DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Improving Wikipedia Knowledge Exploration Capabilities by Similarity and Aspect Based Navigation

Emanuel Johannes Gerber

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Improving Wikipedia Knowledge Exploration Capabilities by Similarity and Aspect Based Navigation

Verbesserung der Wissensexploration in Wikipedia mittels Ähnlichkeits- und Aspektbasierter Navigation

Author:	Emanuel Johannes Gerber
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Tim Schopf, M.Sc.
Submission Date:	15.08.2022

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

Munich, 15.08.2022 Emanuel Johannes Gerber

Acknowledgments

I would like to thank Prof. Dr. Florian Matthes for the opportunity to write my thesis at the chair of Software Engineering for Business Information Systems (sebis). I also want to thank my advisor Tim Schopf. During the last six months, he always guided me in the right direction and immediately helped, when issues appeared during the work of this thesis. Many thanks also to my fellow students and my family who supported my thesis by giving feedback and taking part in my user study.

Abstract

Knowledge Exploration is generally described as a complex, open-ended task of browsing through large amounts of textual information, that requires the reader to quickly navigate between different texts while deciding what content is relevant. Due to its static nature, the current hyperlink navigation structure of Wikipedia does not address the specific information needs of a user in a knowledge exploration scenario.

We address this problem by implementing a content-based recommender system that uses measures of textual similarity and aspect-based similarity to create navigation links between Wikipedia sections that share a given aspect. For this, we train aspect-based document embeddings for an information retrieval task using SBERT and SimCSE. We explore different techniques for training embeddings based on little or no available labeled training data through various data augmentation methods and by exploiting external knowledge from Wikidata.

Our experiments show that data augmentation leads to small improvements only when sufficiently enough data is available while deteriorating the performance of aspect-based embeddings when too little data is available. Furthermore, we show that aspect-specific datasets can be effectively created by using Wikidata in combination with Wikipedia. Finally, our user study compares the user preference for generated navigation links from aspect-based embeddings and baseline Wikipedia links for two knowledge exploration scenarios. The results indicate that users favor aspect-based navigation links that encode the similarity of multiple aspects.

Kurzfassung

Wissensexploration wird im Allgemeinen als eine komplexe, offene Aufgabe beschrieben, bei der große Mengen an Textinformationen durchsucht werden, wobei der Leser schnell zwischen verschiedenen Texten navigieren und entscheiden muss, welche Inhalte relevant sind. Aufgrund ihrer statischen Natur passt sich die derzeitige Hyperlink-Navigationsstruktur von Wikipedia nicht an die entsprechenden aufgabenspezifischen Informationsbedürfnisse eines Benutzers an.

Wir gehen dieses Problem an, indem wir ein inhaltsbasiertes Empfehlungssystem implementieren, das verschiedene Metriken von textueller Ähnlichkeit und aspektbasierter Ähnlichkeit verwendet, um automatisch Navigationslinks zwischen Wikipedia Abschnitten zu erstellen, welche einen bestimmten Aspekt gemeinsam haben. Zu diesem Zweck trainieren wir aspektbasierte Dokumenteinbettungen mit SBERT und SimCSE. Wir erforschen verschiedene Techniken für das Training von Einbettungen auf der Basis von wenig oder gar keinen verfügbaren gelabelten Trainingsdaten durch verschiedene *Data Augmentation* Methoden und durch die Nutzung von externem Wissen aus Wikidata.

Unsere Experimente zeigen, dass *Data Augmentation* nur dann die Qualität von aspektbasierten Einbettungen verbessert, wenn ausreichend Trainingsdaten zur Verfügung stehen, während sich die Qualtiät verschlechtert, wenn zu wenige Ausgangsdaten verfügbar sind. Darüber hinaus zeigen wir, dass aspektspezifische Datensätze durch die Verwendung von Wikidata in Kombination mit Wikipedia effektiv generiert werden können. Zuletzt vergleichen wir in einer Benutzerstudie die Benutzerpräferenz von generierten Navigationslinks aus aspektbasierten Einbettungen und klassichen Wikipedialinks in zwei Explorationsszenarien. Unsere Ergebnisse zeigen, dass Nutzer aspektbasierte Navigationslinks bevorzugen, die die Ähnlichkeit von mehreren Aspekten kodieren.

Contents

Abbreviations

BOW Bag of Words

CWE Insertion Contextual Word Embedding Insertion

CWE Substitution Contextual Word Embedding Substitution

MRR Mean Reciprocal Rank

NSP Next Sentence Prediction

P Precision

PLS Positive Label Substitution

PWC PapersWithCode

R Recall

SBERT SentenceBert

STS Semantic Text Similarity

WD Wikidata

1 Introduction

1.1 Motivation

Knowledge Exploration in text generally describes the task of browsing through large amounts of textual information combined with an open-ended and complex task [1]. It requires the reader to quickly decide what information is relevant and what is not. To effectively explore a large collection of text documents, users need a structure that helps them quickly navigate from one topic to another. Modern encyclopedias such as Wikipedia provide such a navigation structure through hyperlinks that connect pieces of related information. A user that searches for information about "Microsoft" can therefore quickly jump to related pages such as "Bill Gates" or "Microsoft Office". While the hyperlink structure of Wikipedia in particular is frequently updated by the community, the approach of linking pieces of content via static links has several shortcomings. Most importantly, this approach does not address the particular information needs of the user. A user who searches for "the Financial Model behind Scooters" may follow different trails of information through different links than a user that is interested in "The environmental effect of Scooters".

The static nature of Wikipedia links imposes a limit to how effectively a reader can explore the depth of information available. Furthermore, the hyperlink approach from Wikipedia builds on the manual work of a large community. This approach is therefore not applicable to most other text platforms that cannot build on a large community for labeling data.

The goal of this master thesis is to develop and evaluate an approach for enabling navigation within Wikipedia that addresses the specific information needs of the reader. We want to approach this task using content-based recommender systems as has been done in previous work by Ostendorff et al. [2] [3]. For this, we will implement and evaluate different measures of textual similarity and aspect-based similarity to automatically create links between Wikipedia sections that allow the user to navigate from one place to another. The main focus of this work lies in tailoring the type of navigation to the particular user's needs, showing preferably only content that helps the user to find what he searches for.

1.2 Problem Statement

In this work, we will use state-of-the-art approaches in textual similarity and aspect-based similarity to automatically create links between Wikipedia sections that allow the user to navigate from one place to another. While previous approaches like [3] already provided implementations for enabling aspect-specific similarity, they always worked based on large annotated datasets. In most scenarios, such a large amount of training data is not available. Therefore we explore multiple ways how to train aspect-based embeddings using only small amounts of labeled data or using completely unsupervised approaches. Existing benchmark scores from [3] are used to evaluate the performance of the proposed approaches and to derive the best-performing methods.

In particular, we will try out two approaches: Firstly, we will apply Data Augmentation as a way of artificially generating new data from a small existing dataset. Data Augmentation involves creating variations of an existing data source e.g. by applying insertions, deletions, or replacements of words.

Secondly, we will use Wikidata as an external Knowledge Base for creating custom aspect-specific datasets, that require no manual labeling. We will also compare this approach against the performance of gold-labeled datasets.

1.3 Research Questions

Formally, the thesis addresses the following research questions:

1. What is aspect-based similarity?
2. What is the state of the art for semantic textual similarity and aspect-based similarity?
3. How to create embedding representations that capture the similarity for specific aspects?
4. How to evaluate the improvements in knowledge exploration on Wikipedia?

1.4 Structure of the Thesis

In Chapter 2, we analyze the state-of-the-art for Semantic Text Similarity and Aspect-Based Similarity. We give an introduction of all relevant technologies related to this thesis, including Word Embeddings, Document Embeddings, and Wikidata.

Chapter 3 explains the datasets and all preparation and preprocessing steps. We also provide an analysis of the dataset composition.

Chapter 4 explains the methodology of the thesis. We describe the different models that we used for training aspect-based embeddings, including SBERT and SimCSE. Furthermore, we describe different data augmentation strategies that we applied as well as the different experiments that were performed.

Chapter 5 evaluates the results from all experiments and shows different visualizations from our trained embeddings. Furthermore, we show the results from our user study on knowledge exploration.

Chapter 6 discusses the results and concludes from the experiments. We furthermore highlight the limitations of our results and suggest further research directions on the topic.

Finally, in Chapter 7, we summarize our experiments and results and give some recommendations for future work.

2 Fundamentals and Related Work

The problem of aspect-based similarity is strongly connected to the field of Semantic Text Similarity (STS). The following section gives an overview of previous work that we build on for our experiments. The section covers the foundations of modern STS approaches, including Transformer models like BERT, but also highlights previous methods like Word Embeddings, which motivated many state-of-the-art technologies today. Besides the technical foundation and essential work for STS, we will present the most recent attempts for aspect-based similarity and highlight the concept of aspects clearer.

2.1 Semantic Text Similarity

Semantic Text Similarity (STS) is a task that assesses the degree of equivalence between two texts in terms of their meaning. It plays "[...] an important role in information retrieval, automatic question answering, machine translation, dialogue systems, and document matching." [4]. STS is a challenging task for machine learning models because it requires a deep understanding of the text to accurately assess the degree of equivalence. A particular challenge behind STS is the fact that similar meanings can be expressed in a large variety of ways, including different styles, vocabulary, etc.

As shown in Figure 2.1, there exists a broad collection of techniques in the field of STS. Our experiments only focus on **Corpus-Based Similarity** as current state-of-the-art approaches originate from that domain. The general principle behind Corpus Based Text Similarity is to first represent the text as a vector space and then apply distance measures to the vectors to assess the degree of similarity. (*Note: As distance and similarity are inverse to each other, we will use both terms depending on the context*. Different distance measures have been used in the past, such as euclidean distance or hamming distance while the most common distance measure today is cosine distance.

$$cosine\_distance(x, y) = \frac{x \cdot y}{\|x\| \|y\|} \tag{2.1}$$

Cosine distance is a measure of the angle between two vectors. It is in the range $[-1, 1]$, where -1 means that the two vectors are exactly opposite, and 1 means that the two vectors are exactly aligned. The cosine similarity is also known as the dot product.

2.1.1 Evaluation of Semantic Textual Similarity

STS is typically evaluated using a dataset that contains pairs of texts and human-annotated labels indicating the degree of equivalence between the two texts. The most frequently used datasets are listed in [5] and include:

- SICK (Sentences Involving Compositional Knowledge) [6] consists of English sentence pairs that were annotated for relatedness and entailment.
- SNLI (Stanford Natural Language Inference) [7] contains sentence pairs manually labeled for entailment classification.
- STS (Semantic Text Similarity) [8] consists of pairs of sentences with labels annotating their semantic similarity.

2.1.2 Early Work

Very early methods for STS are based on the idea of Bag of Words (BOW). The Bag of Words approach ignores grammar and word order, and simply counts the number of times each word occurs in the text. The resulting word frequency vectors can be then compared using distance measures. Previous approaches also included Matrix Factorization techniques such as Latent Dirichlet Allocation (LDA). LDA is a statistical model that is used to find relationships between topics in a text. It is based on the idea that each text is a mixture of topics, and that words that are close together in meaning tend to be used together in a text. Similar to BOW, LDA computes a vector representation of an input text that can be compared using cosine similarity.

2.1.3 Context Free Word Embeddings

Both BOW and LDA have the essential shortcoming that they disregard word order in a text and do not account for the fact that words can have multiple synonyms. Therefore, they fail at understanding that two texts are similar in meaning when they use different expressions. An improvement for this problem is Word Embeddings:

Word embeddings are vector representations of words that are learned from text data. They are trained on algorithms, including the continuous bag-of-words (CBOW) and skip-gram architectures. CBOW trains word embeddings by predicting a center word by its k surrounding context words. This training has been shown to effectively produce word representations in high dimensional latent space where words with similar meanings or of similar type (e.g. banana and apple), are closer to each other than words with dissimilar meanings (e.g. banana and democracy). The resulting word embeddings can be compared using similarity measures such as cosine similarity or

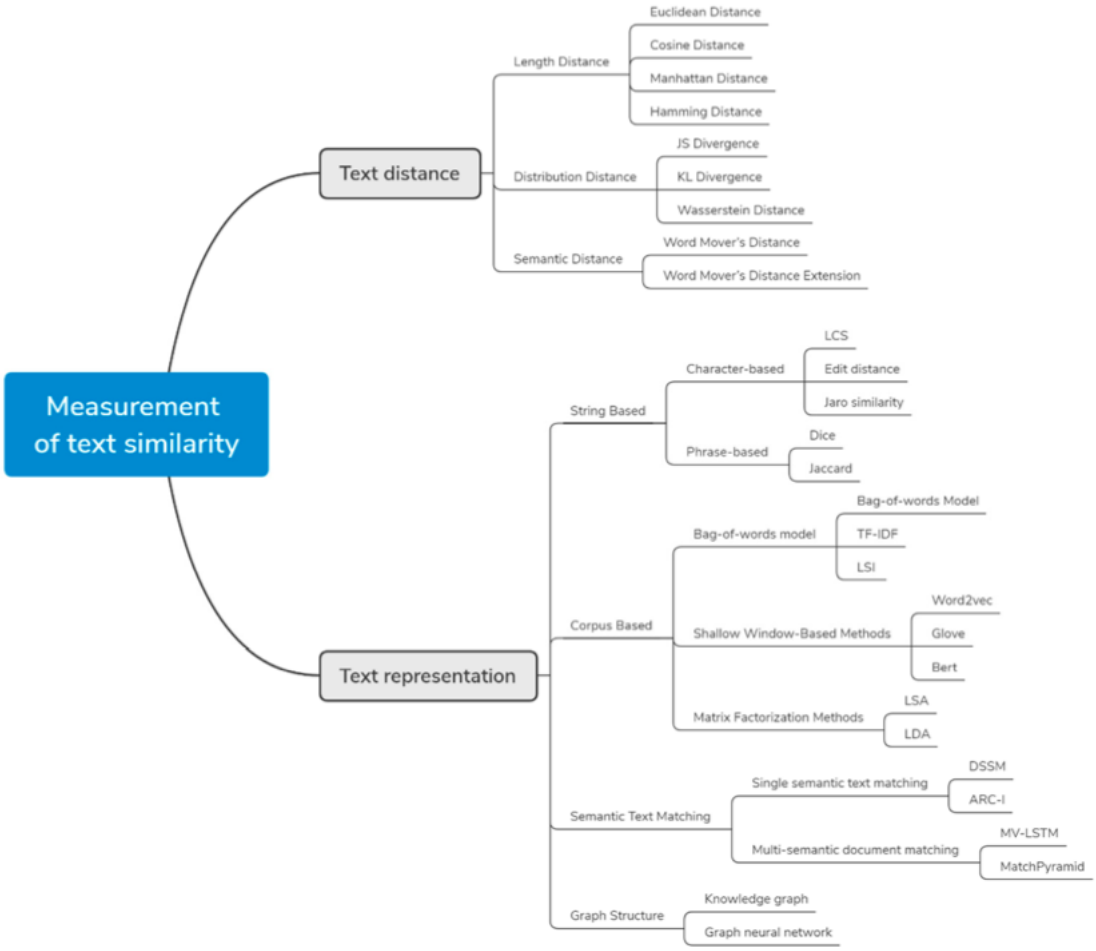


Figure 2.1: Overview of different Measurements of text similarity [4]

euclidean distance. Different word embeddings models have been developed in the past including Word2vec[9] and GloVe[10]. While Word2vec creates word vectors by using a neural network, GloVe is a model that uses a co-occurrence matrix that contains the number of times two words appear together in a corpus of text. Therefore, it can capture word similarity in a more global context ("which words are used frequently in the same document?") compared to Word2vec which only captures word similarity in a local context ("which words are used frequently in the same local neighborhood?"). Different methods exist for computing sentence embeddings or document embeddings using word embeddings. This generally involves using the average of all word embeddings or the average of all word embeddings from the most relevant words in the document.

This approach has been shown to suffer from the conflation of meaning and increasingly noisy data for a growing number of words [11]. For that reason, Doc2vec[12] has been developed as an extension of Word2Vec. Doc2Vec uses the same CBOW architecture and trains word embeddings by predicting a center word by taking as input the surrounding context words and the *document_id*. Thereby the model can learn not only word embeddings but also document embeddings.

2.1.4 Context Sensitive Word Embeddings

Due to the nature of their training, previously described Word Embedding methods have failed at distinguishing the meaning of lexically identical words in different contexts (e.g. bank (financial institution) and bank (place to sit)).

A new method for context-sensitive word embeddings was introduced by the paper *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* [13]. The paper presents a model called BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is a model that is trained on a large amount of text data from Wikipedia and BooksCorpus and can effectively create context-sensitive word embeddings. A major improvement behind BERT was the use of attention and self-attention. The attention mechanism allows the model to focus on relevant inputs when producing output, while the self-attention model allows inputs to interact with each other.

Model Training

BERT is trained completely without labels. Its training is composed of two steps: In the first training step, the model is pre-trained on a large corpus of unlabeled data from Wikipedia and BooksCorpus using a self-supervised learning technique called Masked Language Modeling (MLM). In this phase, the model is given a sentence with some words randomly masked (replaced with a [MASK] token), and the task is to predict the masked words. This pretraining process helps the model learn general language representations that can be applied to a variety of tasks. The second training step is Next Sentence Prediction (NSP). For this task, the model is given two sentences and must predict whether the second sentence typically follows the first. This task helps the model learn the relationship between sentences. The paper shows that BERT can outperform previous state-of-the-art NLP models on several tasks, including question answering and language understanding. Importantly, BERT is also able to generalize to new tasks, showing that it has truly learned generalizable representations of language.

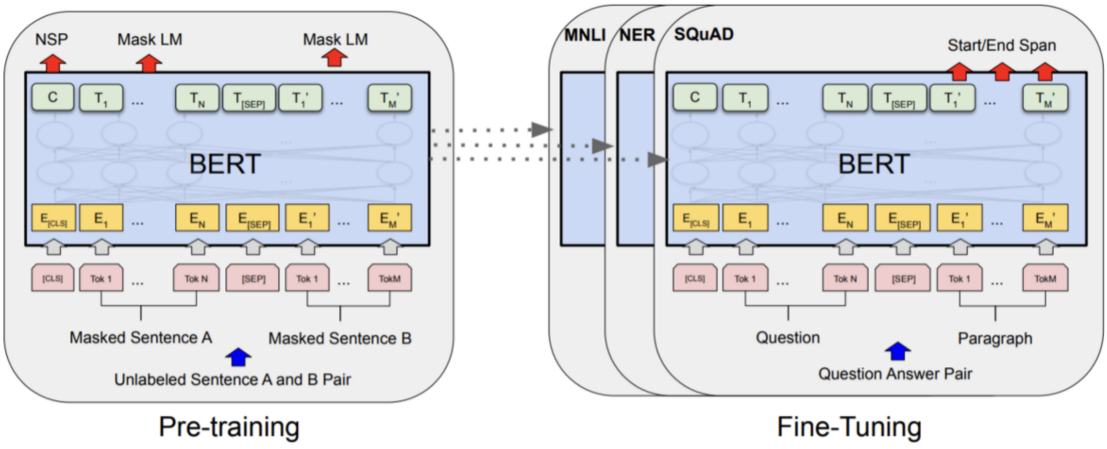


Figure 2.2: Illustration of the Architecture and Training Procedure behind BERT (taken from TowardsDataScience)

Beyond Word Embeddings

While BERT produces context-sensitive word embeddings, there are techniques to obtain sentence-level embeddings from the model as well. The most commonly used approach involves taking the final hidden state of the [CLS] token and using it as a representation of the whole sentence. The CLS token is a special classification token and represents the final hidden state of the model output. In the training phase, this CLS token is used for the Next Sentence Prediction task as it captures the meaning of the entire sentence.

A second approach to obtaining sentence-level embeddings is Mean pooling. This approach simply takes the mean of the hidden states of all the tokens in the sentence.

Sentence Embeddings vs Document Embeddings

While the paper mentions sentences as the inputs to the model, a sentence is loosely defined as "an arbitrary span of contiguous text, rather than an actual linguistic sentence" [13, p. 4]. Therefore, the model allows the encoding of longer sequences of texts (e.g. whole documents) as will be done in our future experiments. Consequently, the terminology "sentence embeddings" and "document embeddings" will be used interchangeably in the following sections.

The only limit to the model input is imposed by the network architecture, which is can represent up to 512 input tokens.

SciBERT

BERT can be applied to a wide range of domains. However, BERT was developed on data from Wikipedia and BookCorpus, which mainly include non-scientific language and wording. This is why SciBERT [14] was introduced. SciBERT is a pre-trained model for NLP that is specifically designed for scientific texts. It is based on the BERT model but has been trained on scientific text data from academic papers. Furthermore, the adapted vocabulary of SciBERT was created to better represent scientific terminology. Similar to BERT, SciBERT can be used for a variety of tasks, such as text classification, information extraction, and question answering. The paper shows significant improvements for downstream scientific NLP tasks compared to the original BERT model.

2.2 Universal Sentence Encoder

The paper Universal Sentence Encoder (USE) [15] provides a model for "encoding sentences into embedding vectors that specifically target transfer learning to other NLP tasks" [15, p. 1]] like text classification, semantic similarity and clustering. "The model takes as input English strings and produces as output a fixed dimensional embedding representation of the string." [15, p. 1]. The pre-trained USE comes with two encoder variations: Transformer Encoders (derived from [16]), and Deep Averaging Networks [17]. While the Transformer Encoder generally shows higher accuracy due to the ability of context-aware word representations, it has a "greater model complexity and resource consumption" [15, s. 3]. The second encoding model with Deep Averaging Network (DAN) on the other hand shows little lower accuracy but is more computationally efficient. This model makes use of a mechanism "whereby input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network (DNN) to produce sentence embeddings" [15, s. 3]. Both encoder variations return a fixed dimensional embedding output of size 512. This embedding represents the meaning of the input sentence in latent space and can be used for computing sentence similarity using cosine similarity (Figure 2.3) or as input to subsequent network layers for training on a more complex task.

Trained on the Stanford Natural Language Inference (SNLI) corpus [7], the model shows strong results on several transfer learning tasks (movie review classification, sentiment analysis, subjectivity evaluation, phrase level, opinion, etc.), especially with only little available training data. Also, the more complex transformer-based encoder achieves close to state-of-the-art results on the STS benchmark (0.782).

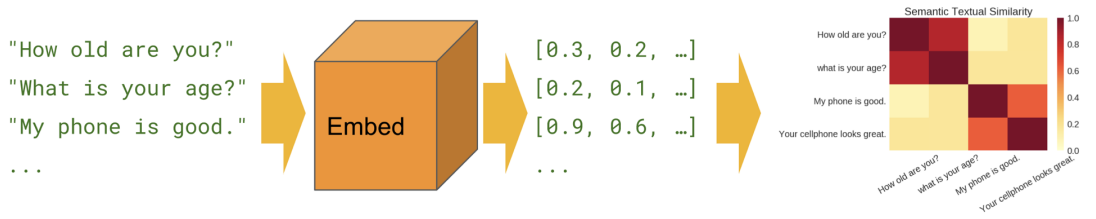


Figure 2.3: Example: Embedding sentences with Universal Sentence Encoder. For every input sentence, the corresponding sentence embedding from USE is compared with every other sentence embedding using cosine similarity and displayed as a heatmap. (picture taken from tfhub.dev)

2.3 Sentence BERT

The paper *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks* [18] introduces a "modification of the pre-trained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity" [18, p. 1]. The paper showed previous state-of-the-art results on sentence-pair regression tasks while drastically reducing the inference time by using a siamese network. SentenceBERT (SBERT) builds up the results from BERT [13] and RoBERTa [19] which used cross-encoders to obtain sentence embeddings. While this procedure improved the score on STS benchmark dataset at the time, the paper argues that "this setup is unsuitable for various pair regression tasks due to too many possible combinations" [18, p. 1]. This problem is directly linked to the use of cross encoders.

Cross Encoders

Cross Encoders use the fact that BERT was already trained on a pairwise sentence classification task during the pretraining with Next Sentence Prediction. In that phase, two sentences were fed into the network, separated by a [SEP] token. The model then learned to predict whether the second sentence followed the first sentence. For the sentence similarity task, the same training setup was used to learn the degree of similarity between two sentences.

`[CLS]␣first_sentence␣[SEP]␣second_sentence␣[SEP]`

While this approach showed solid results, it comes with a large computational overhead as for every pair of sentences, the model needs to perform one forward pass since there are no independent sentence embeddings available. The paper showed that for

a collection of 10.000 sentences, it requires approximately 65 hours to compute all pairwise sentence combinations using modern hardware. They, therefore, proposed Siamese Networks as a new technique to reduce the computational cost of the model.

Siamese Networks

A Siamese Network (sometimes also called Bi-Encoder) is a type of neural network that consists of two sub-networks with tied network weights that are used to infer output vectors that can be. The siamese network takes two sentences as input. Each sentence is propagated through a BERT model and the following pooling layer which uses either mean pooling or CLS pooling to obtain a single output vector. This output vector represents the sentence embedding. The siamese network then computes the cross-product of the two sentence embeddings and uses a softmax classifier to compute a similarity score. Afterward, the model uses triplet loss to update the model weights. As the two models in the network share the same weights this training leads to sentence embeddings where similar sentences lie close to each other and dissimilar sentences lie far away from each other. Consequently, sentence embeddings can be computed independently and efficiently compared using cosine similarity. For the above example with 10.000 sentence pairs, this leads to a computation time of about 5 seconds (compared to 65 hours for the cross encoder).

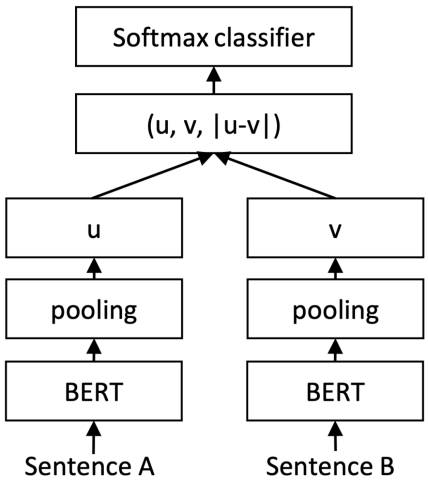


Figure 2.4: Illustration of a Siamese Network Architecture (taken from sbert.net)

Comparison of different Methods

The results show that the BERT CLS vector achieves the lowest scores for all benchmark tasks (avg. 29.19 STS) followed by Average BERT embeddings (54.18) and Average GloVe embeddings (avg 61.32). The best results are obtained from the SRoBERTa-NLI-large model with an average score of 76.68.

Augmented SBERT

Even though the siamese network architecture from SentenceBERT (also called bi-encoder) showed solid results and displayed significant performance benefits over cross-encoders, they typically require a lot of training data to achieve good results. The paper *Augmented SBERT: Sentence Embeddings using Siamese BERT-Networks* [20] evaluates different strategies for data augmentation and pair sampling to improve the performance of the model. The paper exploits the fact that cross-encoders, while being computationally more expensive, generally lead to better results than bi-encoders while also requiring less training data in the process, as was shown by [21]. They propose a new method that uses a small annotated dataset of labeled sentence pairs (named the Gold dataset) and augments it with several augmentation techniques to obtain additional samples of sentence pairs (named the silver dataset). They then use the extended dataset of Gold and Silver samples to train the bi-encoder, thereby increasing the dataset size (see Figure 2.5).

The augmentation technique with cross-encoders involves first training a cross-encoder model on the sentence pairs of the Gold dataset. The cross encoder can then be used to infer the similarity score of unlabeled pairs of sentences that are added to the silver dataset. The paper also evaluates different methods for sampling sentence pairs that should be labeled by the cross encoder, including Semantic Search, Random Sampling, Kernel Density Estimation, BM25, and combinations of these methods.

Additionally, the paper applies standard NLP augmentation techniques for generating a silver dataset using the available methods from the NLPAug Package [22]. These include:

- *Backtranslation*: Translate a text from English to Germany and back from German to English (using a Sequence-to-Sequence model) to obtain a different variation of the original text.
- *Contextual Synonym Replacement*: Replace random words in the text with context-sensitive synonyms using a BERT-based model.

Several experiments on a range of benchmark tasks showed that AugBERT achieves strong results for the soft-labeling approach with cross encoders (in combination with

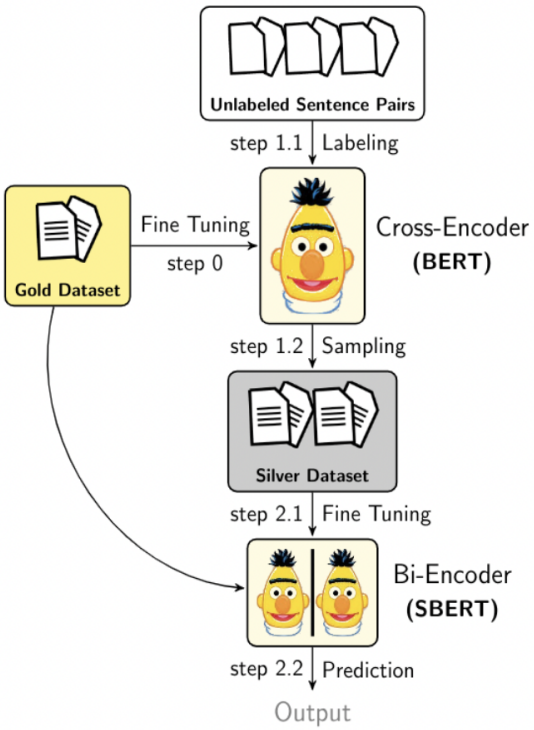


Figure 2.5: Illustration of the training procedure of Augmented SBERT (taken from Github UKPLab)

BM25 for sentence pair sampling) with "an improvement of up to 6 points for in-domain and up to 37 points for domain adaptation tasks compared to the original bi-encoder performance" [20, s.6]. For augmentations with NlpAug, the contextual synonym replacement method was reported to work best. Yet NlpAug was consistently outperformed by cross-encoder augmentation but showed improvements in two of four benchmark datasets over the Gold dataset.

2.3.1 SimCSE

A different approach for dealing with small training datasets was introduced with the paper "SimCSE: Simple Contrastive Learning of Sentence Embeddings" [23]. SimCSE is a method for training sentence embeddings using contrastive learning that showed major improvements over previous STS benchmarks. The idea behind contrastive learning is "to learn effective representation by pulling semantically close neighbors together and pushing apart non-neighbors" [24, s. 1]. The paper provides both a

supervised and an unsupervised approach to train sentence embeddings.

In the supervised approach, the model learns sentence embeddings from contrastive examples of sentences. Each training input is composed of a triplet of sentences (s_1, s_2, s_3) where s_1 is a seed sentence, s_2 is a positive instances of s_1 and s_2 is a negative instance of s_1. The model then learns to find a representation of s_1 that is close to s_2 and far from s_3. Similar to [18], SimSCE uses the NLI (Natural Language Inference) [7] dataset to generate contrastive examples. The NLI dataset is a collection of text pairs that are labeled by the categories "entailment", "neutral", and "contradiction". Entailment means that the first sentence logically entails the second one, neutral means that the two sentences are not related, and contradiction means that the first sentence contradicts the second one. SimCSE uses the NLI dataset to compose contrastive triplets taking the entailment as a positive instance and contradictions as hard negative instances. The paper trained a model based on a total of 994k training examples.

In the unsupervised approach, a model is trained based on contrastive examples of sentence similarity without any labeled data. The paper uses the one million randomly samples sentences from Wikipedia as a training dataset. For each sentence, they compose negative instances using "in-batch negatives", which essentially means that random sentences from the same dataset are taken as hard negatives. Since there exists no information about which sentence is semantically similar to which other sentences, the model generates two variations of each sentence by performing a forward pass through the model (e.g. BERT) while applying different dropout masks each time. Dropout is a method of randomly deactivating a random selection of output units in the neural network, thereby adding noise to the final embedding layer. This technique is usually used to avoid overfitting of neural networks during training. For SimCSE, the different masked dropout variations of the same sentence are used as positive instances and the model is trained to find similar embeddings for the two variations.

The paper showed that the unsupervised SimCSE model (SimCSE-RoBERTa$_{\text{large}}$, avg STS=78.90) performs on par with existing state-of-the-art supervised models (CT-SBERT$_{\text{large}}$, STS=79.39) while supervised SimCSE (SimCSERoBERTa$_{\text{large}}$, STS=83.76) greatly outperforms all previous methods. It shows that in particular for unsupervised SimCSE, the dropout noise performs surprisingly well as a "minimal data augmentation" technique compared to other data augmentation techniques like random word replacement, random word deletion, random word insertion, and random sentence shuffling which have been tested by other papers (see [25],[26]).

2.3.2 The Problem of General Text Similarity

The previous sections discussed various techniques and models for computing semantic textual similarity. The most recent developed models in this sphere have proven very

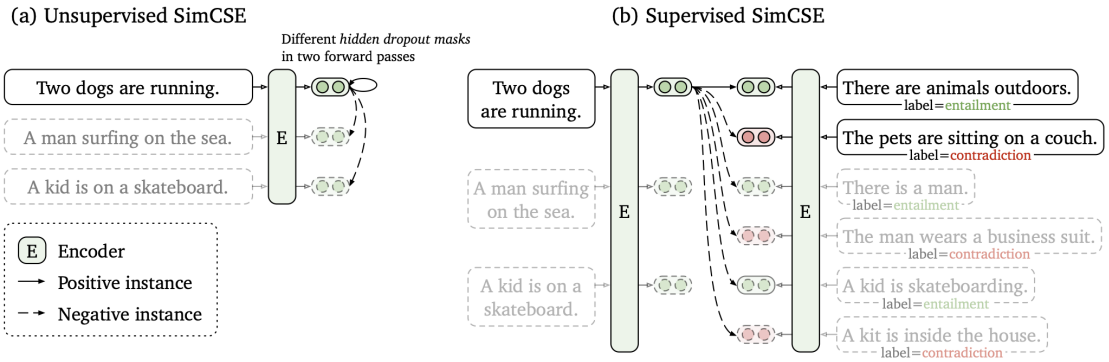


Figure 2.6: Illustration from the SimSCE paper. (a): positive samples from "different hidden dropout masks" on the same input, negatives samples through "in-batch negatives". (b): positive samples from entailment pair from NLI dataset, negatives samples from contradiction pair. (picture taken from Github Princeton-NLP)

strong performance on the SemEVAL benchmark task with performances above 80%. All of these benchmark similarity tasks depend on hand-labeled data from human annotators based on the question of how similar two texts are on a scale from 1 (completely dissimilar) to 10 (completely similar).

However, previous papers have emphasized that this notion of general text similarity is ill-defined ([27], [3], [28], [29]). [28] argues that "...text similarity is often used as an umbrella term covering quite different phenomena. [...] "How similar are two texts?" cannot be answered independently by asking what properties make them similar" [28, s. 1]. Many different aspects can be considered when deciding to which degree texts are similar, such as style, structure, etc. Consequently, the human-annotated SemEVAL dataset introduces a lot of subjectivity to what properties are being evaluated. In the following sections, we use the word "aspect" as a concept of representing different underlying properties of a text. We could not find a generally agreed-upon definition in the literature. Instead, we found that different terminology is used across different domains, such as "facets" or "features" [30] [31] [32].

2.4 Aspect-Based Similarity

Ostendorff et al. (2022) [3] emphasize the problem of general (or generic) text similarity in the context of Literature Recommender Systems (LRS). They show that most recommender systems in the academic literature are content-based recommender sys-

tems, meaning that they make recommendations solely based on the similarity of content to a given seed document. Since the assessment of text similarity underlies the abovementioned problem of "ill-defined text similarity", LRSs are a good candidate for the development of an aspect-based text similarity model. In particular, they argue that aspect-based recommendations "could help to tailor recommendations for specific information needs and increasing their diversity. Especially in the scientific domain, this could help burst filter bubbles or facilitate discoveries" [3].

Ostendorff et al. (2022) evaluate several methods for generating aspect-specific embeddings for the three aspects "task", "dataset", "method" from the PapersWithCode Corpus (3). Given paper abstracts from more than 150k papers, they train a model to project abstracts with the same aspect labels (e.g. dataset: ImageNet) to close positions in the embedding space, thereby effectively allowing a recommender system to choose the most similar papers w.r.t. to a specific aspect (as is illustrated in 3.2).

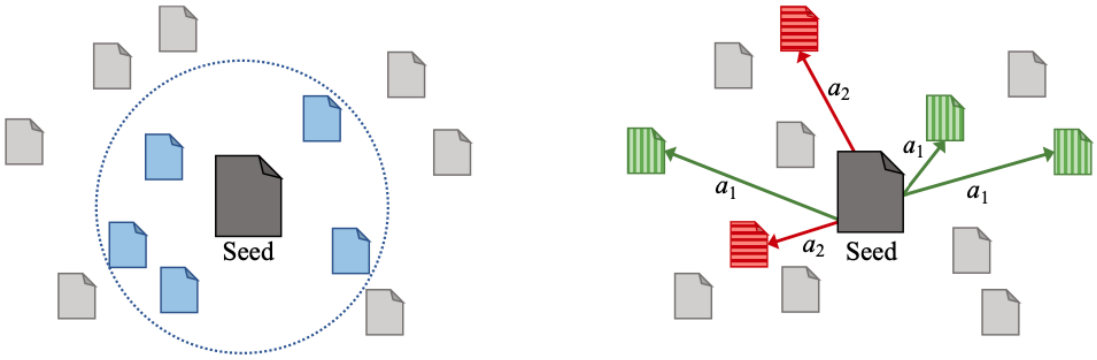


Figure 2.7: Visual Comparison of Generic Similarity (left) vs aspect-based Similarity (right) (picture taken from Github Malteos)

The paper compares several generic embedding methods, including Avg. FastText, SPECTER embeddings, SciBERT embeddings against specialized embeddings. For the aspect specialization, they experiment with Retrofitting, Joined Learning, and Fine-Tuning. The different embeddings are then compared against each other on information retrieval tasks, where the 10 most similar papers are retrieved for each paper and assessed if they share the same aspect with the seed paper. The experiments show that Fine-tuned Siamese Networks based on SciBERT outperform all other embedding methods for all aspects. This shows that Siamese Networks are a good candidate for the development of aspects-based text similarity models.

2.5 Wikidata as Datasource for Aspects

In earlier work, Ostendorff et al. (2020) [27] showed an example of how to use Wikidata (WD) [33] as a data source for aspect classification. They used the properties (relations) from the Wikipedia Knowledge Graph to label pairs of Wikipedia articles according to the relationship between them (e.g. *country of citizenship*, *educated at*, or *employer*). Afterward, they trained a multi-class classifier to predict that relationship from a given set of texts. In their experiments, Ostendorff et al. (2020) used the relationship between two entities as "aspects". However, there are many other ways to use Wikidata as a data source for aspects, such as using the properties of the Wikidata graph to label each Wikipedia page by the properties it has in the graph.

In particular, Wikidata includes a taxonomy how items are related to each other. For example, the property *instance_of* (P31) connects a member of a class to the corresponding class (e.g. "Bill Gates" is an instance of a "Human"). Also, the property *subclass_of* (P279) relates an entity to its corresponding type. By using the SPARQL endpoint from Wikidata, we can retrieve items of a certain type of class to find samples for a particular aspect.

3 Datasets

We used the following two datasets for our experiments. The PapersWithCode [34] corpus was used for evaluating different methods for training embeddings and was compared against the benchmarks from [3]. Finally, the Wikimedia dataset, including a preprocessed version of Wikipedia and Wikidata, was used for building custom training datasets for the exploration task.

3.1 Papers With Code Corpus

The Papers With Code corpus is a publicly available dataset, that is provided by the Chan Zuckerberg initiative under the CC-BY-SA license. The corpus contains a large collection of annotated papers from the academic community, including the most popular papers in the field of computer science. The corpus is available in the form of a set of dump files, which include the paper's metadata as well as labeled information about the used *tasks*, *methods*, and *datasets* that were applied in the paper. In the following, we refer to these labeled categories as aspects and their values as *aspectlabels*, or simply as *labels*. The dump files are openly available via [1] [2] [3].

We derived the code from [3] to aggregate the information from the three data dumps into a single file. The repository is available at Github[4]. The annotated data about the used task, and method are taken from the JSON files from *papers-with-abstracts.json.gz* and *methods.json.gz* respectively and mapped the aspect labels to the correct paper using the unique *paper_id* identifier. Finally, the metadata about the used datasets was collected by analyzing the evaluation metrics behind the *evaluation-tables.json.gz* file. The first column of those evaluation tables contains the corresponding dataset while the remaining columns generally contain corresponding scores. For our purpose, only the dataset names are of interest for our experiments, as they represent our aspect labels.

[1] https://paperswithcode.com/media/about/papers-with-abstracts.json.gz
[2] https://paperswithcode.com/media/about/evaluation-tables.json.gz
[3] https://paperswithcode.com/media/about/methods.json.gz
[4] https://github.com/malteos/aspect-document-embeddings

After aggregating the paper metadata, we obtain a file with 281.996 lines. Analog to how [3] describes, we apply additional data filtering by removing papers that belong to aspect labels with more than 100 instances (e.g. the label "Softmax" from the aspect "method" alone is assigned to more than 6,000 papers).

3.1.1 Dataset Statistics

Our filtered PWC dataset contains 45.918 unique papers. This number is different from the number of papers that were used in the original paper from [3] as the latter used a version of PWC from 2020-10-27 while we use a more recent version from 2022-05-25.

in Figure 3.1 we plot the frequency of aspect labels by aspect. The aspect "dataset" has the most labels but has the fewest number of labeled papers as it is shown in Figure 3.2. In general, the filtered PWC dataset is very imbalanced as there are more than four times more labeled papers for the aspect *task* than for the aspect *dataset*. This imbalance does not pose a problem for our further experiments as we experiment with smaller sub-selections of that dataset.

We keep only abstracts with a minimum of 100 characters abstracts so that our final selection of abstracts has an average character length of 1200 with a standard distribution of 352. This range is suitable for our further work with Transformers, which use a maximum token length of 512.

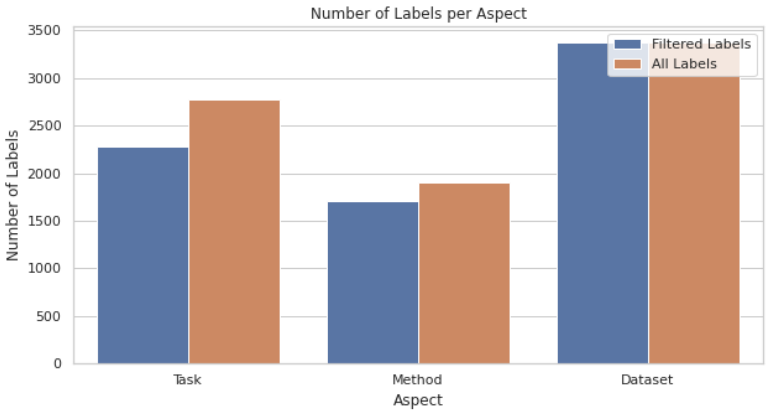


Figure 3.1: Distribution of the number of labels per Aspect

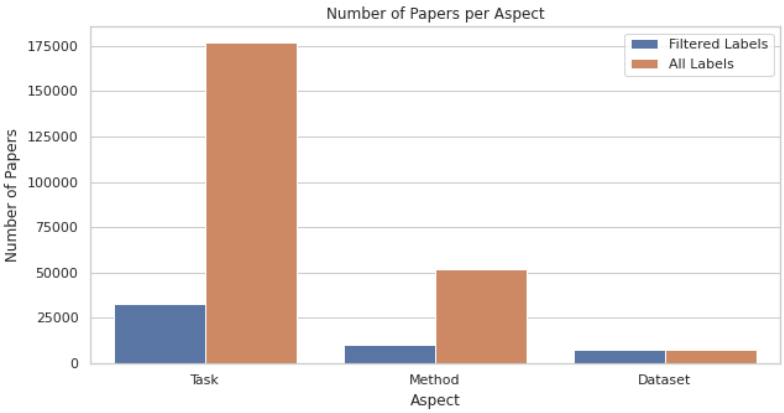


Figure 3.2: Distribution of the number of papers per aspect

Task	**Method**	**Dataset**
Self-Supervised Learning, Machine Translation, Causal Inference, Dimensionality Reduction, Video Summarization	SVM, 3D Convolution, Max Pooling, Autoencoder, ADAM, SGD	ImageNet, Penn Treebank, Cifar-10, Atari 2600 Skiing, Visual7w, COCO

Table 3.1: Example Labels by Aspect

Note about Differences

It is important to note that we could not reproduce the distribution of ground truth labels from [3]. In our experiments, we retrieve several papers per aspect label which is on average more than one order of magnitude away from the original paper. This difference cannot be accounted for by the different versions of the PWC dataset. We assume that the numbers in the original paper are mistaken as the number of labels for the aspect "method" (788) multiplied by the average number of papers per label (12.4) is not identical to the total number of papers for that aspect (given 108,687, should be 9771).

	Task	Method	Dataset
This Work	32,873	10,213	7,305
Ostendoff [3]	154,350	108,687	37,604

Table 3.2: Comparison of the number of papers in the original dataset and our dataset

	#Elements
Wikidata Items	51,450,317
Wikidata Statements	141,206,854
Wikipedia Pages	5,362,175

Table 3.3: Statistics about Wikipedia and Wikidata

3.2 Kensho Derived Wikimedia Dataset

For our second phase of experiments, we used both Wikipedia and Wikidata([33]) for composing aspect-specific datasets. Both Wikipedia and Wikidata are provided as data dumps. As working with Wikipedia requires a substantial amount of data preprocessing and data cleaning, we used a preprocessed version of both datasets through the Kensho Derived Wikimedia Dataset (KDWMD) [35]. The dataset was created based on Wikipedia and Wikidata dumps from 01-12-2019 and is licensed under CC BY-SA 3.0.

The KDWMD dataset provides a filtered selection of 50 million Wikidata items, that are linked to Wikipedia articles. The dataset contains the following aspects:

- *item_id* Wikidata
- *page_id* Wikipedia
- *page_views* Wikipedia - number of visitors in December 2020

Most importantly the KWMD dataset includes a cleaned version of all Wikipedia articles including extracted links between articles as well as links between Wikidata entities and Wikipedia articles. This allows us to quickly find all in-bound and out-bounds links for any given article, which is useful for our experiments. Furthermore, it provides each section from Wikipedia as a separate text, which allows us later to select the introduction sections for our experiences. Figure 3.3 gives an illustration of how Wikipedia and Wikidata are connected. Furthermore, we provide statistics about the dataset in Table 3.3.

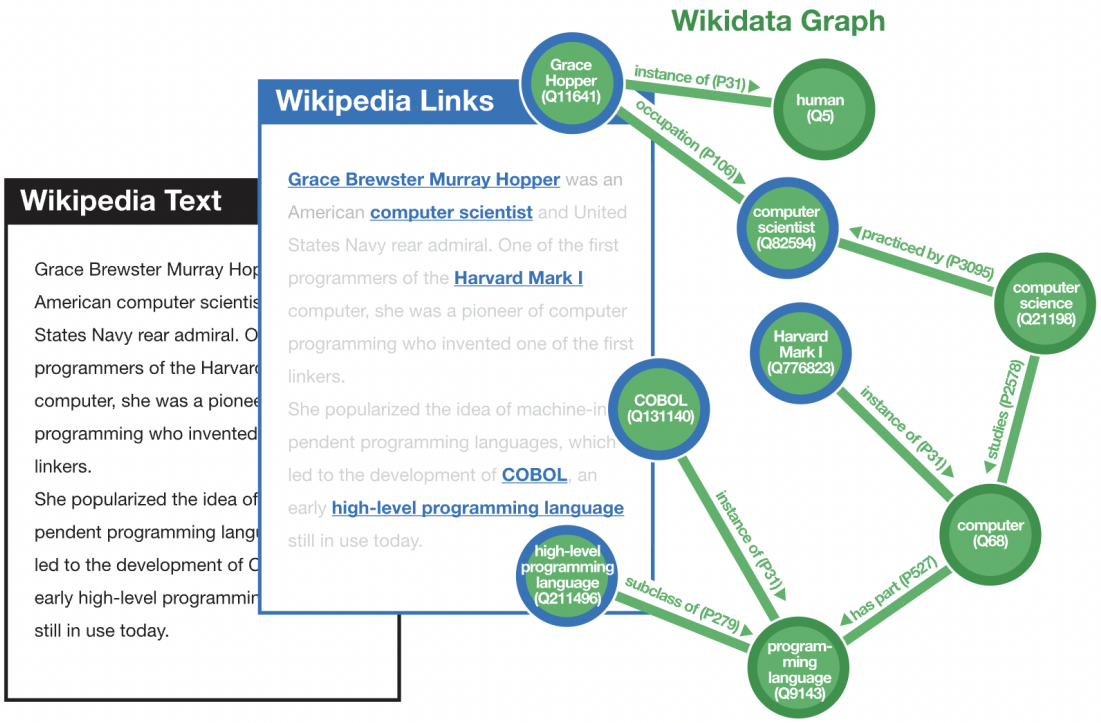


Figure 3.3: Illustration of the KDWMD dataset (taken from Kaggle Kensho-Derived-Wikimedia)

4 Methodology

This chapter describes our methods and experiments for creating aspect-based embeddings. Our methods build on previous work by Ostendorff et al. 2022 [3] who showed that SBERT can be used successfully for training aspect-specific embeddings. Furthermore, we also want to explore the capabilities of SimCSE on training aspect-specific embeddings, as SimCSE has shown state-of-the-art performance for generic document embeddings, outperforming SBERT by a large margin. Therefore, we hope to also push the performance of aspect-specific embeddings by using this more recent architecture. Since the information needs of users can be very diverse and capture many aspects, we need a scalable approach for training aspect-specific embeddings without the need for a large amount of labeled training data which is usually not available. We, therefore, explore several methods for creating embeddings using only small amounts of labeled data or using completely unsupervised approaches.

The first part of this chapter will describe the experiments with the PWC corpus. In this part, we will describe how we train aspect-based embeddings on specific subsets of the original training data using different data sampling strategies. We will then apply data augmentation as a technique to artificially increase the size of our sampled training data and show if data augmentation can help to improve the performance of small training datasets for aspect-specific embeddings.

The second part of this chapter describes the experiments with Wikipedia and Wikidata. Here, we will describe how to create a custom aspect-specific dataset by exploiting the structured information in the knowledge graph.

Both parts in combination aim to derive a new technique for creating custom aspect-specific datasets and training them to address various information needs that a user might have when exploring Wikipedia.

4.1 Models

Since we are training a model on scientific texts, we used SciBERT [14] as the based model for both SBERT and SimCSE.

4.1.1 SBERT

We reused the code from Ostendorff et al. 2022 [3] for training SBERT from their Github Repository [1]. All configurations (except Batch Size) are identical to those from the repository:

Parameter	Value
Base Model	**scibert_scivocab_uncased**
Training Epochs	3
Warmup Steps	100
Batch Size	**14**
Learning Rate	$5e-5$
Max Sequence Length	320
Pooler Type	MEAN
Tempereature for softmax	0.05

Table 4.1: Training Configuration for SBERT: All values that differ from the default values of [3] are highlighted bold

By design, SBERT accepts as training data labeled pairs of documents. The input is composed of triplets (d_1, d_2, score), where score indicates the degree of similarity, or in the case of aspect-based embeddings a binary value if d_1 and d_2 share the same aspect.

4.1.2 SimCSE

Our training code for SimCSE is based on the code from the original SimCSE repository [2] and was modified to fit our needs.

Our imposed maximum input length for all abstracts is 2000 characters (~330 words). We further set the *maximum_sequence_length* for the transformer model to 320 tokens (the original SimCSE paper works with sentences and uses a maximum of 32 tokens). Since BERT-based models use subword tokens (one word can be split up into several tokens) we are aware that some input texts will be truncated because they are too long. However, with regard to the distribution of the input length, we can see that this only concerns a small percentage of the texts. We also changed the behavior of the SimCSE model to not choose the *best model* as the final model, because the evaluation is done based on the STS benchmark which does not necessarily reflect good performance for the aspect-specific embeddings. Table 4.2 gives a summary of our training configuration for SimCSE.

[1] https://github.com/malteos/aspect-document-embeddings
[2] https://github.com/princeton-nlp/SimCSE

For the unsupervised model training, we simply pass a file with lines of text to the SimCSE model. When training the supervised model, we pass a CSV as input, containing triplets of the form (Seed Document, Positive Document, Negative Document).

Parameter	Value
Base Model	**scibert_scivocab_uncased**
Training Epochs	3
Batch Size	**14**
Learning Rate	$5e-5$
Max Sequence Length	**320**
Pooler Type	CLS
Select Best Model after Training	**False**
Tempereature for softmax	0.05
Floating Precision	16

Table 4.2: Training Configuration for SimCSE: All values that differ from the default values are highlighted in bold

4.2 PWC

4.2.1 Data Sampling

As our goal is to be able to train aspect-specific embeddings based on small amounts of data, we assess the outcome of different situations that involve having either a fraction of all aspect labels present in the dataset or having only a few instances of each label.

All Labels + K Instances

This scenario addresses a situation where all aspect labels for a given are known as well as K instances for each of them. We, therefore, sample K=3 instances for each aspect and build training triplets as described above. For 1000 aspect labels and 3 instances per label, we would therefore have 2000 training triplets (we only add positive pairs of the same items once). For our experiments, we use $K = 3$.

K Labels + K Labels

As a further constraint to the above scenario, we also experiment with the performance of aspect-based embeddings if we do not know all aspect labels beforehand and neither know all instances of the given labels. The special challenge for the model is to generalize to previously unseen aspect labels. For our experiment, we use the $K = 30$ for each aspect, which gives us $30 * 30 = 900$ total training items.

Unlabeled Data

The last scenario addresses a situation in which we have no labeled data. In this case, we simply use the entire dataset for training. Since no training item is labeled, we cannot create training triplets as it is done above. Instead, we apply **Self-Augmentation** to create positive variations of the same text and apply negative in-batch sampling. This means that we take a given un-annotated sample s and create a triplet (s, p, n) by taking a random sample n and using it as a negative (negative in-batch sample) and create an augmented variation of s as a positive $p = \text{augment}(s)$. This approach uses the data augmentation techniques described in the next section. Note, that this approach still requires knowledge about what items *have* a corresponding label for a given aspect. For example, it requires that we know that the abstract *"We used AlexNet to train on various datasets..."* mentions some method (*AlexNet*) without the need to know the exact label.

4.3 Data Augmentation

Since the sampled datasets from the previous sections are much smaller than the original dataset, they generally lead to lower performance. Therefore, in this section, we present a range of data augmentation techniques that can be used to create positive variations of a given text to add more data to the training dataset. We will use these data augmentation techniques in combination with the randomly sampled subsets of the original PWC dataset to find out if aspect-specific embeddings can be also trained when little or no data is available. We classify these augmentation strategies as **Token Level Augmentation**, **Sentence Level Augmentation**, and **Document Level Augmentation**.

4.3.1 Token Level Augmentation

Contextual Word Embedding Substitution

We use NlpAUG [22] for substituting synonyms of random words in the text. NlpAUG uses a BERT-based model to find and replace contextual similar words. We use SciBERT as the base model to further improve the quality of these replacements for

academic texts. The intuition behind the Contextual Word Embedding Substitution (CWE Substitution) augmentation strategy is to train the model on a larger variety of vocabulary and prevent the model from overfitting to very specific wording.

Contextual Word Embedding Insertion

Contextual Word Embedding Insertion (CWE Insertion) is very similar to CWE Substitution except that we insert words c instead of replacing them.

Positive Label Substitution

Positive Label Substitution (PLS) is different from the other augmentation techniques in that it works only in combination with a second labeled text. Furthermore, this technique can only be applied when the exact string match of the label is contained in both texts. The idea behind PLS is to create another positive sample by replacing the negative label in a negative sample with the positive label and thereby give more relevance to the aspect label itself while diminishing the importance of the context in which the label occurs.

4.3.2 Sentence Level Augmentation

To perform operations on a sentence level, we first need to apply sentence segmentation to the text. We use the SpaCy library [36] library for this. We use model **en_core_web_md** and enable the parser, and tagger components to find the sentence boundaries. We can then obtain a list of all sentences from each text.

Sentence Shuffling

Sentence Shuffling is another popular data augmentation technique that changes the order of sentences in a text. Besides creating additional positive samples, it also challenges the model to find the relevant information across all places in the document.

Sentence Deletion

The **Sentence Deletion** augmentation technique simply deletes each sentence from the text with a probability of 50%. The goal behind this augmentation technique is to force the model to learn aspect-specific embeddings even when certain pieces of information are missing in the text.

4.3.3 Document Level Augmentation

Document Level Augmentation techniques perform context-sensitive operations on the entire text.

Summarization

The idea behind **Summarization** as an augmentation technique is to create a variation of the text that has a similar meaning to the original text but only lacks detailed information. We use the T5 transformer [37] to generate the summaries. Our implementation uses the model **t5-large** which had been pre-trained on the task of Text Summarization. In order to generate summarizations, we use the Huggingface Transformers library [38] and simply prompt our input text with "summarize: ". We configure the model to return a summary in between 30 and 100 tokens and enable early stopping to prevent the model from generating a summary that is too long.

Backtranslation

Another popular data augmentation technique is **Backtranslation**, also referred to as *Retranslation*. Our used implementation of Backtranslation performs a two-way translation of a given text, one translation from the original English version to German, and afterward from German to English. This returns a version of the original text, that is similar in content but different in wording and can therefore be used as an additional training sample. The implementation for this augmentation technique is based on NlpAUG [22] which uses the models **facebook/wmt19-en-de** for translating from English to German and **facebook/wmt19-de-en** [39] for translating from German to English.

Table 4.3 shows the results of all the above-mentioned data augmentation techniques on one data sample from the PWC dataset.

Original room classification on floor plan graphs using graph neural networks. we present our approach to improving room classification tasks on floor plan maps of buildings by representing floor plans as undirected graphs and leveraging graph neural networks to predict the room categories. rooms in the floor plans are represented as nodes in the graph with edges representing their adjacency in the map. we experiment with house-gan dataset that consists of floor plan maps in vector format and train multilayer perceptron and graph neural networks. our results show that graph neural networks, specifically graphsage and topology adaptive gcn were able to achieve accuracy of 80% and 81% respectively outperforming baseline multilayer perceptron by more than 15% margin.
Backtranslation we present our approach to improving the spatial categorization task on floor plan maps of buildings by presenting floor plans as undirected graphs and using neural graph networks to predict the spatial categories. spaces in floor plan maps are presented as nodes in the graph whose edges represent their neighborhood in the map. we experiment with a house bill set consisting of floor plan maps in vector format and multi-layered perceptrons and neural networks. our results show that neural graphics, especially graphics and topological adaptive gcn, were able to achieve an accuracy of 80% and 81% respectively by exceeding the multi-layered floor plan sketches by more than 15%.: the spatial classification on floor plan plans using neural networks shows our approach to improving the spatial categorization task on floor plan maps of buildings by presenting floor plans as neural plans and predictive graphics.
Summarization room classification task on floor plan maps of buildings using graph neural networks. rooms in the floor plans are represented as nodes in a graph with edges indicating their adjacency in map - our approach aims to improve room category prediction based on graphsage and topology adaptive gcn models

CWE Substitution room classification on floor plan graphs using graph optical networks. we present our technique to improve room classification task using floor plan maps of houses by representing floor plans as undirected graphs and leveraging graph optic networks to infer the room categories. rooms in the floor plans are represented using nodes in the graph with edges representing their neighborhood in the map. we experiment with house - gan dataset that consists of floor plan maps in vector format and train multilayer perceptron and graph neural networks. our results show that graph neural networks, specifically graphsage and topology adaptive gcn were able to achieve accuracy of 80 % and 81 % respectively outperforming baseline multilayer perceptron by more than 15 % margin
CWE Insertion room classification on floor plan graphs using graph neural networks. we present our new approach to improve room classification task on floor plan maps of buildings by simultaneously representing floor view plans as undirected graphs and leveraging graph neural networks to predict the room categories. rooms in the floor plans are represented as nodes in the graph with edges representing their adjacency in the map. we experiment with house - gan dataset that intuitively consists composed of floor plan maps in vector format and train multilayer perceptron and graph neural networks. our results show that graph neural networks, specifically and graphsage and topology adaptive gcn were able to achieve accuracy of 80 % and similarly 81 % respectively outperforming baseline multilayer perceptron by more than 15 % margin
Sentence Shuffling we experiment with house-gan dataset that consists of floor plan maps in vector format and train multilayer perceptron and graph neural networks.rooms in the floor plans are represented as nodes in the graph with edges representing their adjacency in the map.room classification on floor plan graphs using graph neural networks.our results show that graph neural networks, specifically graphsage and topology adaptive gcn were able to achieve accuracy of 80% and 81% respectively outperforming baseline multilayer perceptron by more than 15% margin. we present our approach to improve room classification task on floor plan maps of buildings by representing floor plans as undirected graphs and leveraging graph neural networks to predict the room categories.

Sentence Deletion room classification on floor plan graphs using graph neural networks. we experiment with house-gan dataset that consists of floor plan maps in vector format and train multilayer perceptron and graph neural networks. our results show that graph neural networks, specifically graphsage and topology adaptive gcn were able to achieve accuracy of 80% and 81% respectively outperforming baseline multilayer perceptron by more than 15% margin.
Positive Label Substitution (PLS) bridge to target domain by prototypical ~~contrastive learning~~ graph neural networks and label confusion: re-explore zero-shot learning for slot filling. zero-shot cross-domain slot filling alleviates the data dependence in the case of data scarcity in the target domain, which has aroused extensive research. however, as most of the existing methods do not achieve effective knowledge transfer to the target domain, they just fit the distribution of the seen slot and show poor performance on unseen slot in the target domain. to solve this, we propose a novel approach based on prototypical ~~contrastive learning~~ graph neural networks with a dynamic label confusion strategy for wgan.

Table 4.3: Example Augmentations on one paper Abstract

4.3.4 Labeling using an External Knowledge Base

Finally, we experiment with a situation where we know neither the particular aspect labels in our unlabeled dataset nor do we have annotated instances for each label. However, we know what aspects we want to observe.

In order to create a training dataset, we leverage the Wikidata knowledge base as a source of finding aspect labels, and then find corresponding instances in an unlabeled dataset using string matching (e.g. we string-match the sentence *"We used ImageNet to evaluate the performance of the model"* with the aspect label *ImageNet*).

Querying aspect labels from Wikidata only requires knowing the categories of the instances we want to find. We use the *Wikidata Query Builder*[3] to create a query for finding instance names and afterward execute them in our script. From browsing through different Wikidata entities, we present a list of relevant categories for the PWC dataset in Table 4.4.

One example query for fetching all instances for a given Wikidata item Id is given below for the item "dataset" (*Q1172284*). Here we search both for items that are connected to the given dataset item by either the *instance_of* property (P31) or by the

[3]https://query.wikidata.org/querybuilder/?uselang=de

Aspect	Wikidata Item Ids	Count Items
Task	Q11660 (Artificial Intelligence) Q30642 (NLP) Q182557 (Computational Linguistics)	673
Method	Q192776 (Artificial Neural Network) Q3621696 (Language Model)	771
Dataset	Q1172284 (Dataset)	1335

Table 4.4: Wikidata Queries for given task

subclass_of property (P279), as both properties are often used interchangeably.

```
SELECT ?child ?childLabel
WHERE
{
    VALUES ?p (wd:P31 wd:P279)
    ?child ?p wd:Q12345;
    SERVICE wikibase:label {{ bd:serviceParam wikibase:language "en". }}
}
```

For matching the labels with items, we simply match the lowercase Wikidata title with the lowercase abstract. This returns a total of ~1500 matches for all aspects. When comparing these matches with the ground truth labels, we get an average accuracy of ~25%.

As we only get a small number of pseudo-labeled items as we also use all of the above data augmentation strategies to evaluate the benefits of data augmentation on relatively inaccurately labeled data.

4.3.5 SimCSE Datasets vs SBERT

The previously described triplet structure $(\text{seed}, \text{positive}, \text{negative})$ represents the dataset structure for SimCSE and cannot be used directly for training SBERT. Instead, we need to "transform" it into the triplet structure $(\text{seed}, \text{other}, \text{other})$. We, therefore, apply the following transformation

$$(\text{seed}, \text{positive}, \text{negative}) -> [(\text{seed}, \text{positive}, 1.0), (\text{seed}, \text{negative}, 0.0)] \tag{4.1}$$

Additionally, we enforce the ratio of 2 to 1 for positive samples vs negative samples that were used for training from [3]. For this, we simply remove 50% of all negative triplets from the dataset using random sampling.

4.4 Wikipedia

For the use case of improving Knowledge Exploration on Wikipedia, we want to produce aspect-specific embeddings for Wikipedia sections.

Since Wikipedia contains a broad range of topics, there exist an enormous amount of possible aspects that could be used to generate embeddings. We therefore only work on a subset of topics by limiting our experiments to Wikipedia articles about companies. We consequently define the relevant aspects for these topics as **Industry** (*What type of product/service does the company offer?*) and **Country** (*What country is the company based in?*). Analog to embeddings for the PWC dataset, the objective of this experiment is to train a model so that texts about companies from the same industry (or from the same country) are represented close to each other in the embedding space.

4.4.1 Creating a Custom aspect-based Dataset

While the PWC dataset is manually annotated, Wikipedia is an unstructured source of texts. We, therefore, need to construct an aspect-based dataset ourselves. Similar to how we created a labeled PWC dataset from Wikidata, we use the Knowledge Graph to label our selection of Wikipedia articles. We exploit the fact that some Wikidata items link to the corresponding Wikipedia Item (e.g. [1] (Wikidata Item about Bill Gates) links to [2](Wikipedia article about Bill Gates)). We construct our dataset using the following steps:

1. Find all Wikidata items i of type *business* (Q4830453)
2. Filter items i_k that are linked to a Wikipedia article w_k
3. Find the introduction section s_k of the Wikipedia article w_k
4. Label s_k by the properties *country* (P17) and *industry* (P452) from the Wikidata Item i_k

For steps 1. and 4. we use the *Wikidata Query Builder* to create a query for finding all items and afterward map the returned items to the Wikipedia sections from the Kensho dataset [35]. The SPARQL Query for retrieving all businesses and their corresponding country and industry property looks as follows:

```
SELECT ?child ?childLabel ?country ?industry
WHERE
```

[1] `https://www.wikidata.org/wiki/Q5284`
[2] `https://en.wikipedia.org/wiki/Bill_Gates`

```
{
    {
    VALUES ?p {wd:Q6881511}
    ?child wdt:P31 ?p;
        wdt:P17 ?country;
        wdt:P452 ?industry.
    }
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

This Wikipedia "company" dataset includes ~6000 items. Since we want to compare the aspect-specific recommendation algorithm against the baseline of Wikipedia links, we also add the Wikipedia sections for all outgoing links from that company dataset. Additionally, we add 10.000 randomly selected Wikipedia articles, that serve as "training noise". In total, our Wikipedia-based company dataset contains ~45k items.

The distribution of aspects and aspect labels is highlighted in Table 4.5.

Aspect	Labels	Avg. Instances per Label	Total Items
Industry	97	27.4	6.082
Country	75	62.7	2.062
Outlinks	-	-	24.099
Random Articles	-	-	10.000

Table 4.5: Distribution of aspects and labels for our Custom Wikipedia Dataset

4.4.2 Aspect Configurations

With the custom aspect labeled dataset from Wikidata and Wikipedia, we train four aspect-specific embedding models based on the following contrastive triplets.

- **Country Specific**: $(s, p_{country}, n_{country})$: p and n are positive and negative samples w.r.t. the country aspect.
- **Industry Specific**: $(s, p_{industry}, n_{industry})$: p and n are positive and negative samples w.r.t. the industry aspect.
- **Multi Aspect Specific (Union)**: $(s, p_{country \cup industry}, n_{country \cup industry})$: p is a positive sample if it has **either** the same industry **or** the same country aspect as the seed document.

- **Multi Aspect Specific (Join)**: $(s, p_{county \cap industry}, n_{country \cap industry})$: p is positive sample if it has **both** the same industry **and** country aspect as the seed document.

5 Evaluation

In this section, we will show the results of the experiments for Data Augmentation and Labeling with Wikidata that we described in the previous section. We evaluate the performance of the aspect-based embeddings by framing the evaluation as an information retrieval task. For that, we iterate over each aspect labeled item from the test dataset and retrieve the K=10 nearest neighbors of the item. We then measure how many of the retrieved items match the particular aspect of the seed item. We use the same evaluation metrics as Ostendorff et al. [3] to quantify the results. The following gives a brief explanation of the used metrics since they are not explained in the original paper:

- **Precision@k** (P): The number of nearest neighbors (within the top k candidates) that share the same aspect as the seed document divided by k.
- **Recall@k** (R): The number of nearest neighbors (within the top k candidates) that share the same aspect as the seed document divided by the number of labeled documents with the seed document's aspect.
- **Mean Reciprocal Rank@k** (MRR): Measure of the ranking quality for the nearest neighbors, calculated by averaging the reciprocal ranks ($\frac{1}{\text{rank}}$) of each neighbor. This essentially adds more weight to correctly labeled neighbors the higher they rank.

Different from the original paper, we did not use 4-fold cross-validation but instead used a simple train/test (80/20) split. The reason for this simply lies in the extensive computational resources required for 4-fold cross-validation as is done by [3]. This requires the training of 12 embedding models (one for each aspect of each fold). As the training of one SBERT model takes around 4h on our machine (20 CPU cores + one V100 GPU), this leads to a total training time of around 48h for all folds and all three aspects. Given that we are interested in evaluating the performance of different training dataset configurations, this extensive setup becomes computationally infeasible. With our different evaluation methods in place, our results cannot be directly compared to the results from the original paper. We will therefore run the best performing model from the original paper on our evaluation setup and use the resulting score as an internal benchmark.

5.1 PapersWithCode

The following sections compare the results of different data augmentation strategies on the PWC dataset. Each Table contains the results from the **Gold** dataset (the subset from the original PWC dataset), the results from all **data augmented datasets** in combination with the **Gold** dataset as well as the Lower and Upper Bounds. The Lower Bound is given by the performance of a state-of-the-art (Generic) Text Similarity model (SimCSE trained on the NLI dataset) while the Upper Bound is given by our computed test dataset performance for the best model from the original Ostendorff paper [3] (SBERT model). For all result tables, we highlight the best score in each category in bold.

5.1.1 All Labels + K Samples

The results from Table 5.1 show that the Gold dataset was outperformed by several data augmentation strategies. Both *CWE Substitution* and *Sentence Reordering* outperformed the original dataset in all except one benchmarks. The average improvement for MRR was ~2.5% showing only a slight improvement in the overall performance. *CWE Insert* also outperformed the original dataset in 3 benchmarks but performed consistently worse than *CWE Substitution*. The remaining augmentation strategies showed lower scores for all aspects with *Positive Label Substitution* being the worst performing.

Generally, the SBERT embeddings showed consistently worse performance compared to SimCSE embeddings but displayed similar relative improvements in the above-mentioned data augmentation techniques compared to the Gold dataset. Compared so SBERT, SimCSE performed on average ~4% better on the Gold dataset for the MRR benchmark.

Aspects →	*Task*			*Method*			*Dataset*		
Methods ↓	**P**	**R**	**MRR**	**P**	**R**	**MRR**	**P**	**R**	**MRR**
Upper Bound	0.409	0.424	0.768	0.263	0.302	0.595	0.172	0.418	0.465
SimCSE									
Gold	0.290	0.332	0.626	0.196	0.238	0.488	0.173	0.438	0.509
backtranslation	0.288	0.332	0.625	0.172	0.214	0.433	0.164	0.427	0.514
CWE Insertion	0.295	0.334	0.631	0.178	0.218	0.449	0.169	0.432	0.509
CWE Substitution	0.295	0.333	0.631	0.201	0.241	0.501	0.174	0.447	0.522
PLS	0.246	0.291	0.557	0.127	0.158	0.317	0.173	0.438	0.516
Sentence Deletion	0.284	0.327	0.613	0.170	0.211	0.443	0.171	0.446	0.513
Sentence Reordering	0.300	0.341	0.641	0.200	0.242	0.500	0.170	0.441	0.510
Summarization	0.278	0.322	0.604	0.172	0.216	0.435	0.169	0.440	0.513
SBERT									
Gold	0.221	0.253	0.515	0.095	0.115	0.266	0.138	0.364	0.420
backtranslation	0.198	0.222	0.520	0.142	0.180	0.345	0.130	0.338	0.414
CWE Insertion	0.234	0.275	0.541	0.105	0.130	0.302	0.141	0.365	0.422
CWE Substitution	0.235	0.270	0.543	0.130	0.181	0.381	0.140	0.365	0.425
PLS	0.177	0.214	0.451	0.069	0.088	0.209	0.126	0.348	0.420
Sentence Deletion	0.215	0.241	0.508	0.117	0.146	0.323	0.119	0.341	0.402
Sentence Reordering	0.222	0.255	0.528	0.128	0.166	0.370	0.129	0.345	0.424
Summarization	0.230	0.269	0.536	0.162	0.200	0.335	0.136	0.360	0.425
Lower Bound	0.089	0.152	0.048	0.400	0.039	0.124	0.119	0.316	0.072

Table 5.1: All Labels + Few Instances: Training results for a sampled dataset composed of all available labels for each aspect, but containing only 3 instances per label.

5.1.2 K Labels + K Instances

Table 5.2 shows the results from training embedding models based on 30 labels à 30 samples. The scores from both SimCSE and SBERT indicate clearly that all data augmentation strategies lead to lower scores for all aspects compared to the *Gold* dataset. The consistently worst results were produced by *Positive Keyword Substitution* and *Sentence Deletion* while *CWE Substitution* performed "best" among the data augmentation strategies. We can also see that the dataset performs only slightly better than the Lower Bound based on Generic Text Similarity, showing an overall weak result for the aspect-specific retrieval task.

Surprisingly, all SBERT embeddings for the Gold dataset perform on average less than 50% for all benchmark scores compared to SimCSE.

Aspects →	*Task*			*Method*			*Dataset*		
Methods ↓	**P**	**R**	**MRR**	**P**	**R**	**MRR**	**P**	**R**	**MRR**
Upper Bound	0.409	0.424	0.768	0.263	0.302	0.595	0.172	0.418	0.465
SimCSE									
Gold	0.168	0.198	0.429	0.089	0.109	0.228	0.135	0.346	0.391
Backtranslation	0.120	0.133	0.335	0.068	0.080	0.191	0.131	0.324	0.382
Summarization	0.119	0.139	0.332	0.066	0.084	0.190	0.128	0.323	0.388
CWE Substitution	0.128	0.149	0.342	0.077	0.099	0.223	0.130	0.332	0.393
CWE Insertion	0.142	0.164	0.368	0.071	0.098	0.211	0.124	0.319	0.368
PLS	0.096	0.111	0.282	0.062	0.082	0.183	0.115	0.302	0.359
Sentence Deletion	0.113	0.134	0.309	0.067	0.077	0.172	0.119	0.303	0.353
Sentence Reordering	0.124	0.144	0.352	0.078	0.088	0.212	0.115	0.288	0.326
SBERT									
Gold	0.053	0.063	0.189	0.024	0.029	0.084	0.006	0.015	0.013
CWE Insertion	0.066	0.078	0.229	0.032	0.044	0.108	0.051	0.129	0.180
CWE Substitution	0.007	0.007	0.018	0.028	0.033	0.104	0.004	0.011	0.013
PLS	0.004	0.005	0.015	0.030	0.038	0.112	0.048	0.143	0.175
Sentence Reordering	0.068	0.081	0.237	0.032	0.044	0.125	0.063	0.171	0.224
Summarization	0.072	0.088	0.241	0.022	0.027	0.086	0.058	0.153	0.215
Lower Bound	0.089	0.152	0.048	0.400	0.039	0.124	0.119	0.316	0.072

Table 5.2: K Labels + K Instances: Training results for a sampled dataset composed of only 30 labels and 30 instances per label for each aspect.

5.1.3 Unlabeled Self Augmentation

For the unsupervised self-augmentation task, we show the results of all augmentation strategies in Figure 5.3. Different from the previous results, we do not show results for the augmentation type *Positive Label Substitution* and the *Gold Dataset*, because these two datasets require a labeled dataset. The results show surprisingly strong results given that the embedding was only trained completely without labels. The strongest score was obtained by *Summarization* augmentation on the aspect *task* with an MRR score of 0.417. For the remaining aspects *Sentence Deletion* shows the best results with a MRR score of 0.250 for the aspect *method* and 0.327 for the aspect *dataset*. Overall *Summarization* and *Sentence Deletion* lead to the best results with *Backtranslation* following in the third place, yet being consistently outperformed on all tasks by the two winning data augmentation strategies. Surprisingly, *CWE Substitution* produced comparatively weak results, given its dominance in previous experiments.

Again, we found that SBERT produced lower scores on almost all tasks compared to SimCSE but displayed a very similar pattern of improvement for data augmentation. We only found one data point for the SBERT experiments slightly outperforming the corresponding SimCSE score (Recall score for *Sentence Deletion* on the aspect "method" - 0.102 vs 0.101).

Aspects →	*Task*			*Method*			*Dataset*		
Methods ↓	**P**	**R**	**MRR**	**P**	**R**	**MRR**	**P**	**R**	**MRR**
Upper Bound	0.409	0.424	0.768	0.263	0.302	0.595	0.172	0.418	0.465
SimCSE									
Backtranslation	0.147	0.181	0.390	0.079	0.104	0.241	0.083	0.236	0.301
CWE Insertion	0.073	0.086	0.230	0.067	0.082	0.199	0.039	0.120	0.148
CWE Substitution	0.096	0.114	0.286	0.065	0.078	0.208	0.060	0.185	0.240
Sentence Deletion	0.149	0.179	0.400	0.080	0.101	0.250	0.090	0.261	0.327
Sentence Reordering	0.088	0.109	0.270	0.059	0.077	0.192	0.062	0.177	0.208
Summarization	0.157	0.189	0.417	0.082	0.105	0.249	0.085	0.256	0.319
SBERT									
Backtranslation	0.097	0.101	0.220	0.059	0.094	0.200	0.063	0.199	0.202
CWE Insertion	0.111	0.135	0.242	0.035	0.044	0.125	0.060	0.179	0.243
CWE Substitution	0.086	0.109	0.245	0.056	0.072	0.198	0.055	0.181	0.210
Sentence Deletion	0.139	0.150	0.340	0.077	0.102	0.220	0.080	0.251	0.301
Sentence Reordering	0.088	0.092	0.245	0.057	0.069	0.191	0.006	0.012	0.015
Summarization	0.126	0.148	0.336	0.078	0.097	0.186	0.072	0.224	0.292
Lower Bound	0.089	0.152	0.048	0.400	0.039	0.124	0.119	0.316	0.072

Table 5.3: Self Augmentation: Results for a generated training dataset. We applied various augmentation strategies to create positive samples for each given (unlabeled) item and used in-batch negatives to create negative samples.

5.1.4 Wikidata Labeling

The results from the evaluations for the Wikidata-labeled PWC dataset are displayed in 5.4. In this section, the Gold dataset is composed of items from the PWC dataset that were labeled based on string matches with Wikidata items as described in section 4. Therefore, some annotated items are not correct. While we trained embeddings based on these imperfectly labeled items, we evaluate the performance based on the ground truth test dataset that contains correctly labeled items from the PWC dataset. The results show generally low performance for all benchmark scores compared to the generic similarity model. Also, most data augmentations are not effective on the Gold dataset, outperforming the Gold dataset only for 3 benchmark scores by a small margin. Surprisingly, *Positive Label Substitution* outperforms the Gold dataset on one benchmark (MRR method), while generally representing the weakest augmentation method for all other experiments.

Aspects →	*Task*			*Method*			*Dataset*		
Methods ↓	**P**	**R**	**MRR**	**P**	**R**	**MRR**	**P**	**R**	**MRR**
Upper Bound	0.409	0.424	0.768	0.263	0.302	0.595	0.172	0.418	0.465
Gold	0.128	0.042	0.265	0.058	0.016	0.131	0.072	0.096	0.217
Backtranslation	0.074	0.022	0.180	0.033	0.008	0.096	0.064	0.089	0.190
CWE Insertion	0.083	0.026	0.205	0.023	0.007	0.059	0.026	0.034	0.099
CWE Substitution	0.067	0.020	0.151	0.055	0.015	0.126	0.053	0.068	0.166
PLS	0.082	0.026	0.208	0.057	0.014	0.133	0.068	0.100	0.188
Sentence Deletion	0.088	0.027	0.212	0.034	0.008	0.092	0.068	0.094	0.189
Sentence Reordering	0.077	0.023	0.172	0.023	0.006	0.072	0.044	0.051	0.134
Summarization	0.096	0.031	0.230	0.025	0.007	0.082	0.076	0.126	0.207
Lower Bound	0.089	0.152	0.048	0.400	0.039	0.124	0.119	0.316	0.072

Table 5.4: Wikidata + PWC: Training results for a Wikidata labeled PWC dataset. We used Wikidata items as labels and annotated items from the PWC corpus using string matching.

5.1.5 Complete Dataset

Finally, we show the results for aspect-specific embeddings based on the complete PWC (Gold) training dataset and a selection of augmentation strategies that worked best in the previous experiments (5.5). We find that the Gold dataset outperforms all applied data augmentation strategies for the aspects "task" and "method" while *CWE Substitution*, *Summarization*, and *Sentence Deletion* outperform the Gold standard for the "dataset" aspect with *Summarization* showing the best results on average.

Also, we find that the trained embeddings from SBERT produce generally lower scores than the SimSCE embeddings and did not show any improvements for the augmented training datasets.

Aspects $\rightarrow$	*Task*			*Method*			*Dataset*		
Methods $\downarrow$	**P**	**R**	**MRR**	**P**	**R**	**MRR**	**P**	**R**	**MRR**
SimCSE									
Gold	0.416	0.431	0.776	0.268	0.312	0.606	0.186	0.461	0.507
CWE Substitution	0.400	0.417	0.759	0.262	0.303	0.586	0.186	0.455	0.515
Summarization	0.396	0.416	0.742	0.260	0.297	0.585	0.188	0.463	0.541
Sentence Deletion	0.394	0.410	0.711	0.261	0.291	0.581	0.188	0.466	0.528
SBERT									
Gold*	0.409	0.424	0.768	0.263	0.302	0.595	0.172	0.418	0.465
CWE Substitution	0.407	0.419	0.751	0.262	0.301	0.592	0.170	0.411	0.461
Summarization	0.396	0.416	0.742	0.244	0.285	0.566	0.171	0.416	0.463
Sentence Deletion	0.395	0.411	0.736	0.244	0.284	0.556	0.168	0.412	0.458
Lower Bound	0.089	0.152	0.048	0.400	0.039	0.124	0.119	0.316	0.072

Table 5.5: Complete Dataset: Results on the full PWC train dataset. (*) indicates the results from the original approach proposed by Ostendorff [3]

5.1.6 2D Embedding Space

In Figure 5.1 we visualized selected document embeddings that have been labeled by the aspect "method". To plot the embeddings we used TSNE to reduce the dimensionality of the embeddings from the original 768 dimensions to 2 dimensions. We furthermore colored all data points according to their aspect label. By definition, dimensionality reduction removes information from the original data, but TSNE optimizes the spatial representation of the data in a way that keeps those embeddings close in dimensionality reduced space which are close in high dimensional space and keeps those embeddings far away from each other in dimensionality reduced space which are far away in high dimensional space. Therefore, the visualized embeddings in Figure 5.1 approximate the positions from the original embeddings.

We plot a total of 200 embeddings from 20 aspect labels. (1) shows embeddings from a state-of-the-art text similarity model (SimCSE + NLI), (2) uses the best performing model from the *K Labels + K Instances* experiment, (3) uses the best performing model from the *All Labels + Few Instances* experiment and (4) uses the best performing model from the *Complete Dataset* experiment. The plots (1)-(4) are ordered in ascending order by the corresponding Mean Reciprocal Rank (MRR) score. What can be seen visually is a strong correlation between the separability of data points with the same color and the MRR score. While in (1) we cannot see a visual separation of items by color, (4) shows separated patches of data points with the same color. Since color encodes the aspect label, this means that the embedding space in (4) has learned a representation that keeps items with the same label close in the embedding space, while (1) does not consider the individual aspect.

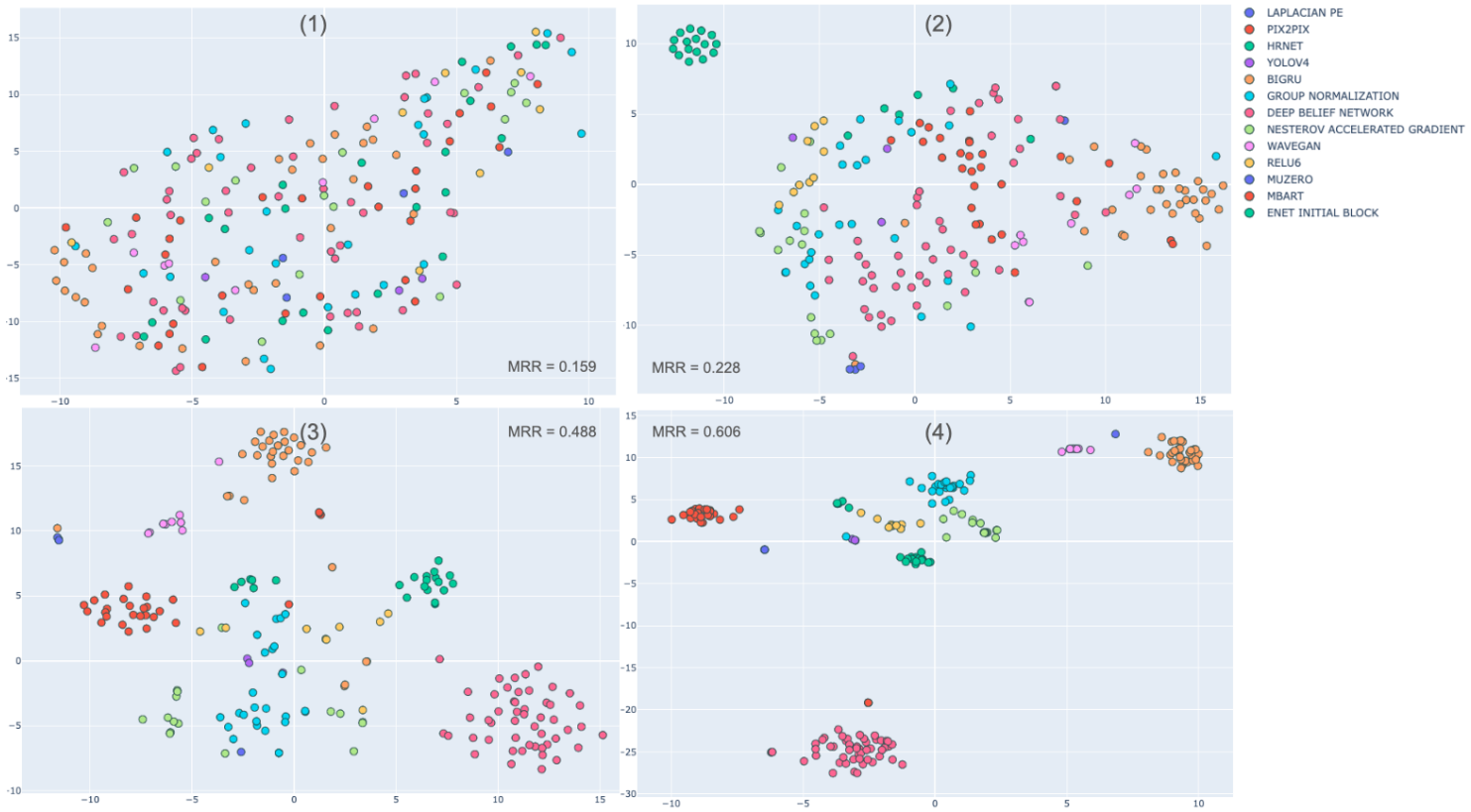


Figure 5.1: Comparison of four different embedding models, showing 200 document embeddings for 20 different aspect labels for the category "method" in 2D space. (1) Generic Embedding (SimCSE + NLI), (2) aspect-specific Embedding trained on 3ß Labels + 30 Instances per label, (3) aspect-specific Embedding trained on All Labels + 3 Instances per label, (4) aspect-specific Embedding trained on the complete training dataset.

5.2 Wikipedia

5.2.1 Evaluation

We evaluate the results of the embeddings using the same metrics as for the PWC dataset (Precision, Recall, Mean Reciprocal Rank). Similar to the PWC experiments, we train the embeddings on the training dataset and evaluate the test dataset using an 80/20 split. The code for calculating scores was adapted from Ostendorff and modified to work with different aspects.

Different from the previous experiments, we do not provide an Upper Bound for our experiments, as there are no existing results available for this dataset. Yet we provide a Lower Bound which indicates the retrieval performance for generic text similarity (supervised SimCSE model [23]).

5.2.2 Results

The results for the Wikipedia Test dataset are shown in Table 5.6. All three training configurations show very strong results for both aspects. While we trained two embedding models for the individual aspect case (one embedding model for aspect country, one for aspect industry), the multi-aspect models are trained on both aspects at the same time. Therefore, one model is used to retrieve the most similar items for both aspects. Surprisingly the best (MRR) score for the aspect country **and** aspect industry were achieved by the multi-aspect (Union) model, outperforming the single aspect models.

Table 5.6: Wiki Evaluation

Aspects $\rightarrow$	*Country*			*Industry*		
Methods $\downarrow$	**P**	**R**	**MRR**	**P**	**R**	**MRR**
Single Aspect	0.390	<u>0.124</u>	0.558	<u>0.625</u>	<u>0.178</u>	0.729
Multi Aspect (Join)	0.444	0.102	0.593	0.622	0.174	0.720
Multi Aspect (Union)	<u>0.555</u>	0.163	<u>0.738</u>	0.538	0.155	<u>0.747</u>
Lower Bound	0.089	0.152	0.048	0.400	0.039	0.124

5.2.3 2D Embedding Space

Similar to section 5.1.6, we visualize the embedding space of randomly sampled Wikipedia articles from our company dataset. Figures 5.2 and 5.3 compare the 2D

embedding spaces of generic text embeddings and aspect-specific text embeddings for the aspects "country" and "industry" for ~1800 items. For both aspects, we can see that the generic text embeddings weakly capture both target aspects, because certain colors (aspect labels) visually dominate certain regions. Yet, there is no clear separation between aspect labels, and we find many colors scattered across the whole space. On the other hand, we find a sharp separation between aspect labels for the specialized embedding space with dense clusters of items with the same aspect label. This finding is consistent with the high scores for the information retrieval task that we showed in the previous section (5.2.2).

We furthermore visually analyzed the local neighborhoods of the aspect-specific embedding spaces (2D) and found that not only items of the same aspect label are close to each other, but also items of semantically similar aspect labels have a smaller distance to each other than semantically dissimilar aspect labels. For example, we found that for country-specific embeddings, items from African countries (Ghana, Egypt, Mali, etc.) share a local neighborhood. The same holds for Arab countries (Saudi Arabia, Bahrain, etc.) as well as South American countries (Trinidad and Tobago, Barbados, etc.), and many more. For industry-specific embeddings, we can also see that the semantic similarity between industries is reflected in the distance of items. For example, we find that items for the labels "Film Industry", "Music Industry", and "Radio Broadcasting" are close to each other. Also, "Rail Transport" and "Maritime Transport" are located in the same neighborhood. We show a selection of these examples in Figure 5.4. Even though we found many instances with a correlation between semantic similarity and local neighborhood of aspect labels, this association is not consistent for all aspect labels. For example, we also found that the items for the aspect label "Austria" are closer to the items from "Japan" than to items from "Germany".

Finally, we also visually analyzed the multi-aspect (Union) embedding space that is shown in Figure 5.5. Bear in mind that these embeddings were trained in a way that keeps items close to each other that share either the same industry or the same country or both aspects. In Figure 5.5 we only color-code the industry aspect. What we find is that the industry aspect is the more dominant aspect for the spatial positioning of items. This can be seen by the homogeneity of the local neighborhoods that mostly contain items of the same color (industry aspect). At the same time, the country aspect determines the spatial positioning of items within the "industry cluster". As highlighted in the figure, the items for "Automotive Industry" are sub-clustered into different patches. We find that different patches for different countries are present (Italy, USA, China, etc.). Furthermore, we observed that items at the boundary between industries are likely to share the same country aspect, as can be seen between the "Automotive Industry (China)" items and the "Consumer Electronics (China)" items that lie next to each other. This result highlights that aspect-based embeddings can be trained on

multiple aspects at the same time while still yielding meaningful results.

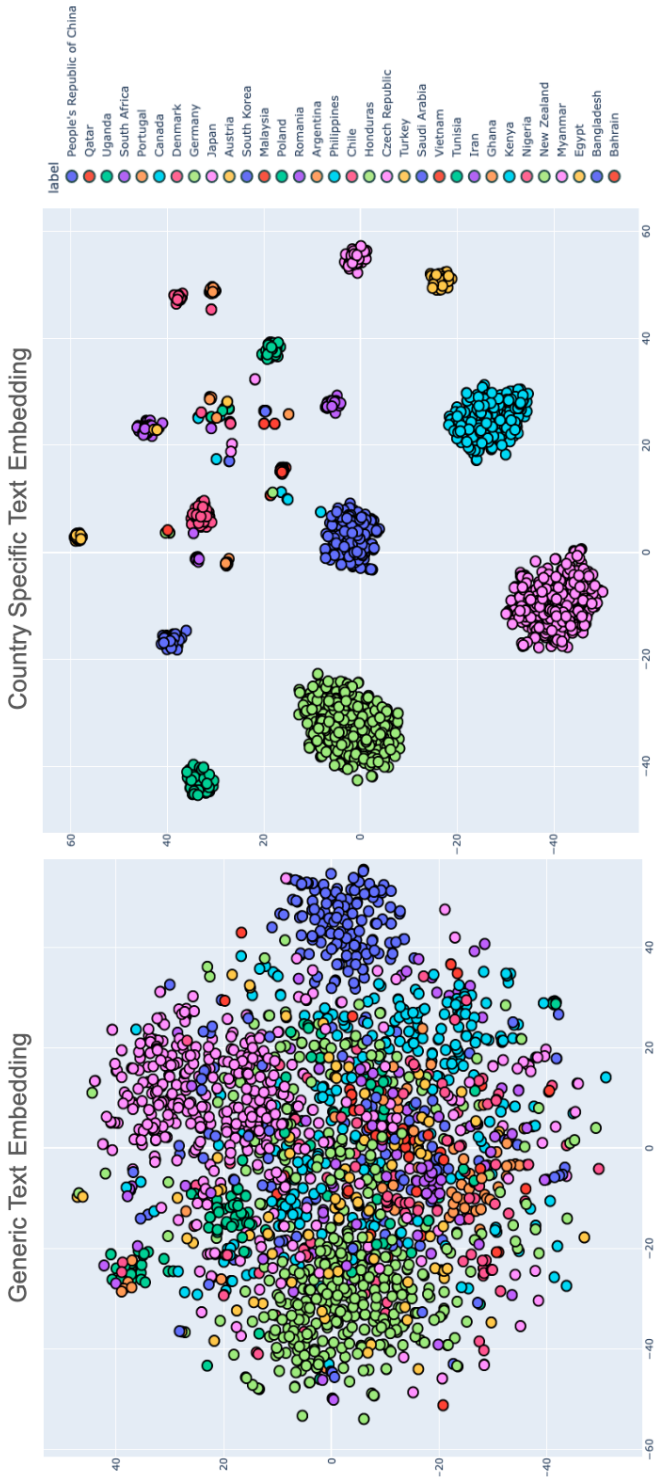


Figure 5.2: Comparison Generic Text Embedding (Left) vs. Country Specific Embedding (Right).

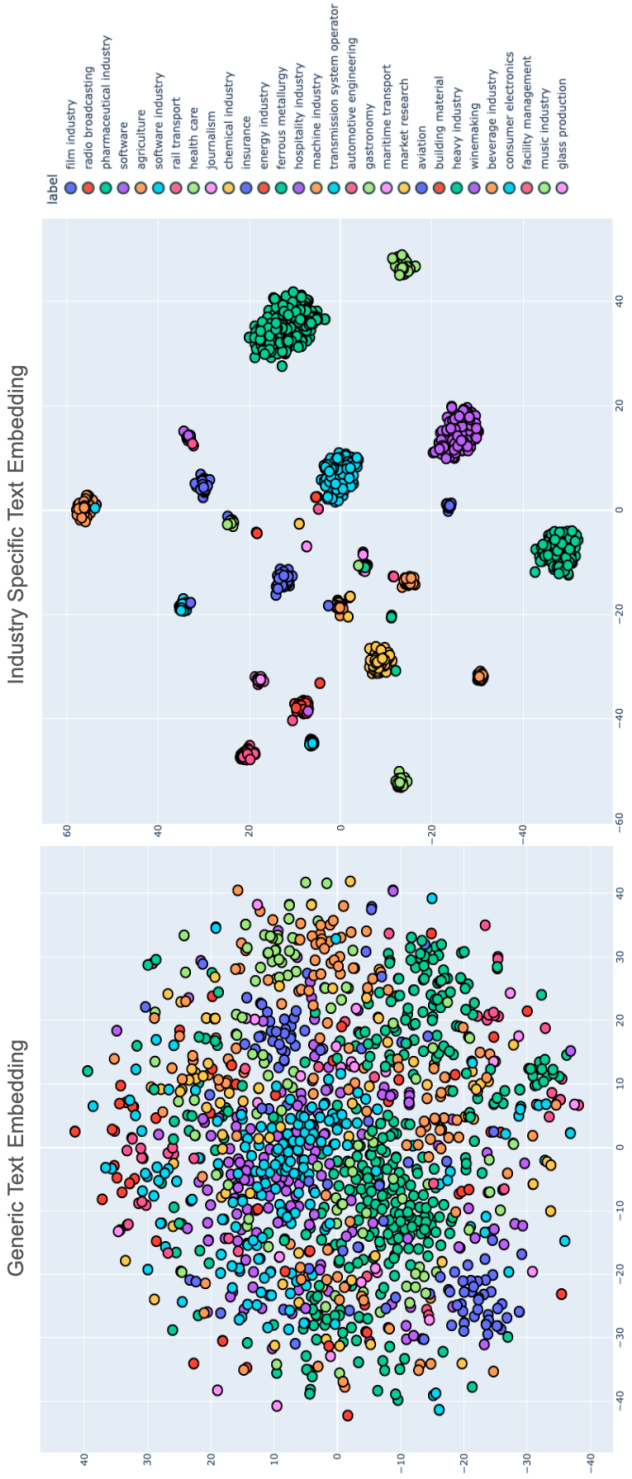
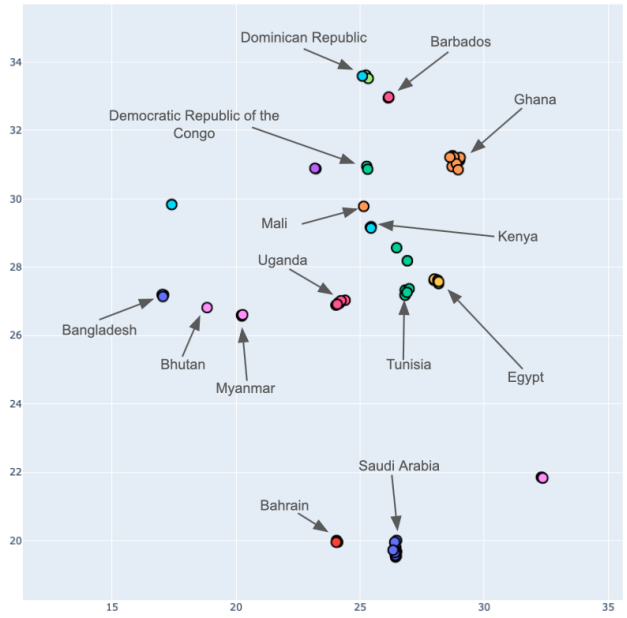
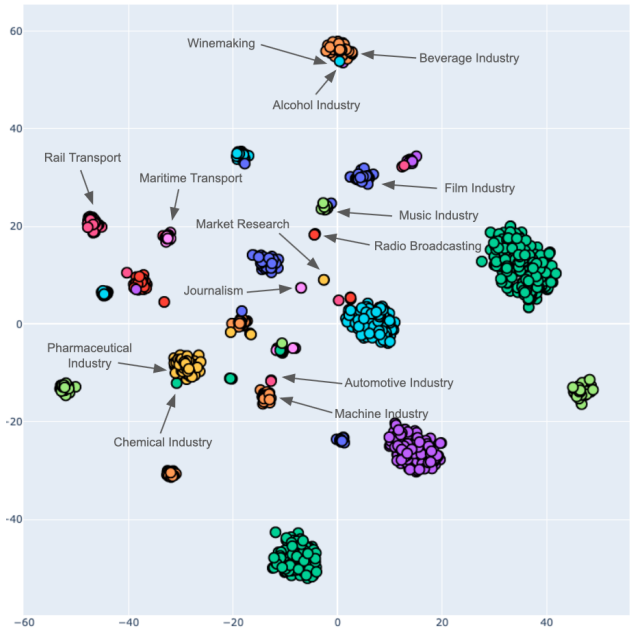


Figure 5.3: Comparison Generic Text Embedding (Left) vs. Industry Specific Text Embedding (Right).



(a) Local 2D Embedding Space for Country-Specific Text Embeddings



(b) Global 2D Embedding Space for Industry-Specific Text Embeddings

Figure 5.4

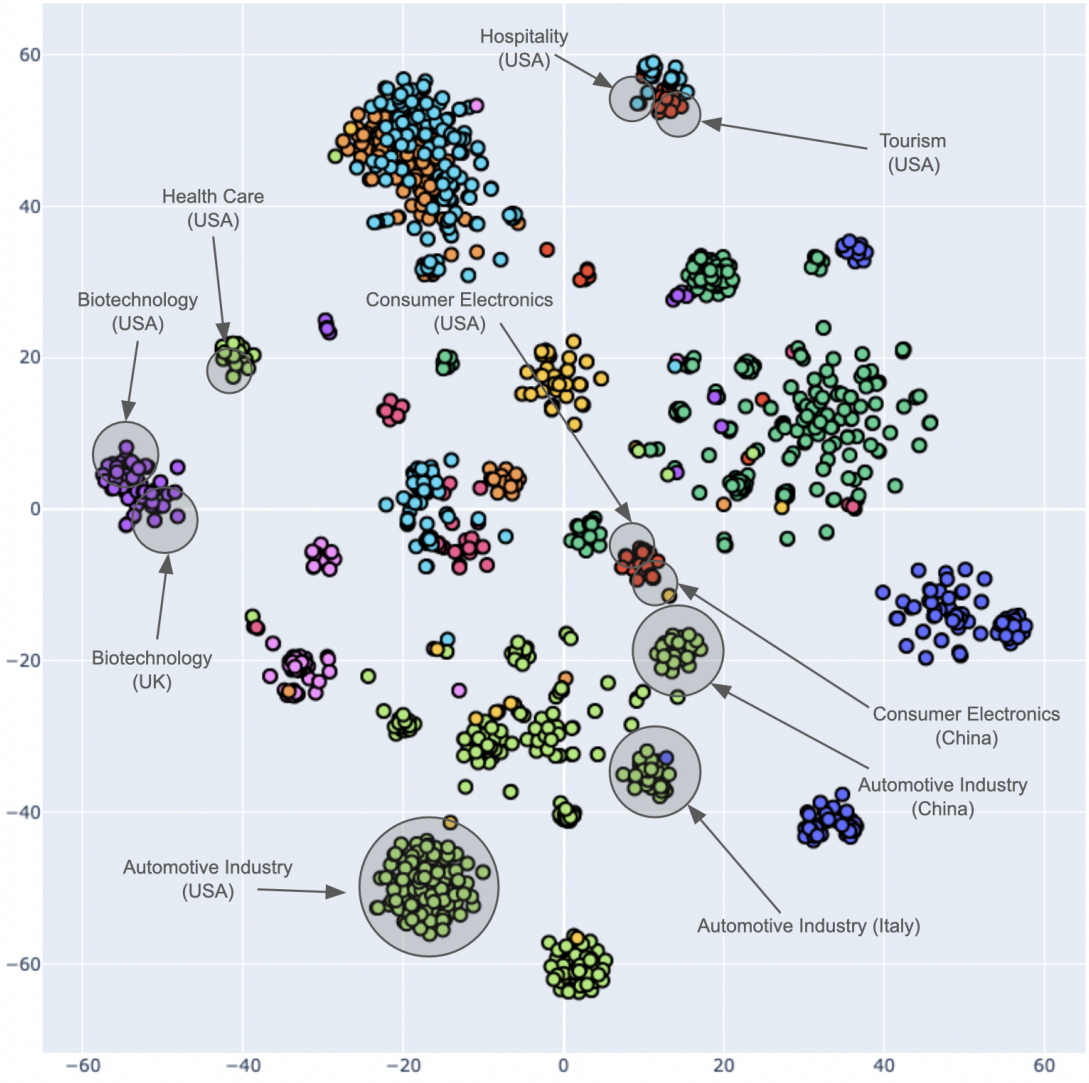


Figure 5.5: Global 2D Embedding Space for Multi-Aspect Specific Text Embeddings (Union).

5.2.4 Example: Nearest Neighbor Search

Finally, we want to give a qualitative comparison of different embedding models for nearest neighbor search. We compare generic embeddings against country-specific, industry-specific, and multi-aspect embeddings based on a seed article about "Amazon". The nearest neighbors are taken from the industry-specific dataset that we trained our embeddings on.

> **Seed Article**: Amazon.com, Inc. is an American multinational technology company based in Seattle that focuses on e-commerce, cloud computing, digital streaming, and artificial intelligence. It is considered one of the Big Four tech companies, along with Google, Apple, and Facebook. Amazon is known for its disruption of well-established industries through technological innovation and mass scale. It is the world's largest online marketplace, AI assistant provider, and cloud computing platform as measured by revenue and market capitalization...
>
> **1. Most Similar (Generic)**: Amazon Prime is a paid subscription service offered by Amazon that gives users access to services that would otherwise be unavailable, or cost extra, to the typical Amazon customer. This includes free two-day delivery, streaming music and video, and other benefits. In April 2018, Amazon reported that Prime had more than 100 million subscribers worldwide...
>
> **1. Most Similar (Generic)**: Amazon Music (previously Amazon MP3) is a music streaming platform and online music store operated by Amazon. Launched in public beta on September 25, 2007, in January 2008 it became the first music store to sell music without digital rights management (DRM) from the four major music labels (EMI, Universal, Warner, and Sony BMG)...
>
> **1. Most Similar (Industry)**: TigerDirect is an El Segundo, California-based online retailer dealing in electronics, computers, and computer components that caters to business and corporate customers. Previously owned by Systemax, which is known for its acquisitions of the intellectual property of the defunct U.S. retail chains Circuit City and CompUSA, and relaunching them as online retailers...
>
> **2. Most Similar (Industry)**: UncommonGoods, LLC is a Brooklyn-based, privately held, American online and catalog retailer, founded in 1999. The UncommonGoods website launched in July, 2000. The company sells small production gifts for children, teens, and adults, home accents, jewelry, accessories, kitchen and home entertaining items, art, games, books, food and drink, and DIY kits...

1. Most Similar (Country): HP Inc. (also known as HP and stylized as 'hp') is an American multinational information technology company headquartered in Palo Alto, California, United States. It develops personal computers (PCs), printers and related supplies, as well as 3D printing solutions...

2. Most Similar (Country): Adobe Inc. is an multinational computer software company headquartered in San Jose, California. It has historically focused upon the creation of multimedia and creativity software products, with a more recent foray towards digital marketing software. Adobe is best known for its Adobe Flash web software ecosystem, Photoshop image editing software, Adobe Illustrator vector graphics edit...

1. Most Similar (Country ∪ Industry): Art Technology Group (ATG) was an independent Internet technology company specializing in eCommerce software and on-demand optimization applications until its acquisition by Oracle on January 5, 2011...

2. Most Similar (Country ∪ Industry): ChannelAdvisor Corp. is an e-commerce company based in Morrisville, North Carolina. The company provides cloud-based e-commerce software solutions to over 2800 customers worldwide, including Dell, Karen Kane, KitchenAid, Under Armour, Timex and Samsung. ChannelAdvisor supports hundreds of e-commerce channels worldwide...

5.3 User Study Knowledge Exploration

5.3.1 Experimental Setup

As a final step, we compared the utility of different embeddings on a next-article recommendation for knowledge exploration on Wikipedia. For this, we show the user a source article from our industry-specific dataset and give him a knowledge exploration goal. We then give the user a selection of different next-article recommendations and ask him to rank the Top 5 links by how useful they are towards the given exploration goal. Each next-article recommendation item displays the first 150 characters of the corresponding Wikipedia article. All next-article recommendations are taken from the industry-specific dataset and include three articles from each of the following categories:

- **Wikipedia Baseline**: The first three linked Wikipedia pages from the Seed Article.

- **Generic Embedding**: The first three most similar Wikipedia pages by generic text similarity (based on the state-of-the-art SimCSE model from [23]).
- **Single-Aspect Embedding**: The first three most similar Wikipedia pages by aspect-based embeddings (**either** country based **or** industry based)
- **Multi-Aspect (Union) Embedding**: The first three most similar Wikipedia pages by a combination of aspect-based embeddings (**both** country **and** industry based)

The user study involves two different exploration tasks. One exploration task primes the user towards exploring companies for a given country (country-based exploration for the countries the USA, China, and Denmark) and another towards exploring companies for a given industry (industry-based exploration for the industries of E-commerce, Gaming, and Mining). The combination of these two exploration types makes up 10 different exploration tasks in total.

We used the platform SoSciSurvey [1] to design the user study and collect the user responses. Afterward, we analyzed the result and evaluated the most popular link categories by type. An example of the user survey questions is shown in Figure 5.7.

5.3.2 Results

Our user study was conducted with a total of **21** users between the ages of 21 and 61 (mean age: **25**). This gives us a total of about **2100** data points (10 tasks, avg. 10 user responses per task)

For the quantitative evaluation of the user study, we aggregated the mean ranks for each type of recommendation article. As we present the user with up to 12 next-article recommendations but only receive user input for the Top 5, we assign the maximum rank of 12 for those recommendations that were not selected by the user.

We plot the responses from the two exploration tasks (country-based exploration and industry-based exploration) in Figure 5.7. We show the type of link on the x-axis and the average rank that the user assigned to it on the y-axis. Note that a lower mean rank indicates a higher user preference, as we asked users to rank the Top 5 articles. For the country-based exploration tasks, we find country-specific embeddings to be the least preferred user choice, being outperformed by both generic embeddings, and the Wikipedia baseline. For the industry-based exploration tasks, we find industry-specific Embeddings as the second most preferred user choice, while the Wikipedia baseline is voted as the least preferred user choice. For both types of exploration tasks, we surprisingly find multi-aspect embeddings as the most preferred user choice, outperforming single-aspect embeddings consistently.

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

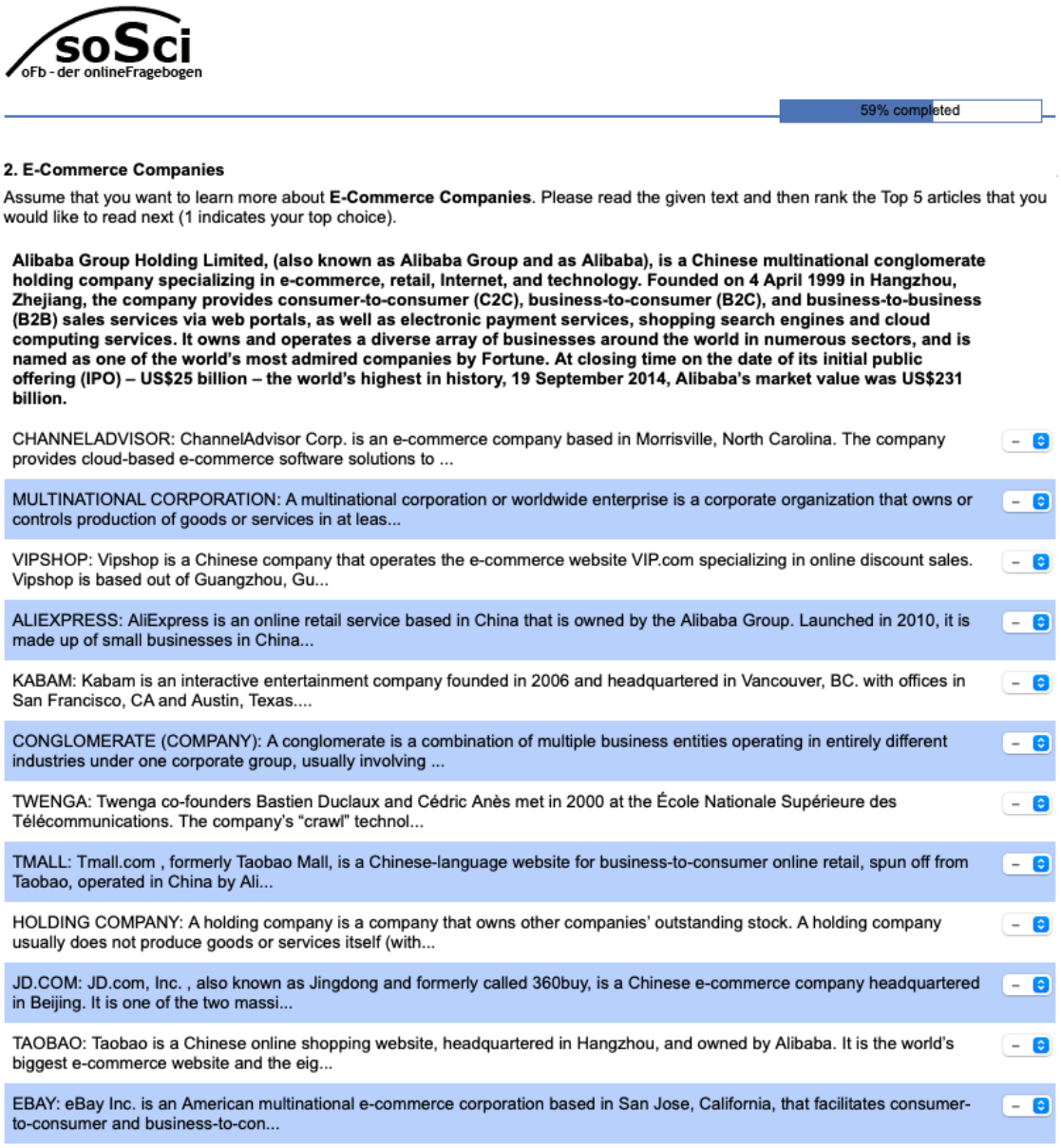
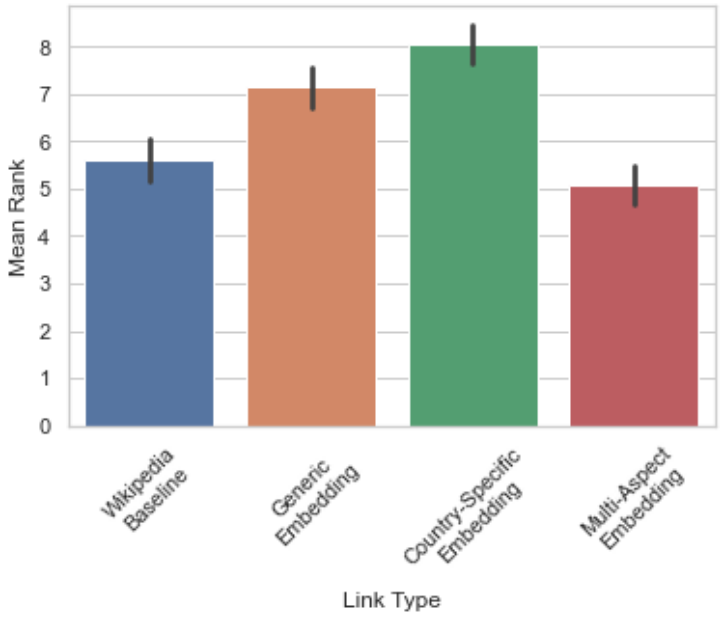
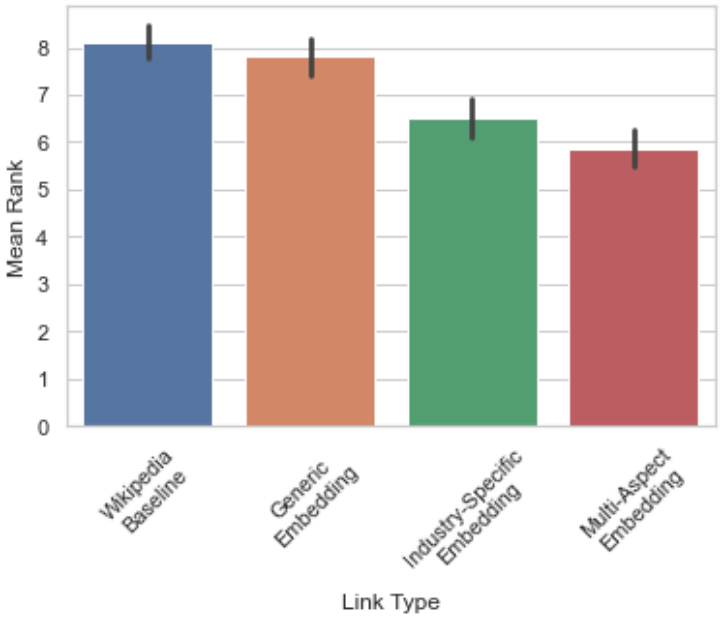


Figure 5.6: Example question from the User Study with SoSciSurvey



(a) Comparison of User responses for the country-based exploration task.



(b) Comparison of User responses for the industry-based exploration task.

Figure 5.7: User Study Results

6 Discussion

This section will discuss the results from the previous chapter and provide explanations as well as draw connections to other literature on the topic.

6.1 PWC

6.1.1 The Effectiveness of Data Augmentation

With the results from 5.1 we showed that data augmentation can be applied to marginally improve the performance of aspect-based embeddings under certain conditions. Using the same ground truth labels, we were able to improve the performance of our information retrieval task by ~2.5%. Furthermore, we were able to outperform generic embeddings using an unsupervised approach with self-augmentation techniques obtaining an average improvement of 0.247 points for MRR across all aspects (5.1.3) compared to generic embeddings. We could also show an improvement for data augmentation on the "dataset" aspect of the full PWC corpus training dataset compared to the Gold dataset 5.5.

Even though these results show the potential of data augmentation, we saw that data augmentation alone is not enough to bridge the gap between the performance of small datasets and the performance of large datasets on our PWC corpus. Furthermore, we need to emphasize that we could not single out the best approach for data augmentation. While we found *CWE Substiution* to work best on a dataset that includes only a few items per label, it was largely outperformed by *Summarization* on the Self Augmentation task. Depending on the task, *Sentence Deletion* and *Sentence Ordering* also produced improvements over the Gold dataset on certain aspects (5.1.1, task+method) while being outperformed on the remaining aspect (dataset) and consistently underperforming for other tasks (5.1.2). Therefore we cannot declare data augmentation alone as the solution for small datasets on aspect-based similarity tasks.

We conclude that data augmentation is only beneficial when working with a moderate size of Gold samples. Both when too little data is available (*K Labels + K Samples* 5.1.2) and when too many data samples are given (*Complete Dataset* 4.4.1) and in particular when the Gold dataset includes incorrectly labeled items (*Wikidata Labeling* 5.1.4) we saw that data augmentation did not improve the performance. In contrast, with a

sufficiently large amount of data as for the *All Labels + K Samples* experiment 5.1.1, we could find improvements in the performance of the embeddings. This conclusion becomes particularly clear when looking at the results of data augmentation on the *Complete Dataset* experiment. Here we found all data augmentation strategies to work for the "dataset" aspect, which contains by far the least training samples compared to the other two aspects (see 3.2). For the remaining aspects with sufficiently much available training data, no improvement could be achieved with data augmentation.

6.1.2 SimCSE vs SBERT

Finally, the comparison between SimCSE and SBERT, in general, showed that SimCSE embeddings outperform SBERT embeddings on aspect-based similarity tasks. We observed that the performance difference between the two approaches is particularly high when we train on small datasets. This becomes clear when we compare the results from *All Labels + K Instances* and *Complete Dataset* (large dataset, small performance difference) and *K Labels + K Instances* (small dataset, large performance difference). For the experiments on the complete PWC training dataset, we could outperform the method from Ostendorff by ~4% using SimCSE and create an additional improvement of ~6% (MRR) for the dataset aspect using *Summarization* as a data augmentation technique. This shows that SimCSE is a better method for training aspect-specific embeddings, particularly for small datasets.

We find this general dominance of SimCSE conclusive with the results from the original SimSCE paper on the STS Benchmark [23].

6.1.3 Limitations

Our proposed explanation for why Data Augmentation is working only when a correct and sufficiently large amount of (Gold) data is available is the fact that data augmentation is not able to add new information to the dataset. While Synonym Replacement, Summarization, Backtranslation, etc. can add variations w.r.t. to phrasing, style, or order of textual information, it alone cannot fill in missing information about the aspect that should be learned when it is never mentioned. This becomes particularly clear with the *K Labels + K Instances* experiment that uses only a small subset of the available aspect labels from the PWC dataset, consequently leading to even worse performance for all Data Augmentation Strategies.

While we tried to apply a comprehensive list of data augmentation techniques, using token-based, sentence-based, and document-based techniques, we could not apply all existing data augmentation techniques that are available.

In particular, we did not apply data augmentation with cross-encoders as described in

[20] because of the high computational complexity for this approach (see 2.3). Therefore we cannot make a statement about the performance of data augmentation in general, but only about the performance of the data augmentation technique that we used.

6.1.4 Connections to other Literature

To the best of our knowledge, we are the first to experiment with data augmentation techniques on aspect-based similarity tasks. Therefore we cannot compare our results against other literature. The only reference we have is to papers for (generic) text similarity tasks such as Augmented SBERT [20]. We find consistency between our results and the results from Augmented SBERT in that *CWE Substitution* was reported as the best performing data augmentation technique. However, we cannot reproduce the findings from CERT [40] on a strong performance for *Backtranslation* since none of our experiments showed consistent improvement for this data augmentation technique. We believe that the performance of a particular data augmentation technique depends heavily on the task and the dataset. Evidence for this can also be found in [20] where data augmentation was shown to improve the performance for the *Spanish-STS* and *MRPC* datasets, yet deteriorated the performance for the *BWS* and *Quora* datasets compared to the respective Gold datasets.

6.2 Wikipedia Experiments

Our results from the experiments on the Wikipedia dataset in 5.2.1 showed very strong performance on the information retrieval task for all aspects. We saw from the visualized 2D embedding spaces that our embedding models were able to effectively separate items by their aspect labels. Furthermore, we found surprising evidence that the similarity between aspect labels is often connected to spatial proximity in the embedding space, even though we did not explicitly train embeddings on the similarity **between** aspect labels but only on the similarity of items for the **same** aspect label. We believe that this is related to the fact that some items from our training dataset are annotated with multiple aspects (e.g. Amazon is annotated with e-commerce, retail, cloud computing, etc.). As the model optimizes the embedding for Amazon to be close to both companies from e-commerce, retail, and cloud computing, this effectively pulls all items from all those industries closer to each other. As a single company naturally engages in related industries (e.g. e-commerce and retail), this could be the reason why we also find related aspects to be close to each other.

Furthermore, we found that training embeddings on multiple aspects at the same time even surpassed the performance of single-aspect embeddings. This observation is surprising as we expected single aspect embeddings to be more specialized. We

attempt to explain our results by the fact that training embeddings on both aspect-specific datasets at the same time provide the model with more training data. As there is a correlation between which industries are dominant in which country (e.g. Arab countries have a higher than average density of oil companies, Asian countries have a higher than average density of video game companies, etc.) this could provide the model with additional data, thereby allowing it more effectively to separate items by their aspect labels. Our results on multi-aspect embeddings motivate us to think about training embeddings in even more aspects, using the available aspect labels from Wikidata. If done exhaustively, this could lead to more explainable models for Semantic Text Similarity, as we can define what aspects in a text should determine similarity and what aspects should not.

In section 4.4.1 we showed a simple, yet effective approach to creating custom aspect-based similarity datasets by exploiting a large number of knowledge triplets from Wikidata and the fact that many Wikidata items link to Wikipedia items, thereby effectively providing a large amount of labeled text data. In our experiments, we only chose the aspects "country" and "industry" as examples for showing how to create a custom dataset. Yet, the general approach is applicable to essentially all properties from Wikidata. In order to quickly find and experiment with aspects, we recommend using the **Wikidata Query Builder** [1] which provides an interface for browsing through Wikidata properties for finding relevant aspects and for composing SPARQL queries.

While Wikidata is one of the biggest knowledge bases in the world, it does not provide a complete set of all entities and aspect information. For example, we saw from the results of the *Wikidata Labeling for PWC* experiment in 5.1.4 that Wikidata captures only a small amount of available datasets, methods, and tasks in the domain of computer science. Furthermore, the described approach is limited to the creation of aspect-based datasets from Wikipedia, which contains texts in an encyclopedic style. For training embedding models on expert domains such as computer science research papers, Wikipedia-based datasets may therefore not yield optimal results. For training aspect-based embeddings for domain-specific tasks that cannot be easily captured by Wikipedia, we recommend follow-up research using Unsupervised Data Generation as described in [41]. The paper showed very promising results from using sequence-to-sequence transformers for creating custom training datasets using few-shot learning [41]. We would have liked to apply this technique for creating datasets on aspect-based embeddings. However, even the smallest model in the paper (T5-11b) is too large for our available hardware with only one V100 GPU. We want to highlight the potential of this approach by providing an example of using the GPT-3 [42] API from OpenAI for generating a custom abstract with the a set of defined aspect labels (**task**=*aspect-based*

[1] https://query.wikidata.org/querybuilder/

Embeddings, **method**=*SimCSE*, **dataset**=*Wikipedia*).

> **Prompt**: *Write an academic abstract for a paper that uses Contrastive Learning for training aspect-based Embeddings on Wikipedia:*
>
> aspect-based Embeddings (ABE) are a type of word embedding that captures the latent relations between aspects and entities in a text. In this paper, we use SimCSE, a tool for training ABE on Wikipedia, to train aspect-based word embeddings for Wikipedia. We evaluate the performance of our trained embeddings on several tasks, including aspect prediction and entity prediction. Our results show that SimCSE can learn high-quality embeddings that capture the latent relations between aspects and entities in Wikipedia.

6.2.1 Impact on Knowledge Exploration

Our user study compared the perceived utility of baseline Wikipedia links, generic text embeddings, and aspect-based embeddings for several knowledge exploration scenarios in the domain of companies with "industry" and "country" as the relevant aspects.

Despite the high scores on our evaluation benchmark for both aspect-based embeddings on the information retrieval task, we found no consistent preference of participants towards single-aspect nearest neighbor recommendations for both scenarios. Instead, country-specific recommendations were on average ranked as least important and industry-specific only as second most important. Yet, we found that multi-aspect recommendations were on average ranked as the most useful for both scenarios. We believe this has two reasons. Most importantly, the evaluation scores show that multi-aspect embedding performs highest for both aspect-specific information retrieval tasks, therefore outperforming aspect-specific embeddings for both scenarios. Secondly, we believe that a combination of the aspects "country" and "industry" form a more natural information need compared to exploring all possible types of companies from a specific country across all industries. Furthermore, we attribute the comparatively high user preference towards Wikipedia baseline links to the fact that Wikipedia users are generally accustomed to the type and structure of links and therefore naturally gravitate towards them, even when they are explicitly guided towards a specific aspect.

Finally, we need to acknowledge that our user study imposed a very specific information need on the participants, proposing that they were interested in exploring the very specific aspect that we provided. Within these constraints, we found that our multi-aspect recommendations were on average the most preferred user choice. However, these aspects represent only a small subset of the possible aspects that can be

meaningful to the user. Addressing those information needs, therefore, requires the identification of those aspects that are relevant to the user. To implement a general purpose recommender system for improved knowledge exploration on Wikipedia, we would therefore need to also be able to detect the user needs based on the interaction history of the user (e.g. "Is the user interested in Chinese industry or is he interested in the country of China in general?"). With our content-based recommender system, we provide only the foundation for implementing such a system but leave it open for future research to automatically detect what aspects are relevant to the user.

7 Conclusion and Outlook

In the previous sections, we described a new approach to making knowledge exploration in Wikipedia more efficient. We implemented and evaluated different techniques for creating links between Wikipedia articles in a way that addresses a specific information need of a user. For this, we used various state-of-the-art measures of document embeddings for text similarity and aspect-based similarly and tested the performance of document embeddings based on an information retrieval task on the PapersWithCode dataset and Wikipedia dataset. Our focus was to work out the best way of training aspect-based embeddings when little or no labeled data is available. We experimented with several Data Augmentation techniques for multiple baseline (Gold) datasets that differed in terms of the amount of labeled data, the precision of annotations, and the distribution of training samples per label. We found that Data Augmentation can marginally improve the performance on a test dataset when sufficiently many data samples are given while deteriorating results when too few data samples are known or the training data contains wrong annotations. We also found that SimCSE produced slightly higher benchmark scores on the PWC dataset compared to SBERT, allowing us to improve the benchmark scores (MRR) from Ostendorff by ~4% on average and making it our preferred architecture for subsequent experiments.

For our second series of experiments, we leveraged Wikidata as an external data source for labeled data and composed a custom aspect-specific dataset by filtering out company-specific Wikipedia articles and labeling them by their country of origin and their industry from the Wikidata knowledge graph. We achieved strong test performance for both aspects and additionally found that our aspect-specific embeddings display a notion of similarity between semantically similar aspect labels. Additionally, we obtained strong results from training embeddings on two aspects at the same time. In general, we showed that our technique for composing custom aspect-specific datasets based on Wikidata and Wikipedia can be generalized for a broad set of aspects and can be used for defining more complex aspects by combining multiple Wikidata properties.

Finally, we conducted a user study that simulated multiple knowledge exploration scenarios and involved participants ranking the utility of recommended "read-next" articles. We provided the user with a seed article and asked them to list the top 5 most useful recommendations from a selection of standard Wikipedia links, a selection of links by generic similarity, a selection of links by single-aspect similarity, and a

selection of multi-aspect similarity. Our experiments showed that users on average preferred multi-aspect-specific embeddings while single-aspect-specific embeddings showed mixed results between the two types of exploration tasks. We believe that our multi-aspect embeddings address a more natural information need of the user and are therefore preferred.

7.1 Outlook

We believe that aspect-based embeddings are not only useful for knowledge exploration tasks but, in general, represent an important step toward more explainable text similarity models. They allow us to define what properties should be considered when comparing two documents. Our experiments mostly included the encoding of a single aspect but we showed that we can also train multi-aspect document embeddings. We believe that the next important step towards improving knowledge exploration in large collections of text documents is to find the best suitable combination of aspects for a particular information need. With our proposed method of creating custom aspect-based datasets from Wikidata and Wikipedia and training them with SimCSE, we have provided a foundation for future experiments that attempt to create multi-aspect embeddings. It will take additional efforts to automatically analyze what combination of aspects is best for a given information need.

We furthermore believe that sequence-to-sequence models like T5 and GPT-3 will play an increasingly important role when it comes to creating custom datasets outside of Wikipedia. While they are immensely expensive to run, they have produced powerful results for generating texts with custom properties and therefore show enormous potential for generating highly custom aspect-specific datasets beyond the constraints of existing datasets or knowledge bases.

List of Figures

List of Tables

Bibliography

[1] É. Palagi, F. L. Gandon, A. Giboin, and R. Troncy. "A Survey of Definitions and Models of Exploratory Search". In: *Proceedings of the 2017 ACM Workshop on Exploratory Search and Interactive Data Analytics* (2017).

[2] M. Ostendorff. "Contextual Document Similarity for Content-based Literature Recommender Systems". In: *ArXiv* abs/2008.00202 (2020).

[3] M. Ostendorff, T. Blume, T. Ruas, B. Gipp, and G. Rehm. "Specialized document embeddings for aspect-based similarity of research papers". In: *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries* (2022).

[4] J. Wang and Y. Dong. "Measurement of Text Similarity: A Survey". In: *Inf.* 11 (2020), p. 421.

[5] D. Chandrasekaran and V. Mago. "Evolution of Semantic Similarity—A Survey". In: *ACM Computing Surveys (CSUR)* 54 (2021), pp. 1–37.

[6] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli. "A SICK cure for the evaluation of compositional distributional semantic models". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 216–223. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf.

[7] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[8] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. DOI: 10.18653/v1/S17-2001. URL: https://aclanthology.org/S17-2001.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[10] J. Pennington, R. Socher, and C. Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[11] M. T. Pilehvar and N. Collier. "De-Conflated Semantic Representations". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1680–1690. DOI: 10.18653/v1/D16-1174. URL: https://aclanthology.org/D16-1174.

[12] Q. Le and T. Mikolov. "Distributed Representations of Sentences and Documents". In: *31st International Conference on Machine Learning, ICML 2014* 4 (May 2014).

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[14] I. Beltagy, K. Lo, and A. Cohan. "SciBERT: Pretrained Language Model for Scientific Text". In: *EMNLP*. 2019. eprint: arXiv:1903.10676.

[15] D. M. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil. "Universal Sentence Encoder". In: *ArXiv* abs/1803.11175 (2018).

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[17] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. "Deep Unordered Composition Rivals Syntactic Methods for Text Classification". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1681–1691. DOI: 10.3115/v1/P15-1162. URL: https://aclanthology.org/P15-1162.

[18] N. Reimers and I. Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. URL: https://aclanthology.org/D19-1410.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: (2019). DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

[20] N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych. "Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks". In: *NAACL*. 2021.

[21] S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston. "Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SkxgnnNFvH.

[22] E. Ma. *NLP Augmentation*. https://github.com/makcedward/nlpaug. 2019.

[23] T. Gao, X. Yao, and D. Chen. "SimCSE: Simple Contrastive Learning of Sentence Embeddings". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6894–6910. DOI: 10.18653/v1/2021.emnlp-main.552. URL: https://aclanthology.org/2021.emnlp-main.552.

[24] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[25] Z. Wu, S. Wang, J. Gu, M. Khabsa, F. Sun, and H. Ma. "CLEAR: Contrastive Learning for Sentence Representation". In: *ArXiv* abs/2012.15466 (2020).

[26] Y. Meng, C. Xiong, P. Bajaj, S. Tiwary, P. Bennett, J. Han, and X. Song. "COCO-LM: Correcting and Contrasting Text Sequences for Language Model Pretraining". In: *ArXiv* abs/2102.08473 (2021).

[27] M. Ostendorff, T. Ruas, M. Schubotz, G. Rehm, and B. Gipp. "Pairwise Multi-Class Document Classification for Semantic Relations between Wikipedia Articles". In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*. New

York, NY, USA: Association for Computing Machinery, 2020, pp. 127–136. ISBN: 9781450375856. URL: https://doi.org/10.1145/3383583.3398525.

[28] D. Bär, T. Zesch, and I. Gurevych. "A Reflective View on Text Similarity". In: *RANLP*. 2011.

[29] N. Goodman. "Seven strictures on similarity". In: (1972).

[30] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar. "SemEval-2014 Task 4: Aspect Based Sentiment Analysis". In: *COLING 2014*. 2014.

[31] J. Vosecky, D. Jiang, K. W.-T. Leung, and W. Ng. "Dynamic multi-faceted topic discovery in twitter". In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (2013).

[32] Y. Chen, Y. Wang, X. Zhao, H. Yin, I. Markov, and M. de Rijke. "Local Variational Feature-Based Similarity Models for Recommending Top-N New Items". In: *ACM Transactions on Information Systems (TOIS)* 38 (2020), pp. 1–33.

[33] D. Vrandečić and M. Krötzsch. "Wikidata: A Free Collaborative Knowledgebase". In: *Communications of the ACM* 57.10 (2014), pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489.

[34] *PapersWithCode*. https://paperswithcode.com. Accessed: 2022-05-30.

[35] Kensho. *Kensho Derived Wikimdia Dataset*. https://www.kaggle.com/datasets/kenshoresearch/kensho-derived-wikimedia-data. Accessed: 2020-05-30.

[36] I. Montani, M. Honnibal, M. Honnibal, S. V. Landeghem, A. Boyd, H. Peters, M. Samsonov, J. Geovedi, J. Regan, G. Orosz, P. O. McCann, S. L. Kristiansen, D. Altinok, Roman, L. Fiedler, G. Howard, W. Phatthiyaphaibun, E. Bot, S. Bozek, M. Amery, Y. Tamura, B. Böing, P. K. Tippa, L. U. Vogelsang, R. Balakrishnan, V. Mazaev, GregDubbin, jeannefukumaru, J. D. Møllerhøj, and A. Patel. *explosion/spaCy: v3.0.0rc: Transformer-based pipelines, new training system, project templates, custom models, improved component API, type hints & lots more*. Version v3.0.0rc1. Oct. 2020. DOI: 10.5281/zenodo.4091419. URL: https://doi.org/10.5281/zenodo.4091419.

[37] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[38] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *ArXiv* abs/1910.03771 (2019).

[39] N. Ng, K. Yee, A. Baevski, M. Ott, M. Auli, and S. Edunov. "Facebook FAIR's WMT19 News Translation Task Submission". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 314–319. DOI: 10.18653/v1/W19-5333. URL: https://aclanthology.org/W19-5333.

[40] H. Fang and P. Xie. "CERT: Contrastive Self-supervised Learning for Language Understanding". In: (May 2020). DOI: 10.36227/techrxiv.12308378.v1. URL: https://www.techrxiv.org/articles/preprint/CERT_Contrastive_Self-supervised_Learning_for_Language_Understanding/12308378.

[41] Z. Wang, A. W. Yu, O. Firat, and Y. Cao. *Towards Zero-Label Language Learning*. 2021. DOI: 10.48550/ARXIV.2109.09193. URL: https://arxiv.org/abs/2109.09193.

[42] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.