Enhancing Content-Based Recommendation Systems: A Comparative Study of AI-Based Embeddings and TF-IDF Models

Jonas Klein, SNR 2040881

January 8, 2024

Master Thesis in Marketing Analytics
Tilburg School of Economics and Management
Tilburg University

Supervisor: Hannes Datta, Associate Professor
Second reader: Joep van der Plas, Ph. D. Student

Contents

Abstract

This thesis presents a comprehensive analysis of AI-based embedding techniques in comparison with traditional term frequency-inverse document frequency (TF-IDF) models, specifically in the context of recommendation systems. Using Tilburg University's course catalogue as a case study, this research explores how AI-driven recommendations, particularly through embeddings, can enhance the effectiveness and personalization of course selection for students. The study involves a practical implementation of these techniques and employs a research-driven application to conduct A/B testing, evaluating the performance of both systems. The results revealed that the GPT algorithm did not outperform the TF-IDF algorithm in both Click-Through and Conversion metrics. Logistic regression analyses indicated that Click-through was not significantly different between the two algorithms. The TF-IDF algorithm displayed statistically significant higher conversion rates compared to the GPT algorithm. However, after controlling for time fixed effects, this significant difference was no longer observed, suggesting that the initial advantage of TF-IDF may be attributed to temporal factors. These findings challenge the assumption that more advanced AI techniques like GPT automatically lead to superior performance in content-based recommendation systems.

1 Introduction

Consumers are overloaded with products or services on nearly every online platform on which they have to decide about what to consume (Dellaert and Häubl, 2012). Examples are which movie to watch, which music to stream, which vacation to pick, which groceries to shop online and this way we can think of many more examples. Consumers therefore frequently find themselves engaging in an extensive search of the various available alternatives before making a purchase (Dellaert and Häubl, 2012). Consequently, consumers often feel overwhelmed by an abundance of information, making it challenging for them to accurately identify products or services that truly meet their individual needs.

Businesses can try to guide customers in their search for products or services and solve the information overload problem. One way to guide customers is by making recommendations. The aim of developing recommendation systems is to reduce information overload by retrieving the most relevant information and services from a huge amount of data, thereby providing personalized services (Lu et al., 2015). Content-based systems, in particular, recommend items whose content matches the items a user has assessed.

An emerging application topic in content-based recommendation systems is considering the semantics of an item description to increase recommendation accuracy and enable word disambiguation in the description (Saia et al., 2016). Utilising text to make content-based recommendations is a challenge in that textual descriptions can provide hundreds, if not thousands, of characteristics (words) that are all possible predictors for the matter of interest (Berger et al., 2020). The challenge lies in capturing relationships among words. Applications of relationship extraction in marketing has been restricted so far, probably due to the relative complexity needed in collecting such relationships and analysing semantic similarity (Hartmann and Netzer, 2023).

Because the great majority of text analysis tools used in marketing relied on Bag of Words (BoW) methodologies such as TF-IDF, the capacity to capture actual linguistic links among words beyond their co-occurrence was restricted (Berger et al., 2020). To obtain complex relationships, one must go beyond simple co-occurrence and bag-of-words techniques (Hartmann and Netzer, 2023). This can be achieved with contextual embedding-based approaches. Contextual text embedding approaches have quickly

gained popularity in the natural language processing industry (Hartmann and Netzer, 2023). These approaches have produced excellent success in producing contextual word- and sentence vector representations (Grootendorst, 2022). These vector representations' semantic features allow the meaning of texts to be represented in such a way that related texts are near in vector space (Grootendorst, 2022).

(Hartmann and Netzer, 2023; Grootendorst, 2022). An example of a contextual embedding model is generative pre-training (GPT) (Asudani et al., 2023), a pre-trained language model for embeddings. Lack of training data has frequently been a limiting problem for classic machine learning approaches (Berger et al., 2020). The large-scale pre-training enables language models to acquire fine-grained linguistic knowledge, which is useful for a variety of downstream tasks such as recommendation (Hartmann and Netzer, 2023). In the case of GPT, the model is pre-trained on a massive amount of text data, such as books, articles, and web pages, to learn the statistical patterns and structures of natural language (Encord, nd).

In this study, we aim to address the challenge of information overload for consumers by comparing the effectiveness of contextual embedding techniques, GPT, and conventional embedding techniques, TF-IDF models, for providing personalized recommendations. The research objectives include evaluating the effectiveness of these recommendation systems with Click-through and Conversion metrics with A/B testing. Click-through is a crucial performance measure of online advertising efficacy, search engine success, and general online behaviour (Lei et al., 2023). Both metrics have been used before to measure the effectiveness of recommendation systems (Gai and Klesse, 2019; Dong et al., 2023). The findings of this study will contribute to the literature on recommendation systems and provide insights into contextual embedding models, like GPT, can be used for improving recommendation accuracy and user experience.

Our data is collected by our own research-driven app for Tilburg University's course catalogue. The app employs two distinct recommendation approaches, based on TF-IDF and GPT. Different versions of the recommendation system are shown to different user groups and it is measured which version leads to higher engagement in terms of clicks an conversion on course recommendations. With this testing, we determine which recommendation approach is more effective in engaging students and aiding them in their course selection process.

With our research-driven app, we distinguish this research from the current literature in how to measure the effectiveness of recommendation systems. The emphasis has traditionally been placed on assessing predictive accuracy using offline datasets. A prime example is the widespread use of the Each-Movie database, a benchmark dataset frequently cited in recommendation studies (Chung and Rao, 2012). There is now however an increasing awareness that a recommendation system should not only be accurate in its predictions but also offer a good user experience. Proper evaluation of the user experience of a recommender system requires conducting a user experiment, either in the form of a lab experiment or a randomized field trial, which includes, but also extends beyond, conventional A/B tests (Knijnenburg and Willemsen, 2015).

Our research shows no superior performance in content-based recommendation systems of advanced algorithms like GPT compared to traditional algorithms. The GPT algorithm did not outperform the TF-IDF algorithm in terms of Click-Through and Conversion metrics. Logistic regression analyses showed that Click-Through rates were not significantly different between the two algorithms. The TF-IDF algorithm initially showed statistically significant higher conversion rates compared to the GPT algorithm. However, after controlling for time fixed effects, this significant difference was no longer observed. This suggests that the initial advantage of TF-IDF may be attributed to temporal factors rather than inherent superiority.

This paper is organized as follows. In Chapter 2 we will discuss relevant literature. In Chapter 3 we will look at the data collection method and the data. Chapter 4 will give the research framework and model explanations. Chapter 5 will report the results and Chapter 6 will give a conclusion and final discussion on their implication for marketing theory and practices.

2 Background and literature

This paper aims to contribute to the literature by comparing GPT, contextual embeddings, with TF-IDF models, conventional embeddings, to determine their effectiveness in providing semantically similar content-based recommendations.

2.1 NLP in content-based recommendation systems

According to Bobadilla et al. (2013) and Lu et al. (2015), recommendation systems are systems that offer individualized suggestions by anticipating a user's interest in an item based on information about the objects, users, their interactions, and other sources of information. These systems make immediate product recommendations based on what clients have already watched, liked, or purchased (Gai and Klesse, 2019). Content-based systems, also known as attribute-based systems, suggest products to customers that are comparable to those that they have previously given high ratings. Content-based filtering recognizes the product features that a client likes and suggests products with those same attributes (Ansari et al., 2000).

One way to find similar items is by employing natural language processing (NLP) techniques, specifically embeddings. In this context, text analysis can be utilized to extract and analyze the descriptive elements of items. Embeddings allow for the comparison and analysis of text strings, unveiling relationships and similarities. Word embeddings are unsupervised learning techniques for capturing latent semantic pattern in language (Rudolph et al., 2017).

2.2 TF-IDF for recommendation

Embeddings are already present in early systems that use relatively simple techniques with TF-IDF weighing (Saia et al., 2016). The TF-IDF algorithm is used to evaluate the importance of words in a textual corpus. The importance is proportional to the number of times the words appear in the document and inversely proportional to the frequency of words appearing in the corpus (Chiny et al., 2022). The rare occurrence of terms in document collections has been regarded as a very important feature in information retrieval methods (Kim et al., 2009).

TF-IDF was first introduced by Salton and Buckley (1988). The idea of Salton and

Buckley (1988) was that retrieving things that are likely to be relevant to the user's needs is the first step towards effective retrieval (TF); rejecting items that are likely to be unnecessary is the second (IDF). The precision of the search is affected when all documents are retrieved, as opposed to just a select number, and this is especially the case when the high-frequency phrases are widespread across the collection (Salton and Buckley, 1988).

2.3 Limitation of current systems

There was however quickly a realization that this method does not capture context. The TF-IDF method cannot do a semantic disambiguation of the keywords in an item description as it is based on a simple bag of words (Saia et al., 2016). This realization mainly originated in the computer science domain (Salton and Buckley, 1991; Pickens and Croft, 2000; Kang et al., 2004; Pazzani and Billsus, 2007). For example Kang et al. (2004) already mentioned that the majority of TF-based techniques struggle to derive semantically precise indexes that accurately represent a document's subjects.

A lot of studies in marketing use text analysis which have not been proven to accurately incorporate context. Netzer et al. (2019) explores the predictive power of textual information in loan applications regarding loan default. The authors use a version of LDA analysis, a topic modeling approach (Sievert and Shirley, 2014), to statistically combine words into topics based on their common co-occurrence in documents. This method looks only at co-occurrence, ignoring context.

Hennig-Thurau et al. (2014) analyzes approximately four million tweets to see their impact on consumers' adoption of new movies. Human coders first manually coded a subset of the tweets. These coders read the tweets and classified them based on their content and sentiment (positive or negative). They used 65% of the coded messages as input to train an algorithm that classifies remaining tweets.

Lee et al. (2018) examines how different types of advertising content on Facebook affect consumer engagement. The authors combine the BoW and TF-IDF methods with human-encoded data from Amazon Mechanical Turk (AMT) to train a natural language processing (NLP) algorithm, try to mimics human content coding.

Hewett et al. (2016) analyses a comprehensive dataset, including social media posts and news stories and measures their influence on business outcomes. The authors

used the Linguistic Inquiry and Word Count (LIWC) software. The purpose of LIWC is to provide a quantitative, or numerical, analysis of the content in a text. This is done scanning files word for word and matching each word against a lexicon of terms specified for various sorts, such as positive and negative emotional, financial, and angry words (Hewett et al., 2016). Context is thus limited to the lexicon.

The kinds of thorough method comparisons that computer science would recommend are not conducted in these marketing researches (Hartmann et al., 2019).

In the domain of recommendation systems, research focused on integrating the contextual semantics of words for the purpose of item comparison is limited. Saia et al. (2016) mentions that methods based on a simple bag of words are unable to perform a semantic disambiguation of the terms in an item description; therefore, in order to improve their accuracy, content-based recommendation systems should develop and begin utilising additional data, semantic analysis tools, and external knowledge sources (such as ontologies). The sparse research on the performance of these systems mainly focuses on using WordNet, a large lexical database of English, for incorporation of contextual semantics of words (Capelle et al., 2012, 2013).

2.4 Contextual embeddings for recommendation

To obtain complex relationships, one must go beyond simple co-occurrence and bag-of-words techniques (Hartmann and Netzer, 2023) and also predefined lexicons (Capelle et al., 2012, 2013). This can be achieved with contextual embedding-based approaches. Contextual text embedding approaches have quickly gained popularity in the natural language processing industry (Hartmann and Netzer, 2023). Contextual word embeddings are created during training by taking into account the words that surround a word—that is, the words in the phrase or text span in which it appears (Pilaluisa et al., 2022). Contextual word embedding techniques have had less uses in marketing, at least thus far, due to their relative novelty and data needs (Berger et al., 2022).

The recent rise of pre-trained language models like BERT (Devlin et al., 2019) and GPT (Radford and Narasimhan, 2018) gives great opportunities to tackle the current challenge of adopting contextual word embeddings in the marketing domain. With these pre-trained models, easily accessible via API calls, the marketing community can greatly benefit by adopting transfer learning, already common in the computer science

domain (Hartmann and Netzer, 2023).

This paper contributes to the current marketing literature by integrating contextual embeddings in content-based recommendation systems, with the use of the pre-trained GPT. With OpenAI's open access, OpenAI fosters transfer learning. Text data can be transformed into embeddings using this pre-trained model, which can then be applied to downstream tasks such as recommender systems and similarity search (Hartmann and Netzer, 2023).

The practical use of embeddings in content-based recommendation systems lies in accessing how similar two phrases' meanings are by measuring the distance between their various embeddings (Takyar, 2023). A vector's relatedness is measured by its distance from another vector. Greater distances indicate lower relatedness, whereas smaller distances indicate higher relatedness. For recommendation, embeddings can be helpful since shorter distances between the embedding vectors indicate greater similarity (OpenAI, 2023).

This paper aims to contribute to the current literature by comparing the GPT algorithm, making use of contextual embeddings, with the conventional TF-IDF algorithm that is not able to capture semantic meaning. We try to answer the question if this algorithm leads to considerable higher clicks and conversions.

3 Data description

In this section, we describe our data collection method and data cleaning process, focusing on (1) the data input for our recommendation context, which includes the collection and cleaning processes, (2) the implementation of our research within our research-driven app, and (3) the data collection and manipulation.

3.1 Data input

Our research on content-based recommendation systems has been conducted within a research-driven application environment. The website was made in Python with the Flask framework (Flask Documentation, 2023). The code for this web-app can be seen in the repository on GitHub created for this research (the Appendix contains the complete URL). Figure 1 shows a screenshot of the homepage of the web-app users entered.

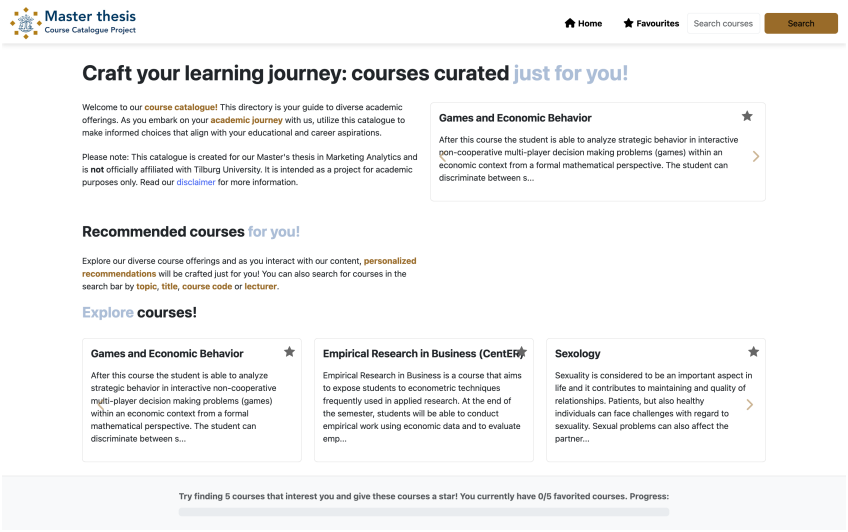

Figure 1: Homepage of the app

In the context of recommendation systems, it is essential to have a variety of items for users to choose from. This is crucial because content-based systems make recommendations based on the characteristics of the items and the preferences of the user. By providing a diverse set of items, we can more accurately assess and enhance the effectiveness and precision of the recommendation system.

In our case, the ”items” are courses offered by Tilburg University. We selected

these courses as the dataset for our recommendation system. We directly scraped these courses from the Tilburg University course catalogue, known as Osiris (see Figure A.1).

We developed an automated scraping system for extracting detailed information about Tilburg University's courses from Osiris and storing this data in a structured database. The system was implemented using Python and Selenium (Selenium with Python, 2023), to automate scraping via a Chrome browser.

We used SQLAlchemy (SQLAlchemy, 2023), to establish connections to a MySQL (MySQL, 2023) database. This allowed for efficient handling of data storage and retrieval operations. Selenium WebDriver, a tool for automating web browser actions, was employed to programmatically control a Chrome browser instance. This enabled the system to interact with web page elements to access course information. This included information such as course names, codes, ECTS credits, languages of instruction, and other details.

The system was designed to insert the scraped data into the database. Once the data was extracted from the website, it was structured into a Python dictionary. This dictionary acted as a temporary storage, holding the key-value pairs corresponding to the various attributes of each course. A connection was established with the MySQL database and then a SQL query was executed to insert the new course data into the database. A screenshot of the database table in SQL can be seen in Figure A.2.

There were mechanisms implemented to handle duplicate entries. In the event of a duplicate entry, the system updated the existing record with the new data. To further enhance the robustness of the scraping process, we implemented other error handling and retry logic. This was particularly crucial for managing interactions with web page elements that may not always behave consistently, such as buttons or links that might fail to respond on initial attempts.

At the time of scraping, there were a total of 2104 courses in Osiris. Since this research focuses on developing a content-based recommendation system for Tilburg University's courses using NLP techniques, it was important to have meaningful and feature-rich course descriptions. Our goal was to create recommendations based on textual characteristics, so we eliminated courses with empty (Figure A.3) or uninformative descriptions, such as those containing only practical details or placeholder text. This process resulted in a refined dataset of 1627 courses, still providing a diverse range of

items for testing our two content-based recommendation systems.

In Table 1 all columns of the courses table in SQL are explained. This information could later be displayed on the website. The courses in the database could be retrieved in our python code. In Figure 2 you can see how this information is displayed on a course page in our web-app.

Column Name	Description
course_name	The name of the course, like ”Business Law” or ”Sociology of Work”.
course_code	A unique code assigned to each course, such as ”100123-B-6”.
language	The language in which the course is taught.
aims	The objectives or goals of the course.
content	Specific topics or material covered in the course.
Degree	The degree program to which the course belongs.
ECTS	The number of ECTS credits the course is worth.
school	The school or department offering the course.
tests	Types of assessments used in the course.
block	The specific time period or term when the course is offered.
lecturers	Names of the instructors or professors teaching the course.
Total Number of Courses (Rows): 1627	

Table 1: Courses table description

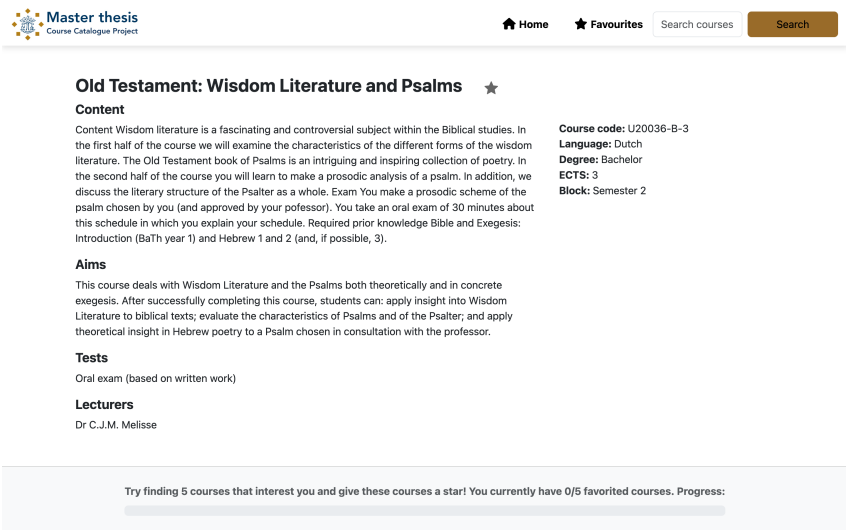

Figure 2: Example course page with course information displayed

3.2 Implementation

Too illustrate how the research was deployed in the research-drive app, we can use Figure 3 to show how users walked through the research-driven app.

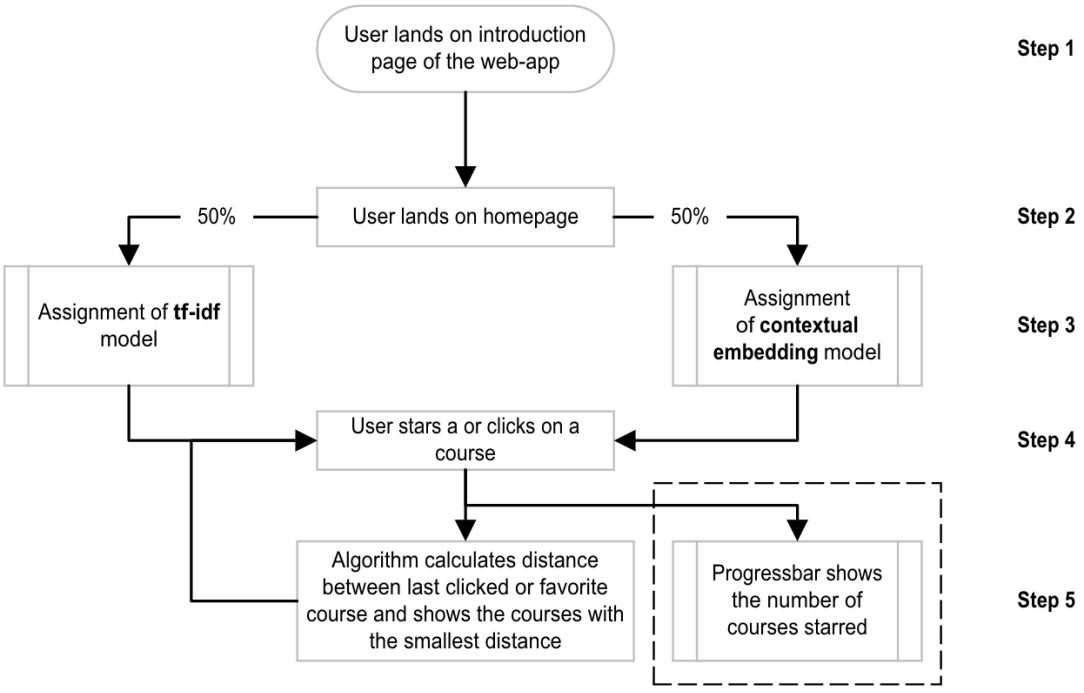

Figure 3: Flow of a user on the website

We guided users to the web-app, but we placed the app in a testing environment. Therefore users did not directly land on the homepage, but we led them to a introduction page of the web-app where we showed instructions of the web-app (Figure A.4, the first step in Figure 3.

We welcomed users to our website with a clear message explaining its purpose and navigation instructions. We guided them on interacting with various features and completing specific tasks. Our instructions directed users to mimic students searching for courses by browsing, clicking, and starring courses, especially noting the site's recommendations and search functionality. We asked them to find and star at least five courses matching their academic interests. Upon clicking 'Show me the courses!', users began a new session, which we tracked using browser cookies. These cookies, small data chunks stored by the web browser (Kennedy, 2023), helped us save user-specific information locally, ensuring no data loss if a user left the site. The users then arrived at the homepage of our app (Figure 1), as depicted in step 2 of our user flow diagram

in Figure 3.

Upon reaching the homepage, we randomly assigned users to one of two recommendation algorithms, either TF-IDF or the GPT model, as illustrated in step 3 of our user flow diagram, Figure 3. Users were free to scroll through the site and discover courses aligning with their interests. A small progress bar at the page's bottom (Figure A.7), provided some guidance, the subprocess at step 5 of Figure 3.

When a user clicked or starred a course (step 4 in our user flow diagram, Figure 3), the assigned algorithm, either TF-IDF or GPT, activated. It then calculated the most similar courses based on the user's last clicked or starred course, as shown in step 5 of our diagram, Figure 3.

The recommendations were shown on either the homepage, as circled in red in Figure A.5, or on the course page, as circled in red in Figure A.6. The courses on the course page are recommendations for the course you are currently watching, since this is the course you last clicked on. Adding these recommendations on the course page is essential, since users like to browse linearly through the course catalogue. This process was repeated as long as the user kept scrolling through the course catalogue.

3.2.1 TF-IDF model

The TF-IDF model works as follows in the background (see Figure 4).The code loads course data from the database and performs text preprocessing. This includes combining the 'aims' and 'content' columns from the SQL table, converting text to lowercase, and removing special characters. A comprehensive list of stopwords in both English and Dutch, along with custom stopwords, is used to filter out unimportant words. The TF-IDF Vectorizer is configured to ignore these stopwords. TF-IDF was first introduced by (Salton and Buckley, 1988). In TF-IDF the first part, TF, stands for term frequency. This measures how frequently a term occurs in its own document.

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in a document } d}{\text{Total number of terms in the document}} \tag{1}$$

The second part, IDF, stands for inverse document frequency. This measures how important a term is over the whole corpus, the collection of documents.

$$IDF(t, D) = \log\left(\frac{\text{Total number of documents in corpus } D}{\text{Number of documents with term } t}\right) \tag{2}$$

The two measurements, Equation 1 and Equation 2, are multiplied to determine the weight of TF-IDF; the higher the weight, the more significant the word in question is within a document.

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D) \tag{3}$$

A text item D_i can be represented by a term vector as $D_i = (w_{i1}, w_{i2}, \ldots, w_{it})$, where w_{ik} is the weight of term T_k assigned to document D_i, provided that t terms in all are accessible. Terms that are not present in a given document are given a weight of zero, whereas terms that are allocated a weight are given a positive value (Salton and Buckley, 1991). These vectors can be represented in TF-IDF matrix. Table 2 shows an example of a TF-IDF matrix. The rows represent the different documents D_i and the columns represent all different words in the total corpus t. This way, every document D_i has a numerical representation, an embedding, of its text with weights to every word t in the corpus. In our web-app the TF-IDF matrix is made in the background by transforming the combined course texts. This matrix represents the importance of words in each course description.

Document	**word 1**	**word 2**	**word 3**	**word 4**	**word 5**	**word 6**	**...**	**word n**
0	0.3010	0	0	0	0	0.1510	...	0
1	0	0	0	0	0.0	0	...	0.201
2	0.0417	0	0	0	0.1	0.0417	...	0
...	...	...	...	...	...	...	...	...
n	0.0209	0.1000	0.1	0.0417	0	0	...	0

Table 2: Sample TF-IDF Weight Matrix for a Corpus of Documents

A vector comparison using the weights of the matched terms may be used to calculate the similarity between documents. With vector elements normalised for vector length, the cosine measure of similarity, which is calculated as the normal inner product between them, has been applied extensively in this context (Salton, 1991). The

formula for the cosine similarity is:

$$\text{Cosine Similarity}(D_i, D_j) = \frac{\sum_{k=1}^{t} w_{ik} \times w_{jk}}{\sqrt{\sum_{k=1}^{t} w_{ik}^2} \times \sqrt{\sum_{k=1}^{t} w_{jk}^2}} \tag{4}$$

Where D_i, D_j are two documents (or text items) for which we are calculating the similarity. In the context of our application, these are two different course descriptions. w_{ik}, w_{jk} are the weights of the k-th term in the documents D_i and D_j, respectively. In the TF-IDF matrix, these weights represent the importance of a word (term) in a document, considering its frequency in that document against its frequency in the entire corpus. t represents the total number of terms in the corpus. In a TF-IDF matrix, this is equivalent to the number of columns, each corresponding to a unique term.

The cosine similarity essentially measures the cosine of the angle between the two vectors representing the documents in a multi-dimensional space (Knox and Datta, 2022). A value closer to 1 indicates high similarity, while a value closer to 0 indicates low similarity.

A cosine similarity matrix is computed in our web-app from the TF-IDF matrix. This matrix is used to find the similarity between courses based on their content. The code iterates through the cosine similarity matrix to identify similar courses for each course. For each course, it sorts other courses based on their similarity score and selects the top similar courses, excluding the course itself. As the last step, the last viewed or starred course from the user's session is identified and the precomputed similarity data is used to recommend similar courses. It fetches the most similar courses from the SQL database, excluding the ones already viewed by the user, and returns them to be shown in the web-app.

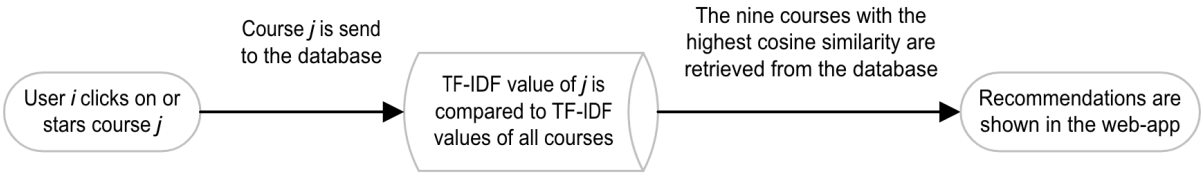

Figure 4: Overview of the TF-IDF algorithm

3.2.2 GPT model

The GPT model works as follows in the background (see Figure 6). For the GPT embeddings we used the description of each course to generate a embedding vector representation for it. Each course's unstructured text description was processed via the model to create the embeddings.

We defined a function in Python to generate text embeddings for each course description using OpenAI's "text-embedding-ada-002" model. To visualize the GPT embeddings made with the "text-embedding-ada-002" model, see Figure 5. Every word is assigned a vector with numbers, determined in the pre-training stage. The output is always a vector with length 1536, no matter the length of the input string. The numbers of the 1536-dimensional vector are always between -1 and 1. This way, the meaning of

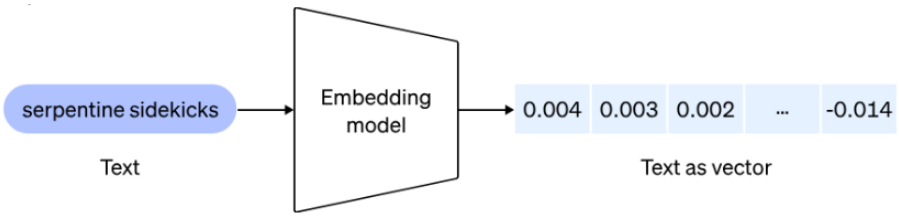

Figure 5: Text to vector conversion: Text is transformed into a numerical vector by the GPT model, allowing the GPT to quantify and interpret language. (OpenAI, 2023)

sentences is represented by a 1536-dimensional vector with numbers between -1 and 1, which you may use to compare it to other sentences (Takyar, 2023).

These embeddings are then saved back to the database, associating each course with its corresponding embedding. By storing these embeddings to the database, we only had to call the OpenAI API once to retrieve embeddings for each course and when a user needed to get recommendations in the web-app, similarities could immediatelly be calculated with the embeddings out of the SQL database.

For the last interacted course, the function finds its index and uses it to obtain its embedding. It then calculates distances between this embedding and others using the same cosine distance metric. Similarity could be calculated in the same way as for the TF-IDF model in Equation 4, since we also are dealing with vectors and need normalization for vector length. Equation 4 can be slightly modified to use it for the embeddings, as shown in Equation 7.

The nearest neighbors are identified, excluding courses the user has already viewed.

As a last step, we are querying the SQL database for detailed information about these recommended courses. It dynamically constructs a SQL query to fetch details of these courses in the order of their relevance, determined by their proximity to the user's last interacted course in the embedding space. This process results in a list of similar courses, which are returned as the recommendation.

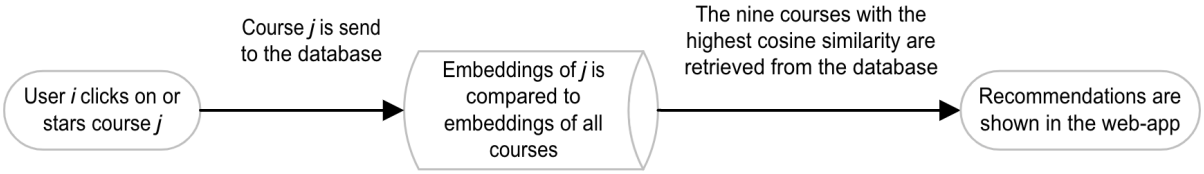

Figure 6: Overview of the contextual embeddings algorithm

3.3 Data

The behaviour of a respondent during their journey was logged in the SQL database. We used this data to examine the effectiveness of the two recommendation algorithms. In Table 3 we show an example of a user's journey as graphically shown in Figure 3.

Table 3: Example of a journey of a user as sent to the SQL database

	ID	timestamp	course_code	activity	algorithm	place
1	f8d...	2023-12-13 13:54:37	none	home	openai	home
2	f8d...	2023-12-13 13:54:51	700398-M-3	clicked	random	random
3	f8d...	2023-12-13 13:54:57	424014-B-6	clicked	openai	rec_coursepage
4	f8d...	2023-12-13 13:55:10	424014-B-6	favorited	openai	rec_coursepage
5	f8d...	2023-12-13 13:55:16	424014-B-6	clicked	favorite	favorite
6	f8d...	2023-12-13 13:55:25	30J102-B-6	clicked	openai	rec_coursepage
7	f8d...	2023-12-13 13:55:27	U11012-B-3	clicked	openai	rec_coursepage
8	f8d...	2023-12-13 13:55:41	U11012-B-3	favorited	openai	rec_coursepage
9	f8d...	2023-12-13 13:55:53	320101-M-6	favorited	openai	rec_coursepage
10	f8d...	2023-12-13 13:56:00	U20036-B-3	clicked	keyword	search
11	f8d...	2023-12-13 13:56:08	U20036-B-3	favorited	keyword	search
12	f8d...	2023-12-13 13:56:00	526406-M-3	clicked	openai	search
13	f8d...	2023-12-13 13:55:27	159890-B-3	clicked	openai	rec_coursepage
14	f8d...	2023-12-13 13:54:37	none	home	openai	home
15	f8d...	2023-12-13 13:55:27	24J006-B-6	favorited	random	random
16	f8d...	2023-12-13 13:55:27	187552-B-3	favorited	openai	rec_home

Note: This is a hypothetical data representation.

In total the behaviour of 175 respondents was logged this way, in the period from November 21, 2023, to December 27, 2023. The distribution over time can be seen in Figure 7.

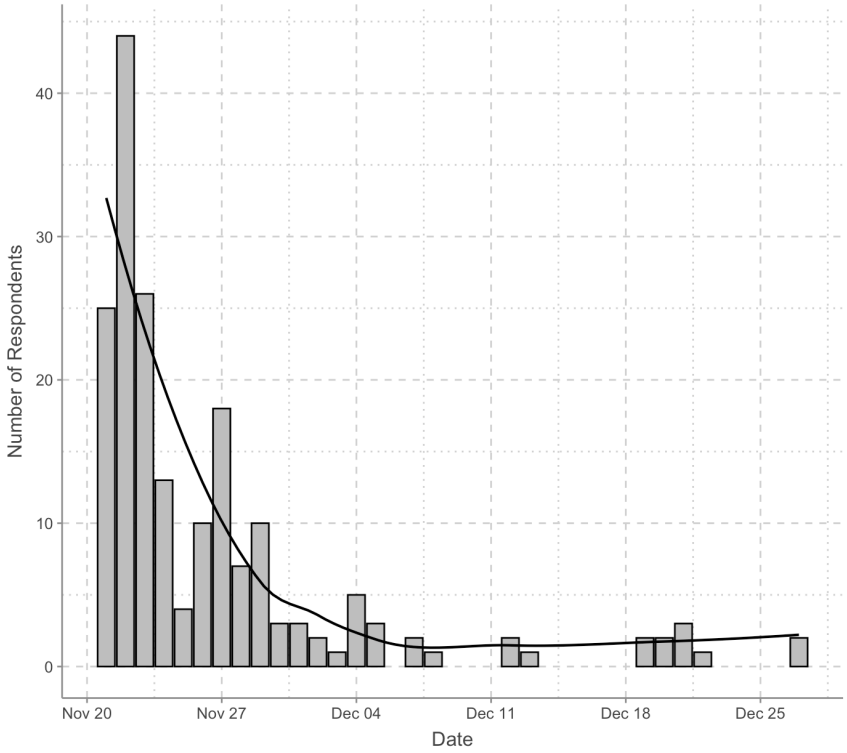

Figure 7: Daily count of respondents

From this 175 respondents 87 people got assigned the TF-IDF algorithm, and 88 people got the GPT algorithm. We excluded respondents that immediately left after landing on the homepage, since these respondents are not yet exposed to any recommendations. After this, we were left with 116 respondents, of which 62 are assigned the TF-IDF algorithm and 54 the GPT algorithm. Therefore, our results reflect the effectiveness measured through CT and CR of respondents that were willing to scroll on the website.

Because the algorithm allocation is random, both the exposed and control groups have comparable unobservable characteristics. The sole variable that distinguishes the two groups is the randomization of exposure to one of the two methods. This indicates that differences in CT and CR can be attributed to the algorithm.

From Table 3 we can make a summary table, Table 4 on the number of 'interactions' of users on the website with courses. Users could either click on a course or give it a star. This could happen on either one of the six places 'random courses', 'recommended courses on the homepage', 'recommended courses on the course page', 'searched courses', 'favorite courses' and 'last viewed courses'. As you can see, the most courses were clicked and starred via the search bar. After this, the most courses were clicked via recommendations made on the coursepage (Figure A.6). In total 1003 interactions

were made with the courses, which means every respondent on average interacted with courses 8.65 times.

Table 4: Summary of the interactions with courses on the website

Place	Activity	Frequency	Cumulative Frequency
Last viewed	Clicked	5	5
Favorite	Clicked	9	14
Random	Clicked	99	113
Random	Starred	124	237
Recommendations on homepage	Clicked	31	268
Recommendations on homepage	Starred	125	393
Recommendations on course page	Clicked	160	553
Recommendations on course page	Starred	78	631
Search	Clicked	221	852
Search	Starred	151	**1003**

3.3.1 Data on Click-through

We looked at the Click-Through (CT) for every single recommendation that was made to all users over the time span of the research. Not every time a user interacted with a course new recommendations were made. When the new interacted course was the same as the course the user interacted with before interacting with this new course, no new recommendations were shown. For example, when a user clicked on a course and also starred this course, no new recommendations were made. In total 650 times new recommendations were shown the users, shown in Table A.1. Nine recommendations were shown at once, resulting in a total of 5850 recommendations made during the time of our research, 50.431 recommendations on average per user.

From Table 3 we were able to get a table with all recommendations made per user and per course, Table 5. This table also shows if the CT for this recommendation was either 0 or 1. For example for user f8d..., when he clicked course U11... (in line 7 of Table 3) nine courses got recommended, thus resulting in nine rows with all different code in the 'rec' column of Table 5. This data thus contains all recommendations made during our research phase.

Table 6 shows descriptive statistics at the recommendation level for the variables from Table 5. As Table 6 indicates, 50.5% of the recommendations were made via the TF-IDF algorithm, and thus 49.5% via the GPT algorithm, based on 5850 observations.

The average CT was 4.6% when look at all the 5850 observations, it being slightly

Table 5: Example of recommendations for a user

	ID	timestamp	algorithm	course	rec	title_len	first_3	place	CT
1	f8d...	2023-12-13 13:55:27	tfidf	U11...	U10...	21	0	course	0
2	f8d...	2023-12-13 13:55:27	tfidf	U11...	U15...	57	1	course	0
3	f8d...	2023-12-13 13:55:27	tfidf	U11...	U20...	10	1	course	0
4	f8d...	2023-12-13 13:55:27	tfidf	U11...	U23...	31	0	course	0
5	f8d...	2023-12-13 13:55:27	tfidf	U11...	U40...	34	1	course	0
6	f8d...	2023-12-13 13:55:27	tfidf	U11...	U46...	38	0	course	0
7	f8d...	2023-12-13 13:55:27	tfidf	U11...	U49...	48	0	course	0
8	f8d...	2023-12-13 13:55:27	tfidf	U11...	J89...	26	0	course	0
9	f8d...	2023-12-13 13:55:27	tfidf	U11...	U45...	37	0	course	0
10	f8d...	2023-12-13 13:55:53	tfidf	320...	400...	39	1	course	0
11	f8d...	2023-12-13 13:55:53	tfidf	320...	500...	41	1	course	0
12	f8d...	2023-12-13 13:55:53	tfidf	320...	502...	41	0	course	0
13	f8d...	2023-12-13 13:55:53	tfidf	320...	550...	40	0	course	0
14	f8d...	2023-12-13 13:55:53	tfidf	320...	553...	40	0	course	0
15	f8d...	2023-12-13 13:55:53	tfidf	320...	710...	28	1	course	0
16	f8d...	2023-12-13 13:55:53	tfidf	320...	800...	14	0	course	0
17	f8d...	2023-12-13 13:55:53	tfidf	320...	840...	17	0	course	1
18	f8d...	2023-12-13 13:55:53	tfidf	J72...	845...	54	0	course	0
19	t79...	2023-11-23 13:08:47	openai	J72...	423...	29	0	home	0
20	t79...	2023-11-23 13:08:47	openai	J72...	583...	18	1	home	0
21	...	...	...	...	...	...	...	...	...

Note: This is a hypothetical data representation. Course codes are truncated so the table fits the page.

higher for the GPT model with 4.7% compared to 4.5% for the TF-IDF model, with 2898 and 2952 observations respectively.

The average title length of the recommendations shown was 30.425 characters, with the smallest title being only 3 characters long and the longest title 99 characters.

One third of the recommendations are among the first three recommendations shown, meaning these are immediately visible. The recommendations were shown in a carousel as Figure A.5 shows, with nine recommendations being divided into three slides in the carousel. Therefore, automatically one third of the recommendations was amongst the first three.

From the 5850 recommendations, only 17.5% were shown on the homepage and the other 82.5% on the course page. This shows that users prefer to navigate through the website linearly. Once they are on a course page users prefer to click on new courses directly from the page they are on and they do not navigate back to the homepage to look at recommendations.

We can also conclude from Table 5 that in total 1274 distinct courses were rec-

ommended, indicating that among the courses that were recommended, a course on average got recommended 4.592 times, with a median of 3. From the 1627 courses that were in the database, thus 1274 courses are discovered, 78.3%. For the two groups GPT and TF-IDF separately a course on average got recommended 3.103 and 3.056 times respectively.

Table 6: Descriptive statistics at the recommendation level

	Mean	Median	St. Dev.	Min	Max	# observations
Algorithm TF-IDF	0.505	1	0.500	0	1	5850
CT	0.046	0	0.210	0	1	5850
GPT	0.047	0	0.212	0	1	2898
TF-IDF	0.045	0	0.208	0	1	2952
Title length	30.425	29	12.609	3	99	5850
First 3	0.333	0	0.471	0	1	5850
Home	0.175	0	0.380	0	1	5850
Recommendations per course	4.592	3	4.652	1	32	1274
GPT	3.103	2	2.741	1	18	934
TF-IDF	3.056	2	3.138	1	26	966

3.3.2 Data on Conversion

The CR will also be extracted from Table 3. Conversion will be determined the moment someone clicked on a course via a recommendation, being 0 if the course was not starred and 1 if the course was starred. This therefore results in 191 usable observations, as Table 4 indicates that 31 times a recommendation on the homepage was clicked and 160 times a recommendation on the course page was clicked. This behaviour is all logged in Table 7.

Table 8 indicates that in 47.6% of the clicks the recommendation was made with the TF-IDF algorithm, thus in 52.4% of the clicks the recommendation was made with the GPT algorithm.

The overall Conversion was 49.7%, but it is only 42% for the GPT algorithm and on the other hand a higher Conversion for the TF-IDF algorithm with 58.2%. For the GPT algorithm, this is based on 100 observations and for the TF-IDF algorithm 91 observations.

Only 16.2% of the recommended courses users clicked on were made via the homepage, the other 83.8% via another course page. The average title length of the 191 courses is 30.654, with the minimum title being 8 characters long and the longest title 85

Table 7: Example of User data on Conversion

	Course Code	ID	Timestamp	Activity	Algorithm	Place	CR	...
1	106254-B-6	96f...	2023-11-23 13:53:33	clicked	tfidf	home	0	...
2	230246-M-6	61a...	2023-11-24 19:06:29	clicked	openai	coursepage	0	...
3	230247-M-6	61a...	2023-11-24 19:06:12	clicked	openai	coursepage	0	...
4	230249-M-6	61a...	2023-11-24 19:06:18	clicked	openai	coursepage	0	...
5	230250-M-6	04d...	2023-11-22 19:32:38	clicked	tfidf	coursepage	0	...
6	230257-M-3	a57...	2023-11-26 14:52:14	clicked	tfidf	coursepage	1	...
7	230273-M-6	61a...	2023-11-24 19:06:35	clicked	openai	coursepage	0	...
8	230273-M-6	d86...	2023-11-24 10:23:56	clicked	tfidf	coursepage	1	...
9	230312-M-6	61a...	2023-11-24 19:06:26	clicked	openai	coursepage	0	...
10	230314-M-6	8f2...	2023-12-22 08:25:40	clicked	openai	coursepage	0	...
11	230315-M-3	901...	2023-11-25 00:39:15	clicked	tfidf	home	0	...
12	230316-M-6	61a...	2023-11-24 19:06:08	clicked	openai	coursepage	0	...
13	230323-M-3	61a...	2023-11-24 19:04:57	clicked	openai	coursepage	0	...
14	230330-M-6	61a...	2023-11-24 19:04:50	clicked	openai	coursepage	0	...
15	230331-M-6	61a...	2023-11-24 19:05:50	clicked	openai	coursepage	0	...
16	230335-M-6	61a...	2023-11-24 19:05:58	clicked	openai	coursepage	0	...
17	230336-M-6	61a...	2023-11-24 19:05:47	clicked	openai	coursepage	0	...
18	230337-M-6	04d...	2023-11-22 19:33:23	clicked	tfidf	home	0	...
19	230337-M-6	61a...	2023-11-24 19:04:43	clicked	openai	coursepage	0	...
20	230337-M-6	8ee...	2023-11-21 12:25:48	clicked	tfidf	coursepage	1	...

Note: This is a hypothetical data representation.

characters. Of the 191 courses, 56% was of the degree bachelor and 44% of the degree master.

Table 8: Descriptive statistics binary variables at the course level

	Mean	Median	St. Dev.	Min	Max	# observations
Algorithm	0.476	0	0.501	0	1	191
Conversion	0.497	0	0.501	0	1	191
GPT	0.420	0	0.496	0	1	100
TF-IDF	0.582	1	0.496	0	1	91
Home	0.162	0	0.370	0	1	191
Title length	30.654	29	13.351	8	85	191
Degree Bachelor	0.560	1	0.498	0	1	191

The course descriptions were categorized into three groups based on their length: the longest third were labeled as 'long', the middle third as 'middle', and the shortest third as 'low'. This way all three groups contain (approximately) the same amount of courses.

The language variable can take on three values. The courses that are clicked by the users are either taught in both English and Dutch, only Dutch or only English. By far most of the 191 recommended courses clicked are being taught in English, namely 161

of the 191 courses.

Table 9: Descriptive statistics categorical variables at the course level

	Frequency	Percentage	Cumulative percentage
Description length			
Low	64	33.508	33.508
Middle	63	32.984	66.492
High	64	33.508	100
Language			
English, Dutch	2	1.047	1.047
Dutch	28	14.660	15.707
English	161	84.293	100

4 Research method

In this section, we describe our research framework. We will focus on (1) the methodology, and (2) the model(s) used to analyse our data.

4.1 Methodology

Our research conducted an A/B-test on the CT and Conversion for both recommendation systems, to measure the effectiveness of both systems. We conducted a between-subjects experiment, where the respondents are randomly assigned to one of the two recommendation systems.

The study has a between-subjects design. Participants in a between-subjects experiment are allocated to one of the experimental conditions at random (Knijnenburg and Willemsen, 2015). Step 3 in Figure 3, is a crucial step in the good execution of the A/B-test; The experiment's sample of participants should ideally be an unbiased (random) representative of the target population (Knijnenburg and Willemsen, 2015). Any variations across individuals are balanced out by randomly allocating them to conditions (Knijnenburg and Willemsen, 2015). In our research randomization happened programmatically the moment users landed on the homepage, so we can assume that there has been proper randomization.

We use logistic regression to model how the recommendation system influences click-through and Conversion. Since logistic regression is ideal for binary outcomes, it effectively handles our dependent variable, which represents the binary states of click-through and non-click-through or conversion and non-conversion. We are interested in how the odds change with a one-unit increase in an independent variable. We will therefore interpret the results with the odds ratio (OR), which is a measure of this change. An OR greater than 1 indicates that the event is more likely as the independent variable increases. An OR less than 1 indicates the event is less likely. To convert our coefficients to OR, we exponentiate them.

Using a logistic regression moreover allows us to add control variables to isolate the effect of the algorithm type on Click-through and Conversion. While we already try to isolate this effect by true randomization, adding control variables helps handling any deviation from complete randomness (Tucker and Zhang, 2010). We will add several

characteristics of the recommendation layout that differs across recommendations and we will add both user fixed effects and time fixed effects.

4.2 Models

We run two separate logistic regression models, one where we regress the algorithm type on Click-through and one where we regress the algorithm type on Conversion.

4.2.1 Model for Click-through

The first model regresses the algorithm with control variables on the click-through for the recommendations j made to user i at time t. We will thus use a logistic regression model, including fixed effects for time and users.

$$\begin{aligned} \text{logit}(\text{CT}_{ijt}) = \beta_0 + \beta_1 \times \text{tfidf}_j + \beta_2 \times \text{visible}_j + \beta_3 \times \text{title\_len}_j \\ + \beta_4 \times \text{homepage}_j + \lambda_t + \alpha_i + \epsilon_{ijt} \end{aligned} \tag{5}$$

Where β_1 is our variable of interest and captures the effect of the TF-IDF algorithm on the log odds of click-through compared to the GPT algorithm. The variable tfidf_j is a binary indicator being 1 if the recommendation j was made with the TF-IDF algorithm, and 0 if it was made with the GPT algorithm.

The second term β_2 captures the effect of a recommendation being among the first three recommendations, and thus being directly visible to the user, on the log odds of click-through. The position of a course can significantly affect its likelihood of being clicked. The items that are immediately visible to the user without scrolling through the carousel generally receive more attention and, consequently, more clicks. This is why we add a dummy variable to account for the position of the recommendation. A control variable for these positions helps to isolate the effect of this increased visibility from the actual performance of the recommendation algorithm.

The third term β_3 captures the effect of the length of the title of a recommendation on the log odds of click-through, it being a continuous variable. A concise title might be easy to understand and more clear, attracting more clicks. However, a longer, more descriptive title may provide more information, attracting more clicks. We need to

control for the effect the appealing nature of the title might have on users intent to click on a course, to not confound with how good the recommendation actually is.

The fourth term β_4 captures the effect of the place where a recommendation is made on the log odds of click-through. A recommendation can either be shown on the homepage or on the page of another course. Including this control variable is relevant, since the layout of the homepage is different form the course page. The homepage offers a variety of options besides recommendations (randomly shown courses, last viewed courses), whereas the course page focuses solely on recommendations. Moreover, users browsing a course page might be in a different stage of the orientation process compared to users on the home page. This difference in intent can affect how they respond to recommendations. By controlling for the place where the recommendation was made, we control for this effect.

The vector λ_t represents time-fixed effects. It captures all unobserved factors that vary over time but are constant across users and recommended items. It includes fixed effects for hour of the day, day of the week and the week. User behavior and responsiveness to recommendations can vary significantly based on these time factors. There may be more extended browsing on both late hours of the day and on weekends due to increased free time. The week is also included since respondents' engagement and interest in surveys can fluctuate over time. Early respondents may be more motivated and later respondents may experience fatigue.

The vector α_i represents user-fixed effects. It captures all unobserved factors that are specific to user i and do not change over time. This way we account for individual user characteristics that might affect their likelihood of clicking through, such as personal preferences or the inherent likelihood to click.

β_0 represents the intercept, the baseline log odds of clicking through when all other variables are zero and ϵ_{ijt} represents the error term.

We will start with reporting the base model of only including algorithm as variable and we will gradually add extra variables. We do this to look at how the effect of the algorithm evolves over this process. Since the type of algorithm is randomly assigned and not related to user characteristics, it should not be correlated with user fixed effects. The algorithm type also does not change over time, so the algorithm type is also not correlated with time fixed effects. However, we do not want to overfit the model.

4.2.2 Model for Conversion

The second model regresses the algorithm with control variables on the conversion for the recommendations j that user i clicked on at time t. We will also use a logistic regression model, including fixed effects for time and users.

$$\begin{aligned}\text{logit}(\text{CR}_{ijt}) = \beta_0 &+ \beta_1 \times \text{tfidf}_j + \beta_2 \times \text{title\_len}_j + \beta_3 \times \text{homepage}_j \\ &+ \beta_4 \times \text{degree\_master}_j + \sum_{k=5}^{6} \beta_k \times \text{description\_len}_{kj} \\ &+ \sum_{k=7}^{8} \beta_k \times \text{language}_{kj} + \lambda_t + \epsilon_{ijt}\end{aligned} \tag{6}$$

Where β_1 is our variable of interest and captures the effect of the TF-IDF algorithm on the log odds of conversion compared to the GPT algorithm. The variable tfidf_j is a binary indicator being 1 if the recommendation j was made with the TF-IDF algorithm, and 0 if it was made with the GPT algorithm.

Terms β_2 and β_3 are the same as terms β_3 and β_4 in the CT model, respectively. However, reason to include β_3 now only are about the different stage of the orientation process users are in and no longer the different layout. The conversion is namely measured after already being landed on the course page via a recommendation, so the layout of the page does not differ. Only the way via which they arrived on the course page differs.

Since users are now on the coursepage as displayed in Figure 2, we can control for more course characteristics that may influence the conversion of users and confound with the actual quality of the recommendation.

The coefficient β_4 captures the effect of a recommendation being of bachelor or master degree on the log odds of click-through. The variable takes on value 1 if the course is of bachelor degree and 0 otherwise. The degree level of a course can signal the level of commitment and depth of study required. This could influence a user's decision to click based on their current education.

The term $\sum_{k=5}^{6} \beta_k$ contains all the $\beta's$ associated with the categorical levels of the description length. The course descriptions were categorized into three groups based on their length: the longest third were labeled as 'long', the middle third as 'middle',

and the shortest third as 'low'. The variable has two coefficients in the model (β_4 and β_5), one for description_len of middle length being either 0 if the description is not of middle length and 1 if it is and one for description_len of high length being either 0 if the description is not of high length and 1 if it is. The first category, low, is used as reference point, to prevent perfect multicollinearity between the three coefficients. It is important to control for the length of a description, since this can influence the likelihood of users starring a course, beyond mere reasons of the algorithm suggesting good courses. Users might lose interest due to the overwhelming length of the text, not starring the course even though it might have fit their interests. On the other hand, long descriptions might signal quality to users, potentially attracting more conversion, without users actually looking if it fits their interests.

The term $\sum_{k=7}^{8} \beta_k$ contains all the $\beta's$ associated with the categorical levels of the language. The variable has two coefficients in the model (β_6 and β_7). One for the course to be taught in both Dutch and English, being either 0 if the language is not Dutch and English and 1 if it is. The second one for the course to be taught in Dutch, being either 0 if the language is not Dutch and 1 if it is. The first category, Dutch, is used as reference point, to prevent perfect multicollinearity between the three coefficients. The language can influence conversion, since courses presented in a language that is native or preferred by the user most likely tend to have higher click-through rates.

We also again include time-fixed effects, λ_t. For this model we do not add user-fixed effects, α_i, since this would overfit the model. There are a large number of users relative to the number of observations. This would make the model too tailored to the specific sample, reducing its predictive power.

We have to mention the exclusion of some course characteristics. As can be seen in Figure 2 there are still some characteristics we did not account for.

The first is the number of ECTS. Its exclusion simplifies the model without sacrificing explanatory power; both the degree level, which is included, and the number of ECTS are likely to be perceived by users as indicators of a course's difficulty. If Degree and ECTS are both included they could be representing the same construct, leading to multicollinearity. The other characteristics, the block in which the course is taught, the tests for the course and the lecturers by whom the course is taught are also excluded since they are not expected to significantly alter the decision for users to star a course.

5 Results

5.1 The effect of recommendation algorithm on Click-through

The key results of the CT model are displayed in Table 10.

The effect of going from the GPT algorithm to the TF-IDF algorithm is negative as long as no fixed effects are added, having an OR of 0.970 (exp(-0.03)), which suggests that items recommended by the TF-IDF algorithm are less likely to be clicked on. When time fixed effects are added the odds are further decreased with an OR of 0.896 (exp(-0.11)) for click-through. With user fixed effects added a recommendation recommended with the TF-IDF algorithm is 2.63 (OR = exp(0.97)) times as likely to be clicked. The effect is then thus reversed. However, the results in all columns are not significant, so there is no significant change in the odds of a click-through happening when going from the GPT algorithm to the TF-IDF algorithm.

There is a significant effect for the place when controlling for time fixed effects and when controlling for user fixed effects. The odds of clicking on a course when this recommendation is made on the homepage instead of on a course page are multiplied by 0.684 (OR = exp(-0.38)) and 0.440 (OR = exp(-0.82)), respectively. This thus suggest that users are more likely to click on a recommended course when this recommendation is made on the course page instead of the homepage, most likely due to the layout of a course page compared to the homepage.

We tried controlling for both time fixed effects and user fixed effects at the same time, but the model could not converge to a solution when adding both. This suggests that the model becomes too complex when adding too much control variables, leading to possible overfit and multicollinearity between levels of the fixed effects variables.

5.2 The effect of recommendation algorithm on Conversion

The key results of the Conversion model are shown in Table 11.

Overall, results in all columns, except the last one where time fixed effects are added, suggest a positive significant effect on conversion for recommendations when this recommendation was made with the TF-IDF algorithm instead of the GPT algorithm.

The effect in isolation, the Null Model, shows a recommendation made with the

Table 10: Click-through results

	Dependent variable:					
	CT					
	Null Model	+ Visibility effects	+ Title Length Effects	+ Place Effects	+ Time Fixed Effects	+ User Fixed Effects
	(1)	(2)	(3)	(4)	(5)	(6)
Tfidf	−0.03	−0.03	−0.03	−0.03	−0.11	0.97
	(0.12)	(0.12)	(0.12)	(0.12)	(0.14)	(1.25)
First_3		0.17	0.17	0.17	0.17	0.17
		(0.13)	(0.13)	(0.13)	(0.13)	(0.13)
Title_len			0.001	0.001	0.001	0.002
			(0.005)	(0.005)	(0.005)	(0.001)
Home				−0.24	−0.38*	−0.82*
				(0.18)	(0.19)	(0.21)
Time fixed effects					✓	
User fixed effects						✓
Constant	−3.01***	−3.07***	−3.11***	−3.07***	−1.50	−3.89***
	(0.09)	(0.10)	(0.18)	(0.19)	(1.22)	(1.02)
Observations	5,850	5,850	5,850	5,850	5,850	5,850

Note: *p<0.1; **p<0.05; ***p<0.01

TF-IDF algorithm to be starred 1.935 times as likely (OR = exp(0.66)). This effect only grows when adding control variables, especially controlling for the place where the recommendation was made. In this case, recommendations are 2.203 as likely to be starred when recommended with the TF-IDF algorithm (OR = exp(0.79)).

The coefficient of whether the recommendation was made on the homepage or a course page is highly significant (at the one percent level) in every column. After controlling for time fixed effects this coefficient is at it highest with the odds being multiplied by 0.005 (OR = exp(-5.22)) when a course is recommended on the homepage instead of another course page. Most likely due to the stage of the orientation process users are in, suggesting users are more likely to star a course when they already saw other courses (later in the orientation process).

The coefficient for the language variable being in the category English is also significant, at the 5 percent level, as long as the time fixed effects are left out. When a course is taught in English instead of Dutch the courses are 2.829 (OR = exp(1.04)) times more likely to be starred. This suggests people preferring English as the language for a course.

Other variables such as the length of the course title, the user's degree, and the description length do not show a significant impact on conversion rates. This indicates that these factors do not play a critical role in the user's decision to star a course

When adding time fixed effects only the significant effect of the place where the recommendation is made remains. The coefficient for tfidf remains positive and significant across models 1 to 6, but becomes non-significant in model 7 when time fixed effects are included. This suggests that some of the explanatory power of the tfidf variable was due to time-based factors. The effect may be partly due to some other time-related trends that affects the dependent variable, such as users starring more courses in the weekend due to more free time.

Table 11: Conversion results

	Dependent variable:						
	CR						
	Null Model	+ Title Length Effects	+ Place Effects	+ Degree Effects	+ Description Length Effects	+ Language Effects	+ Time Fixed Effects
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Tfidf	0.66**	0.67**	0.79**	0.79**	0.78**	0.79**	−0.09
	(0.29)	(0.29)	(0.32)	(0.32)	(0.32)	(0.33)	(0.51)
Title_len		0.01	0.004	0.003	0.003	0.003	−0.001
		(0.01)	(0.01)	(0.01)	(0.01)	(0.01)	(0.02)
Home			−3.08***	−3.06***	−3.08***	−3.34***	−5.22***
			(0.76)	(0.76)	(0.76)	(0.78)	(1.18)
Degree				−0.12	−0.13	0.22	−0.41
				(0.32)	(0.33)	(0.37)	(0.53)
Description_len							
Middle					−0.11	−0.11	−0.30
					(0.39)	(0.39)	(0.56)
High					0.10	0.32	0.18
					(0.40)	(0.43)	(0.62)
Language							
Dutch, English						−14.93	−13.11
						(1,023.65)	(2,788.14)
English						1.04**	0.64
						(0.52)	(0.69)
Time fixed effects							✓
Constant	−0.32	−0.54	−0.15	−0.07	−0.04	−1.13	−12.06
	(0.20)	(0.40)	(0.43)	(0.49)	(0.51)	(0.76)	(3,956.18)
Observations	191	191	191	191	191	191	191

Note: *p<0.1; **p<0.05; ***p<0.01

6 Discussion and conclusion

6.1 Conclusion

Consumers get overwhelmed with items or services on practically every online platform where they must choose what to consume (Dellaert and Häubl, 2012). Businesses can use recommendation systems to assist customers in their search for products or services, therefore addressing the issue of information overload. This research focused on content-based recommendation systems, with a focus on contextual embedding techniques like GPT and the traditional technique TF-IDF. Because the vast majority of marketing text analysis tools depended on BoW approaches such as TF-IDF, the ability to capture true linguistic relationships between terms beyond their co-occurrence was limited (Berger et al., 2020). Complex relationships require more than basic co-occurrence and bag-of-words approaches (Hartmann and Netzer, 2023). Contextual embedding-based techniques can help with this.

This paper investigates whether GPT, contextual embeddings, outperform TF-IDF models in providing semantically similar content-based recommendations. We did not find evidence that the GPT algorithm outperformed the TF-IDF algorithm. The GPT algorithm did not outperform the TF-IDF algorithm in terms of Click-Through and Conversion metrics.

The first logistic regression analyses revealed that there were no significant differences in Click-Through Rates (CTR) between the two algorithms across all models. This conclusion held even after adjusting for potential confounding variables in an attempt to isolate the effect. The analyses consistently found no significant impact of the two different algorithms.

The second logistic regression analyses showed significance in the first six models. The TF-IDF algorithm showed statistically significant higher conversion rates compared to the GPT algorithm. However, after controlling for time fixed effects, this significant difference was no longer observed. This suggests that the initial advantage of TF-IDF may be attributed to temporal factors rather than inherent superiority.

Neither one of the two algorithms thus performed better than the other. These findings challenge the assumption that more advanced contextual embedding techniques like GPT inherently provide superior performance in content-based recommendation

systems. While GPT and similar models are theoretically capable of capturing more nuanced linguistic relationships, our analysis shows that this does not necessarily translate into significantly better outcomes in practical metrics like Click-through and Conversion in recommendation systems.

6.2 Managerial implications

Making recommendations has the advantage of reducing user search effort while rewarding the firm with increased customer loyalty and sales (Ansari et al., 2000). It is thus a good practice to develop recommendation systems, whether content-based, such as the TF-IDF or GPT algorithms discussed in this thesis, or in any other form.

This research shows some important takeaways for managers wanting to implement content-based recommendation systems on their website. The research indicates that more complex algorithms like contextual embeddings do not necessarily result in a significant increase in click-through and conversion, while these are, especially conversion, the main interest for businesses. In cases where the recommendation content is straightforward, simpler models like TF-IDF could be equally effective.

Moreover, implementing contextual embeddings involves additional costs and complexities. These costs could be in terms of financial expenditure for API usage, such as the OpenAI API used in this research, as well as the technical expertise required to integrate and maintain such systems.

6.3 Limitations and future research

Our study has a few limitations that suggests interesting future research topics. The first one being the sample size. The first model has a big sample size of 5850 observations, however, the second model only has 191 observations. From the total 1003 interactions with courses, only 19.04% were direct clicks on recommended courses. This greatly reduced our sample size and therefore the power of our results (Cohen, 1992).

Secondly, the descriptions for the courses often lacked clarity, with many lacking substantive information about the content taught within the course. Efforts were made to filter and refine these descriptions to the best of our ability, but it is important to acknowledge that, overall, the descriptions were poorly structured. If the descriptions

lacks informative features, they may not be useful for our recommendation systems. For the comparative research of our two content-based recommendation systems, the same descriptions data is used for both systems, thus ensuring a fair comparison. However, the impact of the quality of the input data, especially in cases where one algorithm may be more sensitive to the richness and quality of data than the other, is a critical aspect to consider. The GPT algorithm thrives on rich, detailed, and contextually dense descriptions. These models are designed to understand human-like text, so the more informative and feature-rich the input, the better these models can capture the content's nuances. It might suffer more from poor-quality input compared to a more straightforward algorithm, like TF-IDF. Thus, while the current research setup ensures consistency in data (descriptions) for both systems, exploring how each algorithm performs with richer data could be a promising direction for future research.

Lastly, we may not have taken enough steps in ensuring effective randomization. There were 34 dropouts for the GPT algorithm and only 25 for the TF-IDF algorithm. This is a problem when the reasons for dropout are related to the survey content or the characteristics of the respondents. In our case, dropout reasons are unlikely to be attributed to survey content, as the content was identical for both groups up to the point of dropout, and no recommendations were presented by then. Moreover, the randomization protocol was followed correctly without any systematic biases in the assignment process, also since the algorithm was assigned by the computer program. A good practice would still be to examine the distribution of baseline characteristics across the two groups (Lei et al., 2023). We were not able to do this test, since we did not collect data on these pre-experimental characteristics. Future research should therefore always collect such characteristics to be able to ensure randomization statistically.

Bibliography

Ansari, A., Essegaier, S., and Kohli, R. (2000). Internet recommendation systems. *Journal of Marketing Research - J MARKET RES-CHICAGO*, 37:363–375.

Asudani, D. S., Nagwani, N. K., and Singh, P. (2023). Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial Intelligence Review*, 56(9):10345–10425.

Berger, J., Humphreys, A., Ludwig, S., Moe, W. W., Netzer, O., and Schweidel, D. A. (2020). Uniting the tribes: Using text for marketing insight. *Journal of Marketing*, 84(1):1–25.

Berger, J., Packard, G., Boghrati, R., Hsu, M., Humphreys, A., Luangrath, A., Moore, S., Nave, G., Olivola, C., and Rocklage, M. (2022). Wisdom from words: marketing insights from text. *Marketing Letters*, 33:1–13.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, 46:109–132.

Capelle, M., Frasincar, F., Moerland, M., and Hogenboom, F. (2012). Semantics-based news recommendation with sf-idf. *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, 27.

Capelle, M., Hogenboom, F., Hogenboom, A., and Frasincar, F. (2013). Semantic news recommendation using wordnet and bing similarities. *Proceedings of the ACM Symposium on Applied Computing*.

Chiny, M., Chihab, M., Bencharef, O., and Younes, C. (2022). Netflix recommendation system based on tf-idf and cosine similarity algorithms. pages 15–20.

Chung, J. and Rao, V. R. (2012). A general consumer preference model for experience products: Application to internet recommendation services. *Journal of Marketing Research*, 49(3):289–305.

Cohen, J. (1992). Statistical power analysis. *Current Directions in Psychological Science*, 1(3):98–101.

Dellaert, B. G. and Häubl, G. (2012). Searching in choice mode: Consumer decision processes in product search with recommendations. *Journal of Marketing Research*, 49(2):277–288.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dong, B., Zhuang, M., Fang, E. E., and Huang, M. (2023). Tales of two channels: Digital advertising performance between ai recommendation and user subscription channels. *Journal of Marketing*, 0(0):00222429231190021.

Encord (n.d.). Generative pre-trained transformer (gpt) definition. Accessed: December 7, 2023.

Flask Documentation (2023). Welcome to flask. https://flask.palletsprojects.com/en/3.0.x/. Accessed: [Insert the date you accessed this resource].

Gai, P. J. and Klesse, A. (2019). Making recommendations more effective through framings: Impacts of user- versus item-based framings on recommendation click-throughs. *Journal of Marketing*, 83(6):61–75.

Grootendorst, M. (2022). Bertopic: Neural topic modeling with a class-based tf-idf procedure.

Hartmann, J., Huppertz, J., Schamp, C., and Heitmann, M. (2019). Comparing automated text classification methods. *International Journal of Research in Marketing*, 36(1):20–38.

Hartmann, J. and Netzer, O. (2023). *Natural Language Processing in Marketing*, pages 191–215.

Hennig-Thurau, T., Wiertz, C., and Feldhaus, F. (2014). Does twitter matter? the impact of microblogging word of mouth on consumers' adoption of new movies. *Journal of the Academy of Marketing Science*, 43(3):375–394.

Hewett, K., Rand, W., Rust, R. T., and van Heerde, H. J. (2016). Brand buzz in the echoverse. *Journal of Marketing*, 80(3):1–24.

Kang, B.-Y., Kim, H., and Lee, S.-J. (2004). Performance analysis of semantic indexing in text retrieval. volume 2945, pages 433–436.

Kennedy, P. (2023). Sessions in flask. Accessed: 2023-12-01.

Kim, J., Li, J., and Lee, J.-H. (2009). Discovering the discriminative views: Measuring term weights for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics*.

Knijnenburg, B. and Willemsen, M. (2015). *Evaluating Recommender Systems with User Experiments*, pages 309–352.

Knox, G. and Datta, H. (2022). Platform-guided consumption similarity. *SSRN Electronic Journal*.

Lee, D., Hosanagar, K., and Nair, H. S. (2018). Advertising content and consumer engagement on social media: Evidence from facebook. *Management Science*, 64(11):5105–5131.

Lei, X., Chen, Y., and Sen, A. (2023). The value of external data for digital platforms: Evidence from a field experiment on search suggestions. *SSRN Electronic Journal*.

Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems*, 74:12–32.

MySQL (2023). MySQL. https://www.mysql.com/. Accessed: 2023-12-12.

Netzer, O., Lemaire, A., and Herzenstein, M. (2019). When words sweat: Identifying signals for loan default in the text of loan applications. *Journal of Marketing Research*, 56(6):960–980.

OpenAI (2023). What are embeddings? Accessed: November 29, 2023.

Pazzani, M. J. and Billsus, D. (2007). *Content-Based Recommendation Systems*, pages 325–341. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pickens, J. and Croft, W. B. (2000). An exploratory analysis of phrases in text retrieval. In *RIAO Conference*.

Pilaluisa, J., Tomás, D., Navarro-Colorado, B., and Mazón, J.-N. (2022). Contextual word embeddings for tabular data search and integration. *Neural Computing and Applications*, 35(13):9319–9333.

Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

Rudolph, M., Ruiz, F., Athey, S., and Blei, D. (2017). Structured embedding models for grouped data.

Saia, R., Boratto, L., Carta, S., and Fenu, G. (2016). Semantics-aware content-based recommender systems: Design and architecture guidelines. *Neurocomputing*.

Salton, G. (1991). Developments in automatic text retrieval. *Science*, 253(5023):974–980.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513–523.

Salton, G. and Buckley, C. (1991). Global text matching for information retrieval. *Science*, 253(5023):1012–1015.

Selenium with Python (2023). Selenium Python Bindings 2 documentation. https://selenium-python.readthedocs.io/. Accessed: 2023-12-12.

Sievert, C. and Shirley, K. (2014). Ldavis: A method for visualizing and interpreting topics.

SQLAlchemy (2023). SQLAlchemy. https://www.sqlalchemy.org/. Accessed: 2023-12-12.

Takyar, A. (2023). What role do embeddings play in a chatgpt-like model. Accessed: November 30, 2023.

Tucker, C. and Zhang, J. (2010). Growing two-sided networks by advertising the user base: A field experiment. *Marketing Science*, 29(5):805–814.

Appendix

The code for the web-app can be seen in the repository on GitHub created for this research, with URL: https://github.com/KleinJonasUVT/thesis_ma_jonas

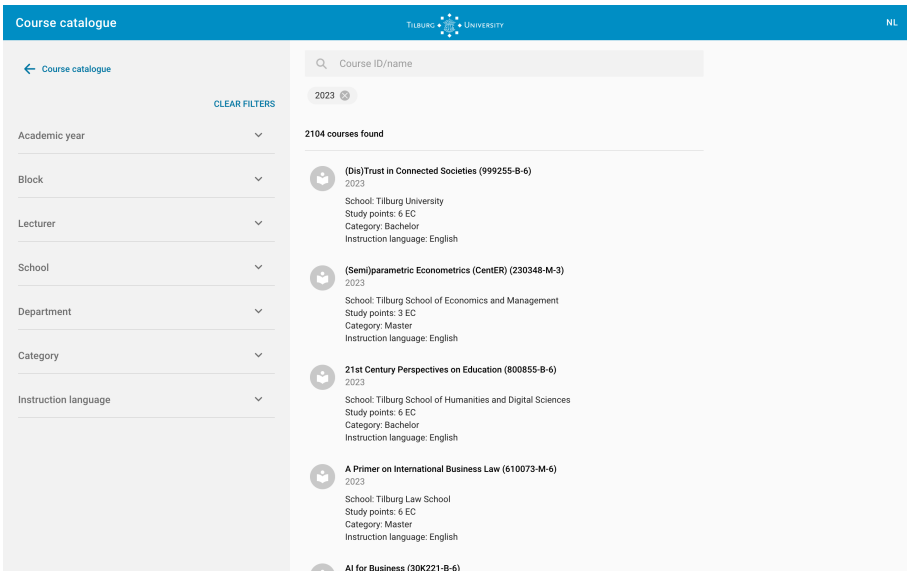

Figure A.1: The current course catalogue of Tilburg University, Osiris

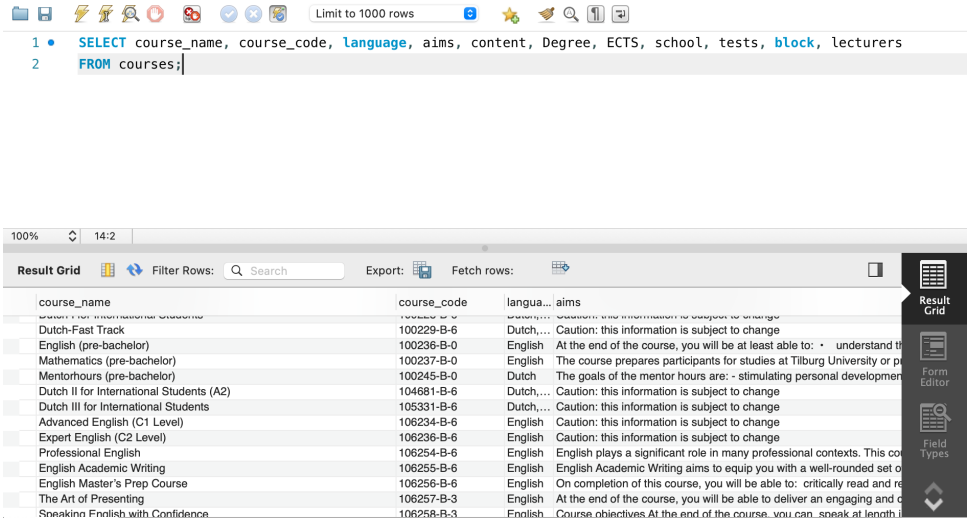

Figure A.2: SQL database query selecting the list of courses scraped from the Tilburg University course catalogue

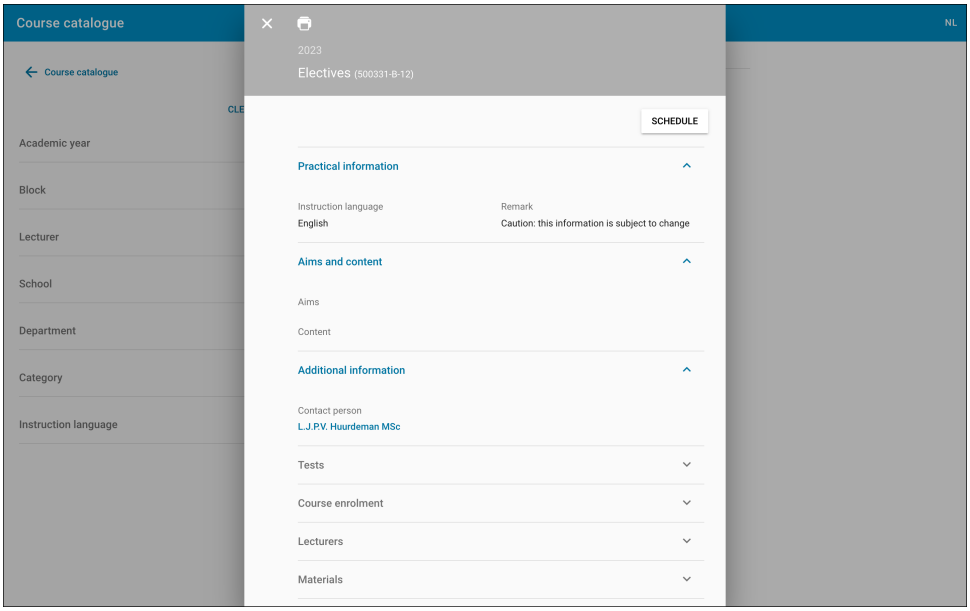

Figure A.3: Example of a useless course for our research, since its content and aim section are empty

Table A.1: Recommendation distribution

Place	Frequency	Cumulative Frequency
Random	153	153
Recommendations on homepage	114	267
Recommendations on course page	176	443
Search	207	**650**

Figure A.4: Introduction page of the website, the first step in Figure 3.

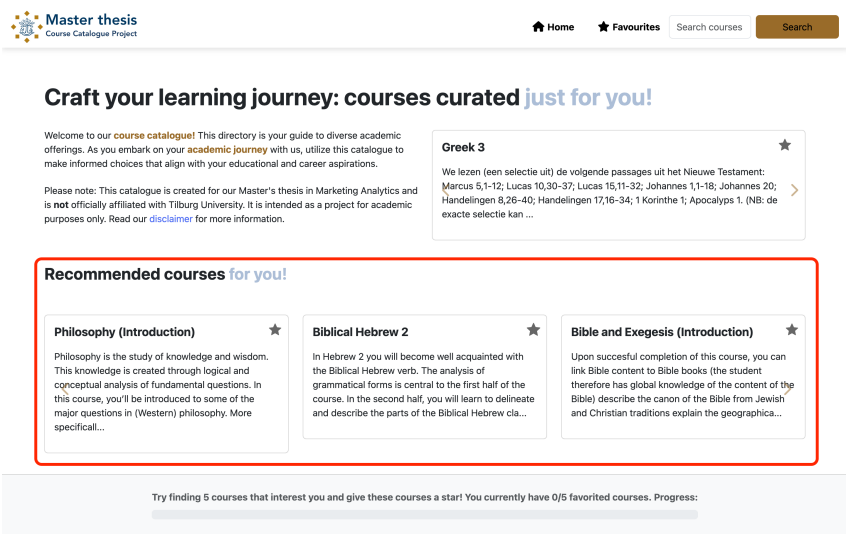

Figure A.5: Recommendations on the homepage based on previous interactions

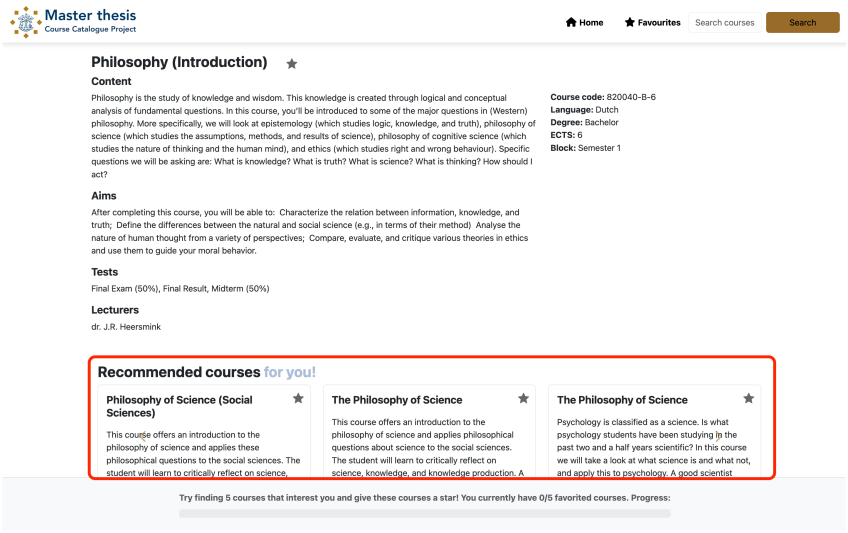

Figure A.6: Recommendations for the course currently on

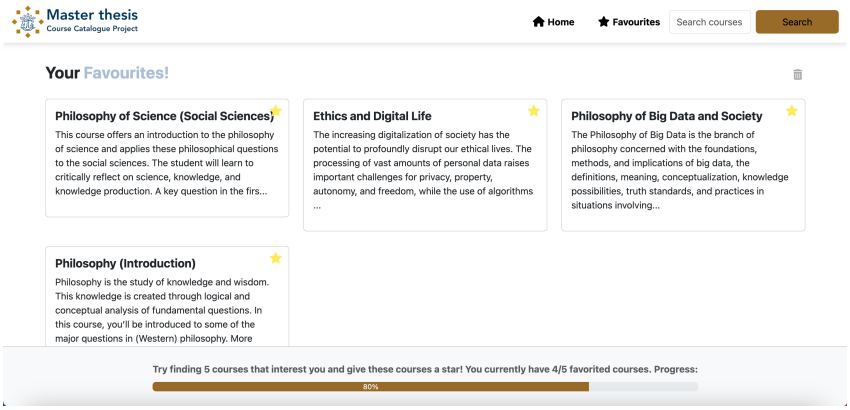

Figure A.7: Favorites page with the progress bar

$$\text{Cosine Similarity}(E_i, E_j) = \frac{\sum_{k=1}^{n} e_{ik} \times e_{jk}}{\sqrt{\sum_{k=1}^{n} e_{ik}^2} \times \sqrt{\sum_{k=1}^{n} e_{jk}^2}} \tag{7}$$

Where

- E_i, E_j represent two entities (like documents, courses, etc.) for which we are calculating the similarity, with each entity represented as an embedding vector.
- e_{ik}, e_{jk} are the elements of the k-th dimension in the embedding vectors E_i and E_j, respectively.
- n represents the total number of dimensions in the embedding vectors. In an embedding space, this is equivalent to the length of the vectors.

$$CI = \bar{x} \pm t_{(0.975, n-1)} \times SEM \tag{8}$$

Where

- $\bar{x}$ = the mean (average) of the Click-Through Rate (CTR)
- $t_{(0.975, n-1)}$ = the t-score that captures the critical value for a 95% confidence interval with $n - 1$ degrees of freedom (df)
- SD_{CTR} = The standard deviation of the Click-Through Rate (CTR)
- n = number of respondents for the algorithm.
- $SEM = \frac{SD_{CTR}}{\sqrt{n}}$ = the standard error of the mean, calculated as the sample standard deviation (SD_{CTR}) divided by the square root of the sample size ($\sqrt{n}$)