This is my title

Bachelorarbeit

Christopher Schütz | 2462248
Bachelor Wirschaftsinformatik

TECHNISCHE
UNIVERSITÄT
DARMSTADT

WIRTSCHAFTS
INFORMATIK

Table of contents

1 Background

This chapter serves to explain the foundations of natural language processing (NLP), especially the subpart of language modeling, needed to understand the problem and methodology used in this thesis. The two main building blocks of this thesis are the examination of a state-of-the-art language model (LM) and its ability to generate high quality, human-like text on the one hand and methods to distinguish such generations from human written text on the other hand. In order to understand the problem and methodology used in this thesis, this chapter explains the foundations of NLP (ch. 1.1), especially the subpart of language modeling, and the foundations of deep learning (ch. 1.2) especially under the aspect of sequence classification.

1.1 History of language modeling

Natural language processing (NLP) is "an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things"[1]. The applications of NLP such as speech recognition, sentiment analysis, question answering and others are numerous[2]. Moreover, these applications are already being heavily used by industry and consumers alike e.g. in the forms of digital voice assistants, sentiment analysis for recommender systems and browser search bars[3]. The subcomponent of NLP needed when it comes to tasks like machine translation, predictive typing or summarization that involve either generating text or estimating the probability of text is called language modeling. The following notation mostly follows the one from the CS224 Stanford Natural Language Processing with Deep Learning lecutre by Chris Manning.

Language modeling is the task of predicting what word comes next. More formally, this means: Given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, compute the probability of the next word $x^{(t+1)}$:

$$P(x^{(t+1)}|x^{(t)}, \ldots, x^{(1)}) \tag{1}$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$
Having a system that does allows us to assign a probability to a snippet of text of length T:

$$\begin{aligned} P(x^{(1)}, \ldots, x^{(T)}) &= P(x^{(1)}) \times P(x^{(2)}|x^{(1)}) \times \cdots \times P(x^{(T)}|x^{(T-1)}, \ldots, x^{(1)}) \\ &= \prod_{t=1}^{T} P(x^{(t)}|x^{(t-1)}, \ldots, x^{(1)}) \end{aligned} \tag{2}$$

Developing and improving language models is a task central to language understanding by which we can measure how well machine learning systems actually comprehend natural language [cite either stanford or ice paper 3 or 4]. This is demonstrated by the fact that "often (although not

[1] Chowdhury (2003).
[2] Gatt & Krahmer (2017).
[3] Valdivia; Luzón; Herrera (2017); Klopfenstein et al. (2017); Pandu (2019).

always), training better language models improves the underlying metrics of the downstream task (such as word error rate for speech recognition, or BLEU score for translation), which makes the task of training better LMs valuable by itself"[4].
Since the first significant language model was proposed back in 1980[5], language models and their architectures have gone through many changes. Especially the rise of Deep Learning and new network models such as RNNs or Transformers have fueled language modeling research in the past few years. The following chapters will cover the most common model types, heuristics and architectures (ch. 2.1.1 - 2.1.3 [STILL HARDCODED]).

1.1.1 N-Gram models

One solution in dealing with the problem of predicting a word after a sequence of $(n-1)$ words in the form of a Markov model, i.e. the probability of each event depends only on the state attained through the previous event, is called an n-gram model. An "n-gram" hereby denotes a chunk of n consecutive words. The core idea is that the probability of a word w_i occurring in the i^{th} instance after a sequence of $(i-1)$ preceding words can be approximated by observing only the preceding context of $(n-1)$ words.

Following this insight we can compute the probability of all n-grams in a corpus of text by simply counting their occurrences. Doing so allows us to calculate these conditional probabilities like so:

$$
\begin{aligned}
P(x^{(t+1)}|x^{(t)},\dots,x^{(1)}) &= P(x^{(t+1)}|x^{(t)},\dots,x^{(t-2+2)}) \\
&= \frac{P(x^{(t+1)},x^{(t)},\dots,x^{(t-2+2)})}{P(x^{(t)},\dots,x^{(t-2+2)})} \\
&\approx \frac{\text{count}(x^{(t+1)},x^{(t)},\dots,x^{(t-n+2)})}{\text{count}(x^{(t)},\dots,x^{(t-n+2)})} \text{(statistical approximation)}
\end{aligned}
\tag{3}
$$

N-Gram Models where n=1 are called Unigram, n=2 Bigram and n=3 respectively Trigram.

[Talk about the text generation process]

Even though n-gram models have been widely used especially due to their simplicity and scalability they do face certain limitations that have led to a decrease in their popularity. One problem that can arise when computing n-gram frequencies/probabilites is that n-grams encountered in a test setting do not appear in the corpus that the model was trained on. This leads to the probability of that n-gram being 0 [insert formula]. In order to counter this different smoothing techniques can be applied.

[4] Jozefowicz et al. (2016).
[5] Rosenfeld (2000).

Sparsity: what if “students opened their w” or “students opened their” never occurred? solutions -> smoothing, backoff [cite stanford textbook chapter]

Storage: Need to store counts for all n-grams you saw in the corpus. NO SOLUTION.

Lack of understanding: Perhaps the biggest drawback of n-gram models, though, is their very limited context size. In practice no n>5 usually. While produced output is often grammatically correct, there is an evident [lack] of coherence.

“today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share”

Even though n-gram language models are still widely used in speech recognition due to their high efficiency in inference, their limitations caused by poor generalization to unobserved n-grams and inability to capture long range dependencies led to the rise of neural language models [ice paper 8].

1.1.2 Neural language models

1.1.2.1 Neural network

This is some text.

1.1.2.2 Recurrent neural networks

This is some text.

1.1.2.3 Convolutional neural networks

This is some text.

1.1.2.4 Attention and transformers

This is some text.

1.2 Deep Learning

This is deep learning.

2 Methodology

This chapter explains the methodology.

2.1 Dataset

In order to create a dataset suited for the classification task a few aspects had to be considered. As deep learning methods were going to be used for the classification task, many labeled training samples were going to be needed [citation for the need of much training data]. The dataset should ideally be comprised of equal amounts of synthetically and human generated texts in order to improve the accuracy of training.

The first idea was to look for already built datasets that are freely available as these would not only reduce the time and amount of work needed for the creation of the dataset, but also provide a benchmark against other models used on them. Because the examined classification task is fairly novel and powerful language models have just started to emerge in recent years [citation], there is a lack of standardized data sets - prior research often focused on detection of artificially generated academic papers instead of short texts [reference to papers that use academic papers]. Furthermore, the incentive of using metadata related to text snippets led to the motivation of building a new dataset.

For this purpose, the following sources of human created text were inspected - taking different aspects like data availability, extensibility by metadata, potential for use of text generation and minimal overlap with the pretrained GPT2-XL Model (1.5B Parameters) into account:

- Wikipedia
- Twitter
- Reviews (e.g. Amazon, IMDb)
- Reddit Comments
- Reuters Corpus

[insert table with different aspects considered]

With everything taken into consideration, Wikipedia articles were chosen as the best fit for this work. Especially the fact that OpenAI did explicitly not train their model on any Wikipedia article [cite the gpt2 paper] and the ease of accessibility to large amounts of (meta-)data led to this conclusion.

2.1.1 Data Extraction

The two most used ways of scraping Wikipedia articles and other data from the Wikimedia Foundation [cite website] are through the APIs [cite] or the database dumps provided by the MediaWiki platform [ref site]. Both of these channels were used for different purposes:

- Database Dumps: Titles, clear text and article id were extracted using a python library called WikiExtractor [ref extractor]. This tool parses the xml-formatted dumps and extracts the aforementioned fields. The output is stored in ".jsonl" (json lines) files [ref json lines], as this is a file format commonly used for nlp data [reference].
- Api: The MediaWiki Api was used to retrieve metadata such as pageviews, edits, the latest edit timestamp and namespaces (i.e. categories) linked to each article, but also to access the latest news listed in WikiNews to generate exemplary news messages.

The extension of text samples by metadata was made in order to examine the following hypotheses: 1. The detection accuracy varies (significantly) across different categories. 2. The higher the edits and/or views on a page are, the lower the detection accuracy will be as the human text will be more sophisticated and better worded. 3. The more recent the last edit timestamp of an article is the lower the detection rate will be as newer information will be less likely to be present in the training data used for GPT-2.

It should be noted that only articles with a minimum length of 1000 characters were taken into consideration in order to filter out many entries that would have diminished the quality of the dataset (e.g. entries that only have a redirect notice [ref]).

2.1.2 Generation Parameter Combination

The output style of GPT2 differs depending on the chosen parameter combination. The parameters that influence the produced output the most are:

- Input length
- Maximum output length
- Temperature - a higher value produces a softer probability distribution over classes which leads to "crazier" or more unlikely text, whereas a lower value does the opposite [cite Hinton Paper no. 23]
- Repetition Penalty

In order to determine a parameter combination that generates convincing text all possible permutations between the parameters given the values listed in [ref table] were used to generate samples for 50 articles. As a metric for evaluation a fine-tuned large (1.5GB weights size as a .pt file) RoBERTa-based model with a mixture of temperature 1 and nucleus sampling outputs was chosen. This configuration was elected as it generalizes well to outputs generated using different sampling methods [citation gpt2 paper]. On top of that, human evaluation was performed while reading through samples created by the best performing parameter combinations according to the RoBERTa evaluation.
The main findings were that a higher repetition penalty as well as shorter output length were key factors for better text generation. In addition to that, choosing a lower temperature such

as [cite Harvard paper] and feeding the LM word split input sentences as opposed to character split input sequences ("tennis is a sport" instead of "tennis is a spo") also improved quality.

[show examples]

The finally selected parameter values were:

Maximum Input Length - 60 characters

The input that was fed into GPT-2 XL was the first sequence of plain text in a Wikipedia article, i.e. no infobox or content table text was considered. Additionally, before feeding the 60 character String into the LM it was split at the last word. This was done because the quality of the generated output tended to increase when input was given in full words rather than in characters and thus having many times split words.

Maximum Output Length - 50 Tokens

This corresponds to an average of about 240 - 260 characters per text (in comparison: the max tweet length is 280 characters). As this is a size that occurs a lot especially in social media or breaking news headlines (with the subtitle), the focus was placed on shorter text snippets [cite techcrunch and socialreport]. [insert figure that shows average character length per platform that achieves the best 'virality']

Temperature - 1.0

Both lowering and increasing (to 0.7 and 1.3) the temperature led to an increased detection of synthetically generated text, which is why the temperature was left at its original value.

Repetition Penalty - 1.3

As repetitions were not desired in the generated output the repetition penalty was increased from its default value of 1.0 to 1.3. Under the assumption that most Wikipedia articles do not contain repetitions (especially in their abstracts) unlike for instance dramatic text, where text repetition can be used as a stylistic device.

Number of beams - 5

Initially thought of as a parameter that would have a big impact on the quality of the generated text, it was found out that altering the number of beams used in the beam sampling strategy when predicting the next most probable tokens was had only little impact on the LM text generation. The values chosen ranged between 5 and 10 as this is currently the de facto standard [cite Stanford lectures] in research.

2.1.3 Data Building

After downloading the Wikipedia dumps the natural text was extracted via the wikiextractor [reference] library (filtering out texts with a character length less than 1000) and the metadata was extracted and parsed by using SAX Parser ("Simple Api for XML") [reference]. In order to finalize the data set creation, synthetic text had to be generated for each article's first 60 characters twice. The double generation was performed in order to select the sample that GPT-2 felt more confident on and improve the data set quality.

Given the aforementioned parameter configurations and using the free Google Colaboratory platform [ref Google Colab] which provides users with a free GPU (Nvidia Tesla K80 link) the text generation of a single sample took about 5 to 6 seconds.
Completing the data set with approximately 100.000 entries, thus, took about 11 days.

The finalized JSON format of a datapoint pair is shown in [Figure 10 - json format of 1 example].

2.2 Approach

2.3 Infrastructure

References

Chowdhury, Gobinda G. (2003): Natural language processing, In: Annual Review of Information Science and Technology, 37 (1), pp. 51–89 https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/aris.1440370103.

Gatt, Albert & Krahmer, Emiel (2017): Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation, In: CoRR, abs/1703.09902 http://arxiv.org/abs/1703.09902.

Jozefowicz, Rafal et al. (2016): Exploring the Limits of Language Modeling,.

Klopfenstein, Lorenz Cuno et al. (2017): The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms, In: Proceedings of the 2017 Conference on Designing Interactive Systems, New York, NY, USA Association for Computing Machinery, DIS '17 https://doi.org/10.1145/3064663.3064672, ISBN 9781450349222, p. 555–565.

Pandu, Nayuk (2019): Understanding searches better than ever before, No address in https://www.blog.google/products/search/search-language-understanding-bert/, Last accessed: 20.02.2020.

Rosenfeld, R. (2000): Two decades of statistical language modeling: where do we go from here? In: Proceedings of the IEEE, 88 (8), pp. 1270–1278, ISSN 1558–2256.

Valdivia, A.; Luzón, M. V. & Herrera, F. (2017): Sentiment Analysis in TripAdvisor, In: IEEE Intelligent Systems, 32 (4), pp. 72–77, ISSN 1941–1294.

A Appendix

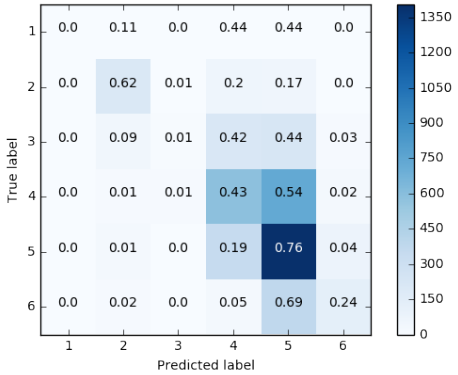
A.1 User groups

A detailed list of all examined Twitter accounts is omitted here for the sake of brevity. User groups were assembled as Twitter lists in the author's profile. Find links to all lists in the following:

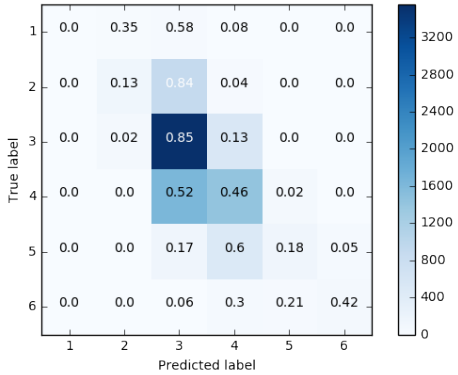
1. Celebrity user group: https://twitter.com/_fpeters/lists/celebrities
2. Politician user group: https://twitter.com/_fpeters/lists/us-politicians
3. Company user group: https://twitter.com/_fpeters/lists/fortune-500

A.2 Confusion matrices

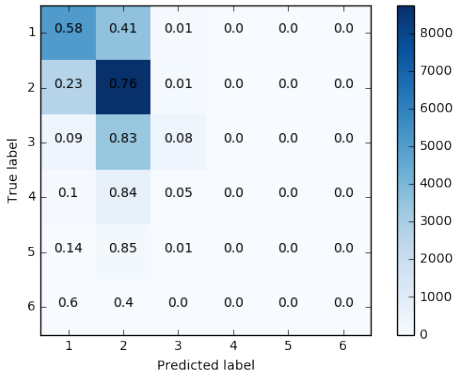
Confusion matrices for favorite classification were omitted in the results chapter. They are displayed in the following.



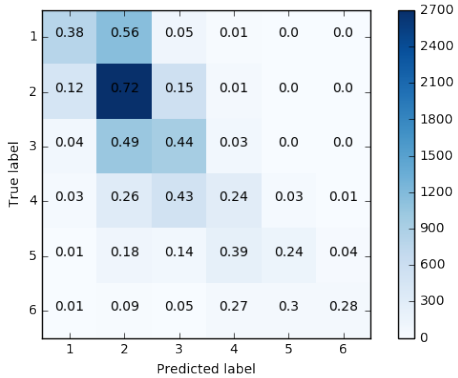
(a) Celebrity data set



(b) Politician data set

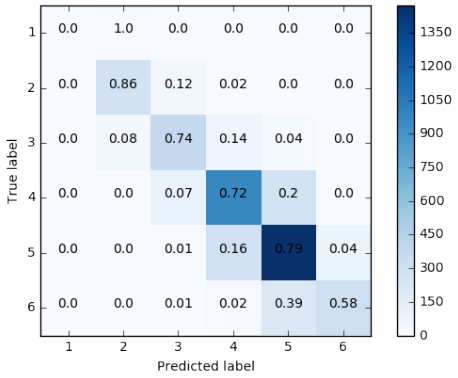


(c) Company data set

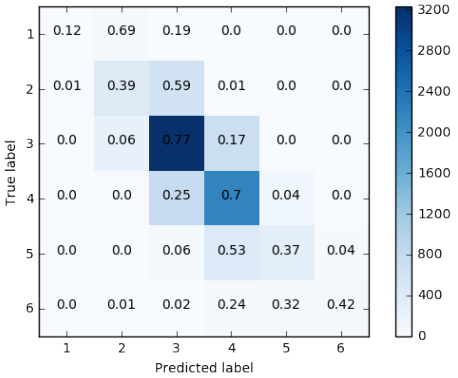


(d) Combined data set

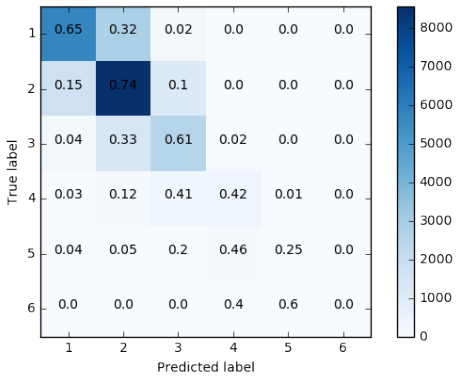
Figure 1: Confusion matrices for linear retweet classification models



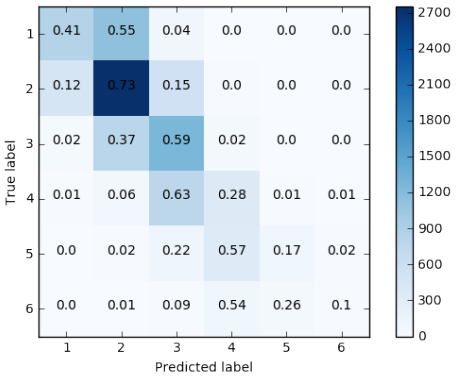
(a) Celebrity data set



(b) Politician data set

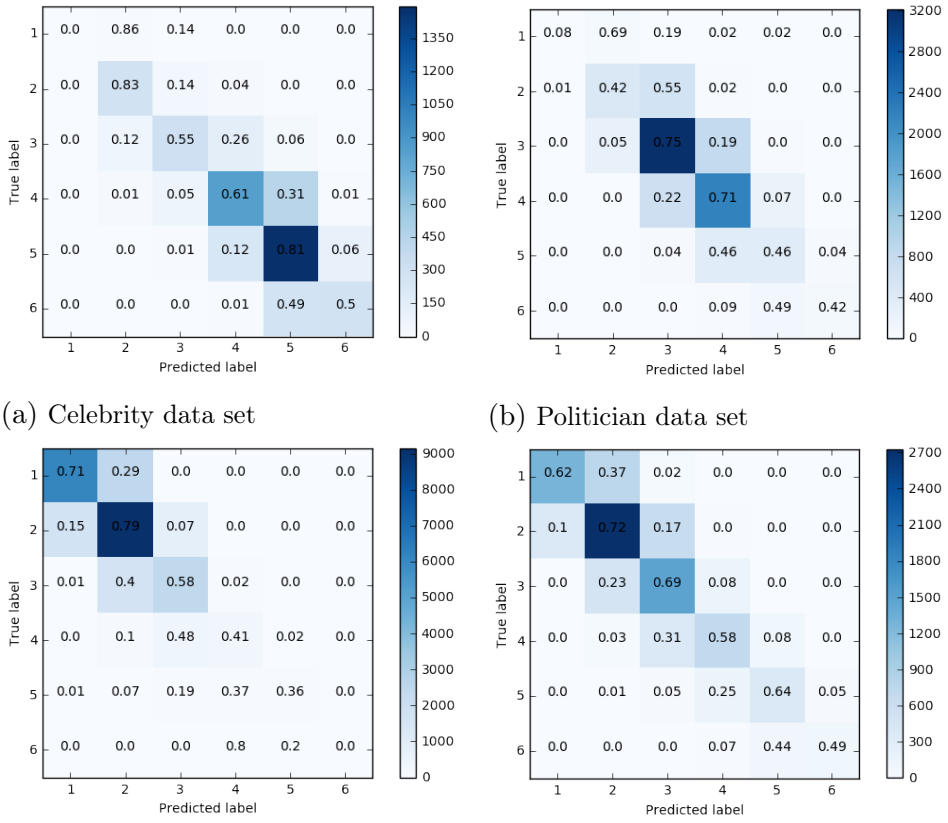


(c) Company data set



(d) Combined data set

Figure 2: Confusion matrices for deep feedforward classification models



(a) Celebrity data set

(b) Politician data set

(c) Company data set

(d) Combined data set

Figure 3: Confusion matrices for multi-input deep neural networks