For example, Speer and Chin, 2016 used retrofitting to improve GloVe and Word2Vec embeddings by combining them with structured knowledge from ConceptNet (Speer & Havasi, 2012). In a similar vein, the word embeddings generated in the course of this work could be enriched with knowledge from an underlying machine ontology.

References

Abecker, A. & van Elst, L. (2004). Ontologies for knowledge management. In *Handbook on ontologies* (pp. 435–454). Springer.

Agirre, E., Alfonseca, E., Hall, K. B., Kravalova, J., Pasca, M., & Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA* (pp. 19–27).

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., … Wiebe, J. (2014). Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (semeval 2014)* (pp. 81–91).

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., …, Mihalcea, R., et al. (2015). Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (semeval 2015)* (pp. 252–263).

Agirre, E., Banea, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Mihalcea, R., … Wiebe, J. (2016). Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)* (pp. 497–511).

Agirre, E., Cer, D. M., Diab, M. T., & Gonzalez-Agirre, A. (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th international workshop on semantic evaluation, SemEval@NAACL-HLT 2012, Montréal, Canada, June 7-8, 2012* (pp. 385–393).

Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., & Guo, W. (2013). * SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity* (Vol. 1, pp. 32–43).

Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2001). Toward conversational human-computer interaction. *AI magazine*, *22*(4), 27–27.

Almuhareb, A. & Poesio, M. (2006). MSDA: wordsense discrimination using context vectors and attributes. In *ECAI 2006, 17th european conference on artificial intelligence, august 29 - september 1, 2006, riva del garda, italy, including prestigious applications of intelligent systems (PAIS 2006), proceedings* (pp. 543–547).

Alvarez, R., Prabhavalkar, R., & Bakhtin, A. (2016). On the efficient representation and execution of deep acoustic models. *arXiv preprint arXiv:1607.04683*.

Arora, S., Liang, Y., & Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*.

Artetxe, M. & Schwenk, H. (2018). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR, abs/1409.0473*. arXiv: 1409.0473

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)* (Vol. 1, pp. 238–247).

Baroni, M., Evert, S., & Lenci, A. (2008). Bridging the gap between semantic theory and computational simulations: Proceedings of the esslli workshop on distributional lexical semantics. *Hamburg, Germany: FOLLI*.

Baumeister, J. (2016). Requirements of affective information systems in the industrial domain. In Nalepa (Ed.), *Proceedings of affective computing and context awareness in ambient intelligence*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Berg, M. M., Isard, A., & Moore, J. D. (2013). An openccg-based approach to question generation from concepts. In *International conference on application of natural language to information systems* (pp. 38–52). Springer.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Bruni, E., Tran, N.-K., & Baroni, M. (2014). Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, *49*, 1–47.

Bundesministerium für Wirtschaft und Energie. (2018). Maschinen- und Anlagenbau. https://www.bmwi.de/Redaktion/DE/Artikel/Branchenfokus/Industrie/branchenfokus-maschinen-und-anlagenbau.html. Accessed: 2018-11-01.

Canalys. (2018). Smart speakers are the fastest-growing consumer tech; shipments to surpass 50 million in 2018. https://www.canalys.com/newsroom/smart-speakers-are-fastest-growing-consumer-tech-shipments-surpass-50-million-2018. Accessed: 2019-02-25.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., & Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the*

11th international workshop on semantic evaluation (semeval-2017) (pp. 1–14). Vancouver, Canada: Association for Computational Linguistics.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., ..., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Chaubard, F., Fang, M., Genthial, G., Mundra, R., & Socher, R. (2017). Lecture notes in natural language processing with deep learning. http://web.stanford.edu/class/cs224n/archive/WWW_1617/lecture_notes/cs224n-2017-notes1.pdf. Accessed: 2018-07-30. Stanford University.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 551–561).

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of ssst-8, eighth workshop on syntax, semantics and structure in statistical translation* (pp. 103–111).

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (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)* (pp. 1724–1734).

Chomsky, N. (Ed.). (1957). *Syntactic structures*. The Hague: Mouton & Co.

Collobert, R. & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(Aug), 2493–2537.

Conneau, A. & Kiela, D. (2018). Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the eleventh international conference on language resources and evaluation (lrec-2018)*.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Dai, A. M. & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems* (pp. 3079–3087).

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th international conference on machine learning-volume 70* (pp. 933–941). JMLR. org.

Dean, J., Patterson, D., & Young, C. (2018). A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro*, *38*(2), 21–29.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391–407.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*.

Dolan, B., Quirk, C., & Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on computational linguistics* (p. 350). Association for Computational Linguistics.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning* (pp. 647–655).

Faruqui, M., Tsvetkov, Y., Rastogi, P., & Dyer, C. (2016). Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on information systems*, *20*(1), 116–131.

Firth, J. R. (1957). A synopsis of linguistic theory 1930-55. *1952-59*, 1–32.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., … Zettlemoyer, L. (2018). Allennlp: A deep semantic natural language processing platform. *CoRR*, *abs/1803.07640*. arXiv: 1803.07640

Gnewuch, U., Morana, S., & Mädche, A. (2017). Towards designing cooperative and social conversational agents for customer service. In *Proceedings of the international conference on information systems - transforming society with digital innovation, ICIS 2017, seoul, south korea, december 10-13, 2017*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech & Language*, *15*(4), 403–434.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the eleventh international conference on language resources and evaluation (lrec-2018)*.

Guha, R., McCool, R., & Miller, E. (2003). Semantic search. In *Proceedings of the 12th international conference on world wide web* (pp. 700–709). ACM.

Hill, F., Cho, K., & Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. In *NAACL HLT 2016, the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies, san diego california, usa, june 12-17, 2016* (pp. 1367–1377).

Hill, F., Reichart, R., & Korhonen, A. (2014). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR, abs/1408.3456*. arXiv: 1408.3456

Hochreiter, S. & Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. In *Advances in neural information processing systems* (pp. 473–479).

Howard, J. & Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (Vol. 1, pp. 328–339).

Hu, M. & Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining* (pp. 168–177). ACM.

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Annual meeting of the association for computational linguistics (acl)*.

Irsoy, O. & Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 720–728).

Iyer, S., Dandekar, N., & Kornél, C. (2017). First quora dataset release: Question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs. Accessed: 2018-12-13.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Jurafsky, D. & Martin, J. H. (2014). *Speech and language processing*. Pearson London.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kenter, T., Borisov, A., & de Rijke, M. (2016). Siamese CBOW: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR, abs/1408.5882*. arXiv: 1408.5882

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems* (pp. 3294–3302).

Kneser, R. & Ney, H. (1995). Improved backing-off for m-gram language modeling. In *1995 international conference on acoustics, speech, and signal processing, ICASSP '95, detroit, michigan, usa, may 08-12, 1995* (pp. 181–184).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Landauer, T. K. & Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, *104*(2), 211.

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

LeCun, Y., Bengio, Y. et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, *8*, pp. 707–710).

Levy, J., Bullinaria, J., & McCormick, S. (2017). Semantic vector evaluation and human performance on a new vocabulary MCQ test. In *Proceedings of the 39th annual meeting of the cognitive science society, cogsci 2017, london, uk, 16-29 july 2017*.

Levy, O. & Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning* (pp. 171–180).

Liddy, E. D. (2001). Natural language processing. In *Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc.*

Lin, T., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: common objects in context. In *Computer vision - ECCV 2014 - 13th european conference, zurich, switzerland, september 6-12, 2014, proceedings, part V* (pp. 740–755).

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *CoRR*, *abs/1801.10198*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Lund, K. & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, *28*(2), 203–208.

Luong, T., Socher, R., & Manning, C. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the seventeenth conference on computational natural language learning* (pp. 104–113).

Maaten, L. v. d. & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Manandhar, S., Klapaftis, I. P., Dligach, D., & Pradhan, S. S. (2010). Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation* (pp. 63–68). Association for Computational Linguistics.

Manaris, B. (1998). Natural language processing: A human-computer interaction perspective. In *Advances in computers* (Vol. 47, pp. 1–66). Elsevier.

Markov, A. A. (1913). Essai d'une recherche statistique sur le texte du roman "Eugene Onegin" illustrant la liaison des epreuve en chain ('Example of a statistical investigation of the text of "Eugene Onegin"illustrating the dependence between samples in chain'). *Izvistia Imperatorskoi Akademii Nauk (Bulletin de l'Académie Impériale des Sciences de St.-Pétersbourg).* 6th ser., *7*, 153–162. English translation by Morris Halle, 1956.

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. In *Advances in neural information processing systems* (pp. 6294–6305).

McRae, K., Spivey-Knowlton, M. J., & Tanenhaus, M. K. (1998). Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, *38*(3), 283–312.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Černocky, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, T., Yih, W.-t., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 746–751).

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, *38*(11), 39–41.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., & Jin, Z. (2016). How transferable are neural networks in nlp applications? In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 479–489).

Mundra, R., Peng, E., Socher, R., & Sohmshetty, A. (2017). Lecture notes in natural language processing with deep learning. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lecture_notes/cs224n-2017-notes2.pdf. Accessed: 2019-01-05. Stanford University.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1717–1724).

Osgood, C., Suci, G., & Tenenbaum, P. (1957). *The measurement of meaning*. Urbana: University of Illinois Press.

Padó, S. & Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, *33*(2), 161–199.

Padó, S., Padó, U., & Erk, K. (2007). Flexible, corpus-based modelling of human plausibility judgements. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (emnlp-conll)*.

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Pang, B. & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on association for computational linguistics* (p. 271). Association for Computational Linguistics.

Pang, B. & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 115–124). Association for Computational Linguistics.

Paulus, R., Xiong, C., & Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). 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)* (pp. 2227–2237).

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf*. Accessed: 2019-05-23.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf*. Accessed: 2019-05-23.

Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE conference on computer vision and pattern recognition, CVPR workshops 2014, columbus, oh, usa, june 23-28, 2014* (pp. 512–519).

Reisinger, J. & Mooney, R. (2010). A mixture model with sharing for lexical semantics. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 1173–1182). Association for Computational Linguistics.

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

Roth, G. & Dicke, U. (2005). Evolution of the brain and intelligence. *Trends in cognitive sciences*, *9*(5), 250–257.

Rubenstein, H. & Goodenough, J. B. (1965). Contextual correlates of synonymy. *Communications of the ACM*, *8*(10), 627–633.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252.

Schnabel, T., Labutov, I., Mimno, D., & Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 298–307).

Schuster, M. & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Shin, H., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., … Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Trans. Med. Imaging*, *35*(5), 1285–1298.

Silberer, C. & Lapata, M. (2014). Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)* (Vol. 1, pp. 721–732).

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. Y. (2008). Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 254–263). Association for Computational Linguistics.

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., & Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, *2*, 207–218.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642).

Speer, R. & Chin, J. (2016). An ensemble method to produce high-quality word embeddings. *arXiv preprint arXiv:1604.01692*.

Speer, R. & Havasi, C. (2012). Representing general relational knowledge in conceptnet 5. In *Proceedings of the eighth international conference on language resources and evaluation, LREC 2012, istanbul, turkey, may 23-25, 2012* (pp. 3679–3686).

Subramanian, S., Trischler, A., Bengio, Y., & Pal, C. J. (2018). Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Taylor, W. L. (1953). "cloze procedure": A new tool for measuring readability. *Journalism Bulletin*, *30*(4), 415–433.

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384–394). Association for Computational Linguistics.

Turney, P. D. (2012). Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, *44*, 533–585.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on machine learning* (pp. 1096–1103). ACM.

Voorhees, E. M. & Tice, D. M. (2000). Building a question answering test collection. In *Proceedings of the 23rd annual international acm sigir conference on research and development in information retrieval* (pp. 200–207). ACM.

Wickens, C. D., Gordon, S. E., Liu, Y., et al. (1998). An introduction to human factors engineering.

Wiebe, J., Wilson, T., & Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, *39*(2-3), 165–210.

Wieting, J., Bansal, M., Gimpel, K., & Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ..., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xiao, H. (2018). Bert-as-service. https://github.com/hanxiao/bert-as-service. Accessed: 2019-05-23.

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, *13*(3), 55–75.

Zadrozny, W., Budzikowska, M., Chai, J., Kambhatla, N., Levesque, S., & Nicolov, N. (2000). Natural language dialogue for personalized interaction. *Communications of the ACM*, *43*(8), 116–116.

Appendix A. Dataset Information

A.1 Human-Generated Question Dataset

This section briefly presents an analysis of the human-generated questions used throughout this work. The questions were collected in a preliminary user study in order to assess how real humans would ask questions about fact values. Participants were asked to formulate questions as they would when conversing with a digital voice assistant. The questions were supposed to be paraphrased in an easy yet varying fashion.

In total, 127 questions were generated. Following Berg, Isard, and Moore, 2013, four categories of questions could be recognized in the generated questions.

- **Wh-Question** e.g., *What is the value of electric current of <x>?*
- **Wh-Request** e.g., *Tell me what is the current of <x>.*
- **NP-Request** e.g., *Tell me the electric current of <x>.*
- **Command (N)** e.g., *Electric current of <x>.*

Table A.1. Distribution of question categories of the human-generated questions over the physical quantity categories.

Question Category	Current	Potential	Pressure	Resistance	**Σ**
Wh-Question	31	31	31	27	**120**
Wh-Request	0	0	2	0	**2**
NP-Request	1	1	0	2	**4**
Command (N)	0	1	0	0	**1**
Σ	**32**	**33**	**33**	**29**	**127**

Table A.1 gives an overview of the distribution of the categories with respect to the physical quantity categories. It shows an extremely uneven distribution in terms of question category with over 90 % being Wh-Questions. To get a closer look, table A.2 depicts the question distribution within the Wh-Question category separated by the respective interrogative word. It tells that most of the Wh-Questions start with *What* followed by questions starting with *How*.

Figure A.1 gives an impression about the phrasing of the human-generated questions. It depicts the distribution of the most frequent prefix trigrams.

Table A.2. Wh-Question distribution by interrogative word over the physical quantity categories.

Interrogative Word	Current	Potential	Pressure	Resistance	Σ
What	19	18	15	21	73
Which	1	2	1	1	5
How	10	10	12	5	37
When	0	0	1	0	1
At what	1	1	1	0	3
At how	0	0	1	0	1
Σ	**31**	**31**	**31**	**27**	**120**

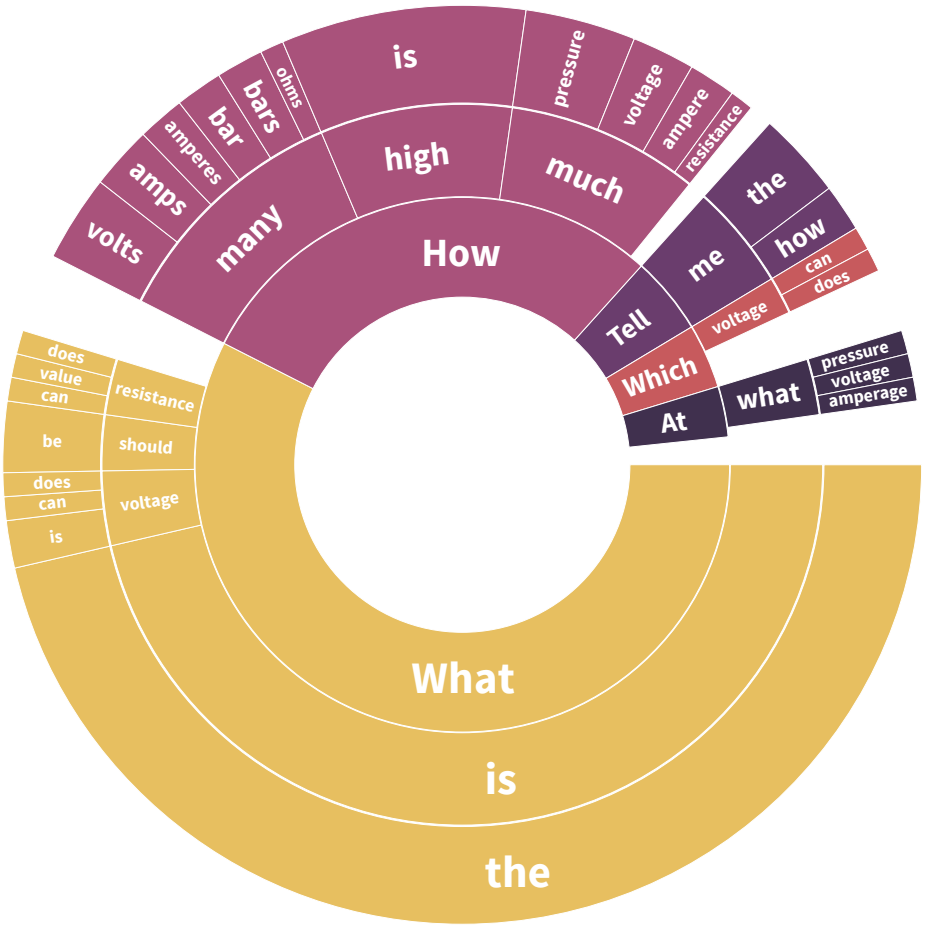


Figure A.1. Distribution of the frequency of question prefix trigrams.

Appendix B. Result Plots

B.1 Sentence Similarity Plots

By analogy with Section 4.1, for each model the following plots showcase the two-dimensional projections of question embeddings obtained by applying Principal Component Analysis (PCA). The embeddings were all generated by average pooling the word embeddings of the questions. All depicted self-trained instances have been trained on the standard dataset, and the fine-tuned ELMo instance on the cased dataset without stopwords.

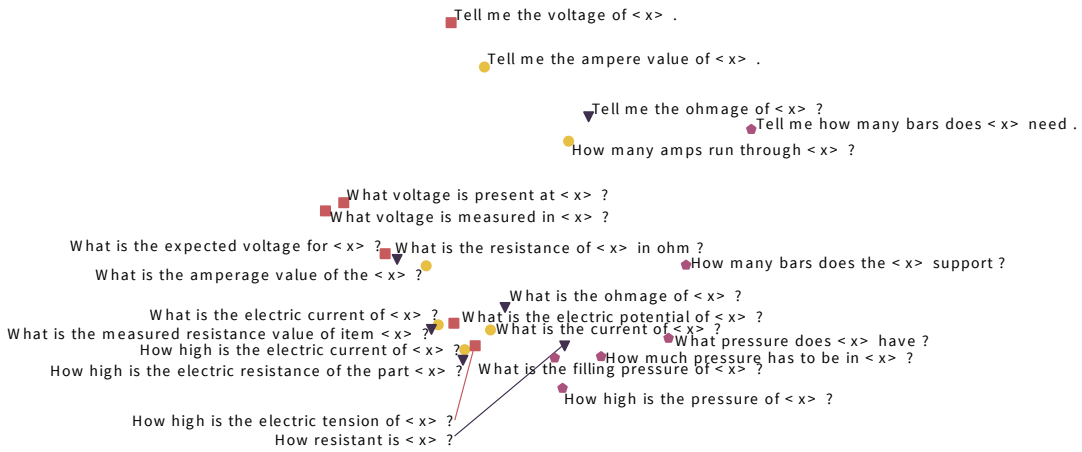


Figure B.1. Pre-trained Word2Vec sentence similarity plot.

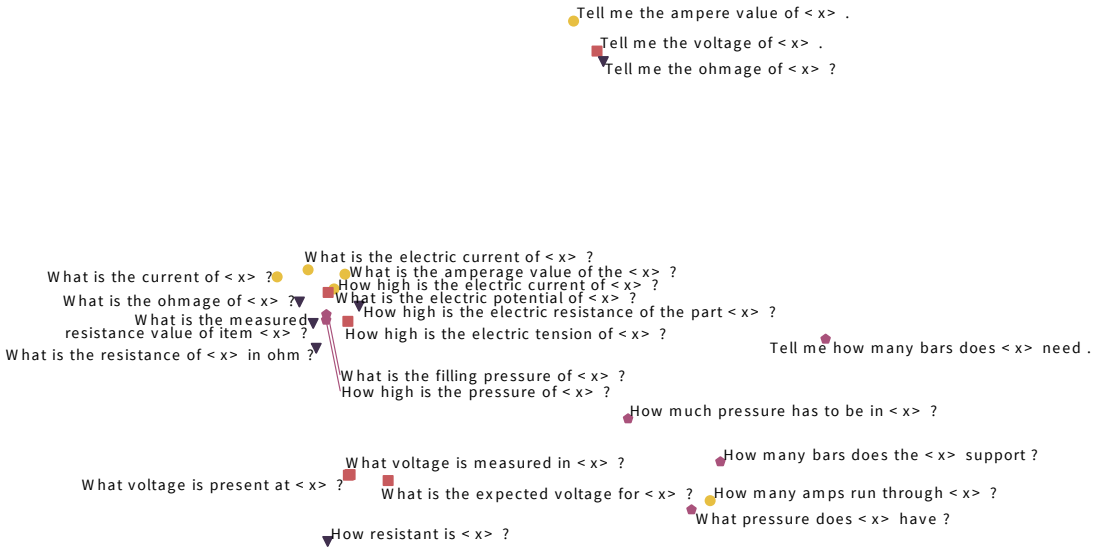


Figure B.2. Self-trained Word2Vec sentence similarity plot.

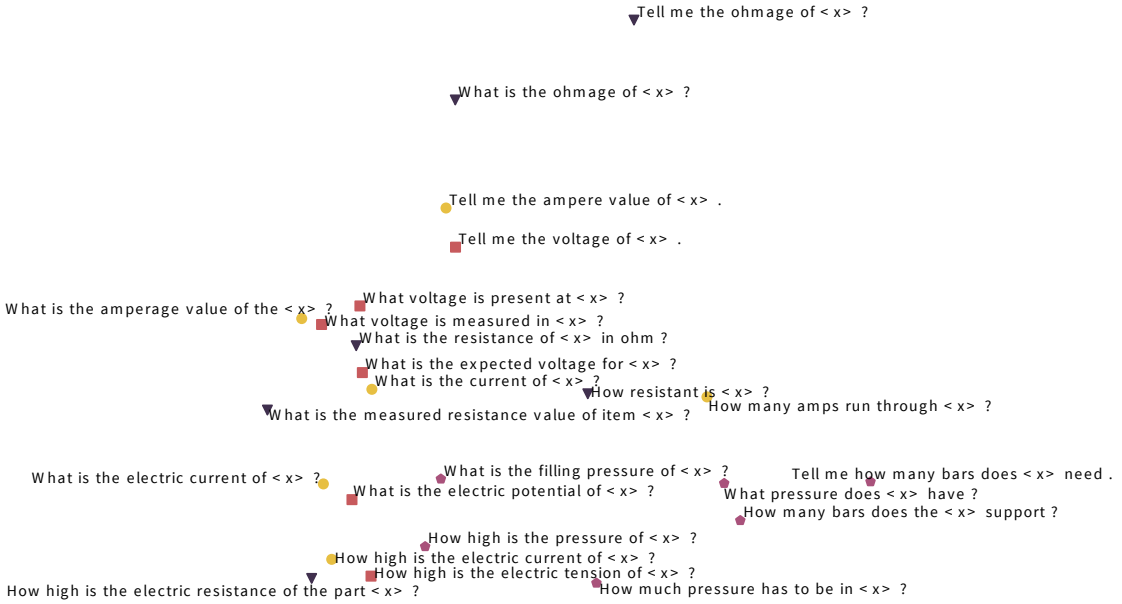


Figure B.3. Pre-trained GloVe Sentence Similarity Plot.

Figure B.4. Self-trained GloVe sentence similarity plot.

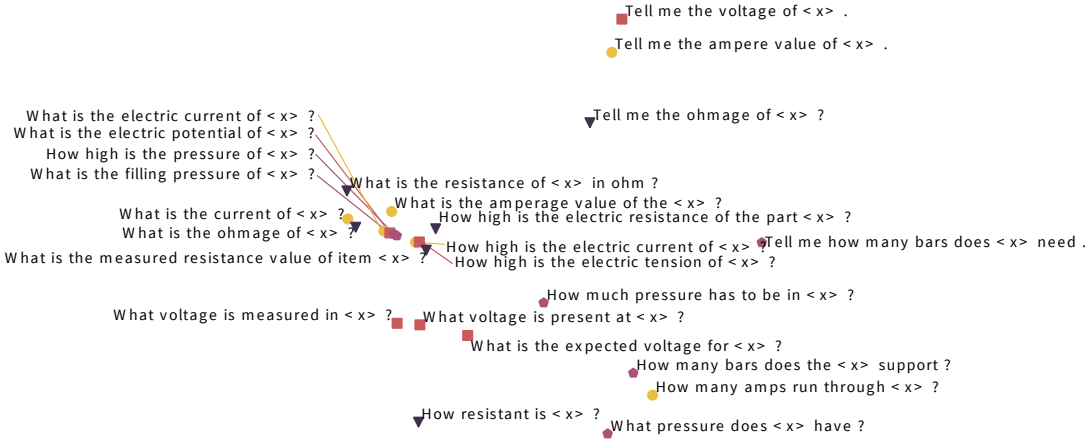


Figure B.5. Pre-trained FastText sentence similarity plot.

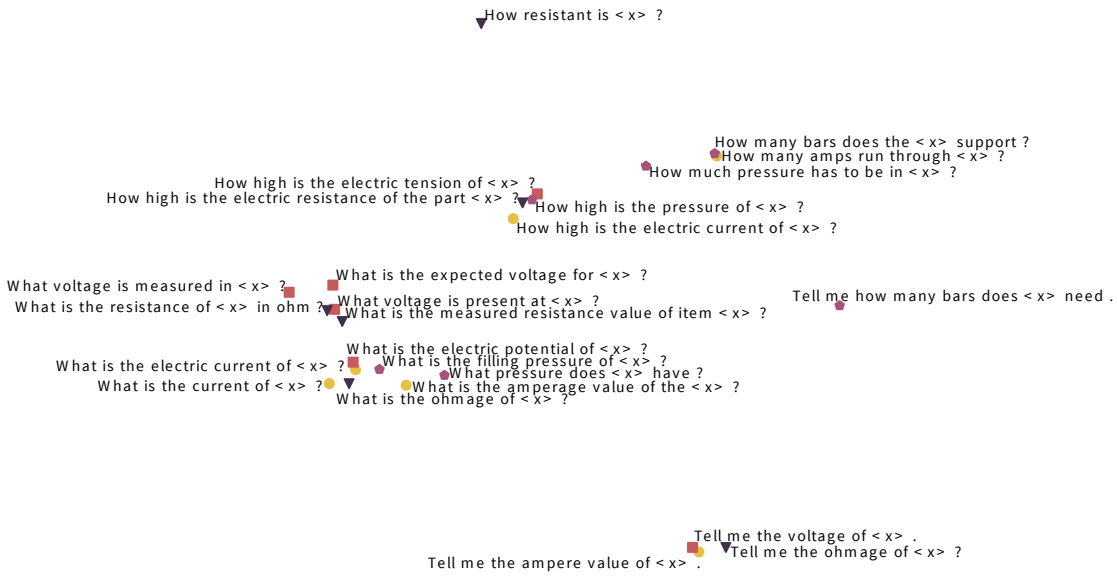


Figure B.6. Self-trained FastText sentence similarity plot.

Figure B.7. Pre-trained ELMo sentence similarity plot.

Figure B.8. Fine-tuned ELMo sentence similarity plot.

Figure B.9. Pre-trained BERT sentence similarity plot.

B.2 ROC Curves of Model Performance on the Question Pair Classification Task

Apart from the F1 metric (see Section 3.5.1), binary classifiers can also be evaluated with the Receiver Operator Characteristic (ROC). This approach maps the relationship of the true and false positive rates of a classifier given different threshold values. In the case of the question pair classification task, the false positive rate can be defined as the probability that the classifier labels two questions as positive, i.e. it determines them as equal category questions, even though they actually originate from different categories. The true positive rate, on the other hand, is the probability that the classifier identifies two questions as equal category questions given that they actually originate from the same category. In general, for a high false positive rate, the true positive rate is also high. In other words, if we bias the classifier towards positive classifications by setting the acceptance threshold low, both the true as well as the false positive rates go up as more items will be labeled positively in the first place. On the other hand, if we bias the classifier towards negative classifications by setting the acceptance threshold high, more items will be classified negatively resulting in lower false and possibly also true positive rates. An accurate classifier, however, shows a large true positive rate even for a very small false positive rate, meaning it can accurately discriminate positive from negative examples.

The area under the ROC curve, also known as Area Under the Curve (AUC), serves as evaluation score identifying the quality of the classifier: an ideal classifier has an AUC of 1, meaning that it has a true positive rate of 1 when the false positive rate is 0. Looking at the plots Figures B.10 - B.13, the 45° diagonal indicates a hypothetical poor classifier randomly assigning questions to either the equal or unequal category. The yellow line indicates the pre-trained and the purple line the best self-trained/fine-tuned instance of each model.

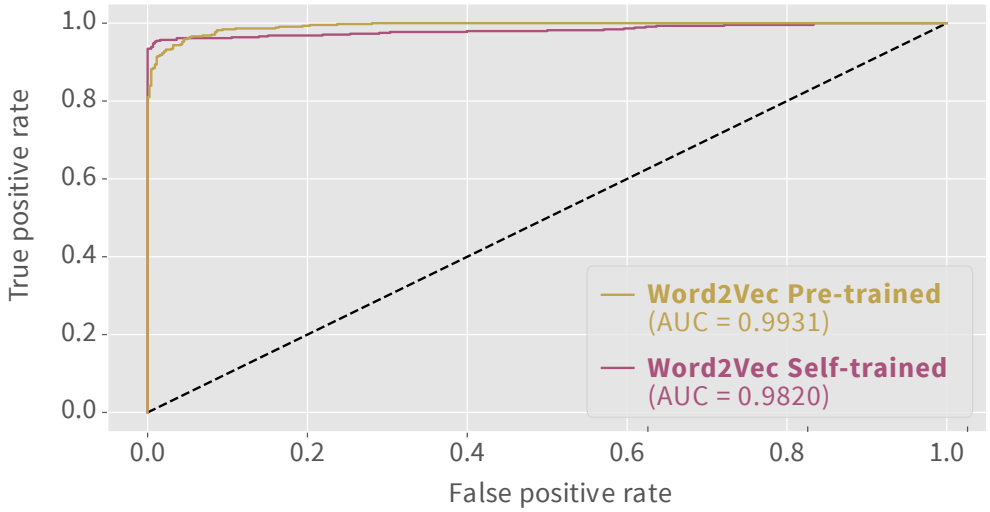


Figure B.10. Comparison between the pre-trained and self-trained Word2Vec model. The curves depict the strongest model and pooling combinations of the respective instances. The best pre-trained combination to a very small degree performs better than the best self-trained combination.

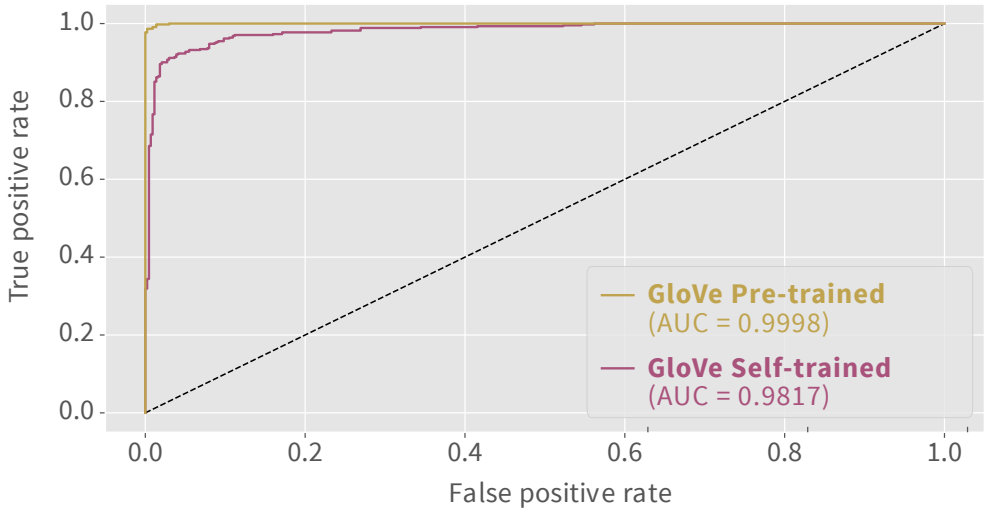


Figure B.11. Comparison between the pre-trained and self-trained GloVe model. The curves depict the strongest model and pooling combinations of the respective instances. The best pre-trained combination performs better than the best self-trained combination.

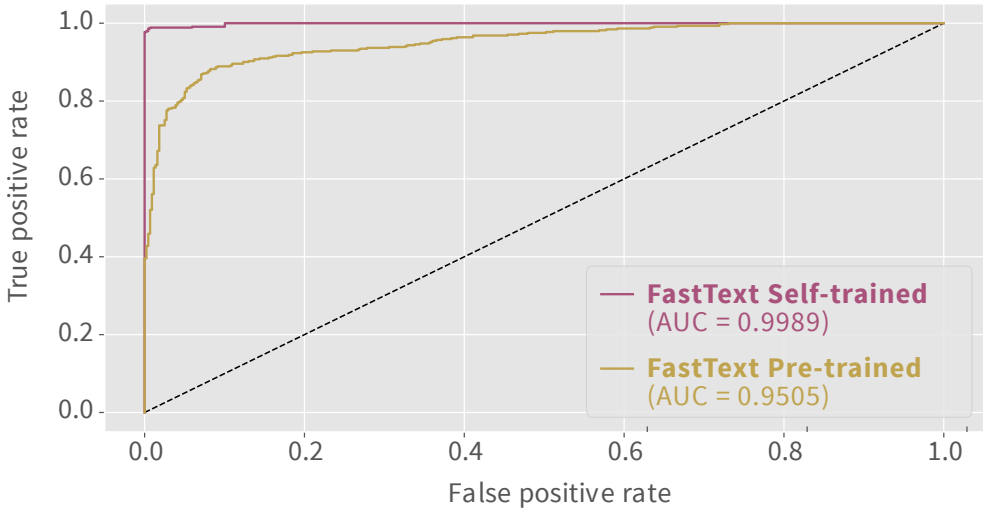


Figure B.12. Comparison between the pre-trained and self-trained FastText model. The curves depict the strongest model and pooling combinations of the respective instances. The best self-trained combination performs considerably better than the best pre-trained combination.

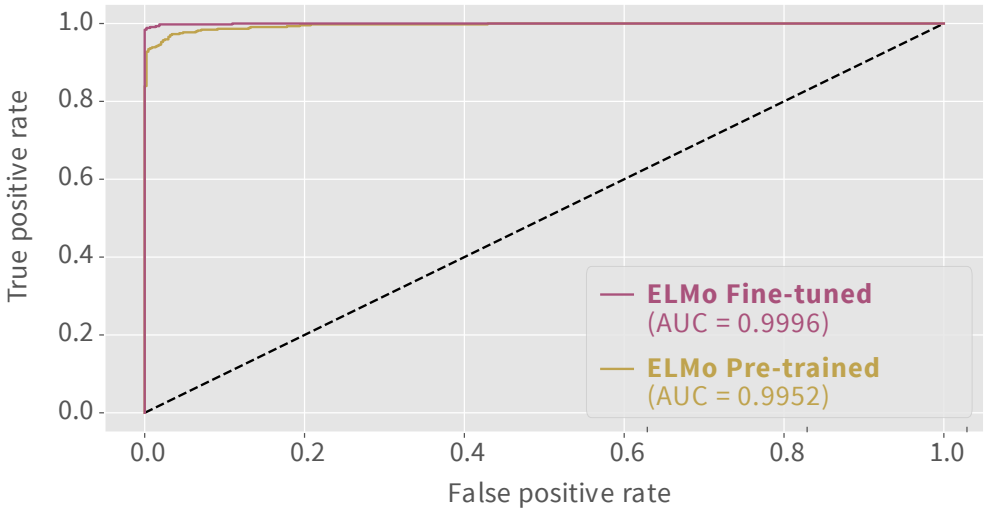


Figure B.13. Comparison between the pre-trained and fine-tuned ELMo model. The curves depict the strongest model and pooling combinations of the respective instances. The best fine-tuned combination performs slightly better than the best pre-trained combination.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Würzburg, 6.6.2019

Jonas Müller