Ludwig Maximilian University of Munich

Department of Statistics

Creating a Customer Centricity Graph for Car Insurances Using Natural Language Processing

Statistical Consulting

Project Partner: Dr. Korbinian Spann, Insaas GmbH

Supervisors: Dr. Christian Heumann
Matthias Aßenmacher

Authors: Naiwen Hou, Statistics with an Economic and Social Science Background, hounaiwen@hotmail.com
Elisabeth Lebmeier, Statistics, e.lebmeier@gmx.de

Place, Date: Munich, 12.11.2020

Abstract

Within the last few years, methods in the field of Natural Language Processing have been improved a lot. This development provides us with tools that make high-quality information extraction from texts possible, which we only knew for numerical data so far. A widely used model class for text is called BERT and there exist many variations for special tasks. Employing these state-of-the-art approaches, we set up a procedure to analyse German customer reviews on car insurances. In particular, we applied aspect and sentiment classification methods on the reviews as well as aspect-based sentiment classification on single aspects. We assigned the aspects to given categories and averaged over their polarities to draw these scores in a radar chart. As this chart depicts the customers' opinions, it is called customer centricity graph. These graphs provide a more differentiating view on car insurance companies than overall star ratings do and make them more easily comparable with respect to certain categories.

Contents

1 Introduction **1**

2 General Setting **3**
2.1 Problem Description 3
2.2 Data 5
2.3 Computing Resources 8

3 Theoretical Background **9**
3.1 BERT 9
3.2 DistilBERT 10
3.3 BERT for Aspect-based Sentiment Analysis 11
3.4 Named Entity Recognition 12

4 Methodology **13**
4.1 Preprocessing 13
4.2 Aspect Detection 14
4.3 Aspect-based Sentiment Classification 16
4.4 Aspect-Entity Matching 17
4.5 Aspect-free Sentiment Classification 21
4.6 Scores 23
4.7 Radar Charts 26

5 Conclusion **28**

List of Figures **30**

List of Tables **30**

Bibliography **31**

Python Requirements **33**

Digital Appendix **34**

1 Introduction

Traditionally, customers did not have a lot of possibilities when they wanted to buy a special product. Over the years, the situation has changed rapidly since competition has grown and, due to the internet, it has become much easier to reach out to customers. Although this may be true for almost all kinds of products, we focus on car insurances here. Of course, customers used this new variety to choose the insurance with the best properties, e.g. best service or lowest price. This has led to a new development within companies: Now, most of them try to find out their customer's needs and wishes in order to keep their clients or even find new ones. This focus on customers is called *customer centricity* and methods to measure it have been developed. In times of online comparison portals, it is easy to both tell your opinion and find reviews from other people. Especially star ratings give a simple number indicating the quality of an insurance company. But this is only a very general approach for a complex issue. Often, people are not objective and they just give their opinion when it is an extreme one. So they might only consider these topics for which their emotions are the strongest - both positive and negative - and they might leave out those they are totally fine with. This results in a great amount of reviews of the same company that talk about completely different topics. So, it is clear that an average star rating cannot cope with the variety within customer reviews.

At this point, our consulting project at Insaas comes into play. It is their business idea to analyse review texts of other companies with respect to topics and emotions and to display the ratings in a differentiating way. This approach is based on publicly available data from the internet and can be extended by internal reviews. The goal is to show in which areas a company performs better than its competitors and where there are still improvements to be made. We are part of this project in order to set up a new approach based on state-of-the-art methods in the field of Natural Language Processing (NLP). In the past few years, in this area there have been a lot of improvements - both conceptual and computational - that make new methods applicable. One of these is the so-called BERT model [3] for which exist lots of variations [7, 8, 15] being tailored for different tasks.

The goal of our work was building a pipeline that takes review data as input and results in a customer centricity graph. The steps in between included information extraction and

grouping it with respect to the area of the company performance. That was followed by turning the information into values that were plotted in the end.

We start our report with a detailed description of the problem, the key terms used throughout the project and the given data. The second part is about the theoretical background of BERT-based models. In the third step, we explain the application of the models, their training process and the results on our data. At the end of that section, we show and compare the final customer centricity graphs, before we come to our conclusion and future ideas.

2 General Setting

First, we will give an overview about the problem and important terms. Then we will continue with a short description of the steps we conducted. Afterwards we will focus on the data at hand and computing resources.

2.1 Problem Description

The goal was to create a customer centricity graph for companies from the car insurance industry using public German text reviews from the internet. A customer centricity graph pictures the opinion of customers based on their feedback. The reviews were provided by Insaas and we will give a detailed description of them in the next section. As they are German, we are going to use German examples throughout this report. Before we start with the outline of the conducted steps, we will shortly explain some key terms.

Key Terms In order to deal with our review texts, we fix some basic terminology. It is not always used with exactly the same meaning in literature. For a better understanding, we illustrate our definitions by an example review:

Der Mitarbeiter war sehr freundlich. Trotzdem ist mir die Versicherung zu teuer.

So-called *aspects* point out topics of a review. For instance, this review is about "Mitarbeiter" and "Versicherung". Aspects are mostly nouns, but some exceptions exist. It may also appear that an aspect is not a part of the sentence literally, but the context indicates this aspect. Here, this could be the aspect "Preis" for the second sentence. Due to the great amount of possible topics to talk about, we introduce *entities*. With that we mean categories in which the aspects can be grouped. The entities were fixed by Insaas and they are *Beratung*, *Erreichbarkeit*, *Freundlichkeit*, *Kompetenz*, *Qualität*, *Preis*, *Leistung* and *Problemlösung*. As we are not only interested in the topics people think about, but also in the emotions they have about them, we consider *sentiments*, too. Here, they are *positive*, *negative* and *neutral*. They can correspond to whole reviews as well as to aspects

only. For instance, the sentiment for the example review is neutral, whereas "Mitarbeiter" is clearly positive and "Versicherung" negative.

Outline With this knowledge, we are able to go into a more concrete overview of the steps that we conducted. They are depicted in Figure 1. Details will be explained in the corresponding sections. The plan included to set up a working pipeline that can be implemented into the existing model of Insaas later. In order to make our explanations easier to understand, we added two example reviews in blue.

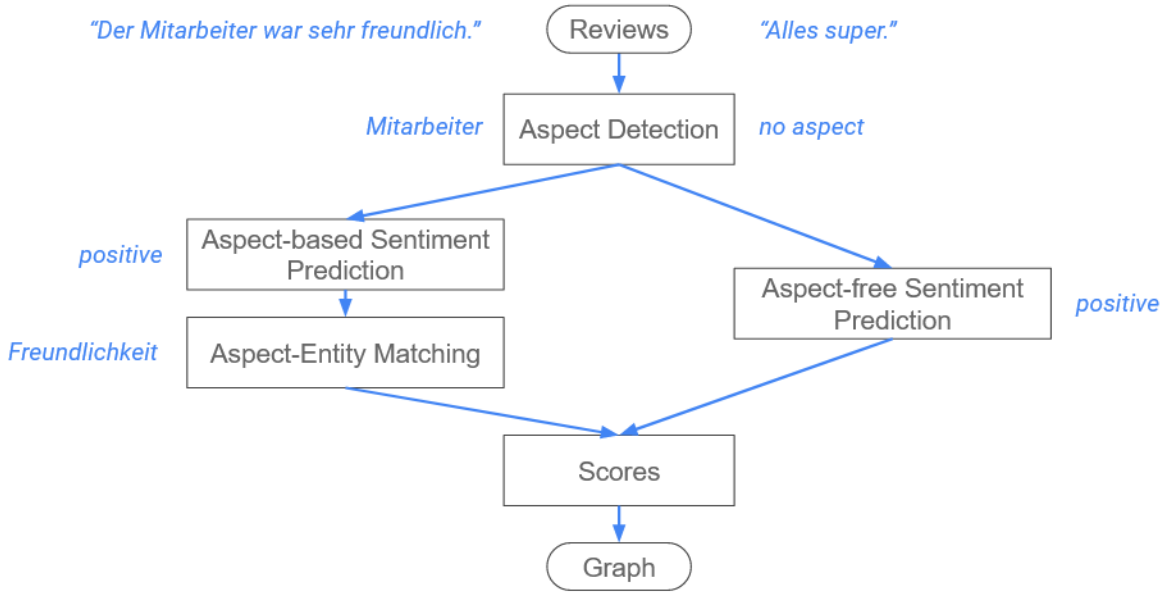

Figure 1: Overview about the steps conducted throughout the project.

At first, we extracted aspects from the reviews (see Section 4.2). For *"Der Mitarbeiter war sehr freundlich."* we found "Mitarbeiter" to be an aspect, while in *"Alles super."* no aspect could be detected. Depending on the results, we split the data into those with identified aspects on the left-hand side and those without on the right. On data with aspects we performed aspect-based sentiment analysis in Section 4.3. This means that the sentiment was determined for each aspect. In the case of "Mitarbeiter", the context suggested the sentiment to be positive. Then the aspects had to be matched to their corresponding entities in Section 4.4. Here, it made sense to assign "Mitarbeiter" to *Freundlichkeit*. So, entities and sentiments are connected via aspects. For the part of the data without any aspects, we just predicted a sentiment for the whole review, which we called aspect-free sentiment (see Section 4.5). This should clearly be positive for our example *"Alles super."*. All this extracted information was then turned into scores (see

Section 4.6), which were depicted in a radar chart, also known as spider diagram, in Section 4.7. In this diagram, polarity scores for each entity were drawn, making graphs for different companies comparable.

2.2 Data

We were given several pieces of data to work with. We will introduce them now and refer to this section later. Due to long training times and the need of label balancing in the training set, we could not always use the whole data sets that we were provided with.

2.2.1 Review-wise Labelled Data

The biggest part of data was review-wise labelled data with about 300,000 reviews. During an exploratory analysis, we realised that about two thirds of the data dealt with dental insurances instead of car insurances. This reduced the number of reviews to 99,673. They were stored in `Review_Aspect_28072020.csv`. Additionally, some parts of the data appeared to be duplicates and very few contained only filling text like "Lorem Ipsum", so these had to be removed as well. Finally, we ended up with 93,543 samples of data which amounted to about 31MB. The final data looked like this:

Date	Company	Source	Rating	Feedback	Aspect	Sentiment
2015-04-28 T14:24:00Z	Ergo_de	Ekomi	5.0	Schnelle und unkomplizierte Bearbeitung	bearbei-tung	positive
2017-09-06 T07:16:00Z	Devk	NaN	4.0	alles super so wie es ist	NaN	positive
2018-04-05 T00:00:00Z	Allianz _Direct	Trust-pilot	1.0	Kaum ereich-bar!! Prämie 2x abgebucht!! und email ...	prämie, reaktion	negative

Table 1: Examples from the review-wise labelled data set `Review_Aspect_28072020.csv`.

The most important information lies in *Feedback* which is the review text that may consist of several sentences. The time stamp in *Date* is currently only used to help identifying

duplicates. The columns *Source* and *Company* indicate where the data come from (e.g. which comparison portal) and which company they are about. Due to different sources, company names may differ in spelling. Thus, the second is necessary for grouping the data company-wise, whereas the first gives information about the scale of *Rating*. If the source of the review provided star ratings, this was stored in *Rating*, either on a 1-5 or 0-10 scale. This information was used by Insaas in order to correct the predictions of their sentiment model. The final sentiments, namely "negative", "neutral" or "positive", can be found in the corresponding column. These sentiments were used as true labels for our aspect-free sentiment prediction in Section 4.5. The *Aspect* column contains the aspects that were predicted by Insaas according to their model. For us, they serve as truth for aspect prediction in Section 4.2.

2.2.2 Aspect-wise Labelled Data

The second data set, named `Review Classifier - Trainingsset-CCG-3.csv`, was constructed during the project as we needed it. It was a subset of 651 rows of the review-wise labelled data which was labelled manually by Insaas. The new labels included sentiments and entities per aspect for up to three aspects. Removing reviews without these labels resulted in 584 rows. Here we see an example:

Feedback	Aspect_1	Sentiment_1	Entity_1	Aspect_2	Sentiment_2	Entity_2
sehr schnelle Bearbeitung freundlicher Service	Bearbeitung	positive	Problemlösung	Service	positive	Beratung
Kommunikation könnte besser dein. Wartezeit am ...	Kommunikation	negative	Beratung	Wartezeit	negative	Erreichbarkeit
Fairer Preis. Super Kontakt	Preis	positive	Preis	Kontakt	positive	Kompetenz

Table 2: Examples from the aspect-wise labelled data set `Review Classifier - Trainingsset-CCG-3.csv`.

For instance, one aspect of the first review is "Bearbeitung" and, as we can see from the

context, it is positive. In this case, the aspect is assigned to the entity *Problemlösung*. This data set was used for training the aspect-based sentiment classifier in Section 4.3 and as additional information for aspect-entity matching in Section 4.4.

2.2.3 Lemmatisation List

A lemmatisation list was needed due to the huge amount of aspects in the review-wise labelled data. Originally there were over 1,000 different aspects which were far too many in order not to have a too complex model. Thus, we assumed that aspects should mostly be nouns. These changes reduced the number of aspects in the data set to 306. For a further reduction of complexity, Insaas provided us with a manually created list where all aspects in the review-wise labelled data set were assigned to a so-called *lemmatised aspect*. According to the definition in [4, p.67], lemmatisation is grouping inflected words according to their lemma which is their dictionary entry. For example, "Beiträge", "Beitrages" and "Beitrags" were assigned to "Beitrag". We extended this approach by also arranging words with similar meaning to a lemmatised aspect. This is that aspects like "Vertragsabschluss", "Vertragsformular", "Unterlagen" and "Vertragswechsel" were allocated to "Vertrag". This procedure kept the meaning of the aspects, but also generalised them. So, 306 aspects from the review-wise labelled data were mapped to only 215 lemmatised aspects. This original list was stored in `Lemmatization_list.xlsx` and used during the aspect detection step in Section 4.2.

During the labelling of the aspect-wise labelled data set, Insaas declared more words to aspects and thus augmented the lemmatisation list to a new length of 262 lemmatised aspects. This list was called `Aspect_Lemmatization_List_9_15-2.xlsx` and included in aspect-based sentiment classification and aspect-entity matching in Sections 4.3 and 4.4, respectively.

2.2.4 Entity Synonyms

For the task of aspect-entity matching we were provided with a list of synonyms for the entities. This list was created by Insaas based on ConceptNet [9], a semantic network that connects words with each other that are related in any possible way. For each entity, there were 50 to 75 synonyms. We restructured the file for easier handling and called it `Synonym_3.0_restructured.xlsx`. In Table 3, one can see synonyms with respect to both meaning (see *Freundlichkeit*) and spelling (see *Qualität*). Also note that some synonyms are not unique for one entity, like "Qualität".

Entity	Synonym 1	Synonym 2	Synonym 3	···
Beratung	Betreuung	Fachberatung	Beratungsleistung	···
Preis	Preises	Ausgangspreis	Sonderpreis	···
Problemlösung	Problemlösungen	Lösung	Lösungsfindung	···
Erreichbarkeit	Ansprechbarkeit	erreichbarkeit	Nicht-Erreichbarkeit	···
Leistung	Gesamtleistung	Leistungsfähigkeit	Qualität	···
Qualität	Qualiät	Qualtität	Qualtät	···
Freundlichkeit	Hilfsbereitschaft	Gastfreundlichkeit	Herzlichkeit	···
Kompetenz	Fachkompetenz	Sachkompetenz	Expertise	···

Table 3: Example synonyms for entities from `Synonym_3.0_restructured.xlsx`.

2.3 Computing Resources

As the models we applied were rather complex, they often needed much time to be trained properly. Thus, we used Google Colaboratory[1], where GPUs are available for free, which accelerated our training process. One cannot choose which GPU to take; this assignment depends on the demand. Consequently, training times may vary between different GPUs, although trained models are quite similar. So, we wrote our code in notebooks from Google Colaboratory, which are basically the same as Jupyter Notebooks. Our code is written in Python.

[1] `https://colab.research.google.com/notebooks/intro.ipynb`

3 Theoretical Background

3.1 BERT

As the basis of almost all models we applied is BERT, we will first explain BERT and then come to its variations. BERT [3] stands for *Bidirectional Encoder Representations from Transformers* as its basic architecture is a multi-layer bidirectional Transformer encoder. Like the authors of BERT, we do not go into the details of Transformers, but refer to [11]. BERT is a language model, i.e. a model that is trained to understand how a language is structured and which vocabulary is used in which way. In order to reach this goal there are two phases: During the pretraining phase, the model learns the structure of the language and the vocabulary in general. This knowledge is used on a certain topic in the fine-tuning phase, where the model learns a specific task and certain terminology.

The authors pretrained BERT on unlabelled data sets of 3,300M words with two tasks: Task one was *Masked Language Modelling* which is about predicting randomly masked words within a text. BERT's specialty is to work in a bidirectional way, i.e. it is able to condition its prediction for the masked word on the context on both sides. This allows BERT to better deal with different meanings of words as more information is captured. The other task was *Next Sentence Prediction* which also includes question answering, for example. In order to use BERT on texts about certain topics, it is necessary to fine-tune it with labelled data. This means that the model is initialised with pretrained parameters and then learns task-specific weights for an extra layer. Fine-tuning BERT can be done for several tasks, like question answering, token classification and sequence classification. The latter is the most important one for us as we want to classify reviews (which are sequences) with respect to both aspects and aspect-free sentiments.

An essential part of the application of BERT during fine-tuning is the preparation of word sequences as inputs. Depending on the task, there is either one sequence on its own or there is a pair of them, e.g. question and answer. First, words are tokenised which means that words that are not part of a vocabulary are split into known tokens. The vocabulary is determined by the pretrained language model. Then, the tokens are turned

into numbers by embeddings. At the beginning of each sequence, a so-called *classification token* `[CLS]` is added. In case of pairs of sequences, the *separator token* `[SEP]` is added between them and at the end, otherwise it is put only at the end. BERT is used to always receive inputs of the same length which is called the *maximum sequence length*. The maximum that can be chosen is 512; smaller values make sense in case of shorter samples. Inputs longer than the given maximum sequence length are truncated, while shorter ones are filled with so-called padding tokens. It is marked which tokens are real and which are there due to padding. In case of sequence pairs, it is also marked to which sequence a token belongs. When all this is done, BERT can be fine-tuned on the data.

We experimented with BERT for aspect detection in Section 4.2 and for aspect-entity matching in Section 4.4 as well. It also serves as basis for aspect-based sentiment analysis methods in Section 3.3.

3.2 DistilBERT

DistilBERT [7] is a smaller version of BERT that was created to deal with memory and computational issues of BERT. It was trained on the same corpus as BERT and it has the same architecture in general. Changes included decreasing the number of layers by a factor of two and removing the pretraining task of *Next Sentence Prediction*. This resulted in a language model being 40% smaller and 60% faster. One might think that these adoptions had severe downsides with respect to the performance, but with only 3% less language understanding capabilities, DistilBERT proved to be almost as good as BERT.

The key technique for reducing the model size is *Knowledge Distillation* for which the authors refer to [2, 5]. The idea behind that is to train a small model to make the same predictions as a large model. To reach this goal, the authors introduced a triple loss combining the losses from language modeling and distillation, and a cosine-distance loss which aims for the same direction of the predictions of both models.

In our project, we used the DistilBERT model both for aspect and aspect-free sentiment classification, as we will explain in detail in Sections 4.2 and 4.5, respectively.

3.3 BERT for Aspect-based Sentiment Analysis

Although BERT and DistilBERT already cover several tasks that are relevant for information extraction from texts, there are some fields where they are only the basis for further methods. One of these is Aspect-based Sentiment Analysis (ABSA). Its aim is to get more detailed information about texts, especially finding sentiments not for a whole text, but for certain aspects within. There are several terms for very similar concepts. More precisely, what we want to do with our data is called *Aspect-based Sentiment Classification (ABSC)* or *Aspect-Target Sentiment Classification (ATSC)*.

For this task, there exist various approaches, so we decided to try only two of them in Section 4.3. We chose LCF-BERT and AEN-BERT as they are based on similar ideas which we will present in this section. Further approaches include [6, 10, 13].

LCF-BERT LCF-BERT [15] introduces a Local-Context-Focus (LCF) mechanism. This means that, in order to identify the sentiment of an aspect, an additional focus is set on words that are close to the aspect. The use of both global and local context requires separate inputs. For the local context input, classification and separator tokens are added before and after the review, respectively, like for BERT. This results in "`[CLS]` + review + `[SEP]`". For global context, the input has the form "`[CLS]` + review + `[SEP]` + aspect + `[SEP]`". In order to work with the aspects correctly, more preprocessing has to be done in advance which will be explained in Section 4.3. On the global input, BERT embeddings and a Feature Extraction Layer are employed.

In order to determine the local context, the *Semantic-Relative Distance (SRD)* is introduced. For every word in a sequence, it is defined as $SRD_i = |i - P_a| - \left\lfloor \frac{m}{2} \right\rfloor$, where i and P_a refer to the position of the word and the aspect, respectively. The length of the aspect is denoted by m. As far as we understand, positions and length m are counted with respect to words. So, the SRD basically measures the distance between words and certain aspects. Its value is calculated for each word around the aspect. If it comes below a certain threshold or is equal to it, this word is said to be part of the local context of the aspect. To clarify this, we use the same example review as above:

Der Mitarbeiter war sehr freundlich. Trotzdem ist mir die Versicherung zu teuer.

Take "Mitarbeiter" as the aspect of choice. We label the positions for all words from 1 to 12, which results in the position of the aspect $P_a = 2$. The aspect itself has a length of $m = 1$, which reduces the last part of the formular to zero. The SRDs for the words in our review can be seen in Table 4. Setting the SRD threshold to 3, "Der", "war", "sehr" and "freundlich" belong to the local context of the aspect "Mitarbeiter". Depending on

Word	Der	Mitarbeiter	war	sehr	freundlich	Trotzdem	ist	mir	die	···
SRD	1	-	1	2	3	4	5	6	7	···

Table 4: SRD values for an example review.

the chosen design, out-of-context words are not taken into account, in case of a *Context Features Dynamic Mask Layer (CDM)*, or with smaller weights, in case of a *Context Features Dynamic Weighted Layer (CDW)*. It is part of the Feature Extraction Layer. Then, in a Feature Interactive Learning Layer and an output layer, the combination of local and global context is used to classify a sentiment for an aspect.

AEN-BERT AEN-BERT [8] stands for *Attention Encoder Network*. This mechanism is introduced to set the focus on words that are semantically closer to the aspect. The model contains BERT embedding layers, an attentional encoder layer, a target-specific attention layer and an output layer. The inputs are of the forms "`[CLS]` + review + `[SEP]`" and "`[CLS]` + aspect + `[SEP]`" for reviews and aspects, respectively. As this approach performs not as good as LCF-BERT in Section 4.3, we do not go further into details here.

3.4 Named Entity Recognition

The aim of classical Named Entity Recognition (NER) [4, p.81f.] is to mark words or short sequences of words that belong to fixed categories. Mostly, these are *Location*, *Person* and *Organisation* and the assignment is context-based. In order to make this more clear, we use the following example review:

> *Mein Versicherungsvertreter Max Mustermann von der Allianz steht mir in seinem Büro in München jederzeit für Fragen zur Verfügung.*

Here, "Max Mustermann" should be labelled as person, "Allianz" as organisation and "München" as location. All other words are usually marked with *Other*.

For this task, a BERT or DistilBERT model has to be fine-tuned for token classification. We experimented with using a modified version of this approach for aspect-entity matching in Section 4.4.

4 Methodology

Here we present the main results of our project. The goal was a code pipeline with data from one company as input and its radar chart as output. In between, the steps we introduced in Figure 1 were conducted. The pipeline was run for data from Allianz and HUK, respectively, for which we also discuss the results of each step. The code for the whole pipeline can be found in the notebook called `pipeline.ipynb`. The text file `overview.txt` contains details about all files that were used. The same overview can also be found in Tables 16 and 17. Throughout the whole project, we used Huggingface's PyTorch implementation [12] of BERT models which can be found in the *transformers*[2] package.

4.1 Preprocessing

First, duplicates were removed from the data sets as well as reviews that only contained filling text like "Lorem Ipsum". Then, we standardised company names that meant the same company but were written differently due to their sources. For example, "Allianz_direct" became "Allianz_direkt". We also removed characters like "_" or "\\" that made text analysis more complicated. This procedure was used on both the review-wise and the aspect-wise labelled data. In order to get our samples for Allianz and HUK, we filtered the review-wise labelled data accordingly. We had 10,680 reviews for Allianz and 11,932 reviews for HUK in the end. For the following steps, specific preprocessing had to be done which will be explained in the corresponding sections.

[2] `https://pypi.org/project/transformers/`

4.2 Aspect Detection

The first real step was to detect aspects in each review. Therefore we trained a classifier in a so-called *multi-label* way. This means that to each review more than one label can be assigned. This seems logical as a lot of reviews cover several topics.

Training For this task we were inspired by this tutorial notebook[3]. Task-specific preprocessing of the review-wise labelled data included building a column with tuples of aspects, removing aspects that appeared less than a certain threshold and using lemmatisation on aspects. Here, the original lemmatisation list `Lemmatization_list.xlsx` was taken which included only lemmas for those aspects appearing within the data. This was sufficient as no other aspects had to be lemmatised. Then, the aspect tuples were turned into numerical values using multi-hot encoding. As loss function we chose the binary cross entropy with logits loss which is standard for multi-label problems.

As there were three German pretrained language models available, we first tested their performance on small data sets of 800 reviews for training and 200 reviews for validation to get a rough overview. The BERT models were `bert-base-german-cased`[4] and `bert-base-german-dbmdz-cased`[5] and the DistilBERT model was `distilbert-base-german-cased`[6]. This short experiment was quite helpful as it showed that the DistilBERT model clearly outperformed the BERT models in terms of training time. Additionally, it also had the highest values regarding performance. All models were trained for 36 epochs, with a learning rate of 8e-05, a maximum sequence length of 512 tokens and a batch size of 12 on a Tesla P100-PCIE-16GB GPU. The observed values can be seen in the following table:

Language Model	Accuracy	F1 (micro)	F1 (macro)	Training Time for 1 epoch (sec.)
`bert-base-german-cased`	0.52	0.68	0.15	48
`bert-base-german-dbmdz-cased`	0.49	0.64	0.13	48
`distilbert-base-german-cased`	0.55	0.70	0.17	25

Table 5: Performance and training time comparison for German language models.

[3] `https://github.com/abhimishra91/transformers-tutorials/blob/master/transformers_multi_label_classification.ipynb`
[4] `https://huggingface.co/bert-base-german-cased`
[5] `https://huggingface.co/bert-base-german-dbmdz-cased`
[6] `https://huggingface.co/distilbert-base-german-cased`

Due to these results, we decided to focus on the German DistilBERT model. The corresponding training notebook was `distilbert4aspects.ipynb`. We included a minimum aspect frequency threshold to sort out aspects that appeared rarely in the whole data set since they were really unlikely to appear in other data sets. We set it to 2 in the end. As this was not sufficient to deal with different aspect frequencies, we added a mechanism to construct nearly balanced data sets. The training set was constructed in such a way that it contained each aspect at least 50 times if possible, otherwise all reviews with this aspect were taken. The validation set was set up analogously with a threshold of 20. No review that was in the training data was allowed in validation data. So in the end, we had training data with 8,291 lines and validation data with 3,311. Although there would have been more data available, we decided against increasing the sizes due to the training time. We trained the model with several hyperparameter settings. The hyperparameters of our choice were maximum sequence length (ML), learning rate (LR), batchsize (BS) and epochs (E). For various combinations of these hyperparameters we observed the following performance measures, all on a Tesla K80 GPU:

E	LR	BS	ML	Accuracy	F1 (micro)	F1 (macro)	Training Time for 1 epoch (min.)
15	8e-05	12	512	0.88	0.98	0.95	13
10	8e-05	16	400	0.84	0.97	0.91	10
10	6e-05	12	512	0.86	0.97	0.93	15

Table 6: Performance and training time comparison of several DistilBERT models for aspect classification.

Due to the sufficient performance, we abstained from further tuning and decided to use the model called `distilbert_ntrain8291_nvalid3311_e15_lr8e-05_bs12_ml512_maf2_ait198_bal` for the pipeline. In an extra file with the same name but extended with "labels", we stored the list of aspects to be used for prediction. There were 198 different aspects including the label "no aspect".

Pipeline In order to get predictions for HUK and Allianz data, we applied the chosen model on preprocessed review texts. As performance should be measured, the labels had to be prepared as well, like we did for model training. When testing the model on HUK data, we reached an accuracy of 91%, F1-Score (micro) of 98% and F1-Score (macro) of 95%. For Allianz data, we achieved performance measures of 92%, 98% and 94%. These values have to be considered with caution as both company data and model training data come from the review-wise labelled data set. Thus, the model might already know some of the reviews and their aspects that should be predicted.

For our next steps we separated the given data into samples with and without aspects. We capitalised all aspects except from "KFZ" which consisted of capital letters only. These transformations were necessary for the following models.

4.3 Aspect-based Sentiment Classification

On the reviews with aspects we used aspect-based sentiment classification methods to predict one sentiment per aspect.

Training Both LCF-BERT and AEN-BERT models were trained on the aspect-wise labelled data set. We used this implementation on Github[7] and modified it for our data. After standard preprocessing procedure, we filtered the data set such that only reviews with at least one aspect-based sentiment label were included. This resulted in a total of 584 lines of data available for this task. The data set was then augmented further as we needed one row per aspect. This meant that each review was duplicated as many times as there were aspects in it. Applying this procedure led to a total number of 1,398 rows.

We applied lemmatisation from the latest lemmatisation list `Aspect_Lemmatization_List_9_15-2.xlsx` on the aspects. We also checked if there were any typos within sentiments to be corrected. Then, the sentiments were transferred into numerical values where "0" stood for *negative*, "1" for *neutral* and "2" for *positive*. The review text had to be transformed as well so that instead of the aspect there was T in the review. This was necessary to tell the model which word should be treated as aspect. If the aspect literally was in the review text, we substituted it with T directly. In case that this did not work, we tried lowercasing and capitalising the aspects and checked if these versions were in the review text and could be substituted. Often none of these approaches were successful, so we included using the lemmatisation list reversely. This means that if an aspect was a lemmatised aspect, we tried to find those words in the text that were assigned to the lemmatised aspect. For example, take the review *"Die Beiträge sind mir zu hoch"* for which the aspect is "Beitrag". Clearly, the aspect itself cannot be found in the review. Lowercasing also does not help. So, taking a look at the lemmatisation list from Section 2.2, shows us that, amongst other words, "Beiträge" is lemmatised to "Beitrag". Consequently, as "Beiträge" is part of the review, it can be substituted with T. The rows without successful T-substitution were deleted from the data. This led to a remaining number of reviews of 1,345 rows. This data was randomly split into training and validation set with a proportion of 0.8, which meant 1,076 reviews for training and 269 for validation. Due

[7] `https://github.com/songyouwei/ABSA-PyTorch`

to the relatively small size of the data remaining after preprocessing, we did not apply balancing with respect to sentiments. The code for LCF-BERT and AEN-BERT training can be found in the notebooks named `lcf_bert.ipynb` and `aen_bert.ipynb`.

The basic model always was `bert-base-german-cased`. We used a semantic relative distance of 3 and a learning rate of 2e-05. For LCF-BERT we fixed a dropout rate of 0 and for AEN-BERT of 0.1. We only focused on the following tunable hyperparameters: number of epochs (E), batchsize (BS), maximum sequence length (ML), L_2-Regularisation (*L2 Reg*), number of hidden dimensions (*hiddim*) and number of embedded dimensions (*embeddim*). Varying these values led to the following results:

Model Type	E	BS	ML	L2 Reg	hiddim	embeddim	Accuracy = F1 (micro)	F1 (macro)
LCF-BERT	10	8	240	1e-05	768	768	0.84	0.76
LCF-BERT	15	5	240	2e-05	300	300	0.82	0.72
AEN-BERT	10	8	170	2e-05	300	768	0.78	0.65
AEN-BERT	10	8	240	0.001	768	768	0.75	0.64

Table 7: Performance of several LCF-BERT and AEN-BERT models.

Due to these performance values, we decided on the LCF-BERT model named `lcf_bert_ntrain1076_nvalid269_e10_lr2e-05_bs8_ml240_lca_dr0_srd3_l2reg1e-05_hiddim768_embdim768`. In addition to the model, we also saved the tokenizer ("tokenizer" plus model name) and the hyperparameters (model name plus "opt") separately.

Pipeline For the application of this model, we needed the same preprocessing as for model training including duplicating reviews and T-substitution. After having received predictions of sentiments, we combined the predictions for the duplicated reviews again so that we had only one row per review. If a review had no predicted sentiment, this review was removed from the data set with aspects and added to the one without aspects.

As we did not have labels for this task in the data sets we used in the pipeline, we could not measure the performance of this model.

4.4 Aspect-Entity Matching

As aspects and corresponding sentiments were available at this point of the project, we had to match aspects and entities as a last step. Originally, the plan was that we received

a list which was provided by Insaas and reviewed by an insurance domain expert. During the project we discovered that there may be ambiguous aspects that belong to different entities based on their context. For instance, "Mitarbeiter" may be allocated to both *Freundlichkeit* and *Kompetenz*.

Training To deal with this issue, we tried to use Named-Entity-Recognition with BERT, like it was done here[8]. Our notebook for this task was named `bert_ner_final.ipynb`. In contrast to the classical approach in Section 3.4, our categories were not *Organisation*, *Location* and *Person*, but the given entities and *Other* for all the remaining words. Therefore, in addition to typical BERT preprocessing, we had to prepare labels for each word in each review, especially mark the words that were aspects and assign the label *Other* to the rest. When applying the model on validation data, we only looked at the predictions for words that were declared aspects. Unfortunately, this approach did not produce good results, maybe because the amount of data was too small. We also tried adding lemmatised aspects and balancing with respect to aspect-entity pairs, but this did not have much impact on the performance. So we focused on the aspect-wise labelled data that we had received. Looking closer at the entity labels showed that there were 163 aspects that were assigned to only one entity. As a model would learn exactly these relationships, we decided to use these assignments directly and made a list of these pairs. We also added aspects that appeared with several entities, but there was one entity that surpassed the others in terms of frequency in almost all cases. So we had a list of 177 aspects for which we did not have to train the model, but only for the remaining aspects. This list could be extracted from `Entity_Aspect_Freq_form_col.csv`, where the manually declared pairs are marked with "x" in the last column. Still, our predictions did not get any better. In general, we did not even have to look at the performance measures, as the predictions were *Other* most of the time.

Thus, we decided to pursue a different approach that made the pipeline work. It can be seen in the notebook called `similarity_fasttext.ipynb` and in Figure 2. For that task we used the list of synonyms for each entity that Insaas had created as described in Section 2.2. In the figure, it is called "lookup file". We used Fasttext Embeddings [1] on aspects, entities and entity synonyms. Then we paired each aspect with all entities and entity synonyms. For each pair of embeddings (aspect, entity/entity synonym) we calculated the cosine similarity. In order to get an entity for each aspect, we took the ten most similar entities or entity synonyms, looked up the entities of the entity synonyms and chose their mode as the final entity for this aspect. We decided to use ten as the amount of entity synonyms to look at because this led to the smallest number of undecided cases in the end. If there was no unique mode, we included the knowledge from the aspect-entity list `Entity_Aspect_Freq_form_col.csv` that we had extracted from the aspect-wise

[8] `https://github.com/billpku/NLP_In_Action/blob/master/NER_with_BERT.ipynb`

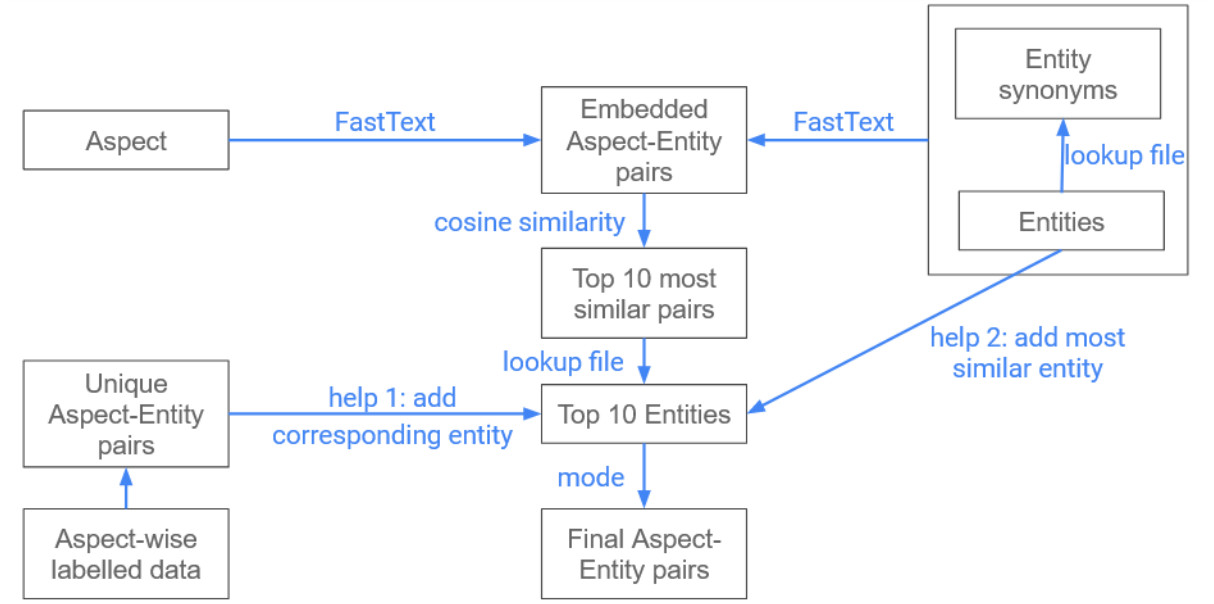

Figure 2: Overview about conducted steps for aspect-entity matching.

labelled data above (see "help 1" in Figure 2). If the entity was still undecided, we added the entity with the highest similarity to the list ("help 2") and tried to take the mode again. Afterwards, there were still two aspects left that had no unique entity assigned. For those we reduced the number of entity synonyms, until we could calculate a mode. This resulted in an aspect-entity matching list which was stored as `aspect_entity_matching.csv`.

As this might seem quite complicated, take the aspect "Mitarbeiter" as an example. It is paired with each entity synonym and for each pair the cosine similarity is calculated. These pairs are sorted in descending order regarding their similarity. The most similar entity synonyms are "Mitarbeiterkompetenz", "Servicebereitschaft" and "Arbeitsleistung". For the sake of simplicity, now we only have a look at the top three instead of the top ten which we actually used for the aspect-entity matching. For these synonyms, we look up their corresponding entities. In this case, they are *Kompetenz*, *Freundlichkeit* and *Leistung*. As there is no most frequent entity in this list, we cannot assign an entity to "Mitarbeiter" for the moment. Thus, we try "help 1" and check whether there is a unique entity based on the aspect-wise labelled data set. In this case, there is none, so we go to "help 2" instead. This means that we take the most similar entity which is *Freundlichkeit* and add it to the list of the three given entities. Now, there is a most frequent entity, namely *Freundlichkeit*, that is the final entity for "Mitarbeiter". The procedure for the entity from "help 1" would have been the same.

One downside of this approach is obviously the negligence of the context. Another flaw can be found when looking at the following relative frequencies of the entities. The table shows that the entities do not have the same probabilities to be assigned to an aspect as they appear with different frequencies in the aspect-entity matching list.

Entity	Relative Frequency
Beratung	0.23
Preis	0.18
Problemlösung	0.18
Erreichbarkeit	0.15
Leistung	0.10
Qualität	0.07
Freundlichkeit	0.05
Kompetenz	0.05

Table 8: Relative frequencies of entities in the aspect-entity matching list.

Pipeline We plainly employed this list of aspects and entities in the pipeline. As we did not have labels for this task in the company data sets, we could not measure the performance of this list. We observed that the unequal entity distribution of the aspect-entity matching from Table 8 has become even more extreme when applied on company data. The resulting frequencies can be found in Tables 9 and 10 for Allianz and HUK data, respectively. Although the order of the entities stayed the same, we found different values of relative frequencies. In addition to the absolute and relative frequencies, we also show how the corresponding sentiments are distributed among the three characteristics *negative*, *neutral* and *positive*. Especially in these columns, the companies strongly differ from each other which we will see in detail in Section 4.6.

Entity	Absolute Frequency	Relative Frequency	Negative Sentiments	Neutral Sentiments	Positive Sentiments
Beratung	8105	0.33	3456	1394	3255
Preis	4598	0.19	2403	852	1343
Problemlösung	3952	0.16	2084	543	1325
Erreichbarkeit	2603	0.11	1277	487	839
Leistung	2047	0.08	1355	277	415
Qualität	1468	0.06	803	319	346
Freundlichkeit	1238	0.05	305	77	856
Kompetenz	621	0.03	338	110	173

Table 9: Entity and sentiment frequencies of predictions for Allianz data, in descending order regarding entity frequencies.

Entity	Absolute Frequency	Relative Frequency	Negative Sentiments	Neutral Sentiments	Positive Sentiments
Beratung	8651	0.35	2386	1422	4843
Preis	4683	0.19	1813	781	2089
Problemlösung	3464	0.14	256	457	1751
Erreichbarkeit	2297	0.09	767	386	1144
Leistung	2070	0.08	1038	271	761
Qualität	1538	0.06	574	295	669
Freundlichkeit	1306	0.05	180	52	1074
Kompetenz	835	222	0.03	138	475

Table 10: Entity and sentiment frequencies of predictions for HUK data, in descending order regarding entity frequencies.

4.5 Aspect-free Sentiment Classification

If no aspects were found within a review text, we could not employ aspect-based methods for sentiment classification. Thus, we used a multi-class classifier to predict the aspect-free sentiment for these reviews. It was multi-class instead of multi-label prediction this time because we wanted to predict exactly one label per review.

Training For this task we trained a German DistilBERT model on those parts of the

review-wise labelled data that were not labelled with any aspects. The training code for this task can be found in the notebook called `distibert4sentiments.ipynb`. We balanced the training set so that there were 500 reviews for each of the three sentiments negative, neutral and positive. The validation set consisted of 100 reviews per sentiment class. No further preprocessing than the usual one plus encoding sentiments was needed. We set the maximum sequence length to 200 for the following models. The hyperparameters we modified were number of epochs (E), learning rate (LR) and batchsize (BS). Varying these hyperparameters led to the following results:

E	LR	BS	Accuracy = F1 (micro)	F1 (macro)	Training Time for 1 epoch (sec.)	GPU (Tesla)
28	2e-05	8	0.67	0.67	18	P100-PCIE-16GB
28	2e-05	64	0.65	0.64	15	P100-PCIE-16GB
30	2e-05	32	0.66	0.66	47	K80
30	4e-05	64	0.65	0.64	49	K80

Table 11: Performance and training time comparison for several DistilBERT models for aspect-free sentiment classification.

As one can see, the performance does not really change for varying hyperparameters and the fixed amount of data. Nonetheless, for the pipeline we chose the model called `distilbert_sentiments_ntrain1500_nvalid300_e28_lr2e-05_bs8_ml200_owoa_bal` because it was slightly better than the rest. It is interesting to see that the training time varies strongly depending on the assigned GPU.

Pipeline On the data we used the same preprocessing methods as usual. In order to measure the performance, sentiment labels had to be encoded. In the following table, we see the number and the proportion of each predicted sentiment for Allianz data:

Sentiment	Absolute Frequency	Relative Frequency
positive	1235	0.61
neutral	447	0.22
negative	331	0.16
sum	2013	1.00

Table 12: Absolute and relative frequencies of aspect-free sentiment predictions for Allianz data.

Comparing these predictions with the true labels, we reached a test accuracy of 76% and a test F1-Score (macro) of 64%. Again, these values might be too optimistic as training

set and company data sets might overlap. We observed a similar distribution of predicted sentiments for HUK which resulted in a test accuracy of 78% and a test F1-Score (macro) of 61%:

Sentiment	Absolute Frequency	Relative Frequency
positive	1632	0.69
neutral	502	0.21
negative	220	0.09
sum	2354	1.00

Table 13: Absolute and relative frequencies of aspect-free sentiment predictions for HUK data.

Although these sentiments cannot be assigned to aspects and entities, we also considered them in the scores in the next section and in our final plots in Section 4.7.

4.6 Scores

We came up with several scores in order to deal with special subgroups of reviews and to see which one shows our results in the best way.

Aspect-free Score For reviews without any aspects, we predicted aspect-free sentiments with the frequencies of Tables 12 and 13. We used the absolute values to calculate an aspect-free score with the following formula

$$aspect\_free\_score = \frac{1}{N\_without\_aspects} \sum_{\substack{review \\ without \\ aspects}} sentiment\ (review),$$

where *review* corresponds to a review without aspects and $N\_without\_aspects$ is the total number of them. With $sentiment\ (review) \in \{-1, 0, 1\}$ we mean the sentiment of each review, encoded for negative, neutral and positive, respectively. This score is basically an average over the sentiments with a lower bound of -1 and an upper bound of 1. It is not part of the radar chart directly as it does not belong to any entity. Thus, it can be found above the chart, as we will also explain in Section 4.7.

For Allianz data the aspect-free score is 0.4491 and for HUK data 0.5998. This means that the reviews without aspects are more positive in average for HUK data than for

Allianz data. For both companies they indicate that the customers of these reviews tend to have a rather positive opinion about the companies in general.

Unweighted Scores For the remaining number of reviews with aspects, we calculated a score for each entity. This formula is actually the same as the one for the aspect-free score, but in this case, we just take those sentiments into account that correspond to the chosen entity. The numbers of positive, negative and neutral sentiments per entity can be found in Tables 9 and 10. As the connecting point between sentiments and entities is the aspect, we sum over the corresponding aspects here:

$$score\ (entity) = \frac{1}{N\_entity} \sum_{\substack{aspect \\ assigned \\ to\ entity}} sentiment\ (aspect).$$

So, N_entity is the number of aspects assigned to the *entity* and the $sentiment\ (aspect) \in \{-1, 0, 1\}$ is the sentiment belonging to an aspect of this *entity*. These scores are depicted as blue lines in the radar charts, more precisely in the left graphs of Figures 3 and 4.

Scores weighted by total As the aspect-entity matching list caused unequal assignments of entities to aspects in Section 4.4, it may appear that there are entities with thousands of corresponding sentiments and entities with only a few hundred. According to Insaas, this inequality should be seen in the scores such that scores with more sentiments get a higher weight. Inspired by [14], we used this formula

$$score_{w\_total}\ (entity) = score(entity)\ \frac{N\_entity}{\sum_{entity} N\_entity},$$

where the weights were the relative frequencies of the entities in the company data sets. These can be found in the third columns of Tables 9 and 10. Unfortunately, these weights seem to be too strong as they draw scores with a relatively low frequency very close to zero. For details, take a look at the third columns of Tables 14 and 15. Because of their very small values we did not draw them in the radar charts of Section 4.7.

Scores weighted by maximum In order to improve the weighted scores, we changed the denominator of the weights from above. We used the maximum of the frequencies instead:

$$score_{w\_max}\ (entity) = score\ (entity)\ \frac{N\_entity}{\max_{entity} N\_entity}.$$

This resulted in a weight of 1 for the most frequent entity, i.e. *Beratung*. Thus, the weighted scores are not as small as above. They can be found in the radar charts on the right-hand side of Figures 3 and 4. Nonetheless, it is still debatable whether it makes

sense to include frequencies of other entities into the score of an entity.

Here are the entity-based scores for Allianz data:

Entity	Unweighted Scores	Scores weighted by total	Scores weighted by maximum
Freundlichkeit	0.4451	0.0224	0.0680
Beratung	-0.0248	-0.0082	-0.0248
Erreichbarkeit	-0.1683	-0.0178	-0.0541
Problemlösung	-0.1921	-0.0308	-0.0937
Preis	-0.2305	-0.0430	-0.1308
Kompetenz	-0.2657	-0.0067	-0.0204
Qualität	-0.3113	-0.0186	-0.0564
Leistung	-0.4592	-0.0382	-0.1160

Table 14: Scores for Allianz data, in descending order regarding unweighted scores.

It is interesting to see that these scores are definitely lower than the aspect-free score of 0.4491. Additionally, they are all negative, except for the ones for *Freundlichkeit*. For HUK data, the opposite can be observed: All scores apart from the ones for *Leistung* are positive. Also the ranking of the entities has changed, but only the highest score surpasses the aspect-free score of 0.5998, as this table shows:

Entity	Unweighted Scores	Scores weighted by total	Scores weighted by maximum
Freundlichkeit	0.6845	0.0360	0.1033
Kompetenz	0.3030	0.0102	0.0292
Beratung	0.2840	0.0989	0.2840
Erreichbarkeit	0.1641	0.0152	0.0436
Problemlösung	0.1429	0.0199	0.0572
Qualität	0.0618	0.0038	0.0110
Preis	0.0589	0.0111	0.0319
Leistung	-0.1338	-0.0111	-0.0320

Table 15: Scores for HUK data, in descending order regarding unweighted scores.

4.7 Radar Charts

The scores from above were plotted into so-called radar charts or spider diagrams. As the scores weighted by total did not differ a lot from each other, we only show the unweighted scores on the left and the scores weighted by maximum on the right. There is one corner and one axis for each entity; the absolute entity frequency is given in brackets. The entity scores are depicted as points on their corresponding axes and then the points are connected by blue lines. On top of the charts, one can find the total amount of reviews and the aspect-free score with its corresponding number of reviews. The remaining reviews form the basis for the radar charts.

Here are the final customer centricity graphs for Allianz:

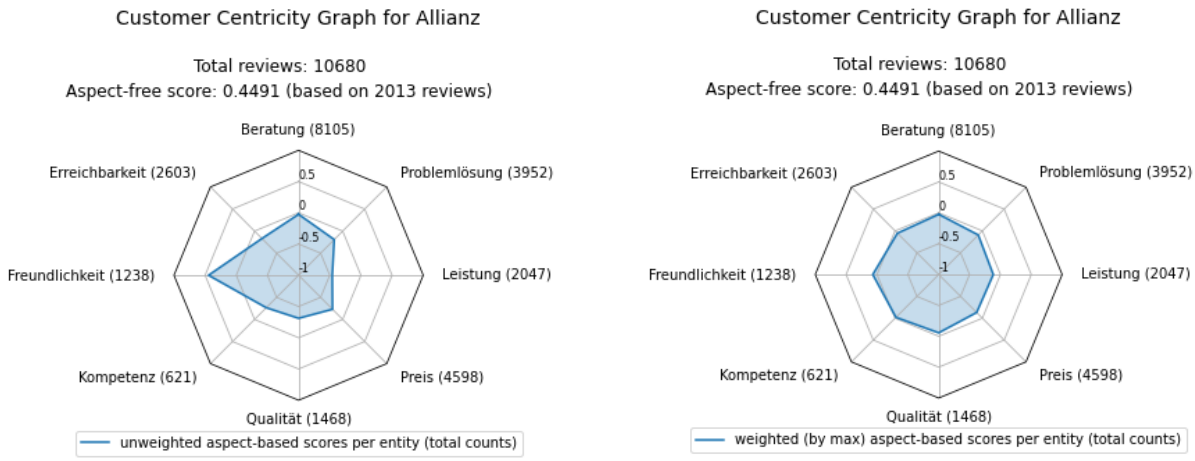

Figure 3: Customer Centricity Graphs for Allianz data; entity scores unweighted and weighted by maximum.

Whereas *Freundlichkeit* can be clearly identified as highest-ranked entity and *Leistung* as lowest-ranked one on the left-hand side, these tendencies are barely visible in the plot on the right-hand side. This strengthens our doubts about the effectiveness of our weighted scores. The same can be observed for HUK data in Figure 4. Comparing only the graphs with unweighted scores, we can see that both companies received the best scores for *Freundlichkeit* and the worst for *Leistung*. That is what we already discovered when looking at Tables 14 and 15. The graphs also show the results that HUK scores were much higher in general than Allianz scores. This is also mirrored in the size of the blue areas: The one for HUK is definitely bigger.

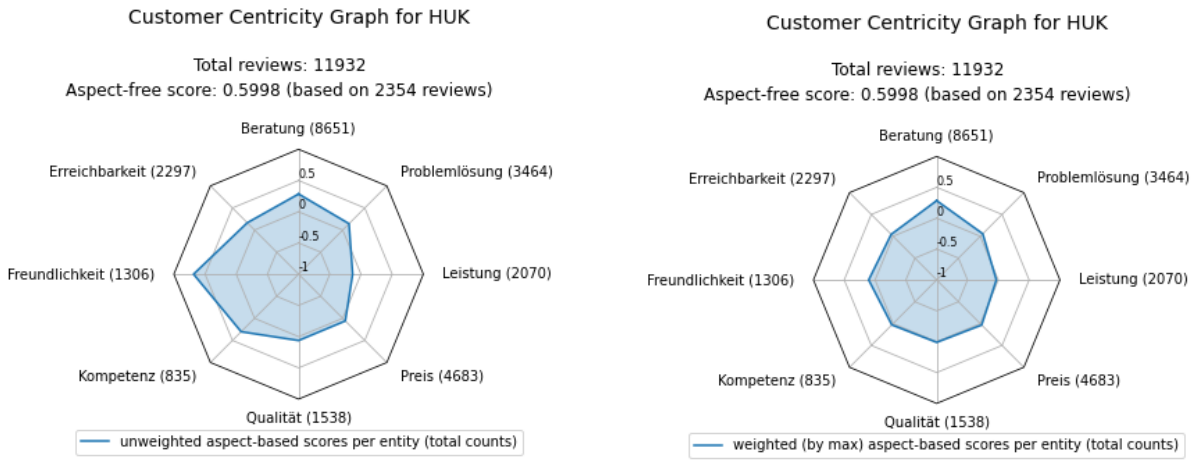

Figure 4: Customer Centricity Graph for HUK data; entity scores unweighted and weighted by maximum.

5 Conclusion

Using state-of-the-art methods from the field of Natural Language Processing, we were able to set up a pipeline that takes review data as input and returns a customer centricity graph. The procedure includes aspect detection and sentiment classification for both single aspects and whole reviews. These goals were reached by employing pretrained German languague models based on BERT. For aspect-entity matching unsupervised similarity methods were chosen. Turning all these information into scores that could be depicted in radar charts was done by more or less simple averaging over the sentiments of a certain group of reviews. The DistilBERT classification models for both aspects and aspect-free sentiments performed decently. Although we could not measure the performance of the ABSA classifier and the aspect-entity matching, they made the pipeline work.

As our aim was to set up a working approach and we did not have to beat any benchmarks, there are still several issues that could or should be improved. First of all, each of the steps (maybe except for the aspect detection) has the potential to perform better. Especially, the aspect-entity matching can be taken to the next level by a context-based approach. This might also solve the issue with unequally frequent entities. In case this problem still persists, the visualisation of the entities could be improved in the graph, for example by adjusting the angles of the entities according to their proportion of all entities. Another way of visualisation could be to add the height as a new dimension to the radar chart. The higher a score is placed in this dimension, the more entities it is based on. Despite these issues, we can say in conclusion that we managed to establish a working infrastructure that is able to extract valuable and differentiating information from review data. Based on our work, Insaas may improve the results by more tuning or more sophisticated models.

In the following, we mention some possible extensions that came up during the project and might be included in the future. One idea is to integrate the quality of each review into the scores. Following the hypothesis that reviews with good grammar and spelling show a more fine-grained and reliable opinion, we might get interesting new results. This information could be added in form of weights. What is also related to the style of writing are emojis and emoticons. These can be used to further improve the sentiment predictions.

Another interesting point could be the comparison of entity scores over time. This might be helpful for companies to see whether, for instance, a change in their service system goes down well on the customers. This may be depicted by a new dimension in the plots.

With the current state of the graphs, companies are comparable for each entity. In order to make overall comparison easier, an overall score would help. This could be calculated as a weighted sum over the entity scores. The weights may depend on entity frequencies or their importance within the field of car insurances.

It would also be interesting to create a data set out of reviews from several companies and take the received graph as an average benchmark for the whole car insurance industry. To build this data set, it would be necessary to determine which companies contribute in which amount to the standard of car insurances. Maybe these are the five biggest companies or all companies depending on their proportion of sold insurances or customers. The average benchmark could then be used to centre the company-wise scores. This may also solve the issue of reviews being mainly about extreme opinions that we mentioned in the introduction.

Another idea deals with the set of entities: Having the given set of eight entities made our work rather easy. Of course, this does not always suffice to cover realistically all the categories people are talking about. So it might be necessary to add new entities, for example from the field of Marketing and Sales. Generalising these entities may also make the approach applicable to other fields of industries.

So, there are still a lot of ideas left to be turned into reality. We are interested to see how Insaas will include them into their model in the future.

List of Figures

1 Overview about the steps conducted throughout the project. 4

2 Aspect-Entity Matching 19
3 CCGs for Allianz 26
4 CCGs for HUK 27

List of Tables

1 Review-wise labelled data 5
2 Aspect-wise labelled data 6
3 Entity Synonyms 8

4 SRD Example 12

5 Performance German models 14
6 Comparison of Aspect Classifcation Models 15
7 Performance of ABSA Models 17
8 Entity Frequencies in Aspect-Entity List 20
9 Entity Frequencies for Allianz 21
10 Entity Frequencies for HUK 21
11 Comparison of Aspect-free Sentiment Models 22
12 Aspect-free Sentiment Prediction Frequencies for Allianz 22
13 Aspect-free Sentiment Prediction Frequencies for HUK 23
14 Scores for Allianz 25
15 Scores for HUK 25

16 Pipeline Files 34
17 Training Files 35

Bibliography

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[2] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. 2006.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] Yoav Goldberg. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. 2015.

[6] Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. Adapt or get left behind: Domain adaptation through BERT language model finetuning for aspect-target sentiment classification. *CoRR*, abs/1908.11860, 2019.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[8] Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. Attentional encoder network for targeted sentiment classification. *CoRR*, abs/1902.09314, 2019.

[9] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975, 2016.

[10] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. *CoRR*, abs/1903.09588, 2019.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[12] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[13] Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. BERT post-training for review reading comprehension and aspect-based sentiment analysis. *CoRR*, abs/1904.02232, 2019.

[14] Boya Yu, Jiaxu Zhou, Yi Zhang, and Yunong Cao. Identifying restaurant features via sentiment analysis on yelp reviews. *CoRR*, abs/1709.08698, 2017.

[15] Biqing Zeng, Heng Yang, Ruyang Xu, Wu Zhou, and Xuli Han. Lcf: A local context focus mechanism for aspect-based sentiment classification. *Applied Sciences*, 9:3389, 2019.

Python Requirements

Package	Version
Python	3.6.9
attrdict	2.0.1
fasttext	0.9.2
keras	2.4.3
matplotlib	3.2.2
numpy	1.18.5
pandas	1.1.4
scipy	1.4.1
sklearn	0.0
tensorflow	2.3.0
torch	1.7.0+cu101
tqdm	4.41.1
transformers	3.4.0

Digital Appendix

Here, we give a short overview about the files that we worked with and that are provided in the folder `PipelineFiles_v2`. Table 16 deals with the pipeline, i.e. the application of trained models on specific data. Table 17 shows which notebooks were used for which training step and which other files they required. A similar overview is also included in `overview.txt` in the same folder.

A. Files used in Pipeline

File Name	Description
`Review_Aspect_28072020.csv`	review-wise labelled data
`distilbert_ntrain8291_nvalid3311_e15_lr8e-05_bs12_ml512_maf2_ait198_bal.pt`	aspect detection model
`distilbert_ntrain8291_nvalid3311_e15_lr8e-05_bs12_ml512_maf2_ait198_bal_labels.txt`	labels (list of aspects) for aspect detection model
`lcf_bert_ntrain1076_nvalid269_e10_lr2e-05_bs8_ml240_lca_dr0_srd3_l2reg1e-05_hiddim768_embdim768_full.pt`	LCF-BERT model
`tokenizer_lcf_bert_ntrain1076_nvalid269_e10_lr2e-05_bs8_ml240_lca_dr0_srd3_l2reg1e-05_hiddim768_embdim768_full.pt`	LCF-BERT tokenizer
`lcf_bert_ntrain1076_nvalid269_e10_lr2e-05_bs8_ml240_lca_dr0_srd3_l2reg1e-05_hiddim768_embdim768_opt.txt`	config file for LCF-BERT model
`Aspect_Lemmatization_List_9_15-2.xlsx`	latest lemmatisation list
`distilbert_sentiments_ntrain1500_nvalid300_e28_lr2e-05_bs8_ml200_owoa_bal.pt`	aspect-free sentiment model
`aspect_entity_matching.csv`	aspect-entity pairs for aspects from aspect detection model

Table 16: Files needed to run `pipeline.ipynb`.

B. Files used for Training

Pipeline Step	Notebook Name	Needs
Aspect Detection	distilbert4aspects	Review_Aspect_28072020.csv, Lemmatization_list.xlsx
Aspect-based Sentiment Classification	lcf_bert	Review Classifier - Trainingsset-CCG-3.xlsx, Aspect_Lemmatization_ List_9_15-2.xlsx
Aspect-based Sentiment Classification	aen_bert	Review Classifier - Trainingsset-CCG-3.xlsx, Aspect_Lemmatization_ List_9_15-2.xlsx
Aspect-free Sentiment Classification	distilbert4sentiments	Review_Aspect_28072020.csv
Aspect-Entity Matching	similarity_fasttext	distilbert_ntrain8291_nvalid3311_e15_lr8e-05_bs12_ml512_maf2_ait198_bal_labels.txt, Synonym_3.0_restructured.xlsx, Entity_Aspect_ Freq_form_col.csv
Aspect-Entity Matching	bert_ner_final	Review Classifier - Trainingsset-CCG-3.xlsx, Aspect_Lemmatization_ List_9_15-2.xlsx

Table 17: Files used for training single models.