HAW HAMBURG

Masterarbeit

André Soblechero Salvado

Language Agnostic News Recommendations Using Deep Learning Based Causal Transformer Decoder

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

André Soblechero Salvado

Language Agnostic News Recommendations Using Deep Learning Based Causal Transformer Decoder

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang *Master of Science Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Tim Tiedemann

Eingereicht am: 21. Februar 2022

André Soblechero Salvado

Title of Thesis

Language Agnostic News Recommendations Using Deep Learning Based Causal Transformer Decoder

Keywords

Recommender System, Natural Language Processing, Next Click Prediction, Sentence Embeddings, Language Agnostic, Causal Transformer Decoder, Regression, News Recommendations, LaBSE, BERT, Approximate Nearest Neighbor Search

Abstract

Machine learning is currently widely used for recommender systems. Previous work often represents users using unordered feature vectors while ignoring the sequential nature of users' actions. In addition, classification models are often used which are limited by the time that they are trained. User history has grown exponentially since the age of Big Data, which implies the need to constantly retrain classification models. This thesis proposes a method which uses representations of users as time-series of embeddings, replacing inflexible standard feature vectors and considering the sequential nature of users' actions. These user representations are then used to train a causal transformer decoder regression model which outputs the embedding of the next entry in the time-series, making it unnecessary to constantly retrain the model. The A/B testing results show that the proposed method works.

André Soblechero Salvado

Thema der Arbeit

Language Agnostic News Recommendations Using Deep Learning Based Causal Transformer Decoder

Stichworte

Recommender System, Natural Language Processing, Next Click Prediction, Sentence Embeddings, Language Agnostic, Causal Transformer Decoder, Regression, News Recommendations, LaBSE, BERT, Approximate Nearest Neighbor Search

Kurzzusammenfassung

Maschinelles Lernen wird heutzutage häufig in Empfehlungssystemen verwendet. Bisherige Arbeiten haben die Benutzer oftmals als nicht sortierte Feature-Vektoren dargestellt. Dies ignoriert die sequentielle Verhaltensweise eines Benutzers. Ebenfalls werden in vielen Fällen Klassifikationsmodelle verwendet, welche durch den Zeitpunkt des Trainings beschränkt sind. Seit Big Data steigt die Anzahl der Interaktionen von Benutzern exponentiell. Diese Tatsache impliziert, dass Klassifikationsmodelle konstant neu trainiert werden müssen. In dieser Arbeit wird eine Methode vorgestellt, die Benutzer als eine Zeitreihe von Einbettungen repräsentiert. Diese Zeitreihen ermöglichen es nicht flexible Feature-Vektoren zu ersetzen sowie die sequentielle Verhaltensweise eines Benutzers mit einzubeziehen. Die erwähnten Zeitreihen werden verwendet, um ein kausales Transformer Dekodierer Regressions Modell zu trainieren, welches die jeweils nächste Einbettung vorhersagt. Hierdurch wird verhindert, dass das Modell konstant neu trainiert werden muss. Die A/B-Tests demonstrieren, dass die vorgestellte Methode funktioniert.

Contents

List of Figures

1 Introduction

Recommender systems today are heavily used in various services, including platforms like Netflix, which provide personalized movie recommendations, and Facebook, whose feed is built around recommender systems to increase user engagement and provide content which is meaningful for the user [19, 26]. Recommender systems can be approached using varied methods, from statistics to machine learning or, more specifically, deep learning. The goal is usually to provide meaningful content to the user and thus increase the engagement of the user with the service. It is common to use similar users for this purpose so that a user is recommended items that he has not seen but that a similar user has seen.

This thesis can be viewed as an extension of the idea of Chen et al. 2019 [13], which enabled classifying the likelihood of the next click of a user using a time-series representation of the user that considers the sequential nature of the user's behavior. This work in contrast to Chen et al. 2019 [13], does not use classification to make the algorithm more flexible since classification models are limited by the time that they are trained. In this thesis, a method using news-article representations for entries in a time-series is used to train a causal transformer decoder, which is capable of predicting the next news-article representation [29, 34]. These news-article representations are vector representations of the news-articles' text generated by the pretrained transformer encoder model Language-Agnostic BERT Sentence Embedding (LaBSE) [16]. The predicted news-article representations are then used to determine which news-article the user is recommended using a nearest neighbor algorithm. This method makes the recommendations independent of the number of news-articles published on a daily basis and considers the taste direction of a user regarding the reading history. The method is evaluated using a proprietary news dataset, but it should work on all datasets where users can be represented as time-series of vectors.

This thesis consists of four main chapters. The analysis chapter includes a basic analysis of the proprietary dataset and related work forming the basis of this thesis and the research question. The design chapter explains the design of the whole recommender system, including technical aspects, evaluation scores, and model architecture. The evaluation chapter shows the results of training the causal transformer decoder and A/B testing. In the conclusion and limitation chapter, the limitations are explained and possible future research is proposed.

2 Analysis

This chapter analyzes the dataset used for this task and explains related work such as metrics, machine learning models, and algorithms that are fundamental for the objective of this thesis. This chapter examines the basics to build a recommender system which predicts news article representations for users, where users are represented by a time-series of news article representations using a causal transformer decoder and language agnostic sentence representations.

2.1 Data Inventory

Real data are used in this thesis to build a recommendation system consisting of anonymized user and news article data. The data are gathered from several German news portals. In this section, the available data are presented without detailing their usage. The data are divided into two categories: articles and pageviews. A detailed analysis of the columns can be found in the appendix chapter.

Articles

publisher_id	article_drive_id	...	article_full_text
khhs	iohhtm456fbi	...	"Omicron is ... no happy end."
khhs	345fgkzu56	...	"Pi is ... no happy end."
⋮	⋮	⋮	⋮
khhs	rtoiszhuif	...	"Sigma is ... no happy end."
khhs	rtoiszhuif	...	"Omega is ... happy end."

The news articles are saved in six parquet files. Apache Parquet is a column-oriented compressed data storage format used in multiple applications such as Apache Hadoop. The size of all files together is 1.59 GB and there are 823,948 news articles. There are two kinds of columns: The first holds general information about the news article, such as the article's full text and release date, while the second holds information extracted from these general information columns using machine learning, such as the sentiment of a news article.

Pageviews

user_id	article_drive_id	...	time_engaged_in_s
uskljng	iohhtm456fbi	...	12332
uskljng	345fgkzu56	...	12
⋮	⋮	⋮	⋮
uskljng	rtoiszhuif	...	321
uskljng	rtoiszhuif	...	4321

The second part of this dataset regards pageviews using a table with 313,565,551 entries, with 41.8 GB divided into 1,295 parquet files. A pageview is the interaction of a user with an article on a website. Each article occurs on average 183.21 times, and thus each article received on average 183 reads when of the pageviews occurred. The standard deviation is 6,775.87, which is high considering the average pageview per article. A news portal has an average of 18,445,032 pageviews and the standard deviation is 21,215,710, which is also high and could indicate that the pageviews are from portals with highly diverse page traffic.

2.2 Productive Metrics

This section describes two metrics which are commonly used to indicate a performance increase or decrease of recommender systems while A/B-Testing.

Click-through Rate

This metric click-through rate (CTR) shows how often a recommendation is clicked relative to the number of recommendations shown to a user [40].

$$CTR = \frac{Clicks}{Recommendations} * 100 \tag{2.1}$$

When the score of this metric increases for a new recommendation method during A/B testing, then this indicates better recommendations.

Session length

This metric calculates the individual or average session length of a user on a platform. A session is a temporary and interactive information exchange between the user and platform, which describes the length of time which a user spends continuously on a platform. This metric can indicate how satisfied and engaged a user is with the platform and thus its content. Since the content is also determined by recommender systems, an increased session length can indicate an improved recommender system.

2.3 Model-based Collaborative Filtering

Model-based collaborative filtering is a widely used method of building a personalized recommendation system [31]. This section describes a possible approach to applying model-based collaborative filtering to the news dataset. Model-based collaborative filtering produces recommendations using users with similar interactions with items based on a scoring of the interaction. A user-item matrix is usually used which describes the interaction between users and items.

2.3.1 Example Use-Case

Using model-based collaborative filtering requires an indicator to describe the interaction between a user and an item. For this purpose, a score that indicates the relative time spent on an article is used.

We use the time $t = readtime/words$ because longer articles require more time to read and vice versa.

There are four different scores:

1. Significantly below average read time
2. The user spent below-average time but the time spent is statistically common
3. Above-average read time
4. Significantly longer read time

A score of 0 indicates that the user has not yet read the article, and these scores are candidates for the recommendation.

The calculation is performed as follows:

Average time per word $\mathbf{t}$ in a text, Overall mean μ and standard deviation σ	Score
$\mathbf{t} = 0$	0
$\mathbf{t} < \mu - \sigma$	1
$\mu - \sigma \leq \mathbf{t} < \mu$	2
$\mu \leq \mathbf{t} < \mu + \sigma$	3
$\mu \leq \mathbf{t}$	4

Using this relationship between users and items, we can create the user-item matrix as a target and train a model which takes a users matrix and an items matrix and transforms them to the user-item matrix. The users matrix is a matrix in which each row represents a user, and the items matrix is a matrix in which each row represents an item using embeddings. We now train a model with the user matrix and item matrix as inputs with the aim to reproduce the user-item matrix using the mean squared error (MSE).

The output would be the following matrix:

$$\begin{matrix} User1 & User2 & User3 & User4 & User5 & \\ \left(\begin{matrix} 1 & 3 & 1 & 1 & 4 \\ 0 & 3 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 4 \end{matrix} \right) & \begin{matrix} Article1 \\ Article2 \\ Article3 \\ Article4 \\ Article5 \end{matrix} \end{matrix}$$

We also know the real relationship between a user and each item from the production system. Using the model, we may acquire a high score for a user-item element although the user has not yet interacted with the item. We could then recommend this item because the model predicts that the user might want to interact with the item or, more specifically, that similar users have interacted with this item.

2.3.2 Advantages and Disadvantages

This method of creating recommendations includes the following advantages and disadvantages:

Advantages

- Personalized recommendations. Each user receives recommendations based on his taste rather than based on similar items, for example, and thus the recommendations are personalized.
- This method is simple to train due to its simplicity.

Disadvantages

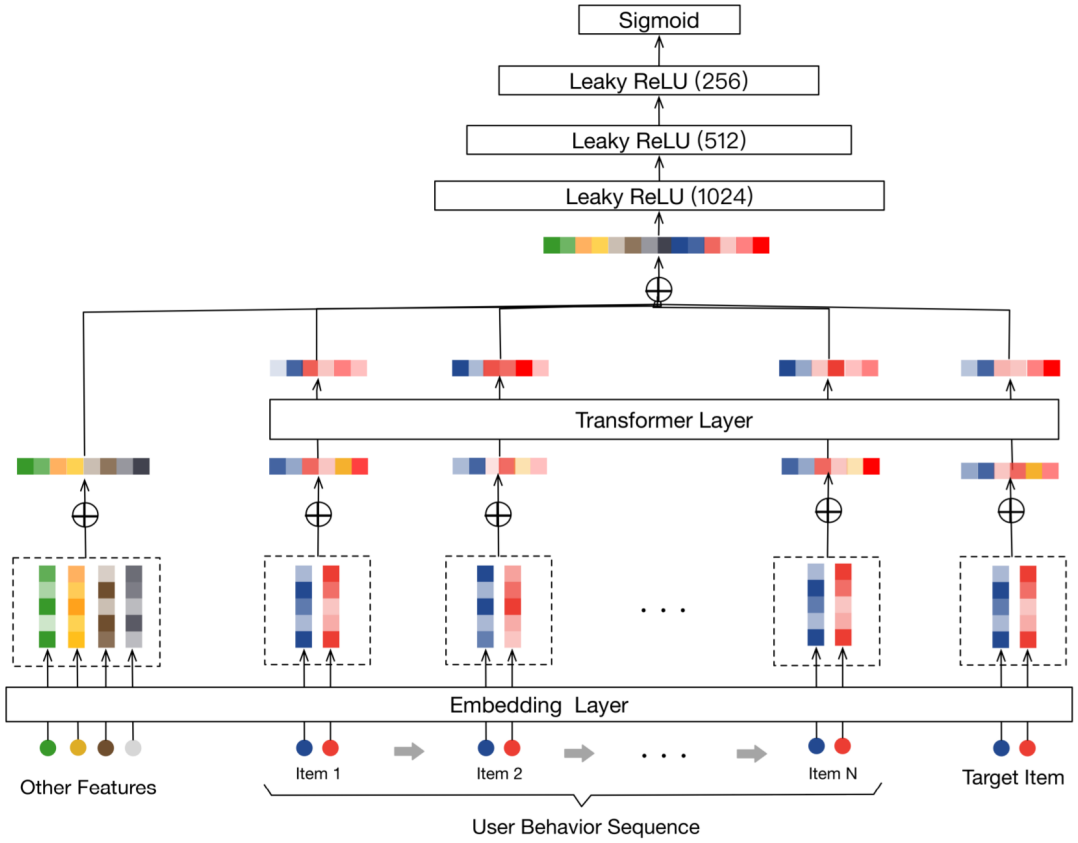
- Collaborative filtering cannot flexibly handle new items because embeddings are generated while training.
- It is not easy to include meta data such as the location of a user.
- The order of items does not matter in the user-item interaction matrix, although the order might impact what the user would like to interact with next.
- Users without an interaction would receive random recommendations.

2.3.3 Conclusion

Since model-based collaborative filtering does not consider the sequence of items or, more specifically, the order in which articles have been read, and because the constantly increasing amount of data would require constantly retraining the model, we will not proceed with this approach.

2.4 Behavior Sequence Transformer for E-commerce Recommendation in Alibaba

Figure 2.1: Architecture of the Behavior Sequence Transformer.[13]



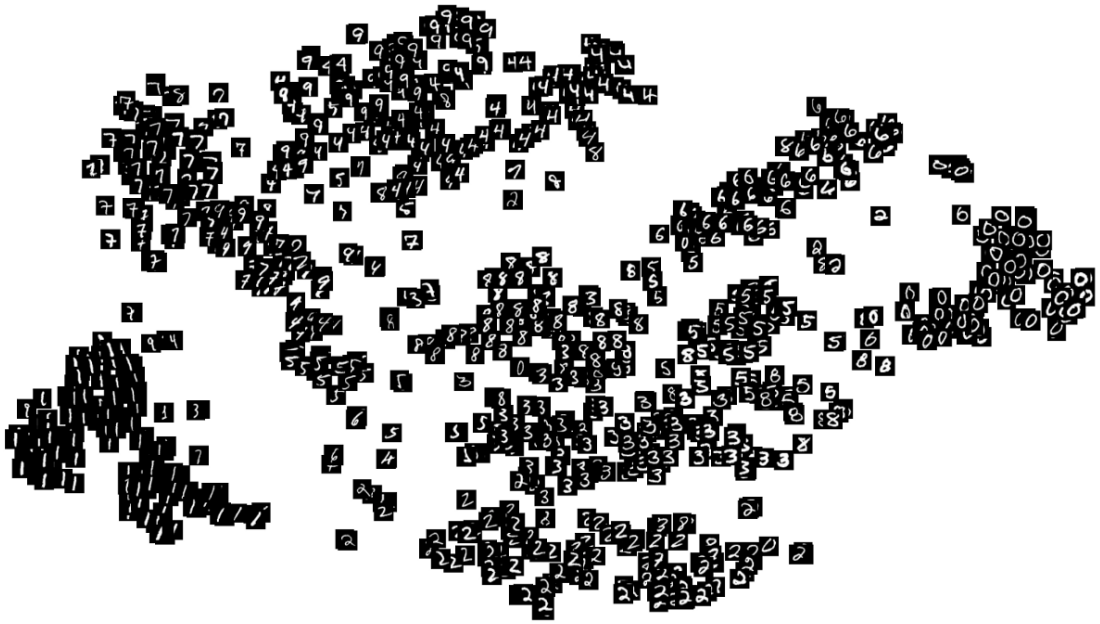
The method introduced by Chen et al. 2019 [13] suggested a transformer model capable of classifying the likelihood of the next click of a user for a given target item while considering the sequence of items [34]. A user is represented as a time-series of items and other features such as the user profile. Each item and other features become embedded to a low dimensional vector [18]. These vectors are then feed into a neuronal network to classify the likelihood that the user will click on the target item. One major drawback of this method is that the model needs to be retrained every time an item unknown to the model is introduced, because embeddings are fixed mappings from a high dimensional one-hot vector to a low dimensional vector representation of the one-hot vector.

2.5 Latent Space in Machine Learning

Pictures, texts, and sounds need to be represented in a mathematical way to function with machine learning algorithms.

Pictures are mapped into the latent space to enable increased efficiency since the scale of Big Data requires consideration in machine learning. The latent space is usually a multi-dimensional vector-space containing vectors which cannot be interpreted directly but have the characteristic that similar items embedded to the vector space should have similar vectors.

Figure 2.2: A latent space for the MNIST Dataset which shows how similar pictures achieve similar representations in a two-dimensional latent space [7]



The compressed vectors are part of the latent space in which similar pictures have similar vector representations. There are several ways to map inputs to a latent space, such as training an autoencoder which takes the inputs, compresses them to a vector, and tries to reproduce the input from the vector again. After training, we can use the encoder part of the autoencoder to map inputs to a latent space. These vectors can now be used for a variety of tasks such as classification, nearest neighbor search, clustering, etc.

2.5.1 Sentence Embeddings

Since the data used in this thesis consist of news which are usually formulated using natural language, we need a method to represent the news articles in a mathematical manner [25]. A sentence embedding is a sentence mapped into a latent space. Similar sentences are similarly represented in the latent space; for example, the sentence ”I hate you” and ”I do not like you” should have a similar vector.

Similar vectors are defined as vectors with a high cosine similarity, while vectors of sentences which are less similar have a smaller cosine similarity. The cosine similarity is used because other distance measurements can be problematic since latent spaces are usually high dimensional and not always normalized.

Models capable of compressing sentences to vectors are trained similarly to pictures in latent spaces. A special aspect of sentences is the variable length of the input.

The method introduced in this thesis uses sentence embeddings to substitute fixed embeddings for each item.

2.5.2 Language-agnostic Sentence Embeddings

Language-agnostic sentence embeddings are sentence embeddings created by a model capable of encoding the ”sense” of a sentence while ignoring the language in which the input sentence is formulated [16].

This means that the sentence ”How are you?” in English and the German-translated sentence ”Wie geht es dir?” have a similar vector in the latent space.

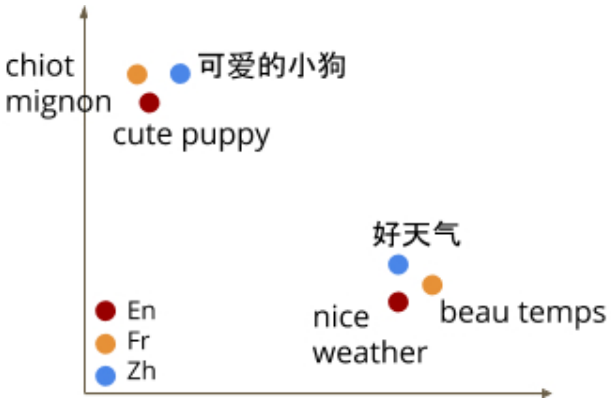


Figure 2.3: Visualization of latent space with language-agnostic sentence embeddings. Similar sentences are encoded to similar vectors, where similarity is defined by cosine similarity [4]

The main advantage of this kind of embedding is that models trained on embeddings of sentences in one language can also be used for sentences in other languages. Data are currently one of the most valuable assets, and this kind of sentence embedding implies that data are needed in only one language since one classifier trained on these embeddings could also perform well on sentences in other languages.

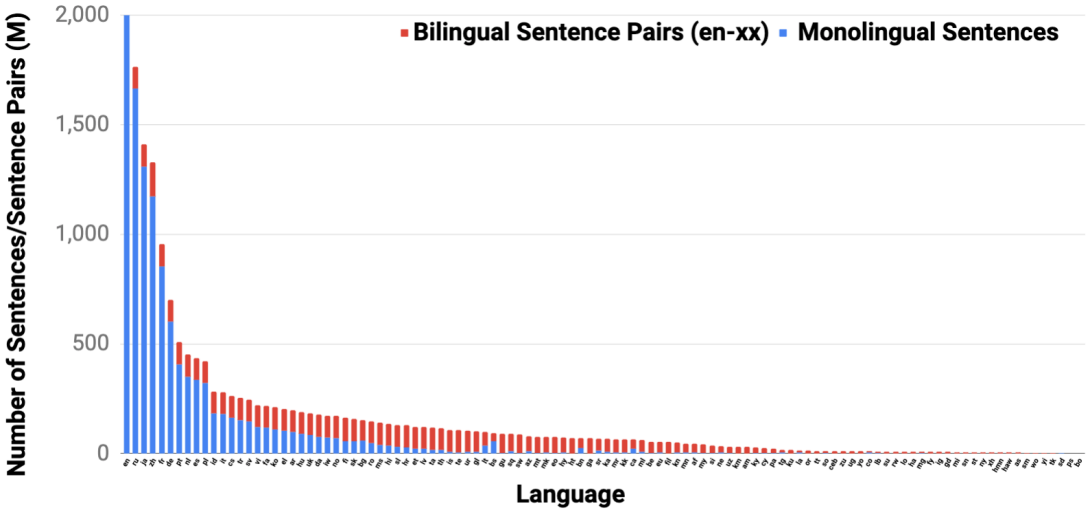
The recommender system in this thesis is based on a model capable of effectively encoding sentences in 109 languages.

Language-agnostic sentence embeddings are used because most effective open-source embedding models are trained on English data or are language agnostic. The reason for this is that the internet is English centric, and thus the datasets needed to train a model capable of encoding text to sentence embeddings are often English centric. In addition, different languages profit from each other since many languages share the same origin.

2.6 LaBSE: Language-agnostic BERT Sentence Embedding

LaBSE is a finetuned BERT model capable of mapping texts to language-agnostic sentence embeddings [14, 16]. LaBSE supports 109 languages and was trained using contrastive learning [23]. BERT is a transformer encoder model which is explained in the next section.

Figure 2.4: LaBSE bilingual and monolingual sentence pairs per language [4]

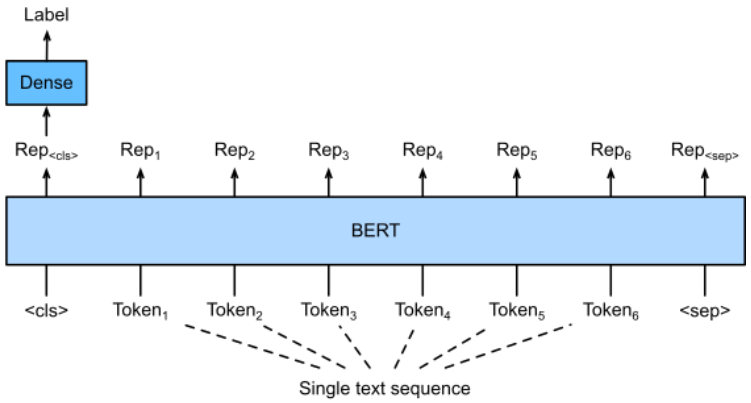


The language-agnostic sentence embeddings produced by LaBSE will be used for the recommender system. LaBSE was trained on 17 billion monolingual sentences and 6 billion bilingual sentence pairs. A sentence pair consists of two sentences where one sentence is, for example, a sentence in English whose pair is the German translated sentence. This model was chosen because it is an efficient transformer sentence encoder while performing well in cross-lingual text retrieval.

2.6.1 BERT Model

Single recurrent neural network (RNN) models are difficult to distribute over multiple GPUs due to their recurrent nature, where the next forward pass depends on the previous one. Another negative aspect is the vanishing gradient problem of RNNs [22, 34, 20]; as a result, the transformer architecture was introduced in 2017 to enable processing sequences more efficiently and solving the vanishing gradient problem using self-attention [34]. While the vanilla transformer consists of an encoder and decoder, BERT only uses the encoder to encode text with a variable-length bidirectionally [14]. The concept of bidirectionally is the same as with bidirectional LSTMs, where the corresponding output of a token (token representation) is determined by the whole sentence rather than only the tokens before the token [21, 30].

Figure 2.5: Simplified Bert Architecture. Visualizes the connection between input tokens ($< cls >/Token_i$) and output vectors ($Rep_{<cls>}/Rep_i$), which should represent the input token in the context of the text. [3]



The first input token of BERT is often a so-called classification token ($< cls >$) whose corresponding token embedding is used as a sentence representation and is thus useful for sentence-based tasks. LaBSE is a BERT-based multilingual model, which was finetuned to produce language-agnostic sentence embeddings at the $< cls >$ tokens' corresponding token embedding. The model consists of twelve transformer encoder layers, twelve attention heads in each layer, a 500,000 token vocabulary (words and subparts of words), and a dimension of 768 for each token representation.

2.6.2 Contrastive Learning

Contrastive learning has been used to archive the capability of LaBSE to encode text to a language-agnostic latent space [23].

The objective of contrastive learning is to maximize the similarity of vectors between equal inputs and minimize the similarity of unequal inputs; for example, the vector representation of the German sentence "Wir geht es dir?" should ideally be equal to the vector representation of "How are you?". At the same time, we want to minimize the similarity between the vector representation of "Wie geht es dir?" and "I am a student" since they have different meanings.

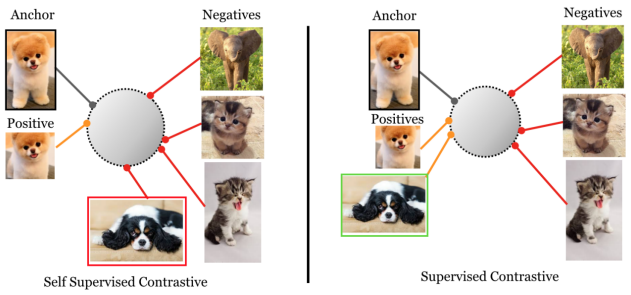


Figure 2.6: Contrastive Loss Visualization [23]

A dual encoder with shared parameters was used to train LaBSE, which means that the encoder encodes a batch of texts in several languages with one feedforward and a batch of translations of these texts with another feedforward. The objective is to maximize the similarity between sentence pairs and minimize the similarity between their uncorresponding translations in a batch.

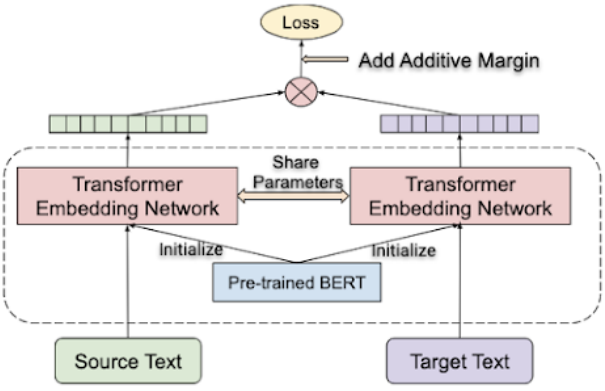
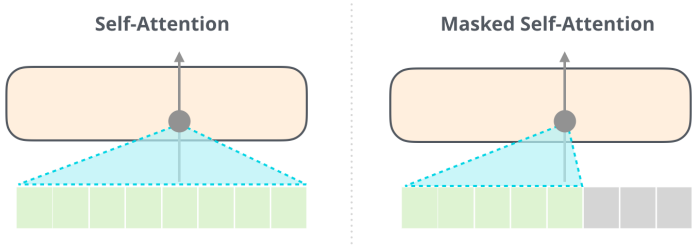


Figure 2.7: LaBSE Training Visualization [4]

2.7 Time-Series Forecasting and Causal Transformer Decoder

Since the recommendation model will be based on time-series forecasting or, more specifically, on forecasting of the next language-agnostic embeddings, this section explains time-series forecasting and the possibility to use a GPT2-based causal transformer decoder for this purpose [29]. Time-series forecasting involves making predictions using historical data ordered by timestamps. The historical data in this case are the news article reading history of each user, where the simplified objective is to predict the next news article. Since transformer encoders such as BERT can only be used bidirectionally, and thus the prediction would always depend on both future and past entries in the time-series, a GPT2 causal transformer decoder will be used. The main difference between a transformer encoder and a causal transformer decoder is that the latter's predictions only depend on past time-series entries and can be compared with a simple RNN without recurrence and many parameters. Causal transformer decoders always need at least one input of a time-series and can use time-series with a maximum length determined by the longest time-series in the training data. To archive the ability to only depend on past time-series entries without recurrence the self-attention is masked so that while generating the corresponding output vector of an entry in the time-series, only information from past entries of the time-series will be attended to while future tokens will not.

Figure 2.8: Self-Attention vs Masked Self-Attention [6]



This approach enables training a model capable of predicting the next news article from one past news article while being trained on time-series with a minimal length of ten entries and a maximal length of two hundred entries.

2.8 Regression Problem

If the objective is concretized, then the training of the causal transformer decoder becomes a regression problem. The objective is to predict language-agnostic embeddings for news articles produced by LaBSE, which represents the content of the news articles. By predicting embeddings, the recommendation system gains flexibility because, as long as the article is not out of bounds of the training dataset, new articles can be processed without retraining a new causal transformer decoder in contrast to news article classification models; for example, if the model is only trained on corona news-articles then the model might perform poorly on predicting news-article embeddings with different content.

The MSE [35] is used as a loss function to minimize the error between output vectors of the causal transformer decoder and the language-agnostic sentence embeddings which it should predict:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - x_i) \tag{2.2}$$

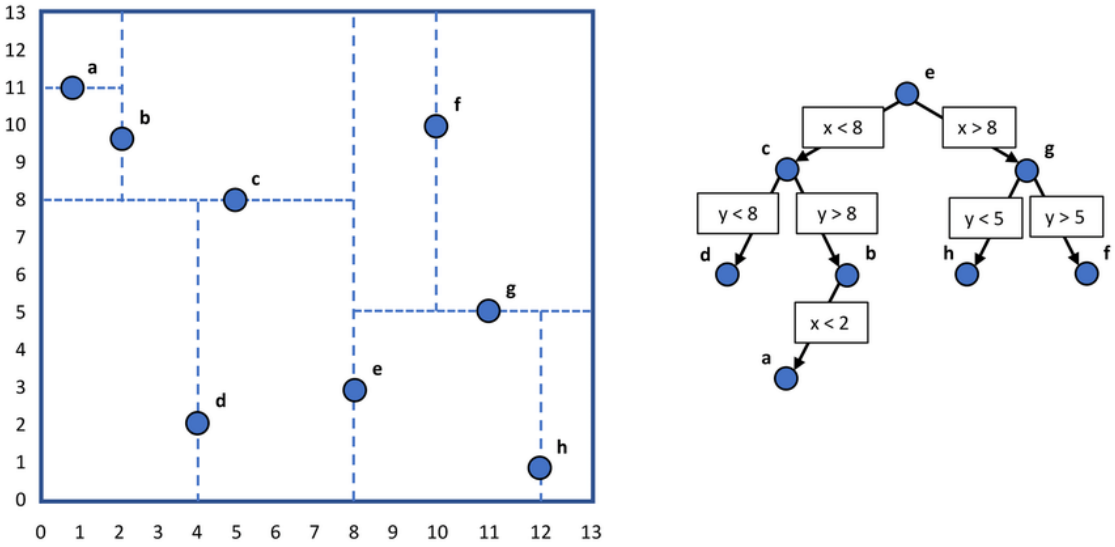
2.9 Approximate Nearest Neighbors Search

Since the causal transformer decoder is trained using a regression error, the predicted language-agnostic embeddings produced by the model only include vectors without specific news articles bound to them. These generated vectors can be viewed as taste directions showing what the user might want to see next. Providing proper recommendations requires a method to acquire news articles for the predicted vector. For this purpose, an approximate nearest neighbor search is used to retrieve the most similar vector to the user's taste direction [9], where the best recommendation for a user needs to be selected from a news pool. All these news articles are encoded using LaBSE. The algorithm takes all embeddings of the news articles in the pool and a predicted embedding of the model and outputs the top k nearest neighbors of the vector using an angular metric.

2.9.1 k-Dimensional Tree

If the news pool consists of ten billion news-article embeddings and the algorithm aims to find the top ten neighbors of a predicted vector every ten milliseconds, the naive approach is to compare the predicted vector every ten milliseconds with each of the ten billion news-article embeddings, which would be highly inefficient.

Figure 2.9: Example of a 2-Dimensional Tree [32]



To make the algorithm more efficient, an approximate nearest neighbor algorithm is used to circumvent the need to compare the predicted vector with each news article embedding; instead, the algorithm uses decision trees to find the best fitting partition. The news-article embeddings are partitioned into different parts of the latent space and the algorithm uses decision trees to choose in which partition we need to search for the nearest neighbors [11].

2.10 Research Question

Since the reading history might have an impact on the news-article a user would like to read next and the fact that news-article domains are very volatile the question arises:

> Is it possible to build a recommender system which uses time-series representing a user, where each entry of the time-series is an embedding, by training a causal transformer decoder regressions model which receives a time-series of embeddings and outputs the next embeddings representing the article that the user would like to read next, which is then used to retrieve the item out of all candidates using a nearest neighbor algorithm?

3 Design

This chapter explains design decisions and combines the previous chapter's techniques to a recommender system, starting with the cleaning and transformation of the dataset to fit the needs for training the causal transformer decoder.

3.1 Data Cleaning, Transformation, & Analysis

Since the anonymized and proprietary news dataset has columns which are not necessary for training the causal transformer decoder, the dataset first needs to be cleaned of unused columns and rows with null entries as well as users with fewer then ten pageviews.

Only the columns user_id, article_drive_id, page_view_start_local, portal_id from the pageviews are needed. The user_id was used to generate users with article_drive_-ids. The article_drive_id were sorted ascending by page_view_start_local.

All pageviews were removed where at least one of the used columns is null.

The columns article_drive_id and article_full_text of the articles were used. The article_drive_id was used to map a pageview of a user to an article_full_text, which will be embedded using LaBSE to generate a vector representation of the article. Only users with 10-200 pageviews were considered. The minimum of ten pageviews was chosen since each training step should maximize the improvement of the model, and smaller time-series are implied in each step since the model is a causal transformer decoder. The maximum of two hundred is used due to hardware limitations. Larger time-series require more VRAM in combination with a decent batch-size, and there needs to be a maximum size since VRAM is limited. Some facts about the transformation are listed as follows:

- The number of users without null values is 35,048,127, and the largest group of users have only one pageview;

- The number of users with 10 to 200 pageviews is 1,806,951;
- The number of articles without null values is 471,913;
- The mean sequence length of a user time-series is 31.63 and the standard deviation is 32.27.

3.2 Recommender System Architecture

This chapter introduces the three components of the recommender system and how they are used. Every user is described as a series of articles or, more specifically, a series of language-agnostic embeddings produced by LaBSE and ordered by time. A model is trained to generate the next embedding of a news article using the past embeddings in the context of the time-series. The whole architecture can be divided into three parts:

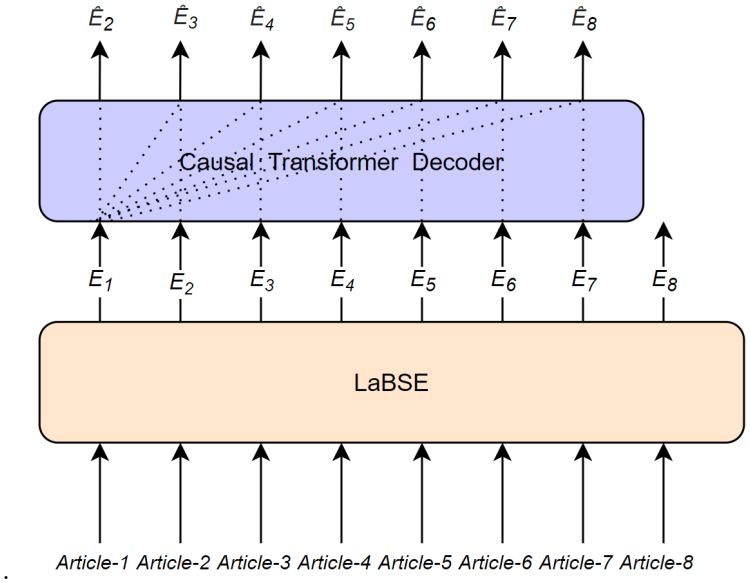
- **LaBSE:** A model which encodes the content of a news article to an embedding;
- **Causal transformer decoder:** A model which takes an ordered series of embeddings and outputs vectors with content encoded which the user wants to read next based on their reading history;
- **Approximate nearest neighbor search:** An algorithm which enables retrieving the top k news-article recommendations for a user using the embeddings of a news pool and the predicted vector.

This method of producing recommendations is chosen due to the volatility, velocity, and variety of news articles. An approach is needed that enables quick handling of new news articles which may have little in common with older articles.

It thus might not be sufficient to train a classifier that predicts the next news article since new news articles are constantly released.

3.2.1 Model Architecture

Figure 3.1: Causal Transformer Decoder Architecture & LaBSE. $Article - i$ is a text in natural language. All $Articles - i$ are ordered by the reading history of a user. E_i is the embedding for the full text of the $Article - i$. $\hat{E}_i$ is the predicted embedding by the causal transformer based on past embeddings. The dotted lines underline that $\hat{E}_i$ depend only on the embeddings $[E_1, E_{i-1}]$



The pictured architecture visualizes training the causal transformer model.

1. Each text of an $Article - i$ is encoded to a language-agnostic sentence embedding E_i using LaBSE.
2. These ordered article embeddings are fed into the causal transformer decoder to predict the embedding of the next article $\hat{E}_{i+1}$.

The trained causal transformer decoder consists of 85.9 million trainable parameters.

The MSE is used as an error function between the output vectors of the causal transformer decoder and the target vectors.

The LaBSE model will not be finetuned and is only used to generate language-agnostic sentence embeddings for each news-article text.

3.2.2 Approximate Nearest Neighbor Search

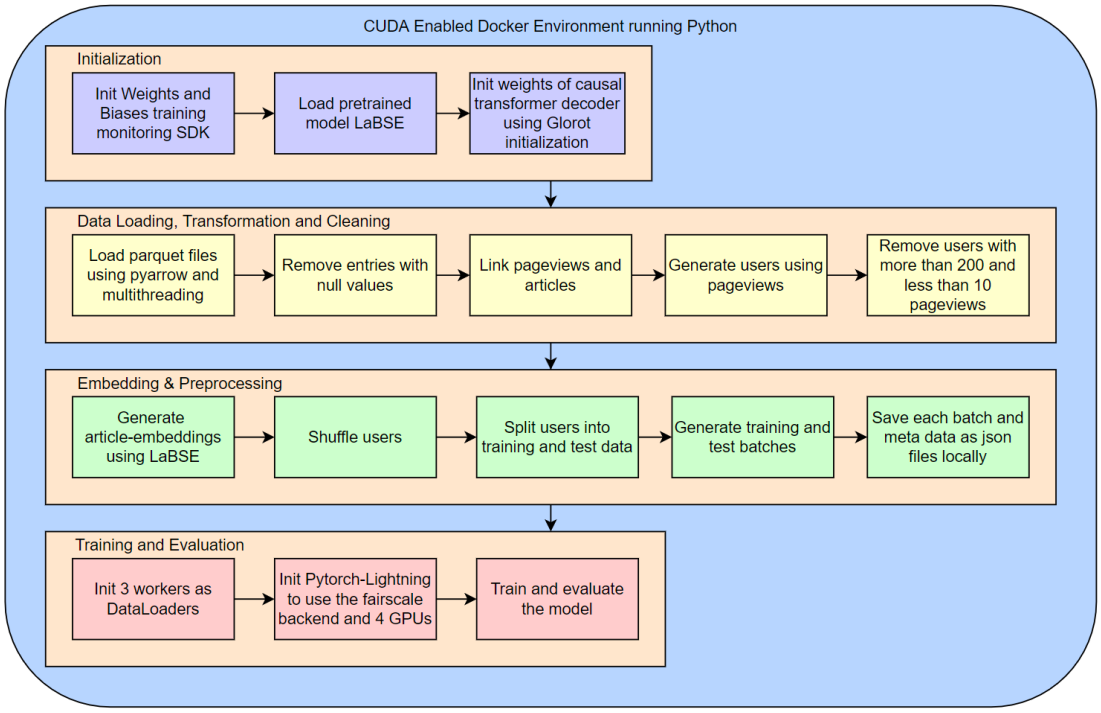
During testing between epochs and A/B testing, the approximate nearest neighbor search is used to retrieve the best recommendation for a specific user. This method is necessary because the causal transformer decoder only outputs a vector that may indicate the topic or content of the next news that a user would like to read rather than the article itself. All news articles are embedded using LaBSE, and a 768-dimensional tree is built for each news portal. During testing between the epochs for each output of the causal transformer decoder, the best-fitting article out of the news-article pool is searched for and included in the score calculation. To receive a recommendation during A/B testing, the last output vector of the causal transformer decoder is used to search for the best-fitting article out of the news-article pool.

3.3 Technical Design

This section explains the experiment procedure and most technical dependencies used for experimentation and A/B testing.

3.3.1 Experiment Procedure

Figure 3.2: This chart explains the experiment procedure



The experiment procedure can be divided into four steps:

1. **Initialization:** In this step, the weights and biases training monitoring framework is initialized, the causal transformer decoders parameters are initialized using xavier_uniform_ and LaBSE is downloaded from the huggingface hub [17, 38].

2. **Data Loading, Transformation, and Cleaning:** In this the step, the 1,295 parquet files in which the pageviews are located are loaded using pyarrow and threading. The six parquet files in which the articles are stored are loaded sequentially. The data will be cleaned from rows with null values, and afterwards users

will be generated from the tables. A user is a list of articles ordered by time, and not used users are removed for training.

3. **Embedding and Preprocessing:** This step describes how the training and test data are generated. A language-agnostic sentence embedding is initially generated for each article using LaBSE. The training/test split of 99.9%/0.1% is applied to the data. After generating all batches, each batch is stored locally as a json file, because the amount of data would be too great to hold in memory since each data loader holds the same data, and thus all data would be held in memory with a redundance of four.

4. **Training and Evaluation:** The data loaders are separate processes to increase the training speed since Python is not known for its threading speed. The initialization of the training loop includes the definition to use the fairscale backend (explained in the next section) and the defined training parameters.

3.3.2 Technical Dependencies

Python

Python is the chosen programming language due to its widespread usage for machine-learning tasks and large community [33].

Jupyter Notebook

The most basic tool used for training, data exploration, cleaning, and transformation is the Jupyter Notebook [24], which enables running Python in a browser and thereby making it easy, flexible, and fast to try different solutions on remote machines.

PyTorch

PyTorch is the chosen deep learning framework due to its fast releasing speed regarding community-released pretrained models and its flexibility to change in-depth parts of the training procedure with ease; for example, the optimizer LAMB was used during training although PyTorch does not support LAMB natively [28, 39, 5].

Transformers

The transformers framework is currently one of the most popular deep-learning natural language processing frameworks [37], which also enables using and sharing pretrained transformer models. The framework also supports a wide range of transformer variations.

LaBSE was used directly from the transformers framework, and the causal transformer decoder is a modified GPT2 model from the transformers framework. The transformers framework supports tasks ranging from natural language generation to natural language understanding and includes a trainer which was not used.

PyTorch Lightning

PyTorch Lightning is a framework that combines PyTorch with other libraries which are able to reduce the training time [15]. This framework tries to minimize the boilerplate code and offers support for a wide range of extensions. The two PyTorch Lightning integrations wandb and fairscale were used to monitor and accelerate the training process.

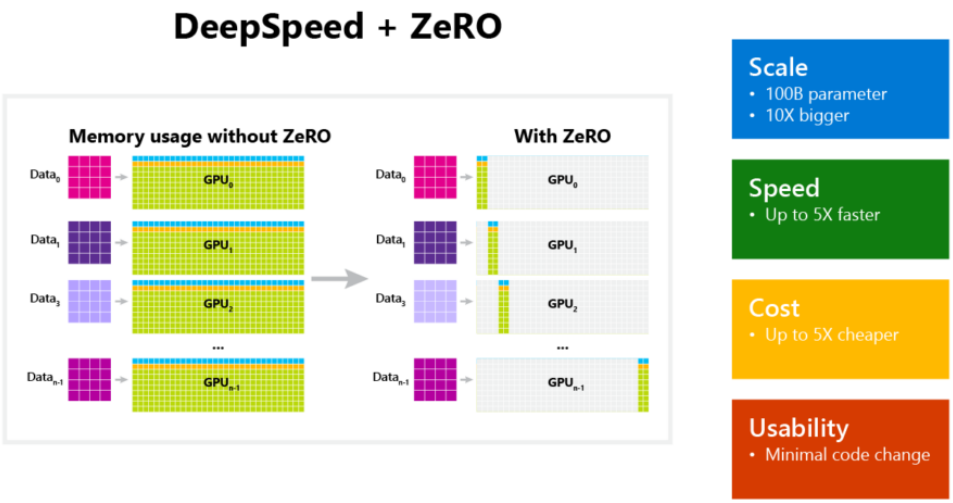
Wandb

Wandb is a tool that allows the user to monitor their machine learning training [12], it offers an easy API and enables the user to monitor the live training online. Since the training is being run on remote machine this tool is quite powerful and speeds up the whole integration process.

Fairscale

To speed up training, deep learning models are trained using multiple GPUs. The conservative approach of realizing this is to hold the deep learning model redundant on all GPUs or to shard the model manually over all GPUs. When the model is held redundantly on all GPUs, which is less effort compared to sharding, each GPU processes a part of the batch. The main disadvantage, especially for deep learning models with many parameters, is that such large models need significant VRAM, representing a waste of resources.

Figure 3.3: Deepspeed [8]



Fairscale divides the model into several parts and evenly distributes them over the VRAM of all GPUs. Like in parallel forward, each GPU processes a part of the batch, and the parameters of other parts of the model are requested on demand during the feedforward from other GPUs [10].

The ZeRO optimizer from deepspeed was the first implementation of this solution. The author decided to use fairscale instead of deepspeed due to the more PyTorch-native method of handling tensors.

Annoy

Annoy is a framework for nearest neighbor searching from Spotify [1] and is mostly written in C++ since searching in large vector spaces is a time-consuming task and C++ applications have a lower resource consumption than python programs. In addition, this framework offers a python API which enables its use with python natively while taking advantage of the speed of C++.

Pyarrow

Pyarrow allows rapidly loading compressed data in the form of parquet files. The main advantage during data exploration is the possibility to only load certain columns and directly filter entries of columns while loading, because the whole dataset is too large to load directly into the memory, and usually the whole table needs to be loaded before filtering [2].

Pandas

Pandas is an easy-to-use and flexible data analysis tool that offers significant functionality for tables and time-series [27, 36]. After filtering and loading, only specific columns were transformed to pandas using the pyarrow tables to gain insights faster.

3.4 Evaluation Scores

Instead of the standard evaluation metrics, special metrics were used during training to mimic productive usage. This section explains the three metrics used between epochs. During training, no distinction is made between different news-portals while during evaluation, each prediction is evaluated on the news articles of their target news portal instead of all news portals, which is important because different news portals can publish similar news articles and the recommendations in production are portal specific.

3.4.1 Cosine Similarity Score

The cosine similarity score indicates the similarity of the predicted vector of the causal transformer decoder model to the target vector produced by LaBSE. It computes an average over each cosine similarity between the output vector and target vector, which can be described as follows:

$$\frac{1}{N}\sum_{i=1}^{N}(c(x,y)) \tag{3.1}$$

where N is the sequence length, x is an output vector, y is a target vector, and c is defined as follows:

$$c : \mathbf{R}^{768} \times \mathbf{R}^{768} \to \mathbf{R}, \tag{3.2}$$
$$x, y \mapsto \frac{x * y}{||x||||y||}$$

3.4.2 Search Cosine Similarity Score

Instead of using corresponding targets, this score uses the most similar vector retrieved from the approximate nearest neighbor search using each output vector of the causal transformer encoder as input for the search algorithm. If the similarity between the retrieved and target vectors is higher than 0.5, it counts as true positive (tp) otherwise as false positive (fp). This approach should simulate the productive behavior of the trained model between the epochs.

$$precision = \frac{tp}{tp + fp} \tag{3.3}$$

3.4.3 Search Precision@100

The metric Search Precision@100 builds on the two previously mentioned scores. An approximate nearest neighbor search is used to determine the 100 most similar vectors to each output vector. A sample is counted as tp when the target news articles' corresponding vector is found in the 100 most similar vectors; if not, then it counts as an fp.

The score is calculated the same way as the search cosine similarity score.

It is more difficult to achieve a high score when applying this approach than with both previously explained scores since there can be more than 100 news articles that are appropriate candidates to recommend; for example, the news portal has more than 1,000 news articles about football and the target news article is about football, so football recommendations would generally be a favorable fit, but the score would only count the prediction as a tp if the target news article is part of the top 100 most similar vectors.

4 Evaluation

The training required 46 hours and 43 minutes for almost 80,000 steps with a batch size of 150 per GPU. The machine used for training has four A100-SXM4 40GB VRAM GPUs, one TB of RAM, and 64 logical CPU cores. The cleaned data were randomly divided into training and test data, where the training data is 99.9% of the data and the test data 0.1%. The sample size is statistically sufficient due to the amount of datapoints. The test data were used for a test loss and to calculate the custom metrics between the epochs.

4.1 Loss

This section presents and analyzes the training and test loss.

Figure 4.1: Train loss for each step

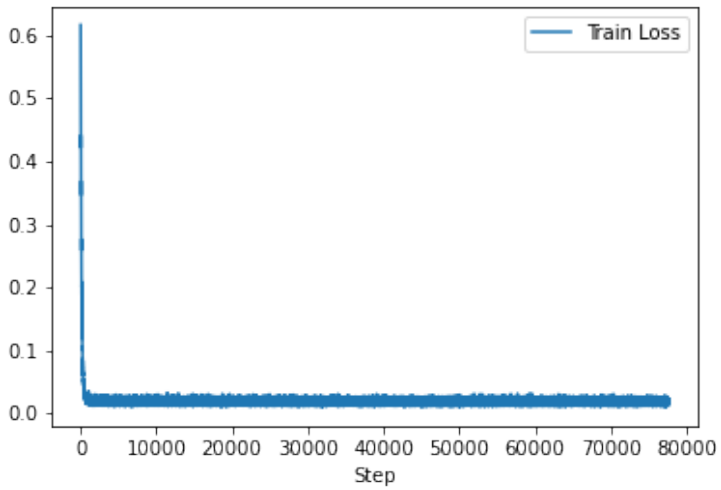
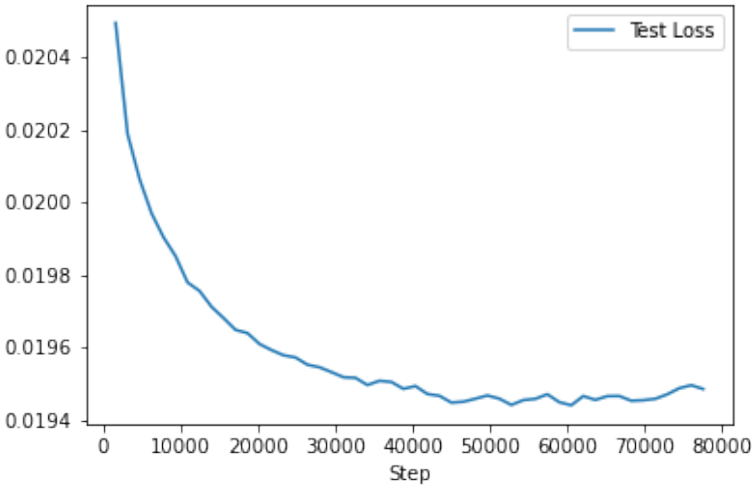


Figure 4.2: Test loss after each epoch

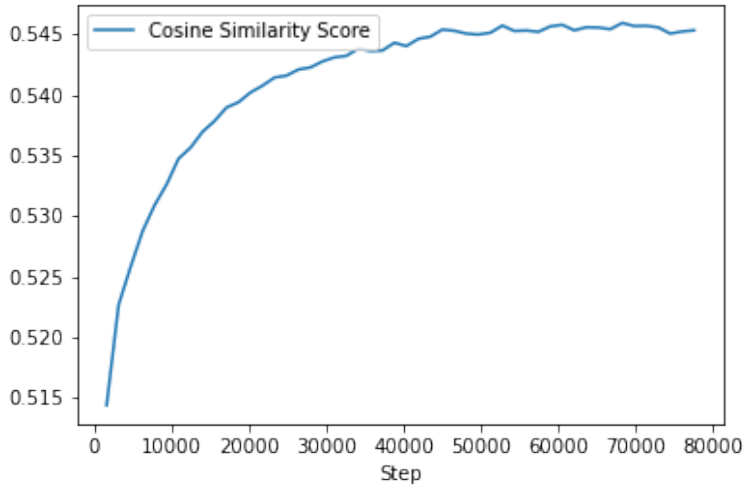


During the first steps, the training loss converges drastically, which shows that the model quickly adjusts to the latent space. In contrast to the training loss, the test loss converges slower. The model with the lowest test loss will be used for A/B testing.

4.2 Evaluation Scores

This chapter presents and analyzes the score results of the training.

Figure 4.3: Cosine similarity score after each epoch



This graph shows that the model generally produces vectors more similar to the target vectors. The score seems generally low, which may be the result of insufficient data.

Figure 4.4: Search cosine similarity score after each epoch

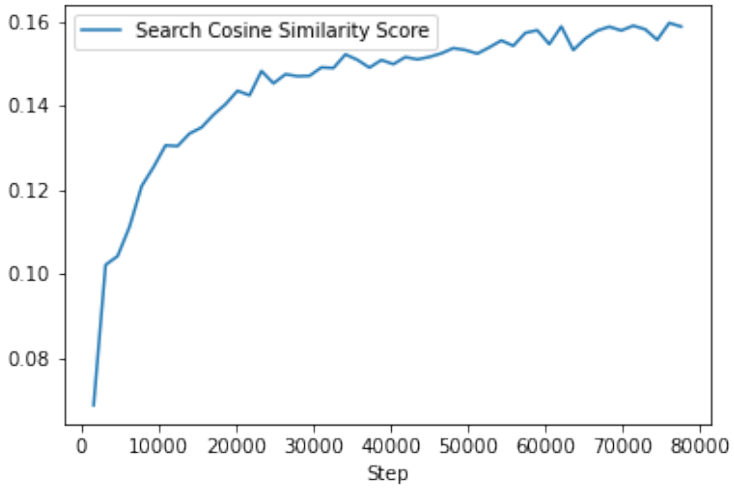
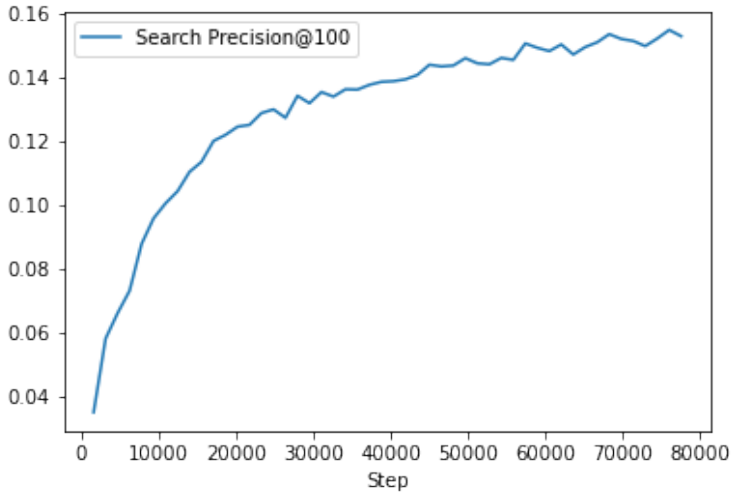


Figure 4.5: Search Precision@100



The two graphs above show that when the similarity between the most similar vector and the target vector is not above 0.5, then the target vector will not be found in the 100 most similar vector to the predicted vector. In contrast to the cosine similarity scores, these results could indicate that the model might not work perfectly in production. These

less-promising scores may also be the result of having insufficient users to generalize since the complexity of predicting embeddings with a dimension of 768 is quite high.

4.3 A/B Testing

The recommender system introduced in this thesis was A/B tested against a proprietary recommender system on four news portals in Germany. The A/B testing was not supervised by the author. The users are divided into the following three groups:

- **A:** Users which receive recommendations using the method introduced in this thesis;
- **B:** Users which receive recommendations using a proprietary recommender system;
- **Fallback:** Users experiencing their first pageviews and thus not receiving recommendations since the recommender system introduced in this thesis requires at least one pageview to produce a recommendation. This group is the control group.

The following three scores were used to measure the performance of each recommender system after the user has seen the recommendations:

- **Click rate:** A score which indicates how often the user clicked on recommendations;
- **User bounced:** A score which indicates how often a user closed the portal after seeing a recommendation;
- **Media time after click:** The time that a user spent on the portal after clicking on the recommendation.

262.721 pageviews were generated for A/B-Testing by real users.

4.3.1 Click Rate

Figure 4.6: Click Rate Mean

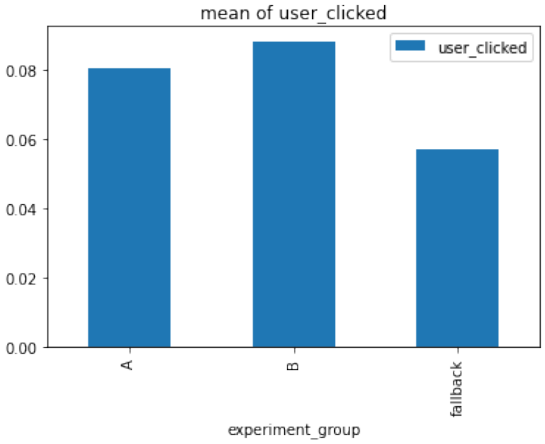
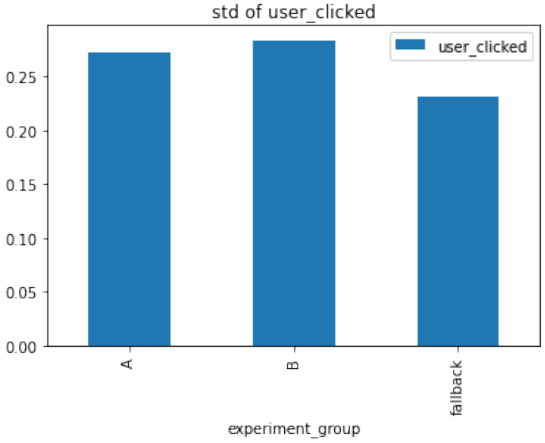


Figure 4.7: Click Rate Standard Deviation



The first iteration of the recommender system introduced in this paper seems to perform worse than the proprietary recommender system. The standard deviation of B is higher, which could imply more stable results of the recommendations received by user group A.

Because this test occurred using the first iteration of the recommender system, the results are promising.

4.3.2 User Bounced

Figure 4.8: User Bounced Mean

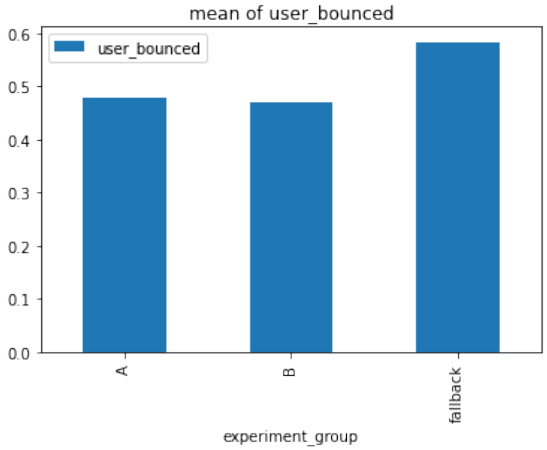
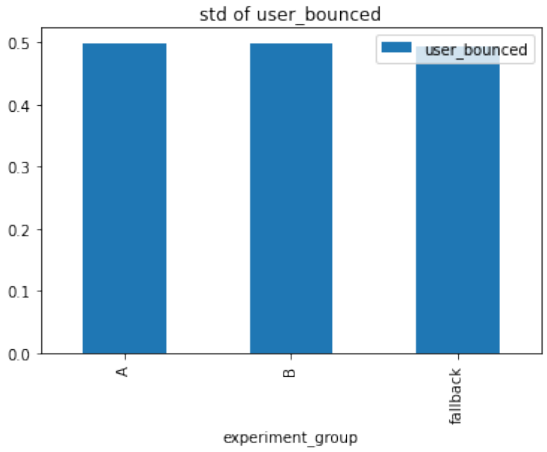


Figure 4.9: User Bounced Standard Deviation



Slightly more users of group A terminated their session on the website after seeing recommendations. Groups A and B generally perform equally, and more than 50% of the users of groups A and B continued their session on the website.

4.3.3 Media time after click

Figure 4.10: Media time after click Mean

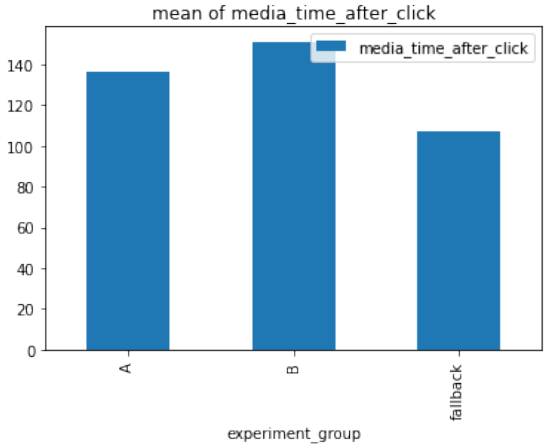
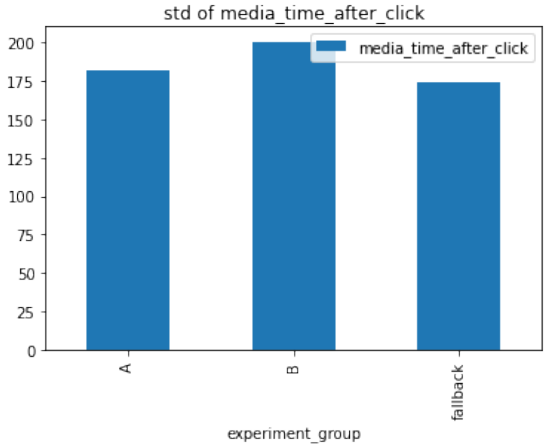


Figure 4.11: Media time after click Standard Deviation



The B group has a higher mean media time after clicking on recommendations, which indicates a better performance of the proprietary recommender system. The graph might indicate a correlation between the user click rate and the media time after clicking. It thus might be possible that if users click on recommendations more often, they might spend more time on the portal.

4.3.4 Conclusion

Group A and group B generally received recommendations which improved their time spent on the portal considering the control group fallback. Group B might have received better recommendations than group A, which might indicate that the proprietary recommender system is better than the recommender system introduced in this thesis. Considering that the first iteration of this system was used for A/B testing, the performance seems impressive. The A/B testing occurred more than half a year after the data for the training were generated, which could imply that training on the latest data could have substantially improved the performance of the algorithm. This favorable performance and the fact that the data were old might justify further research and iterations.

The graphs grouped by portal can be found in the appendix.

5 Conclusion & Limitations

It can be concluded that the pipeline including the model can produce useful recommendations for users considering the A and control group in A/B testing. The recommender system can be applied to varied domains since it only depends on the embeddings of items, which can currently be generated using deep learning for texts, pictures, videos and sounds. The following list consists of possible limitations and outlooks for future research possibilities:

- Transformer models are usually trained with more data than were used in this thesis. The model was trained on approximately 1,806,951 sequences, while a model such as LaBSE has been trained on several billions of sequences. This implies that the causal decoder transformer may not cover the whole domain of news-article content. By including users with at least two pageviews in the training, the model could have performed better since more data would have been used for training.
- The model requires constant retraining due to the velocity and variety of news articles and to counter out-of-domain data. New news articles become published on a daily basis which may lack similarities with older articles besides general similarities, such as articles about football. A model trained before the coronavirus pandemic, for example, might be unable to produce sufficient recommendations during the coronavirus pandemic.
- The read time of an article was not considered while cleaning the data. A user which only reads the first lines of an article still receives a similar article recommendation, although their interest was not sufficient to finish the article. This could be one reason for a decreased media time after clicking.
- No k-fold cross-validation was used, and thus the resulting model may not always be as positive or negative as observed in the experiment.

- Instead of a causal transformer decoder, other models with less parameters could have been used for performance comparison.
- A user with no reading history will not receive a recommendation using this pipeline, although general information about the user is given in the dataset. It would have been possible to add information about the user as a prefix to the time-series of news articles, which would have provided recommendations to a user without the need to read at least one article.
- The number of users could have been increased by truncating the sequences of users with a sequence length longer than 200.
- A benchmark could have been used to compare the technique proposed in this thesis with other techniques since the proprietary recommender system could not be described.
- The model could have been pretrained with randomly generated time-series of all article embeddings. This approach would have given the model a general understanding of the latent space before learning how to predict the next embedding.
- Instead of language-agnostic sentence embeddings, feature vectors/multi-hot vectors could have been used to represent each article, which would have decreased the complexity of the training and thus possibly increased the performance.

In future experiments, the causal transformer decoder will be trained on a much larger dataset.

6 Acknowledgement

I thank Kai von Luck for providing guidance, assistance, constructive critics, knowledge, and interesting discussions through the Master of Science Computer Science program.
I thank the Creative Space for Technical Innovations, which is part of the FTZ Smart Systems at the Hamburg University of Applied Sciences, for providing access to their server infrastructure.

Bibliography

[1] Annoy (Approximate Nearest Neighbors Oh Yeah), howpublished = https://github.com/spotify/annoy, note = accessed: 2022-01-30 .

[2] Apache Arrow, howpublished = https://github.com/apache/arrow, note = accessed: 2022-01-30 .

[3] Fine-Tuning BERT for Sequence-Level and Token-Level Applications, howpublished = https://d2l.ai/chapter_natural-language-processing-applications/finetuning-bert.html, note = accessed: 2022-01-30 .

[4] Language-Agnostic BERT Sentence Embedding, howpublished = https://ai.googleblog.com/2020/08/language-agnostic-bert-sentence.html, note = accessed: 2022-01-30 .

[5] pytorch-lamb, howpublished = https://github.com/cybertronai/pytorch-lamb, note = accessed: 2022-01-30 .

[6] The Illustrated GPT-2 (Visualizing Transformer Language Models), howpublished = https://jalammar.github.io/illustrated-gpt2/, note = accessed: 2022-01-30 .

[7] Understanding Latent Space in Machine Learning, howpublished = https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d, note = accessed: 2022-01-30 .

[8] ZeRO & DeepSpeed: New system optimizations enable training models with over 100 billion parameters, howpublished = https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/, note = accessed: 2022-01-30 .

[9] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 257–264, New York, NY, USA, 2014. Association for Computing Machinery.

[10] Mandeep Baines, Shruti Bhosale, Vittorio Caggiano, Naman Goyal, Siddharth Goyal, Myle Ott, Benjamin Lefaudeux, Vitaliy Liptchinsky, Mike Rabbat, Sam Sheiffer, Anjali Sridhar, and Min Xu. Fairscale: A general purpose modular pytorch library for high performance and large scale training. https://github.com/facebookresearch/fairscale, 2021.

[11] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, sep 1975.

[12] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[13] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. Behavior sequence transformer for e-commerce recommendation in alibaba, 2019.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[15] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

[16] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding, 2020.

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[18] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method, 2014. cite arxiv:1402.3722.

[19] Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), dec 2016.

[20] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[22] L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, Inc., USA, 1st edition, 1999.

[23] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021.

[24] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[25] Danqi Liao. Sentence embeddings using supervised contrastive learning, 2021.

[26] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems, 2019.

[27] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett,

editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

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

[31] Lalita Sharma and Anju Gera. A survey of recommendation system: Research challenges. 2013.

[32] Paul Ullrich and Colin Zarzycki. Tempestextremes: A framework for scale-insensitive pointwise feature tracking on unstructured grids. *Geoscientific Model Development*, 10:1069–1090, 03 2017.

[33] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

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

[35] Dennis D. Wackerly, William Mendenhall III, and Richard L. Scheaffer. *Mathematical Statistics with Applications*. Duxbury Advanced Series, sixth edition edition, 2002.

[36] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[37] 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, 2020.

[38] 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. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[39] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2020.

[40] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. Deep learning for click-through rate estimation, 2021.

A Appendix

A.1 Articles

The following items describe each column of the general information columns:

- **publisher_id** An id of the publisher of the specific news article. A publisher can own multiple news portals. There are ten unique publisher_id.
- **article_drive_id** A unique article id. There are 823,944 unique article_drive_id.
- **article_header** The article header. There are 755,922 unique article headers. Each header consists of between one to thirty words with a mean of 6.12 words.
- **article_teaser** An article teaser. There are 738,363 unique article teasers. Each article-teaser consists of 1-1,475 words with a mean of 23.73 words.
- **article_full_text** The full text of the article. There are 821,750 unique article teasers. Each article-teaser consists of 1-29,832 words with a mean of 387.43 words.
- **is_plus_article** A bool which is set to true if only premium users have access to the article. When the bool is false, then every user can read the article. 149,395 articles are only available for premium users and 656,183 for all users.
- **is_dpa** This bool holds the information if the article is from the news media company Deutsche Presse Agentur (DPA). There are 98,222 articles from the DPA and 725,726 are not from the DPA.
- **article_dpa_id** This is a unique identifier of the DPA. There are 47,290 unique DPA identifiers in this dataset.
- **published_at_local** The local date when the article was published. The time range is 2017-01-01 02:21:11 – 2020-11-10 05:22:08.

- **modified_at_local** A date which indicates when the last modification of the specific article occurred. The time range is 2017-01-01 02:21:11 – 2022-02-22 01:00:00. This date is officially marked as not reliable what explains that the range ends in the future.
- **article_header_contains_quote** A bool which indicates whether the header of an article contains a quote or not. 22,724 articles contain a quote in the header and 801,224 do not.
- **article_header_contains_question** A bool which indicates whether the header of an article contains a question or not. 24,604 articles contain a question in the header and 799,344 do not.
- **article_header_contains_doppelpunkt** A bool which indicates whether the header of an article contains a colon or not. 140,217 contain a colon in the header and 683,731 do not.
- **article_header_contains_pronoun_writer** This bool indicates whether the writer of the article header wrote in the writer's perspective or not. There are 14,626 article headers in the writer's perspective and 809,322 that are not.
- **article_header_contains_pronoun_reader** This bool indicates whether the writer of the article header wrote in the reader's perspective or not. There are 3,159 article headers in the reader's perspective and 820,789 that are not.
- **article_preview_contains_quote** A bool which indicates whether the preview of an article contains a quote or not. 43,290 articles contain a quote in the header and 780,658 do not.
- **article_preview_contains_question** A bool which indicates whether the preview of an article contains a question or not. 62,107 articles contain a question in the header and 761,841 do not.
- **article_preview_contains_doppelpunkt** A bool which indicates whether the preview of an article contains a colon or not. 235,333 contain a colon in the header and 588,615 do not.
- **article_preview_contains_pronoun_writer** This bool indicates whether the writer of the article preview wrote in the writer's perspective or not. There are 39,728 article previews in the writer's perspective and 784,220 that are not.

- **article_preview_contains_pronoun_reader** This bool indicates whether the writer of the article preview wrote in the reader's perspective or not. There are 27,447 article previews in the reader's perspective and 796,501 that are not.

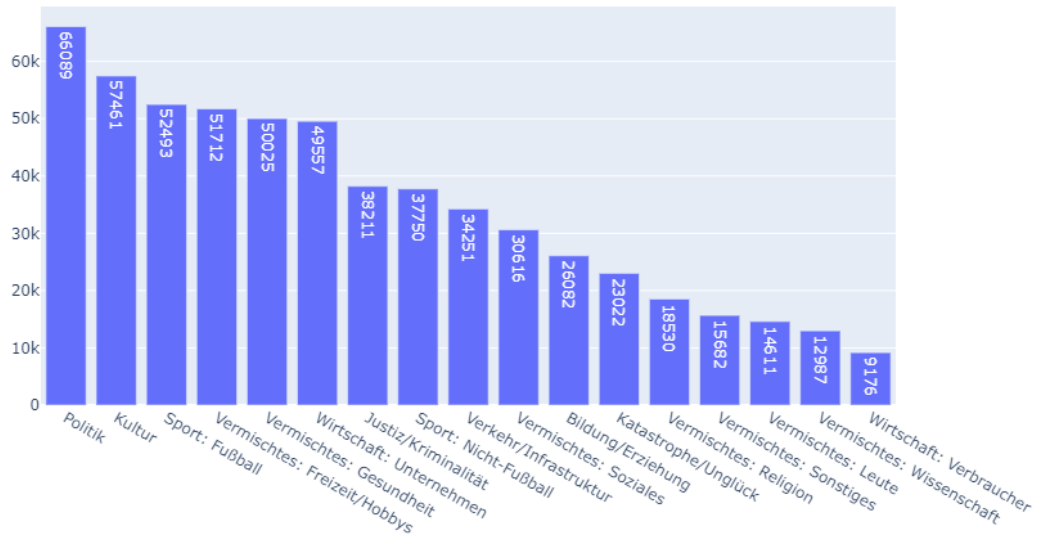
The following items describe each column which were filled using classifiers with most entries presented as floats in ranging from -1 to 1. Every entry smaller than zero will be counted as negative and every other entry as positive:

- **pad_pleasure** This column indicates that the article triggers the emotional state of pleasure. 296,561 articles tend to give pleasure and 9,765 do not.
- **pad_arousal** This column indicates that the article triggers the emotional state of arousal. 4,908 articles tend to cause arousal and 301,418 do not.
- **pad_dominance** This column indicates that the article triggers the emotional state of dominance. 290,812 articles tend trigger dominance and 15,514 do not.
- **preview_pad_pleasure** This column indicates that the article preview triggers the emotional state of pleasure. 334,473 articles tend to cause pleasure and 50,907 do not.
- **preview_pad_arousal** This column indicates that the article preview triggers the emotional state of arousal. 73,006 articles tend to cause arousal and 312,374 do not.
- **preview_pad_dominance** This column indicates that the article preview triggers the emotional state of dominance. 307,074 articles tend to trigger dominance and 78,306 do not.
- **emo_aerger** This column indicates if the article triggers the emotion anger. According to a machine learning model, 81,449 articles tend to trigger this emotion and 224,877 do not.
- **emo_erwarten** This column indicates if the article triggers the emotion expectations. According to a machine learning model, 172,631 articles tend to trigger this emotion and 133,695 do not.
- **emo_ekel** This column indicates if the article triggers the emotion disgust. According to a machine learning model, 68,888 articles tend to trigger this emotion and 237,438 do not.

- **emo_furcht** This column indicates if the article triggers the emotion fear. According to a machine learning model, 87,552 articles tend to trigger this emotion and 218,774 do not.
- **emo_freude** This column indicates if the article triggers the emotion joy. According to a machine learning model, 114,705 articles tend to trigger this emotion and 191,621 do not.
- **emo_traurigkeit** This column indicates if the article triggers the emotion sadness. According to a machine learning model, 174,547 articles tend to trigger this emotion and 131,779 do not.
- **emo_ueberraschung** This column indicates if the article triggers the emotion surprise. According to a machine learning model, 121,711 articles tend to trigger this emotion and 184,615 do not.
- **emo_vertrauen** This column indicates if the article triggers the emotion trust. According to a machine learning model, 243,172 articles tend to trigger this emotion and 63,154 do not.
- **article_preview_emotion** This item is not further analyzed because it consists of the next eight items and is redundant.
- **preview_emo_aerger** This column indicates if the article preview triggers the emotion anger. According to a machine learning model, 255,472 articles tend to trigger this emotion and 129,908 do not.
- **preview_emo_erwarten** This column indicates if the article preview triggers the emotion expectations. According to a machine learning model, 249,997 articles tend to trigger this emotion and 135,383 do not.
- **preview_emo_ekel** This column indicates if the article preview triggers the emotion disgust. According to a machine learning model, 281,310 articles tend to trigger this emotion and 104,070 do not.
- **preview_emo_furcht** This column indicates if the article preview triggers the emotion fear. According to a machine learning model, 236,373 articles tend to trigger this emotion and 149,007 do not.

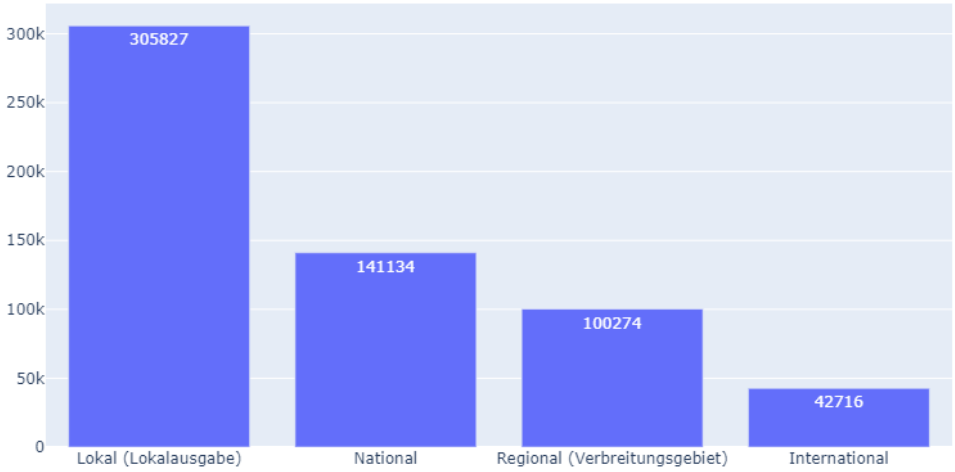
- **preview_emo_freude** This column indicates if the article preview triggers the emotion joy. According to a machine learning model, 205,822 articles tend to trigger this emotion and 179,558 do not.
- **preview_emo_traurigkeit** This column indicates if the article preview triggers the emotion sadness. According to a machine learning model, 278,898 articles tend to trigger this emotion and 106,482 do not.
- **preview_emo_ueberraschung** This column indicates if the article preview triggers the emotion surprise. According to a machine learning model, 266,127 articles tend to trigger this emotion and 119,253 do not.
- **preview_emo_vertrauen** This column indicates if the article preview triggers the emotion trust. According to a machine learning model, 252,792 articles tend to trigger this emotion and 132,588 do not.
- **topic** This column is the classified topic of the article.

Figure A.1: Topic Distribution



- **locality** This column is the classified geographical scope of an article. International articles are internationally relevant while local articles are more likely to only be published in local newspapers.

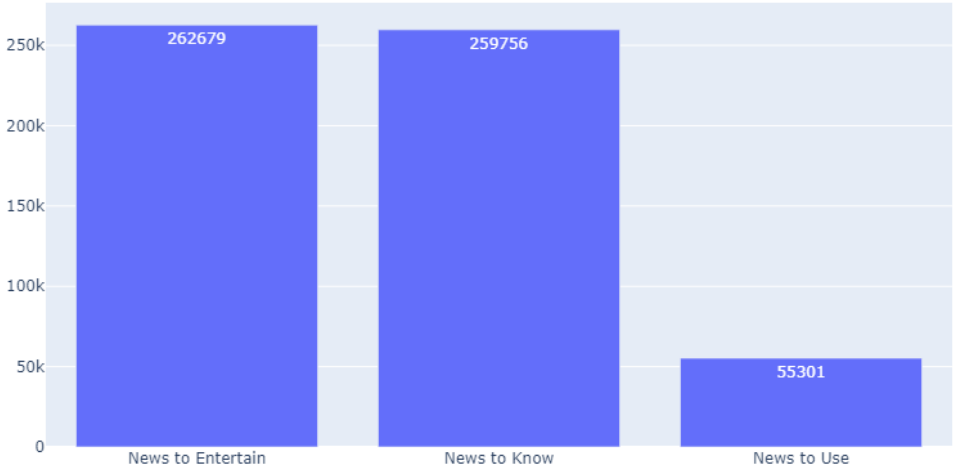
Figure A.2: Locality Distribution



The chart indicates that the portals favor publishing local news articles.

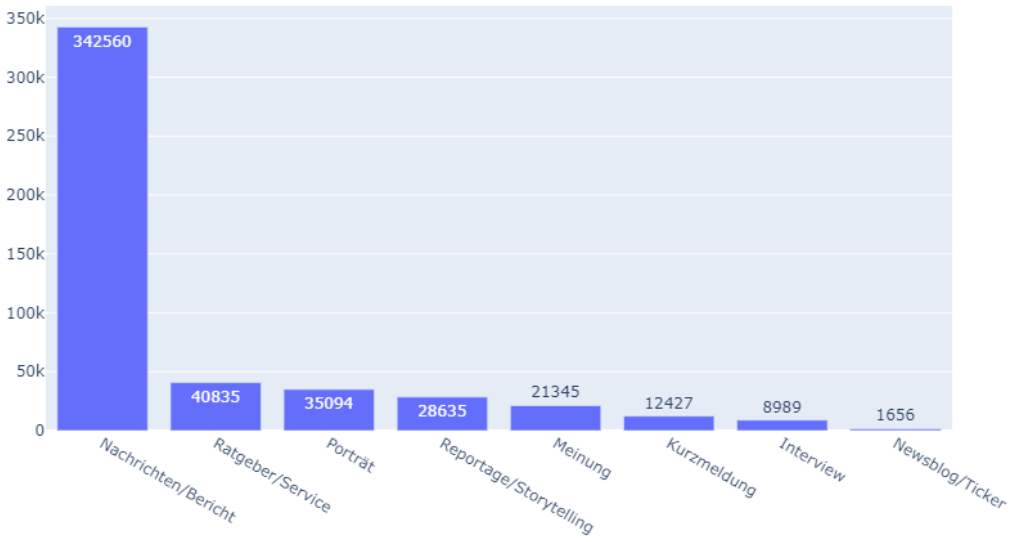
- **newstype** This column is also provided by a machine learning model to classify whether an article tries to tell the reader something for entertainment, knowledge, or usage reasons; for example, an article about something funny occurring in the world may be more likely published to entertain readers, while an article about a local catastrophe tries to inform them.

Figure A.3: Newstype Distribution



- **genre** This classification indicates the genre of an article.

Figure A.4: Genre distribution

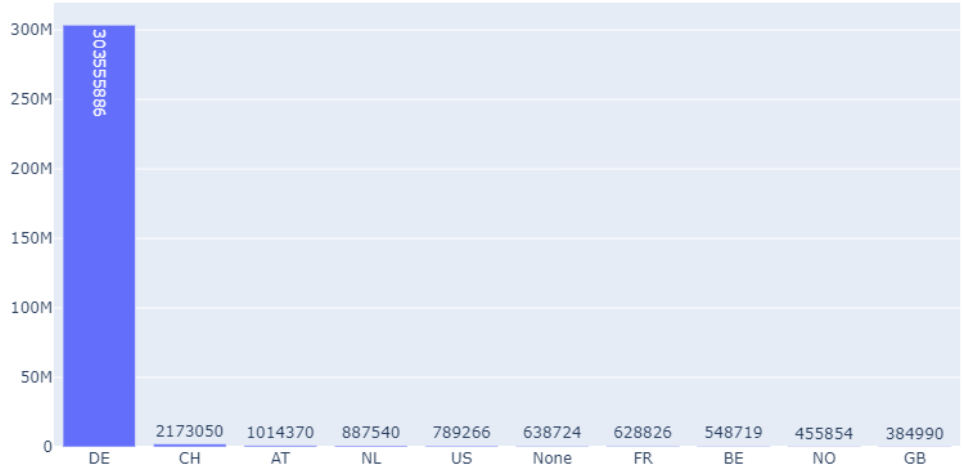


A.2 Pageviews

- **user_id** A unique user identifier. There are 56,199,311 unique user identifiers.
- **session_id** A unique session identifier. There are 167,711,970 unique session identifiers. It indicates which pageview belongs to the same session.
- **session_referer_medium** It indicates how the session was started; for example, the user came to the page over a browser. There are seven unique session_referer_-medium.
- **session_referer_source** This columns adds details to sessio_referer_medium. There are 88 unique session_referer_sources such as "Google"."
- **geo.city** This column shows the city in which the pageview occurred. There are 55,971 unique cities.

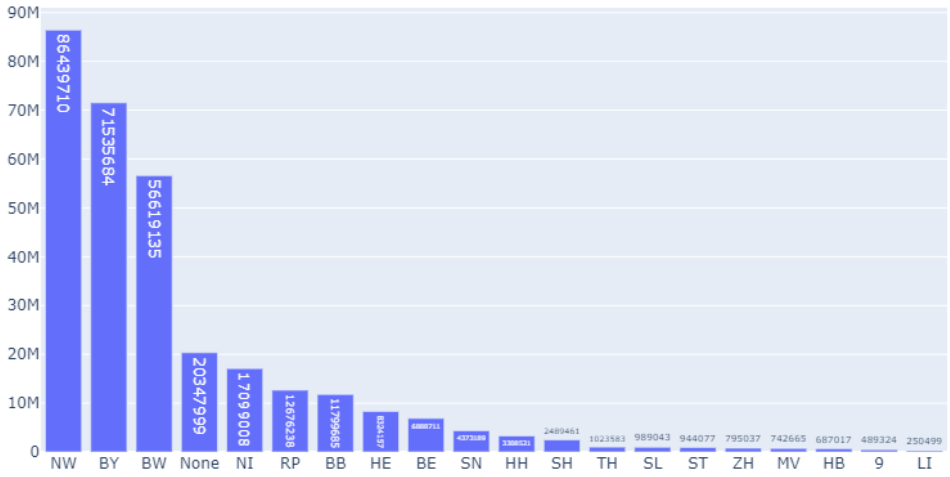
- **geo.country** This column shows the country in which the pageview occurred.

Figure A.5: Top ten countries



- **geo.region** This column shows the region in which the pageview occurred.

Figure A.6: Top twenty regions



- **geo.region_name** This column contains the names of the regions.
- **geo.zipcode** This columns contains 55,412 unique zip codes.
- **browser** This column describes properties of the browser used while accessing the article. It consists of doc_width, doc_height, view_width, and view_height.
- **os** This column describes properties of the operating system used while accessing the article. This column is anonymized.
- **useragent.device_name** This column shows the phone type used while accessing the article. There are 10,617 unique phone types.
- **publisher_id** There are ten publishers of articles with unique identifiers. Each publisher can have multiple portals.
- **portal_id** There are 17 unique portal identifiers. A portal belongs to a publisher and can be understood as a website where articles are published.

- **page_view_id** This columns consists of the unique identifiers of each pageview. There are 312,908,725 unique identifiers. Considering that the length of the whole table is 313,565,551, this identifier is not necessarily correct.
- **url_ebene_1** Anonymized URL structure of the pageview's URL. There are 3,273 unique parts.
- **url_ebene_2** Anonymized URL structure of the pageview's URL. There are 69,909 unique parts.
- **url_ebene_3** Anonymized URL structure of the pageview's URL. There are 89,922 unique parts.
- **referer_url_ebene_1** Anonymized URL structure of the previous pageview's URL. There are 3,913 unique parts.
- **referer_url_ebene_2** Anonymized URL structure of the previous pageview's URL. There are 40,687 unique parts.
- **referer_url_ebene_3** Anonymized URL-structure of the previous pageview's URL. There are 26,384 unique parts.
- **user_type** This column consists of strings which describe the user type. There are three different user types. 267,929,300 pageviews are from users with the user type "anonym", 31,609,829 from the user type "premium", and 14,026,422 from the user type "registered".
- **x_scroll_pct** x pct scrolling of a user on the article.
- **y_scroll_pct** y pct scrolling of a user on the article.
- **x_scroll_pct_min** Max x pct scrolling of a user on the article.
- **y_scroll_pct_min** Min y pct scrolling of a user on the article.
- **time_engaged_in_s** This column indicates how much time the user spent engaged with an article in seconds. The minimum is 0 seconds, the maximum 489,825 seconds, and the mean is 30.87 seconds.

- **article_type** This column indicates whether the article is only for plus users, metering, or free. Metering means that this article belongs to the class of articles where non-premium users can only access those articles a certain amount of times per month. There are 80,521,347 plus articles, 78,200,855 free articles, and 31,081,643 metering articles. This column is based on the user's behavior.
- **is_paywall** This column indicates whether the article accessed has a paywall or not.
- **content_type** This column shows what type of content the user viewed in this pageview. There are 5 different kinds of content. 189,803,845 of the pageviews are "article", 111,773,153 "overview", 6,108,329 "other", 3,980,703 "media", and 1,899,521 "shop".
- **user_engagement_segment** This column shows how engaged a user usually is. There are eight engagement types, such as other which stands for new users or loyal for users who visit the portal regularly.
- **article_drive_id** This identifier links the pageview to a specific article. There are 823,945 unique article identifier in this table.
- **page_view_start_local** This column shows the local date when the user started interacting with the article. The dates are range from 2020-10-01 00:00:00.169000 to 2021-05-21 02:05:19.343000
- **page_view_end_loca** This column shows the local date when the user stopped interacting with the article. The dates range from 2020-10-01 00:00:02.841000 to 2021-05-21 02:05:19.343000
- **completion_time_in_s** This column shows how much time a user needed to complete the article. It ranges from 0.4 seconds to 11,932.4 seconds with an average read time of 199.725 seconds.
- **fraction_article_read** This column shows what fraction of the article the user has read. It ranges from 0 to 1 and the average is 0.24.
- **completion** This column is true if the user completed the article and false if not. 9,201,597 entries are true and 141,750,981 false, and thus most users are not completely reading articles.

A.3 AB-Testing

Figure A.7: Mean click rate grouped by portal

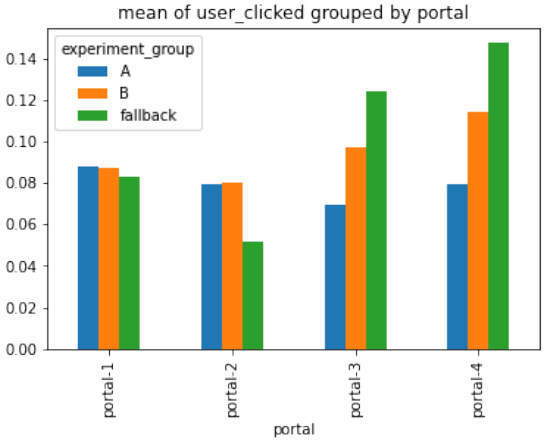


Figure A.8: Click rate standard deviation grouped by portal

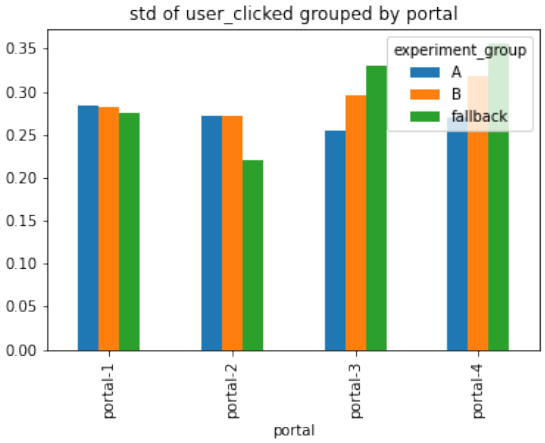


Figure A.9: Mean user bounced grouped by portal

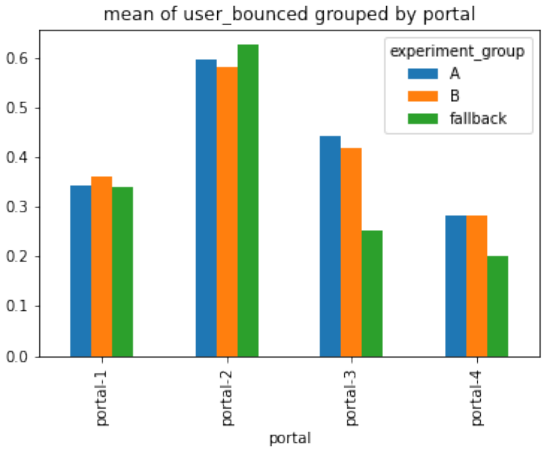


Figure A.10: User bounced standard deviation grouped by portal

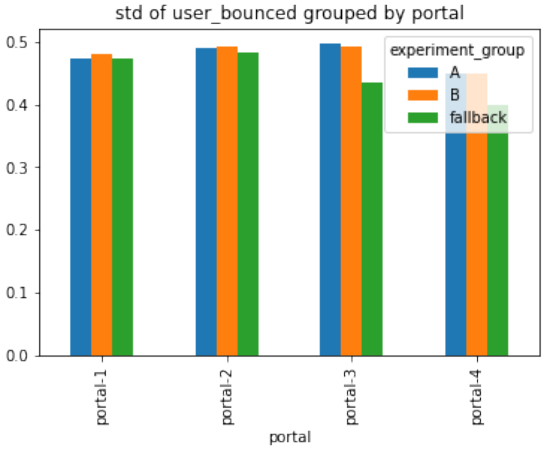


Figure A.11: Mean media time after click grouped by portal

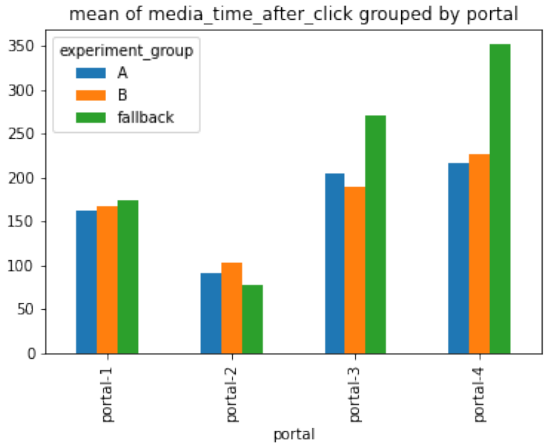
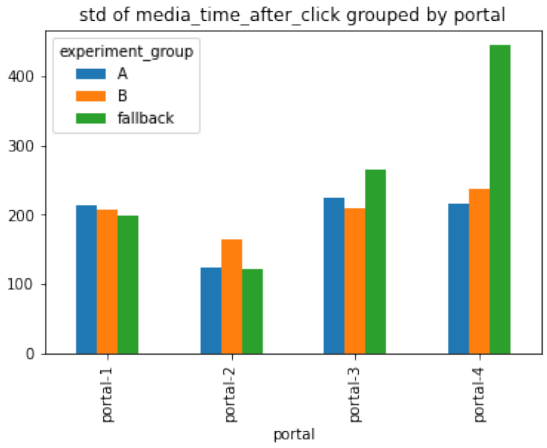


Figure A.12: Media time after click standard deviation grouped by portal



Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original