That's what the data said: An NLP Analysis of Script Lines from the US TV-Show "The Office"

Ausarbeitung zur Vorlesung Aktuelle Data Science-Entwicklungen: Intelligent Text Analysis

für die
Prüfung zum Bachelor of Science

an der Fakultät für Wirtschaft
im Studiengang Wirtschaftsinformatik

an der
DHBW Ravensburg

Verfasser:	Moritz Schlager (2780007)
	Timo Heiß (3739454)
Kurs:	WWIDS120
Dozent:	Oliver Sampson
Abgabedatum:	20.03.2023

1 Introduction

> ***Holly***: *So you missed something really big.*
> ***Michael***: *It was bigger than I thought.*
> ***Holly***: *Well that's what you said-*
> ***Michael***: *No I didn't say it. Oh no no I didn't say it. Why are you smiling so much?*
> *[...]*

The scene above is a short conversation between Michael Scott and Holly Flax, two characters in the US TV-show "The Office". A fan of the show may, however, note that this scene does not appear in any of the released episodes. Instead, it is generated by an AI. Even though this scene does not have much content, some elements of "The Office" can be found, such as the fact that the AI is (almost) referencing the famous catchphrase "That's what she said". The model that generates scenes like the above is only one achievement of this paper.

More generally, we analyze a dataset containing script lines of the US TV-show "The Office" with various traditional and modern approaches of Natural Language Processing (NLP). "The Office" is a comedy TV series that follows the daily lives of a group of office workers at the "Dunder Mifflin" paper company. The show is shot in the style of a mockumentary with a camera-crew following the office workers and showing interactions, conflicts and humor.

Our objective now is to apply methods of NLP in order to gain interesting insights into the show and its characters by only looking at "what the data says". More specific, we analyze characters, relationships, sentiments and topics to identify speaking styles and developments. We train models to generate scenes (such as the scene above) and to classify the speaker of a line. The results of this paper should be consistent with the storyline of the show and provide additional insights both for fans and for people who did not watch the show.

The structure of the paper can be described as follows: In Chapter 2, we present the dataset we used and conduct a short exploratory analysis to understand the data and assess its quality. In chapter 3, we prepare the data for a further analysis and modeling. This includes various preprocessing approaches of NLP as well as feature extraction methods. In Chapter 4, we conduct an analysis with NLP methods including speaker analysis, word analysis, sentiment analysis, topic modeling and network analysis. In Chapter 5, we train Large Language Models to, on the one hand, generate new scenes for the TV-show and, on the other hand, to predict the speaker for a given line. In the final Chapter 6, we summarize our results and draw a conclusion.

2 Data Understanding

For our analysis, we use a dataset from Ralhan (2018). The first five rows are shown in **Tab. 2.1**. The dataset contains lines from all seasons of the US TV-show "The Office", including some deleted scenes.

Tab. 2.1: First five rows of the dataset

id	season	episode	scene	line_text	speaker	deleted
1	1	1	1	All right Jim. Your quarterlies...	Michael	FALSE
2	1	1	1	Oh, I told you. I couldn't close...	Jim	FALSE
3	1	1	1	So you've come to the master...	Michael	FALSE
4	1	1	1	Actually, you called me in here...	Jim	FALSE
5	1	1	1	All right. Well, let me show you...	Michael	FALSE
...	...	...	...	...	...	...

In its unprocessed, raw format, the dataset contains 59911 samples or lines. All columns with their respective data types and descriptions are shown in **Tab. 2.2**. A further analysis of the values in the dataset verifies that our data covers all nine seasons of the show. The maximum number of episodes per season is 26 with up to 116 scenes. 52187 of the aforementioned 59911 lines are unique. "Yeah." is the most frequently spoken line with a total of 486 occurrences. There are 794 unique speakers throughout the show. The most frequent speaker is "Michael" with 12137 lines. 57975 lines of the dataset actually made it into the final version of the show.

Tab. 2.2: Data Schema

Column	Data Type	Description
id	Integer	unique identifier for each line
season	Integer	season of the show in which the line appeared
episode	Integer	episode in which the line appeared (within the respective season)
scene	Integer	scene in which the line appeared (within the respective episode)
line_text	String	content of the line (with additional directorials in square brackets)
speaker	String	speaker of the respective line in the show
deleted	Boolean	shows if the line has been deleted (not in the final version of the show)

In the next step, we check the data quality of our dataset. The only major issue is that some line breaks in the original CSV are not working correctly, which we fix manually. Moreover, we identify limitations of our dataset: In very few scenes some lines are missing, which could be discovered by watching the show and comparing it to the dataset. Also, only few directorials are included, which can by far not describe everything that happens on screen. Thus, there is a semantic gap with some context being lost. That could make the analysis more difficult. Furthermore, we face issues when there is a "show within a show" (e.g. when characters are watching TV). These scenes are displayed in an unclear, inconsistent way. Also, sometimes a speaker has different names, which can be considered a data quality issue.

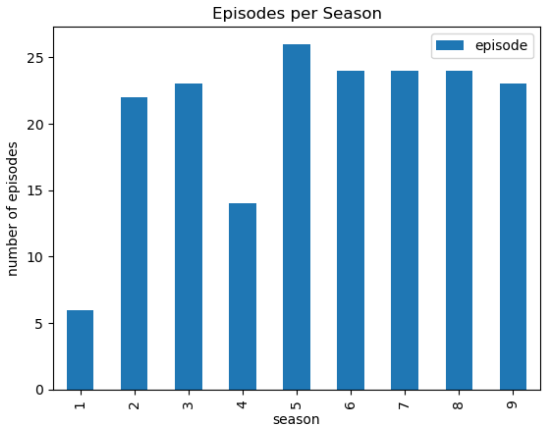

Fig. 2.1: Number of episodes per season

When plotting the number of episodes per season in (see **Fig. 2.1**), we identify differences compared to numbers given by Wikipedia (2023b). Further research, however, showed that there are no missing, but double episodes[1] are counted as one in our dataset, whereas they are counted as two on Wikipedia.

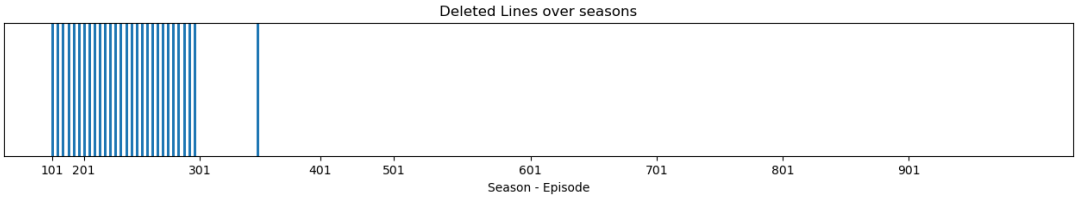

Fig. 2.2: Deleted lines in the dataset over the seasons

When looking at the distribution of the deleted lines over the seasons (**Fig. 2.2**), we can see that they only occur in the first two seasons (and once in season 3). Therefore, and since they may provide an distorted picture of a scene if their content extremely differs from the actual lines, deleted scenes will not be considered in our analysis.

For some analysis tasks, we also use data containing the full name of each character in the show mentioned on Wikipedia (2023a). In addition, we use a list of compound words that appear in "The Office" such as "Accounting department" or "Dunder Mifflin" generated by ChatGPT[2]. Both of the additional datasets are used in preprocessing steps, which will be described in the next chapter.

[1] Double episodes are episodes that belong together in terms of content and share the same title.
[2] https://chat.openai.com/

3 Data Preparation

In the course of this paper, we mainly use the libraries nltk as proposed by Loper and Bird (2002) and sklearn as proposed by Pedregosa et al. (2011). To make reading more convenient we will not cite the respective documentations in the text. Other libraries and their documentations will be cited when used.

3.1 Preprocessing

The preprocessing of the previously described data involves many independent optional steps. It heavily depends on the analysis task, which kind of preprocessing is needed. To standardize this process and make the handling easier, we create a parameterizable preprocessing function. The following paragraphs describe the distinct functions orchestrated depending on the preprocessing configuration.
Generally, as justified above, we remove all deleted scenes as well as the column "deleted". As a first option, we offer to **concatenate scenes** by appending all lines of a scene with its respective speaker to one string and grouping the dataset to only contain one row per scene. This allows an analysis of speech in a larger unit and can for instance be used to look at term frequencies. Moreover, we provide a function to **extract directorials**, i.e. all words written in square brackets, into a separate column. Usually, these are descriptions of what the characters do or how they behave in certain scenes. Additionally, preprocessing includes the following functions, which can all be categorized as bare string processing tasks:

- **remove punctuation**: only matches word and whitespace characters using the regex [^\w\s]
- **lower**: changes all letters to lowercase
- **remove stopwords**: uses the english nltk stopwordlist to filter the corresponding words from the lines
- **expand contractions**: uses the contractions package by van Kooten (2022) to separate combined words (for example *don't* to *do not*)

Moreover there are two normalization methods to choose from. **lemmatize** calls the nltk *WordNetLemmatizer*, which uses the built-in morphy function of WordNet to only return the base of the given word. Another option is to use the *stem* function, which utilizes *PorterStemmer* to reduce given words.
To conclude the set of preprocessing options, we implement two types of conversion that can be applied to the data. The first one is to **tokenize** the lines by means of different tokenizers. We implement nltk's *TreebankWordTokenizer*, *WordPunctTokenizer* and *WhitespaceTokenizer*, which can be selected depending on the requirements of the task.

Additionally, we provide the option to specially tokenize certain words that occur in "The Office". This includes converting first and last name of characters as one token using the character names dataset described in Ch. 2. Also, special compound words that occur in the TV-show are treated as one token (e.g "Dunder Mifflin" oder "Regional Manager") making use of the compound word list (Ch. 2). The second conversion is part-of-speech tagging (**POS-tagging**), which is used to tag words according to their lexical categories. Here we apply nltk's pos_tag on tokenized lines.

3.2 Feature Extraction

In addition to the preprocessing function, we provide a separate method for **feature extraction**. This method returns a matrix of token counts by applying one of the following sklearn vectorizers:

- *CountVectorizer* returns a matrix of either binary or count representation of word appearances
- *TfidfVectorizer* returns a matrix of TF-IDF features
- *HashingVectorizer* returns a matrix of token occurrences using the hashing trick to map token strings to integer indices and normalizing the matrix

Feature extraction is a powerful technique for converting unstructured textual data into structured numerical data suitable for machine learning algorithms. Therefore, this method is utilized in the following analysis for calculating scores such as TF-IDF as well as for model training in sentiment analysis and other applications.

4 NLP Analysis

4.1 Speaker Analysis

For the following analysis, we only extract directorials in terms of preprocessing. First, we perform a frequency analysis of the speakers. We count the number of scenes in which they appear as well as the number of lines and words they say.

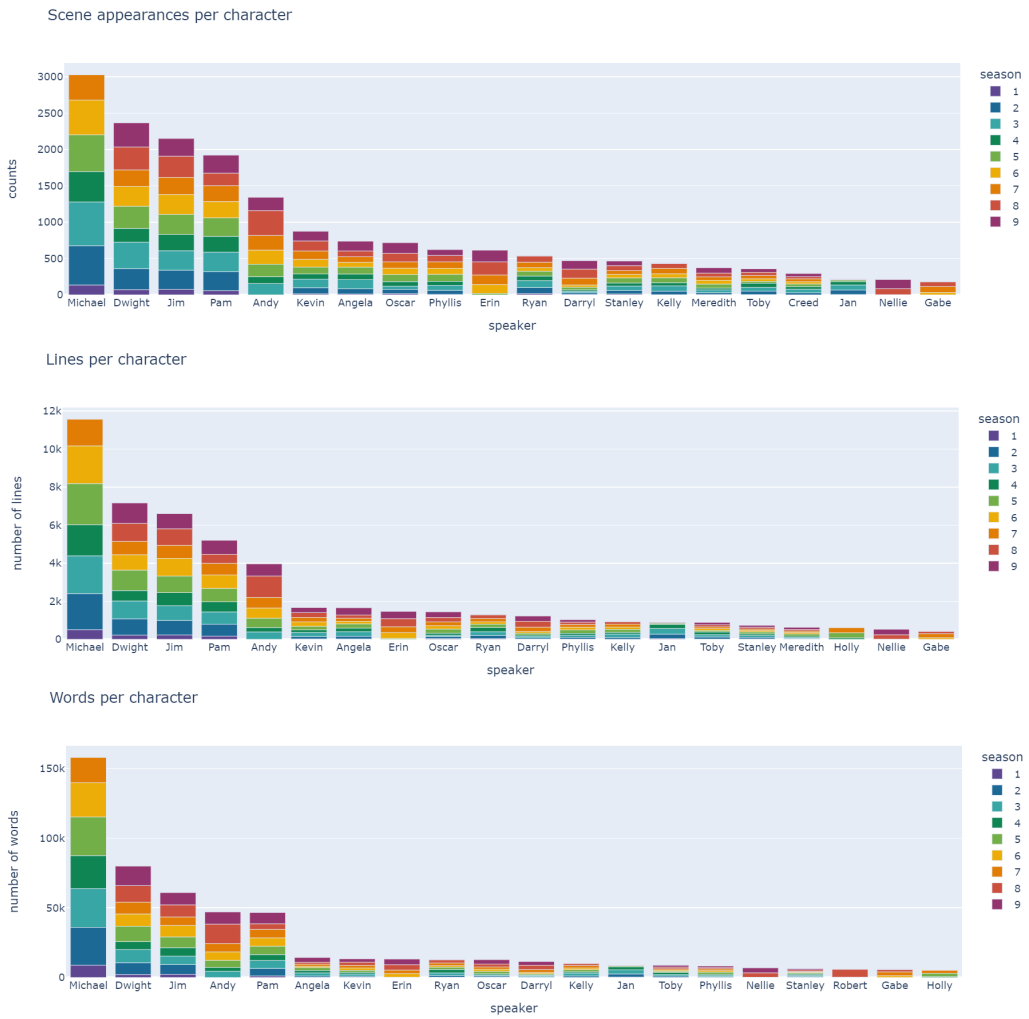

Fig. 4.1: Frequency analysis of scene appearances, lines and words per speaker and season

Fig. 4.1 shows that Michael is overall the most frequent appearing character in every analysis followed by Dwight, Jim, Pam and Andy. These five can thus be identified as the main characters. Moreover, we observe that the order of the supporting characters differs depending on the analysis. For example Kevin has more lines than Angela, while she speaks more words. This is an indication of different speaking styles. While some characters tend to narrate at length, others are rather short-winded. Furthermore, some characters only appear in specific seasons such as Michael in the seasons 1-7 or Robert in season 8.

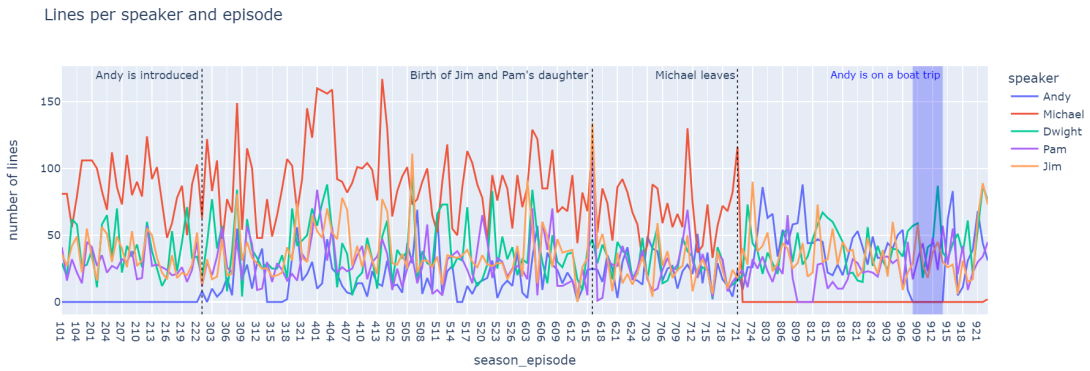

Fig. 4.2: Frequency analysis of lines per speaker in the course of the show

To include a temporal dimension, we take the previously identified main characters and plot their number of lines for each episode to display developments in the course of the show. Through this visualization, certain events in the show can be identified (see **Fig. 4.2**). Moreover, we can see that Michael clearly dominates in terms of speech portions in the first seven seasons and is then "replaced" jointly by Dwight, Jim, Pam and especially Andy, who has more lines in last two seasons.

4.2 Word Analysis

In this section, we analyze occurences of single words in lines. For this purpose, we preprocess the data by extracting directorials, removing punctuation, transforming to lowercase, expanding contractions and tokenizing lines using nltk's *TreeBankWord* and our own list special tokens (character names and compound words).

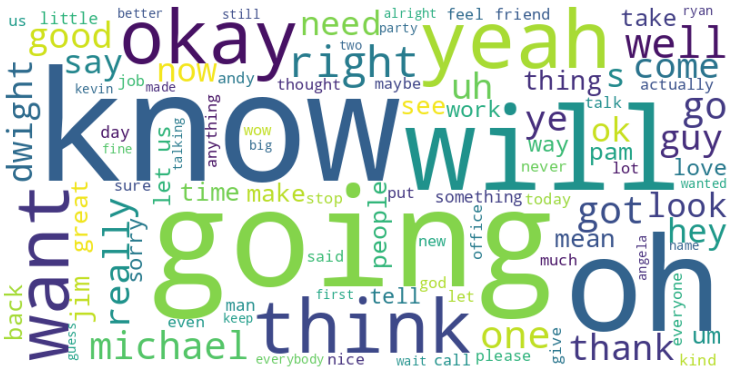

Fig. 4.3: Word cloud

First, we create a word cloud[3] from all words occurring in the script (see **Fig. 4.3**). We observe that most of the words are commonly used in everyday language (going, know, will, oh) or some character names. Words that relate to specific topics, however, are relatively rare (e.g. job, office, work, call, party, friend, love). This can be confirmed by counting the words in the script, which leads to generally used words like "I", "you"

[3]implementation by Andreas Mueller et al. (2018)

or “is“. After stopword removal, we receive similar most frequent words (e.g. know, oh, going). In an attempt to find more meaningful important words, we use POS-tagging in our preprocessing function to tag words with their respective lexical category [4]. When filtering for the most frequent nouns, we do not receive sufficient additional information compared to the word cloud. For the adjectives, however, we get results shown in **Fig. 4.4**. These adjectives are mainly positively-connoted. From this, we can derive that the show generally has a positive tone, which aligns with the genre as a comedy.

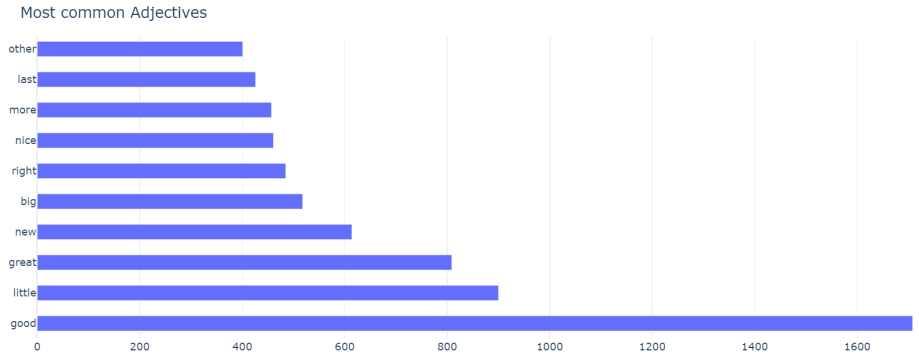

Fig. 4.4: Most frequent adjectives (uses POS-tagging)

In a next step we compare the speaking styles of Dwight and Micheal by looking at their most fequent words, we can notice only small differences. More detailes are revealed by looking at the Trigrams found by nltk (see **Fig. 4.5**). We can detect characteristic sentences for Michael like ”oh, god, oh” or nonsense like ”beep, beep, beep”. Dwight’s speaking style features repeating names like ”jim, jim, jim” or ”michael, michael, michael” and also things that suit to him as a character (”volunteer, sheriffs, deputy”).

ngram Michael	ngram Dwight
((let, us, go), 75)	((let, us, go), 38)
((let, us, get), 35)	((hey, hey, hey), 21)
((hey, hey, hey), 26)	((yes, yes, yes), 17)
((come, let, us), 23)	((let, us, get), 15)
((oh, god, oh), 21)	((ha, ha, ha), 13)
((beep, beep, beep), 21)	((go, go, go), 13)
((let, us, see), 18)	((jim, jim, jim), 12)
((god, oh, god), 17)	((wait, wait, wait), 10)
((stop, stop, stop), 16)	((whoa, whoa, whoa), 9)
((na, na, na), 16)	((la, la, la), 9)
((go, let, us), 13)	((let, us, see), 7)
((yeah, yeah, yeah), 13)	((michael, michael, michael), 7)
((right, right, right), 13)	((zero, zero, zero), 7)
((blah, blah, blah), 13)	((volunteer, sheriffs, deputy), 6)
((go, go, go), 13)	((one, two, three), 6)

Fig. 4.5: Trigrams comparison for Michael and Dwight

Lastly, we use a yellowbrick dispersion plot[5] to visualize the occurances of certain words throughout the show. First, we consider locations: Scranton, where most of the show takes place, is mentioned throughout the show. Stamford, another branch of Dunder Mifflin,

[4] We also applied a self-trained Brill tagger, but the results were almost undistinguishable.
[5] implementation by Bengfort and Bilbro (2019)

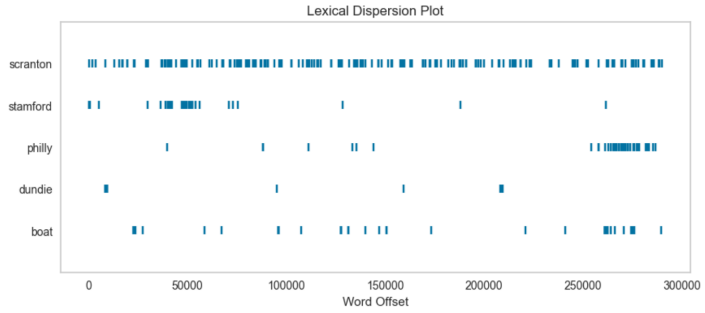

Fig. 4.6: Dispersion plot for occurrences of certain words throughout the show

mainly occurs in the earlier part, as Jim (one of the main characters) is transferred to this branch for some time. "Philly" (short for Philadelphia) is often said towards the end of the show as Jim temporarily joins a company there. The "dundies" (annual award ceremony to acknowledge the achievements of employees), appear at single occasions, mostly when this event actually takes place in the show. The word "boat" is frequently said towards the end of the show when Andy leaves for a boat trip.

4.3 Sentiment Analysis

In the following analysis, we want to find out which sentiments dominate in the show and how they develop over time. Sentiment analysis identifies and quantifies affective states and subjective information (Mejova, 2009). There are several libraries and pre-trained models to perform this task. We first use nltk's SentimentIntensityAnalyzer to detect negative, neutral and positive sentiments. In **Fig. 4.7**, we perform an analysis of the episode "Goodbye Michael". Note, that a moving average is applied to flatten peaks. Since Michael is one of the most influential characters in the show, his leaving causes much emotion. Positive and negative sentiments often change in this episode as a goodbye causes mixed feelings. We also notice a positive peak at the end when Michael says goodbye to Pam, which is a really positively-connoted scene.

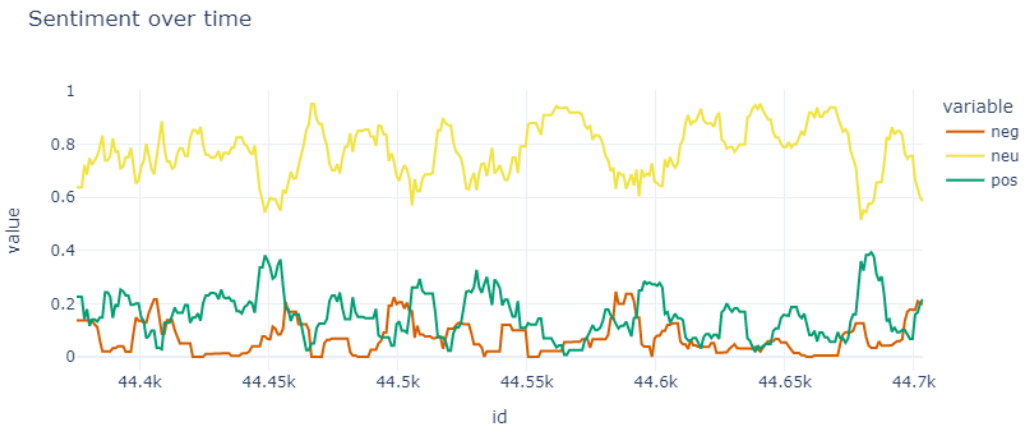

Fig. 4.7: Sentiment Analysis for the episode "Goodbye Michael" (S7E22)

For even more details, we use a pre-trained DistilRoBERTa-base-transformer model, which is already fine-tuned on transcripts from the show "Friends" Li (2022). It can predict six emotions and a neutral class. These emotions include anger, disgust, fear, joy, neutrality, sadness and surprise. For the following analysis, we filter the data to contain only scenes in which the two considered characters appear exclusively and thus talk to each other. After applying the model to the filtered lines data, we use a softmax function on the returning tensors to retrieve a probability distribution of all classes for the corresponding lines. To visualize the results, we apply a moving average to the data to emphasize a trend throughout the overall values.

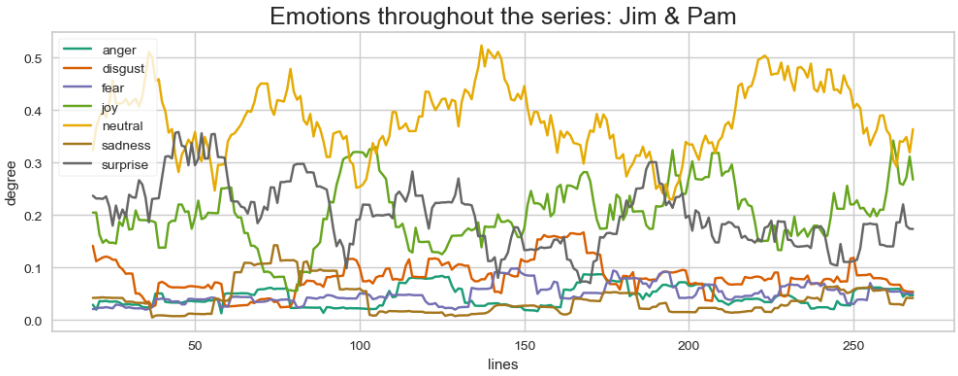

Fig. 4.8: Relationship of Jim and Pam throughout the show

The analysis of the relationship between Jim and Pam (see **Fig. 4.8**) reveals that their interaction appears to be an emotional rollercoaster, reflecting the evolution of their romantic involvement. Notably, the plot shows a distinct peak in joy at line 101 of their common scenes, when Jim asks Pam out on a date. However, due to the smoothing effect of the moving average, specific events are not always distinguishable in the plot.

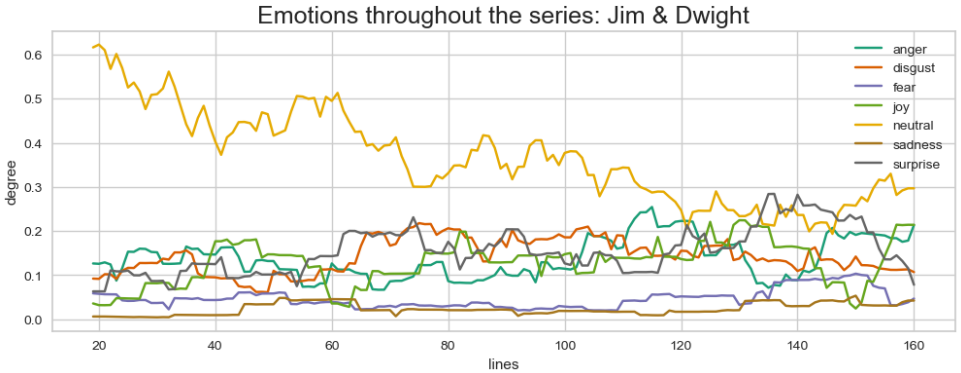

Fig. 4.9: Relationship of Jim and Dwight throughout the show

When looking at Jim and Dwight (see **Fig. 4.9**), we notice that the relationship is becoming less neutral in the course of the show. This is consistent with the storyline where they develop a special kind of "friendship" towards the end.

4.4 Topic Modeling

In this section, we want to discover what topics are talked about in "The Office". For this task, we use different approaches of topic modeling. Our first more traditional method uses Latent Dirichlet Allocation (LDA). One time, we use LDA on a vector with TF-IDF scores and once on a vector with word counts. Both times, we run the LDA for ten components, which does not give sufficient results. In our next approach, we use a SentenceBERT-transformer to create embeddings, reduce the dimensionality with UMAP and cluster the results with HDBSCAN. We then analyze the topics by retrieving the top words per topics using c-TF-ID scores. However, this does not bring interesting insights, which is mainly due to the large number of topics (ca. 60). Thus, we iteratively reduce the number of topics by merging the most similar ones (by cosine similarity). This leads to less but also more general topics, which hardly adds value to our analysis.

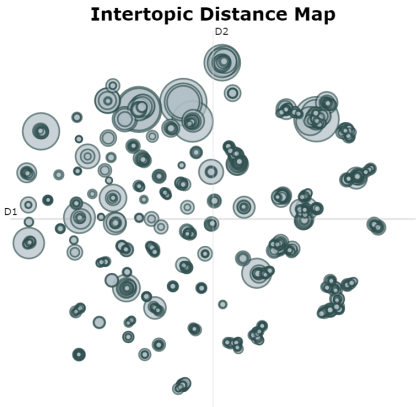

Fig. 4.10: Intertopic distance map for the topics by BERTopic

Our third approach is using BERTopic[6], which implicitly performs the previously described steps, but in an optimized way (Dua, 2021). We then visualize our topics in an *Intertopic Distance Map* (see **Fig. 4.10**). Even though there are already some interesting topics (e.g. music, guitar, tunes, piano, play), over 900 topics are still confusing. Also, with overlapping clusters, the plot indicates that the number of topics can be further reduced (e.g. (sing, song, sang, singing, soul) overlaps with the above mentioned music-topic). Therefore, we reduce the number of topics to 30, which is much easier to analyze. However, this comes with the tradeoff of more impure topics.
We also used BERTopic to visualize selected topics over time. The results are shown in **Fig. 4.11** for three selected topics. Christmas, birthdays, etc. are relevant throughout the show with peaks for the season 2 and 6. Also, there is the wedding topic with three peaks, all in seasons when there are actual weddings in the series. The third topic about printers, copiers and other office supplies appears throughout the seasons (as the show is

[6]implementation by Grootendorst (2022)

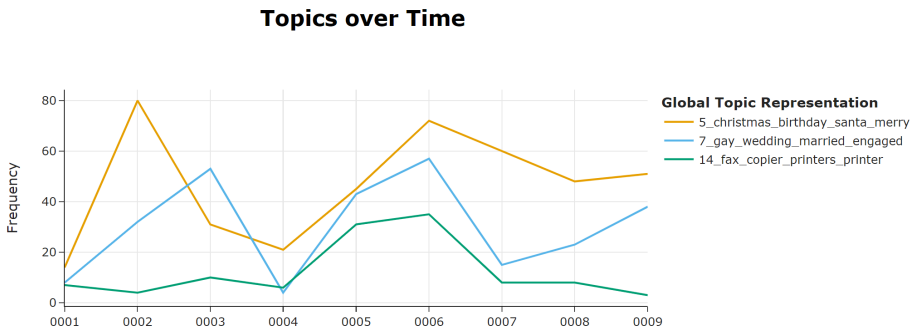

Fig. 4.11: Frequency of selected topics over time

about an office), but with a peak at the seasons 5 & 6, where the paper company "Dunder Mifflin" (aka "The Office") is bought by a printer company.

4.5 Network Analysis

For our last analysis, we want to detect which characters interact with each other and how this changes throughout the show. The data is prepared by selecting the most prominent characters. We then aggregate the scenes and extract the speakers as a set. Based on this, we create a graph that represents their relationships. The edge weight between two speakers counts shared scenes. The degree indicates how often the speaker appears. We use the spring_layout algorithm of networx[7] to position the nodes in space and visualize the graph using plotly. The width of nodes and edges shows their degree and weight.

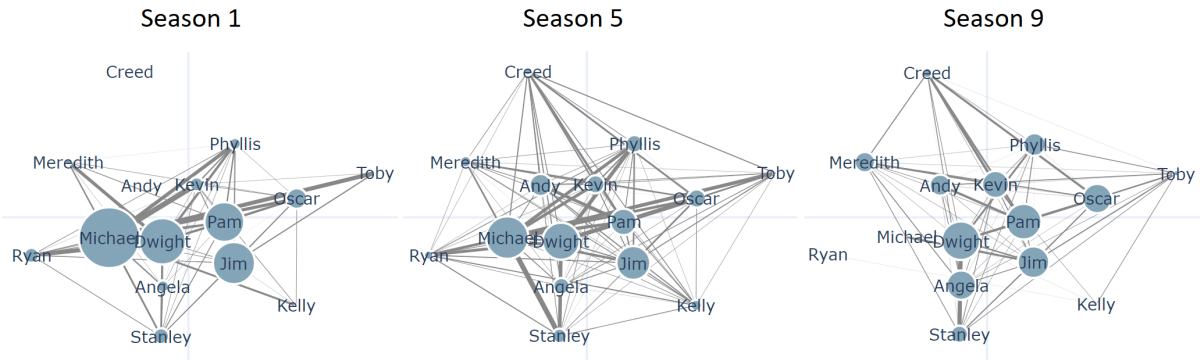

Fig. 4.12: Frequency of selected topics over time

Our analysis reveals that the main characters Michael, Dwight, Jim, Pam, and Andy are at the center of the network due to their strong connections to each other and many scenes they are featured in. Over the course of the show, the network becomes more densely connected, and the conversation share of the supporting characters increases, resulting in a more balanced network. This development can be seen in **Fig. 4.12** when comparing seasons 1 and 5. When Michael leaves the show, the other main characters take on a greater share of the conversation. This fact is displayed in the network of season 9.

[7] implementation by Hagberg et al. (2008)

5 Modeling

5.1 Speaker Classification Model

In this section, we describe and evaluate a transformer-based model to predict the speaker of a given line. We start by filtering the dataset to only contain the five most occurring characters (Andy, Dwight, Jim, Michael and Pam). Other characters were excluded from the model training due to their lack of conversation share, making it difficult to determine their special speaking style. Prior to fine-tuning the pre-trained model, we encode the training and validation data using a DistilBertTokenizer. For the model architecture, we utilize a pre-trained DistilBertForSequenceClassification[8] model. In the training process, we use an AdamW optimizer with a learning rate of 4e-5. We also try to use class weights for further optimization, which does not improve the results. A test using the sentence "Good to see you" (prediction: Pam 0.62, Michael 0.24, Jim 0.08, Andy 0.04, Dwight 0.02) shows good results, because Pam is a friendly character and therefore likely to say this sentence, while Dwight is rather quirky.
The results of the training show an accuracy score of 0.42 on the validation data. To further evaluate the performance of the model, we use a confusion matrix (see. **Fig. 5.1**). Compared to a confusion matrix of randomly distributed guesses, effects of the training can clearly be seen. There is still a bias towards the more present classes, but the model clearly learned about the specific speaking styles of the main characters.

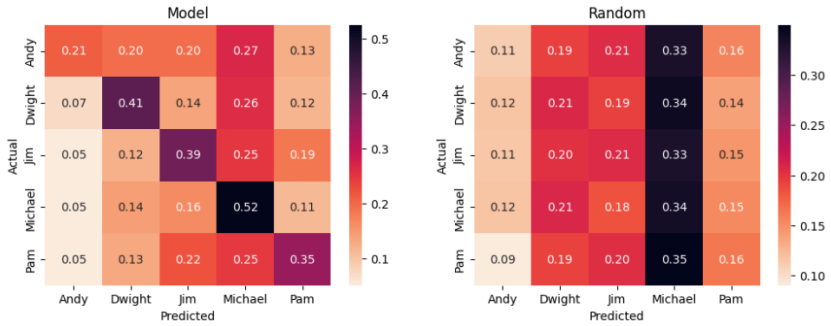

Fig. 5.1: Speaker classification model evaluation

We also use SHAP[9], an Explainable AI tool, to better understand how the model is working. This analysis reveals that the model's predictions are based on the presence or absence of certain keywords and phrases unique to each character's speaking style. In the example in **Fig. 5.2**, it becomes clear that the model has only a slight preference for short lines like "okay". As those can be said by every character, this does not represent the strengths of the model. The second example makes clear how the model is working. The magenta highlighted words contribute to the confidence of the model to predict a certain

[8] implementation by Sanh et al. (2019)
[9] implementation by Lundberg and Lee (2017)

character. For this line, the words “dundie“, ‘fat“ and “winners“ are most influential. All of these expressions represent things Michael (LABEL_3) cares about in the series and so he is predicted by the model for this line. In a similar analysis, we look at a randomly picked subset of lines. The bar chart in **Fig. 5.3** indicates that Jim seems to use “favorite“ more often than Pam, who is rather talking about “emily“ and “daughter“.

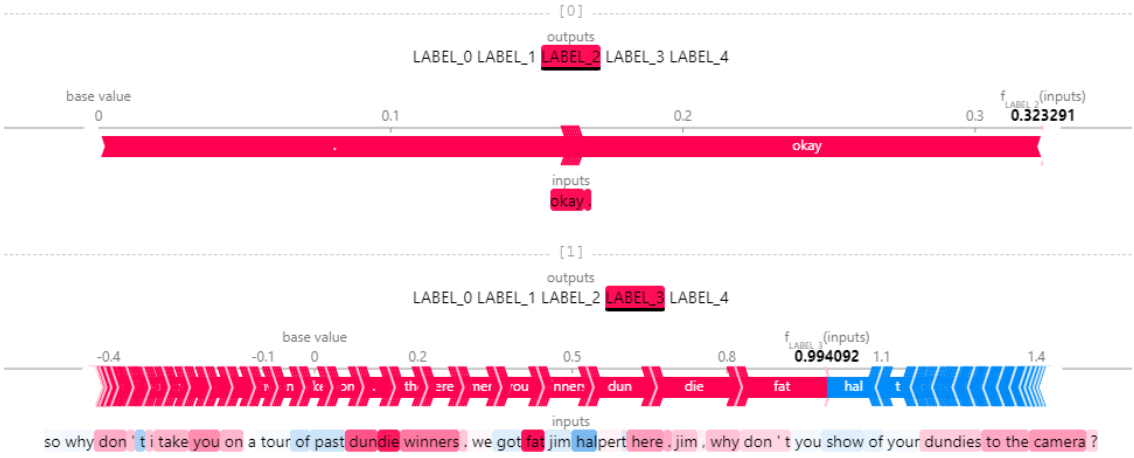

Fig. 5.2: Influence of specific words on speaker classification

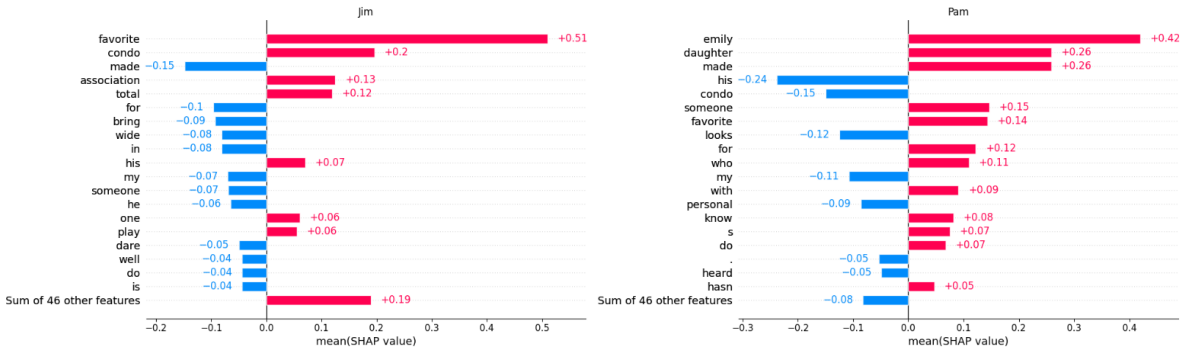

Fig. 5.3: Comparison of Pam and Jim’s speaking style

To summarize, our speaker classification model provides an approach to classify lines by their respective speakers based on their unique speaking style. However, the model has some limitations: Entirely new sentences that are not similar to any spoken line in the series are difficult to classify accurately. Additionally, for very short and generic sentences, it is impossible to make a prediction as the style of speaking does not differ between the characters in this regard. To further improve the model, we suggest to filter the data for a certain minimum line length to assure that there are less ubiquitous lines.

5.2 Scene Generation Model

To close the circle to the introduction, we now describe our approach to generate own scenes. The model used is a pre-trained transformer based on the GPT-2 language

model[10]. The data preparation involves concatenating each scene into one row, keeping directorials as they provide additional information about a scene. The language model is then fine-tuned using an AdamW optimizer on the tokenized data with a learning rate of 5e-4. We evaluate the model by looking at its training and validation loss, which indicates overfitting (see **Fig. 5.4**). This may be due to the limited available training data of a few thousand scenes.

Fig. 5.4: Learning curves for the scene generation model

In addition to the introduction, we provide another generated scene below. Every character featured in this short excerpt has its own style of speaking and interacting with others (e.g. Angela as strict, stubborn and cold or Creed as mysterious, shady and bizarre). Although there is no real content, this shows that the model successfully learned to imitate the characters.

***Michael**: Thats it! I am back!*
***Dwight**: Ah, thank you! So long.*
***Angela**: Extension 128.*
***Creed**: Hiya Pumpkin.*
***Angela**: You doing great?*
***Creed**: Yeah. So, do you want to hear me sing in your sexy sexy sexy highness?*
***Angela**: No.*

A limitation of the model is the scarcity of available data, along with highly varying scene structure and content. Despite these challenges, our model is able to generate scenes that capture the mood of different characters and create entirely new conversations. The generated scenes may not always be meaningful, but this is the case for many real scenes in the show as well.

[10]implementation by Radford et al. (2019)

6 Conclusion

In this last chapter, we will put all the results of the different analysis and models together into one conclusion of our work. The most important achievements are the following:

- We identified main and supporting characters of the show in our speaker analysis.
- We detected key moments and topics in the course of the show with topic modeling as well as speaker, word and sentiment analysis.
- We discovered sentiments in the show, which in particular revealed more information about character relationships and their developments.
- We visualized characters and their interactions and showed how this changes over the seasons.
- We identified unique speaking styles of characters by using both traditional methods and transformers, and we are able to predict the speaker of a line with an accuracy of 42%.
- We trained a generative model, which captures distinct features of the show and can generate new scenes.
- We detected that mainly colloquial language is used in the show, which is indicated by word analysis, speaking styles and generated scenes.

Therefore, we can conclude that we could not only give results that are consistent with the storyline of the show and known to fans of "The Office". We also provide additional insights, which might not be directly discovered by only watching the show. In particular the last point is interesting as an important message of the show is based on its ordinary everyday style. As Pam points out with the very last line of the show: "[...] an ordinary paper company like Dunder Mifflin was a great subject for a documentary. There's a lot of beauty in ordinary things. Isn't that kind of the point? ".
Finally, to complete our paper, we will discuss the limitations of the dataset and our work. The main difficulty here was that we only have the spoken lines. Due to the small number of directorials, we lack much of the visual component such as facial expressions, gestures, or atmosphere. A comedy like "The Office" and its jokes, however, thrive on aspects like timing, surprise and visual accompaniment. This is also a reason why our generated scenes may seem a bit weird and not really funny. Yet, if we picture the scene from the introduction with a hysterically screaming Michael Scott in our mind's eye, we can already imagine how visuals and tone of voice can actually make these scenes funny.

References

[Andreas Mueller et al. 2018] Andreas Mueller ; Jean-Christophe Fillion-Robin ; Raphael Boidol ; Font Tian ; Paul Nechifor ; YoonsubKim ; Peter ; Remi Rampin ; Marianne Corvellec ; Juan Medina ; Yuchao Dai ; Baze Petrushev ; Karol M. Langner ; Hong ; Alessio ; Ian Ozsvald ; Vkolmakov ; Terry Jones ; Eric Bailey ; Valentina Rho ; IgorAPM ; Divakar Roy ; Chandler May ; Foobuzz ; Piyush ; Low Kian Seong ; Jeroen Van Goey ; James Seden Smith ; Gus ; Feng Mai: *Amueller/Word_Cloud: Wordcloud 1.5.0*. 2018. – URL `https://zenodo.org/record/1322068`

[Bengfort and Bilbro 2019] Bengfort, Benjamin ; Bilbro, Rebecca: Yellowbrick: Visualizing the Scikit-Learn Model Selection Process. 4 (2019), Nr. 35. – URL `http://joss.theoj.org/papers/10.21105/joss.01075`

[Dua 2021] Dua, Sejal: Dynamic Topic Modeling with BERTopic - Towards Data Science. In: *Towards Data Science* (2021). – URL `https://towardsdatascience.com/dynamic-topic-modeling-with-bertopic-e5857e29f872`. – Access Date: 19.03.2023

[Grootendorst 2022] Grootendorst, Maarten: *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. 2022. – URL `https://arxiv.org/pdf/2203.05794`

[Hagberg et al. 2008] Hagberg, Aric A. ; Schult, Daniel A. ; Swart, Pieter J.: Exploring Network Structure, Dynamics, and Function using NetworkX. In: Varoquaux, Gaël (Ed.) ; Vaught, Travis (Ed.) ; Millman, Jarrod (Ed.): *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, 2008, pp. 11 – 15

[van Kooten 2022] Kooten, Pascal van: *kootenpv/contractions*. 2022. – URL `https://github.com/kootenpv/contractions`. – Access Date: 20.03.2023

[Li 2022] Li, Michelle: *michellejieli/emotion_text_classifier*. 2022. – URL `https://huggingface.co/michellejieli/emotion_text_classifier`. – Access Date: 20.03.2023

[Loper and Bird 2002] Loper, Edward ; Bird, Steven: *NLTK: The Natural Language Toolkit*. 2002. – URL `https://arxiv.org/abs/cs/0205028`

[Lundberg and Lee 2017] Lundberg, Scott M. ; Lee, Su-In: A Unified Approach to Interpreting Model Predictions. In: Guyon, I. (Ed.) ; Luxburg, U. V. (Ed.) ; Bengio, S. (Ed.) ; Wallach, H. (Ed.) ; Fergus, R. (Ed.) ; Vishwanathan, S. (Ed.) ; Garnett, R. (Ed.): *Advances in Neural Information Processing Systems 30*.

Curran Associates, Inc., 2017, pp. 4765–4774. – URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[Mejova 2009] Mejova, Yelena: Sentiment analysis: An overview. In: *University of Iowa, Computer Science Department* (2009)

[Pedregosa et al. 2011] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830

[Radford et al. 2019] Radford, Alec ; Wu, Jeff ; Child, Rewon ; Luan, David ; Amodei, Dario ; Sutskever, Ilya: Language Models are Unsupervised Multitask Learners. (2019)

[Ralhan 2018] Ralhan, Abhinav: *The Office Scripts Dataset - dataset by abhinavr8*. 2018. – URL https://data.world/abhinavr8/the-office-scripts-dataset. – Access Date: 27.01.2023

[Sanh et al. 2019] Sanh, Victor ; Debut, Lysandre ; Chaumond, Julien ; Wolf, Thomas: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In: *ArXiv* abs/1910.01108 (2019)

[Wikipedia 2023a] Wikipedia: *List of The Office (American TV series) characters*. 2023. – URL https://en.wikipedia.org/w/index.php?title=List_of_The_Office_(American_TV_series)_characters&oldid=1144719986. – Access Date: 19.03.2023

[Wikipedia 2023b] Wikipedia: *List of The Office (American TV series) episodes*. 2023. – URL https://en.wikipedia.org/w/index.php?title=List_of_The_Office_(American_TV_series)_episodes&oldid=1136894492. – Access Date: 19.03.2023

Selbständigkeitserklärung

Wir versichern hiermit, dass wir die vorliegende Ausarbeitung zur Vorlesung „Aktuelle Data Science-Entwicklungen: Intelligent Text Analysis“ mit dem Thema

That’s what the data said: An NLP Analysis of Script Lines from the US TV-Show “The Office“

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum	Unterschrift
Ort, Datum	Unterschrift