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

Jonas Klein, SNR 2040881

December 19, 2023

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 Rate (CTR) and Click Rate (CR) metrics. The TF-IDF algorithm displayed slightly higher mean values, but the differences were not statistically significant, suggesting that neither algorithm performed distinctly better than the other. 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 described above. One way to guide customers is by making recommendations. The information overload can be mitigated by guiding customers to a personalized recommended choice. 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. In the context of content-based recommendation systems, text analysis can be utilized to extract and analyze the descriptive elements of items. Text analysis has the potential to personalise customer-firm interactions (Berger et al., 2020).

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. While there are several applications for relationship extraction in marketing, their use has been restricted so far, probably due to the relative complexity needed in collecting such relationships and analysing semantic similarity (Hartmann and Netzer, 2023).

Text can be represented using word embedding techniques, which create a multi-dimensional distributed representation aiming to capture the meanings of words. The fundamental idea of the most traditional technique to capture relationships with word embeddings, bag of words (BoW), is that the text may be described using an unordered list of frequencies of words (terms) in text (Marcińczuk et al., 2021). This way, relationship extraction can be captured by the mere co-occurrence of words (Hartmann and Netzer, 2023). BoW lacks the ability to differentiate between common and rare words; When the high frequency phrases are not contained in a few specific papers, but are common across the collection, all documents are more likely to be retrieved (Salton and Buckley, 1988). Incorporating IDF into BoW addresses this limitation by assigning a weight to each word that reflects its importance in the corpus, resulting in a TF-IDF representation of documents (Salton and Buckley, 1988; Marcińczuk et al., 2021).

Because the great majority of text analysis tools used in marketing relied on 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).

The advantage of contextual embedding approaches lies in transfer learning; being able to us pre-trained language models for embeddings (Hartmann and Netzer, 2023; Grootendorst, 2022). An example of a contextual embedding model is generative pre-training (GPT) (Asudani et al., 2023). 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). The marketing community can greatly benefit by adopting transfer learning from the NLP and computer science fields, where sharing models, data, and code is common, enabling researchers to build upon existing work rather than starting anew for each project (Hartmann and Netzer, 2023).

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. By leveraging the nuanced understanding enabled by contextual embeddings and integrating it into recommendation systems, we can sift through vast datasets to offer semantically similar and personalized recommendations. The research objectives include evaluating the effectiveness of these recommendation systems through A/B testing and measuring user engagement in terms of click-through-rate (CTR) and conversion rate (CR). CTR is a crucial performance measure of online advertising efficacy, search engine success, and general online behaviour (Lei et al., 2023). both CTR and CR 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 how pre-trained models can be used for transfer learning in 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 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); that is where our research-driven app can be used.

Our research shows that...

This paper is organized as follows. In chapter 2 we will discuss relevant literature. Chapter 3 explains our research framework. chapter 4 will give the empirical findings and Chapter 5 will give a final discussion on their implication for marketing theory and pratices.

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.

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).

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).

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 pure marketing literature that uses text analysis still used methods 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, Huang and Luo (2016) uses text analysis for understanding consumer preferences for complex products, Hennig-Thurau et al. (2014) analyzes approximately four million tweets to see their impact on consumers' adoption of new movies, Lee et al. (2018) examines how different types of advertising content on Facebook affect consumer engagement, Hewett et al. (2016) analyses a comprehensive dataset, including social media posts and news stories and measures their influence on business outcomes and Tirunillai and Tellis (2012) investigates the relationship between user-generated content (UGC) and stock market performance. 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).

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).

To visualize GPT embeddings, see Figure 1. Every word is assigned a vector with numbers, determined in the pre-training stage. This way, the meaning of sentences is

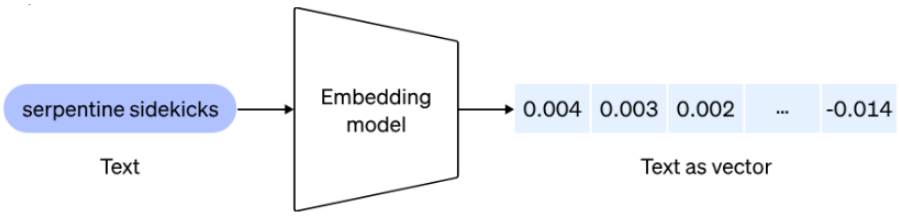

Figure 1: Text to vector conversion: Text is transformed into a numerical vector by the GPT model, allowing the GPT to quantify and interpret language. (OpenAI, 2023)

represented by numbers, which you may use to compare it to other sentences (Takyar, 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).

3 Method and data description

In this section, we describe our research framework, 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, (3) the evaluation of the performance of the search algorithms and data collection and manipulation, and (4) the research method.

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.

In this setting, 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. These courses were selected as the dataset for our recommendation system. The courses were directly scraped 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 quiry 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 an IntegrityError, indicative 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. As part of our input data cleaning process, we paid special attention to the course descriptions, since we are focusing with this research on developing a content-based recommendation system for Tilburg University's courses, utilizing NLP techniques. This necessitated ensuring that the course descriptions were meaningful and feature-rich, aligning with our objective to create recommendations based on textual characteristics. It was essential that these descriptions contained substantive content relevant to the course's nature and content. We aimed to avoid empty descriptions, descriptions with mere practical details or placeholder text lacking in informative features. See Figure A.3 for an example course that does not contain content or aims information and is thus deleted from the dataset.

Besides the removal of the courses without meaningful descriptions we refrained from deleting any courses, since we wanted an environment with a rich set of items for users to choose from, to be able to properly test the two content-based recommendation systems.

After removing all of these courses, we were left with a dataset that consisted of courses with descriptions that provided valuable information about the course's nature, content, and objectives. This resulted in a total of 1627 courses used for our recommendation systems to be built on.

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

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.

The users were guided to the actual app, but the app was placed in a testing environment. Therefore users did not directly land on the homepage, but they were being led to a introduction page of the web-app where instructions of the web-app were shown, the first step in Figure 3. A screenshot can be seen in Figure 4. Users landed on a welcome page with a message explaining its purpose and instructions. Participants were guided on how to navigate the prototype website, interact with various features, and complete specific tasks. Participants were instructed to visit the prototype website, mimicking the behavior of students searching for courses. They were encouraged to

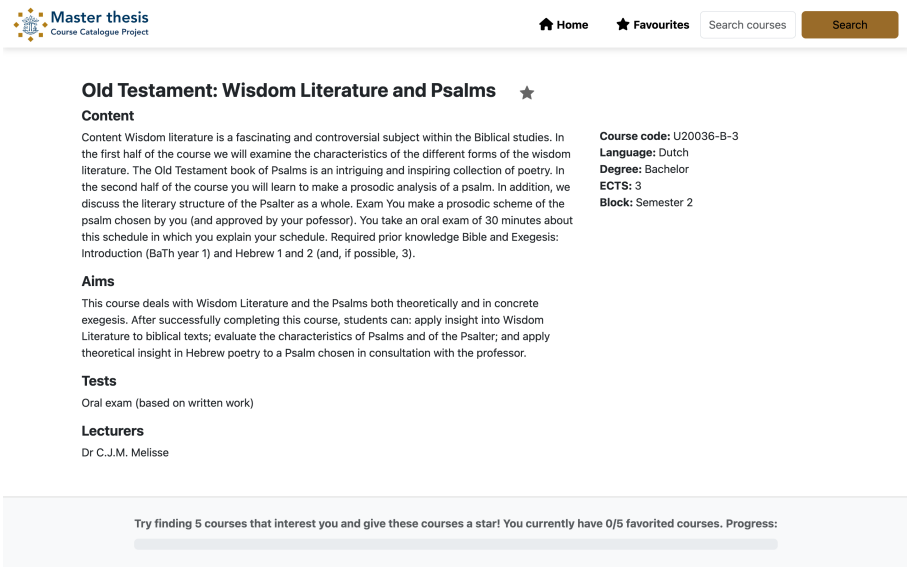

Figure 2: Example course page with course information displayed

browse, click, and favorite courses, paying attention to the recommendations provided by the site. The research required participants to use the search functionality to find and star at least five courses that aligned closely with their academic interests. After clicking on 'Show me the courses!' participants were starting a new user session. Such a session is stored client-side in browser cookies. Cookies are small chunks of data stored on your computer by the web browser (Kennedy, 2023). This way, user-specific information was saved in the browser and therefore not lost when a respondent closed their computer. They were led to the homepage of the app, step 2 in Figure 3. A screenshot can be seen in Figure 5.

Immediately after landing on the homepage users were randomly assigned one of the two recommendation algorithms, either TF-IDF or the GPT model, step 3 in Figure 3. Users were then able to scroll on the website freely and discover courses that most closely aligned with their interests. They were still guided a little bit by the small progress-bar at the bottom of the page, which can be seen in Figure A.4. After a user clicked or starred a course, step 4 in Figure 3, either the TF-IDF algorithm or the GPT algorithm, dependent on which model was assigned, began to calculate the most similar courses to the last clicked or starred course, step 5 in Figure 3.

The TF-IDF model works as follows in the background (see Figure 6).The code loads course data from the database and performs text preprocessing. This includes

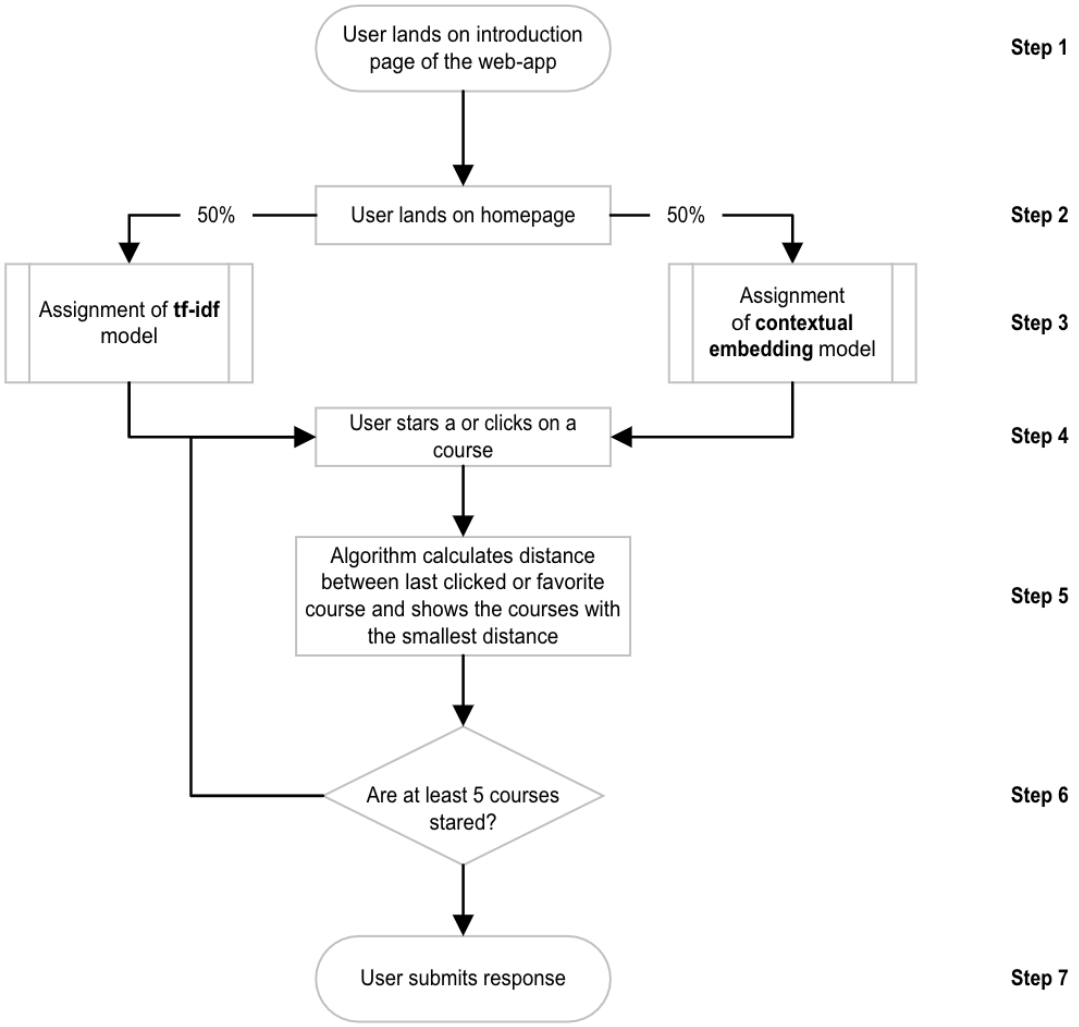

Figure 3: Flow of a user on the website

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

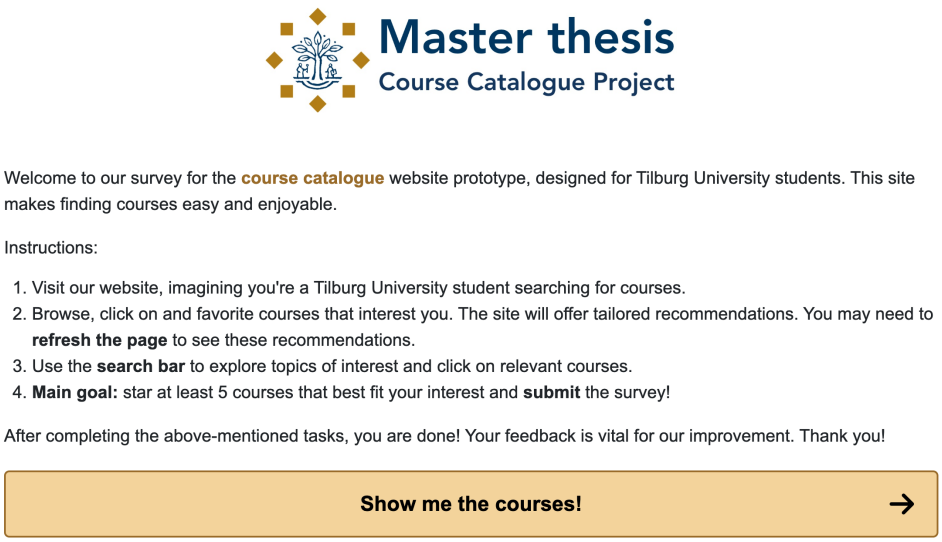

Figure 4: Introduction page of the website, the first step in Figure 3.

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.

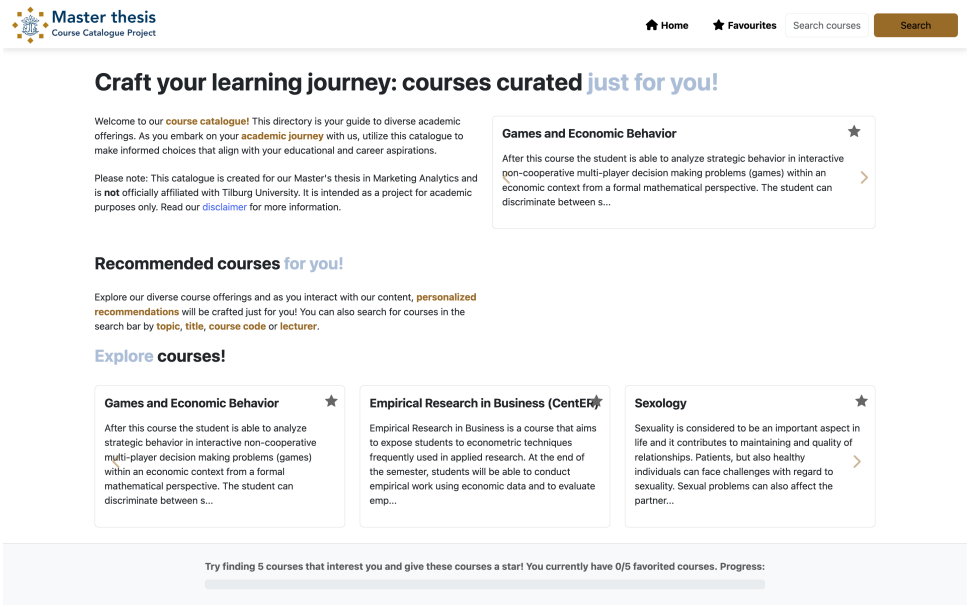

Figure 5: Homepage of the app

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.

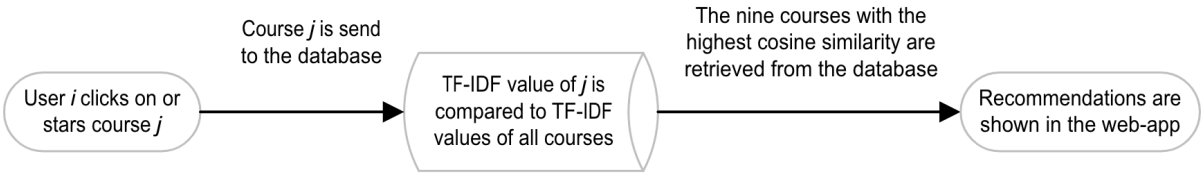

Figure 6: Overview of the TF-IDF algorithm

The GPT model works as follows in the background (see Figure 7). 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. Embeddings for a course description look like this:

```
[   -0.006929283495992422,
    -0.005336422007530928,
    ...
    -4.547132266452536e-05,
```

```
    -0.024047505110502243
]
```

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.

$$\text{Cosine Similarity}(E_i, E_j) = \frac{\sum_{k=1}^{n} e_{ik} \times e_{jk}}{\sqrt{\sum_{k=1}^{n} e_{ik}^2} \times \sqrt{\sum_{k=1}^{n} e_{jk}^2}} \tag{5}$$

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.

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.

The recommendations were shown on either the homepage, as circled in red in Figure 8, or on the course page, as circled in red in Figure 9. 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

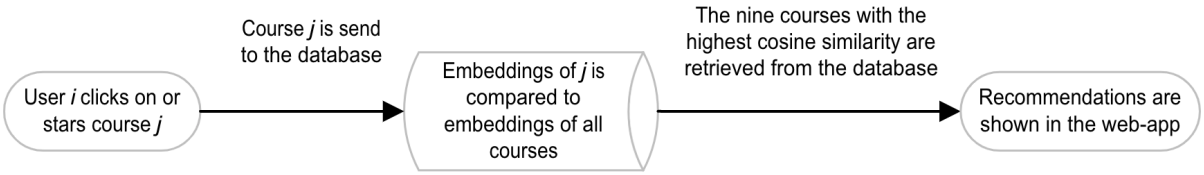

Figure 7: Overview of the contextual embeddings algorithm

essential, since users like to browse linearly through the course catalogue.

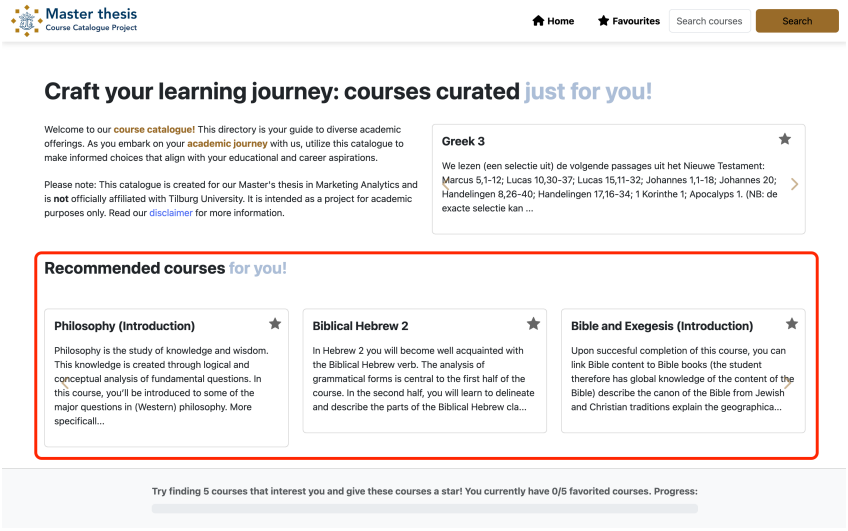

Figure 8: Recommendations on the homepage based on previous interactions

This process was repeated as long as the user kept scrolling through the course catalogue. As long as not at least five courses were stared by the user, the user was not able to submit their response. At the bottom of the page a progress-bar was shown. In Figure A.4 the favorite page is shown with four courses given a star, thus the progess-bar is at 80%. The moment at least five courses are stared, the progress-bar changes to a submit button for users to submit their response, as shown in Figure A.5. This process is shown in step 6 in Figure 3 where the flow branches into different paths, either back to clicking and starring courses, or submitting the response. After user submit their response, step 7 in Figure 3, they are redirected to a page where they are thanked for their contribution and their journey ended.

3.3 Performance evaluation and Data manipulation

We use click-through rate (CTR) and conversion rate (CR) as metrics for performance evaluation of the two recommendation systems. We obtain the CTR and CR through

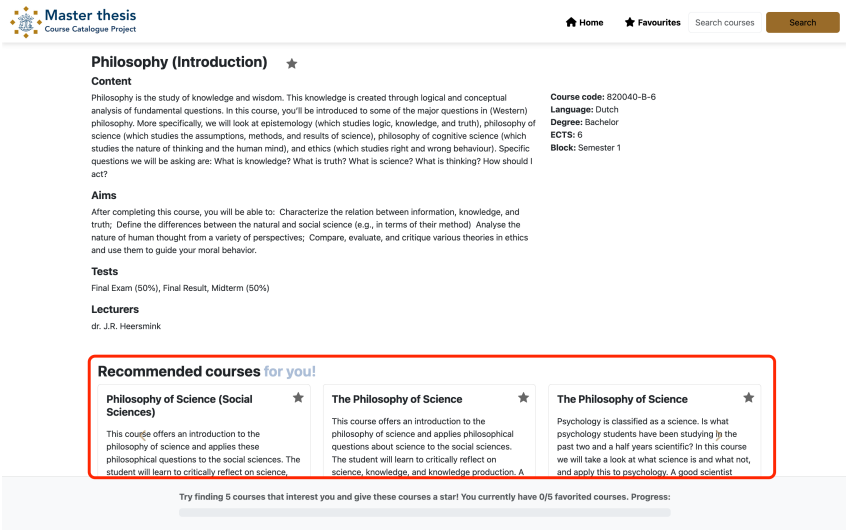

Figure 9: Recommendations for the course currently on

the interaction of users with the website. The behaviour of a respondent during their journey was logged in the SQL database. In Table 3 we show an example of a user's journey as graphically shown in Figure 3, from which CTR and CR can be extracted.

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	recommendations
4	f8d...	2023-12-13 13:55:10	424014-B-6	favorited	NULL	NULL
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	recommendations
7	f8d...	2023-12-13 13:55:27	U11012-B-3	clicked	openai	recommendations
8	f8d...	2023-12-13 13:55:41	U11012-B-3	favorited	NULL	NULL
9	f8d...	2023-12-13 13:55:53	320101-M-6	favorited	openai	recommendations
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	NULL	NULL
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	recommendations
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

Note: This is a hypothetical data representation.

We define a click as a 'clicked' activity in Table 3 where the corresponding algorithm is either 'openai' or 'tfidf' and 'recommendations' for the place column. We also count a click when a recommended course was directly 'favorited' without first clicking on the course, as for example happened in line 9 of Table 3. The user than namely have

inspected the 'thumbnail' and based on this favorited the course. Since we try to assess the effectiveness of the recommendation system as a whole, we measure the ratio of clicks on any of the recommendations in the whole system as outlined in red in Figure 8 and Figure 9 to the number of times the recommendation section was shown to users. The formula for the CTR is:

$$CTR = \frac{\text{Number of Clicks on Recommendations}}{\text{Total times Recommendations were Displayed}} \times 100\% \tag{6}$$

A conversion is the desired outcome, which in our case is a favorite of a course. Therefore, a conversion is defined as a 'favorited' activity in Table 3 where the corresponding algorithm is either 'openai' or 'tfidf' and 'recommendations' for the place column. Since users are able to give courses a star before they click on a course, the denominator also has to be the total times recommendations were displayed. The formula for the CR then is:

$$CR = \frac{\text{Number of Favorites of Recommendations}}{\text{Total times Recommendations were Displayed}} \times 100\% \tag{7}$$

For the example journey for the user in Table 3 we can calculate the CTR and CR for better understanding. The user clicks on a recommendation in line 3, 6, 7 and 13. The user favorited a course immediately from reading the thumbnail without first clicking in line 9, which we will also count as a click. We thus have 5 clicks in total. The user get's to see a new recommendation section after line 2, 3, 5, 6, 7, 9, 10, 12 and 13, so in total 9 times.

$$CTR = \frac{5}{9} \times 100\% = 55.56 \tag{8}$$

The user favorited recommended courses in line 4, 8 and 9. We already determined the total times recommendations were shown, 9 times.

$$CR = \frac{3}{9} \times 100\% = 33.33 \tag{9}$$

In the way we have calculated the CTR and CR for the example above, we can create a table where we store this data for every user of the website. We then get Table 4. This table contains data for every user on the algorithm they were assigned, the number of clicks on the recommendations, the number of conversions on the recommendations,

the total number of recommendations, the Click-Through Rate (CTR), the Conversion Rate (CR) and the fact if the user is an early dropout or a valid respondent. A number of users landed on the homepage and got assigned an algorithm, but after that never clicked on any course. They thus never saw any of the recommendations and we can not say anything about their behavior.

Table 4: User Interaction Data

	ID	Algorithm	Clicks	Conversion	Num Recs	CTR	CR	Dropout
1	013...	TF-IDF	2	2	5	0.4	0.4	Valid respondent
2	01b...	GPT	3	3	6	0.5	0.5	Valid respondent
3	040...	TF-IDF	0	0	5	0	0	Valid respondent
4	04d...	TF-IDF	0	0	0	NaN	NaN	Dropout
5	08d...	GPT	5	3	9	0.556	0.333	Valid respondent
6	0c7...	GPT	4	4	5	0.8	0.8	Valid respondent
7	11e..	TF-IDF	10	4	12	0.833	0.333	Valid respondent
8	1j7...	TF-IDF	2	2	7	0.286	0.286	Valid respondent
9	145...	GPT	4	3	6	0.667	0.5	Valid respondent
...	...	...	...	...	...	...	...	...

Note: This is a hypothetical data representation.

From Table 4 we can calculate the respondents per algorithm, distinguished between valid and non-valid respondents, the dropouts (see Figure 10). In total we had 168 respondents from the start of our research. From this group, 85 respondents got the GPT algorithm and 83 got the TF-IDF algorithm. From the GPT group, 32 respondents never clicked a course, leading to 53 valid respondents for the GPT algorithm. From the TF-IDF group, 24 respondents never clicked a course, leading to 59 valid respondents for the TF-IDF algorithm.

3.4 Research method

In our research we will conduct an A/B-test on the CTR and CR for both recommendation systems, to measure the effectiveness of both systems. We will conduct a between-subjects experiment, where the respondents are randomly assigned to one of the two recommendation systems.

Traditionally, the emphasis has been on testing forecast accuracy using offline datasets. One prominent example is the extensive usage of the Each-Movie database, a benchmark dataset commonly used in recommendation research (Chung and Rao, 2012). However, there is a growing realisation that a recommendation system should

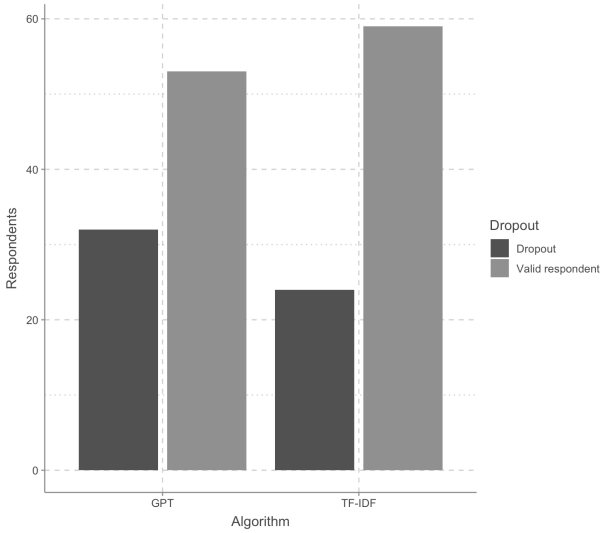

Figure 10: Number of participants per algorithm, adjusted for dropouts

not only make accurate predictions but also provide a pleasant user experience. Proper evaluation of a recommender system's user experience necessitates conducting a user experiment, either in the form of a lab experiment or a randomised field trial, which includes conventional A/B tests (Knijnenburg and Willemsen, 2015); this is where our research-driven app was used.

The data collection for our A/B-test was conducted over a specific timeframe, from November 21, 2023, to December 19, 2023. We collected participants via LinkedIn, by sharing the link to the web-app, this way trying to get as many respondents as possible. Data about users on the web-app was anonymous and could not be traced back.

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.

In our analysis, we will conduct a one-tailed independent samples t-test, testing if the GPT algorithm has a significantly higher mean CTR and CR than the TF-IDF algorithm. The TF-IDF is regarded the baseline algorithm, against which the new system is tested. We can do a power analysis to find the sample size we need for this

test with the help of G*Power (Faul et al., 2009). We will conduct a one-tailed t-test on the difference between two independent means. We will use an effect size d of 0.5, a significance level of 0.05 and a power of 0.8. The power analysis from G*Power returns a required sample size of 102, thus 51 respondents per group (see Figure A.6). With 53 respondents for the GPT algorithm and 59 for the TF-IDF algorithm we thus have a big enough sample size.

4 Results

We conducted two one-tailed independent samples t-tests for both the CTR and CR between the two algorithms. We will first show the comparison through mean calculation and their 95% confidence interval. Then we will show the two t-tests and show their results.

4.1 Means confidence interval

We give a statistical comparison of Mean Click-Through Rate (CTR) and Mean Click Rate (CR) among the two different algorithms: GPT and TF-IDF.

In Table 5, the algorithms are compared based on their mean CTR. The table includes the mean CTR ($Mean\ CTR$), standard deviation of the CTR (SD_{CTR}), the sample size (n), the standard error of the mean (SEM), and confidence intervals ($CI\ Upper$ and $CI\ Lower$). Equation 10 shows how $CI\ Upper$ and $CI\ Lower$ are calculated, leading to a confidence interval around the means of 95%.

GPT shows a mean CTR of 0.338 (**33.8%**) with a standard deviation of 0.297, based on a sample size of 53. The SEM is 0.0408, and the 95% confidence interval ranges from 0.257 to 0.420. TF-IDF has a mean CTR of 0.407 (**40.7%**) with a standard deviation of 0.276, based on a sample size of 59. The SEM is 0.0359, and the 95% confidence interval ranges from 0.335 to 0.479.

Table 5: Algorithm Comparison: Mean CTR

Algorithm	Mean CTR	SD_{CTR}	n	SEM	CI Upper	CI Lower
GPT	0.338	0.297	53	0.0408	0.420	0.257
TF-IDF	0.407	0.276	59	0.0359	0.479	0.335

In Table 6, a similar comparison is made for the Mean Click Rate (CR). GPT has a mean CR of 0.237 (**23.7%**) with a standard deviation of 0.265, based on a sample size of 53. The SEM is 0.0364, and the 95% confidence interval ranges from 0.164 to 0.310. TF-IDF shows a mean CR of 0.291 (**29.1%**) with a standard deviation of 0.270, based on a sample size of 59. The SEM is 0.0351, and the 95% confidence interval ranges from 0.220 to 0.361.

Figures 11 and 12 are visual representations of the confidence intervals for the mean CTR and CR of the algorithms, respectively. The points indicate the mean values, and

Table 6: Algorithm Comparison: Mean CR

Algorithm	Mean CR	SD_{CR}	n	SEM	CI Upper	CI Lower
GPT	0.237	0.265	53	0.0364	0.310	0.164
TF-IDF	0.291	0.270	59	0.0351	0.361	0.220

the error bars represent the 95% confidence intervals.

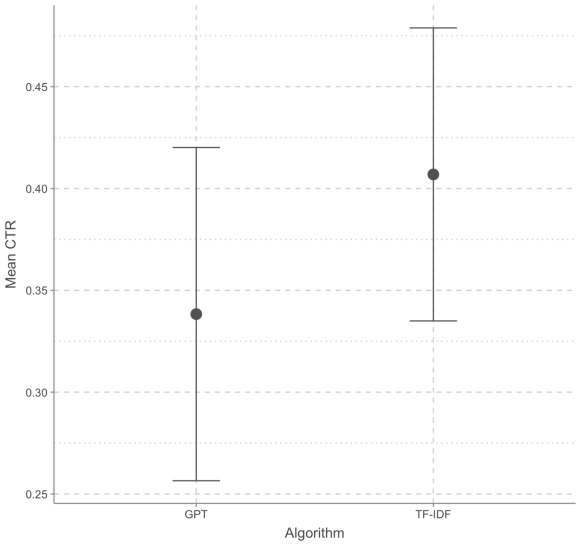

Figure 11: Comparison of Mean CTR with Confidence Intervals

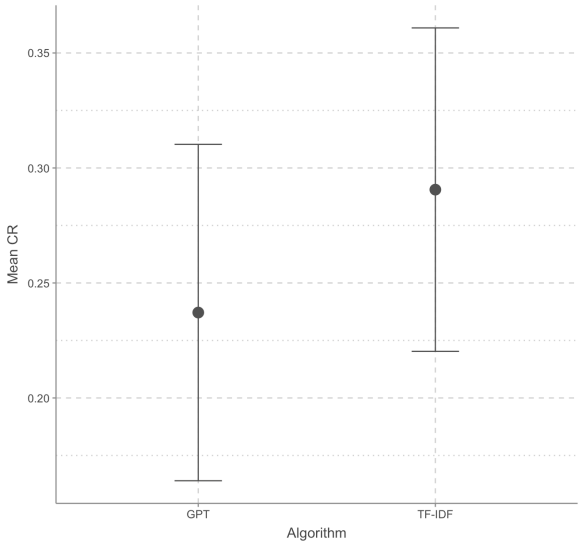

Figure 12: Comparison of Mean CR with Confidence Intervals

The comparison of Mean *CTR* and Mean *CR* between GPT and TF-IDF algorithms indicates that TF-IDF has a slightly higher mean *CTR* as compared to GPT, with mean values of 0.407 and 0.338, respectively. However, both algorithms demonstrate overlapping confidence intervals, which suggests that the difference is not statistically significant at the 95% confidence level.

For the Mean CR, GPT also has a lower mean of 0.237 compared to TF-IDF's mean of 0.291, but again, the confidence intervals overlap.

The visual representations in Figures 11 and 12 further support these findings, with the error bars overlapping for both algorithms. This indicates that there is no clear superiority of one algorithm over the other in terms of the CTR and CR metrics.

4.2 T-tests

Since the calculation and comparison of the means of CTR and CR pointed out that TF-IDF actually had a higher mean CTR and CR, there is no point in conducting a

one-tailed independent samples t-test, testing if the GPT algorithm has a significantly higher mean CTR and CR than the TF-IDF algorithm. Actually, we have to reverse the test and see if the CTR and CR for the TF-IDF is significantly higher than the CTR and CR for the GPT algorithm.

Table 7 shows the results of the one-tailed independent samples t-test comparing the CTR of the TF-IDF and GPT system. The t-value is 1.2622, which measures the difference between the two sample means relative to the variation in the data. A higher t-value indicates a greater difference between the groups. The critical t-value for our reseach van be calculated with the help of G*Power (Faul et al., 2009), which shows the critical t-value is 1.6602343 (see Figure A.6), thus suggesting there is not a great difference between the two groups.

The p-value is 0.1048. This value indicates the probability of observing the test results, or more extreme results, if the null hypothesis is true. In this context, the null hypothesis states that there is no difference in means between the two groups. A p-value of 0.1048 suggests that there is a 10.48% chance of finding a difference as large as (or larger than) the one observed if the true difference in means is zero or less. We therefore do not have sufficient evidence to reject the null hypothesis at the 5% significance level.

The lower bound of the confidence interval includes negative values, which also suggests that the difference in means might not be significantly greater than zero. This is shown in Figure 13, where you can see that the 95% confidence interval crosses the zero (dashed) line, it including negative values.

Table 7: Welch Two Sample t-test Results for CTR

Statistic	**Value**
t-value	1.2622
Degrees of Freedom (df)	106.54
p-value	0.1048
95% Confidence Interval	
Lower Bound	-0.02157628
Sample Estimates	
Mean of TF-IDF	0.4069000
Mean of GPT	0.3383104

The results for the one-tailed independent samples t-test comparing the CR of TF-IDF and GPT shown in Table 8 lead to the same conclusion as for the CTR metric. The

t-test comparing the CR of TF-IDF and GPT systems yields a t-value of 1.0557, even though the critical t-value from the G*Power analysis is 1.6602343. The p-value of 0.1048 also indicates no significant difference between the two systems at the 5% significance level. Moreover, the confidence interval's lower bound including negative values implies that the mean difference is not significantly greater than zero, as shown in Figure 14, where the 95% confidence interval crosses the zero (dashed) line, it including negative values.

Table 8: Welch Two Sample t-test Results for CR

Statistic	**Value**
t-value	1.0557
Degrees of Freedom (df)	109.09
p-value	0.1467
95% Confidence Interval	
Lower Bound	-0.03053312
Sample Estimates	
Mean of TF-IFD	0.2905736
Mean of GPT	0.2371324

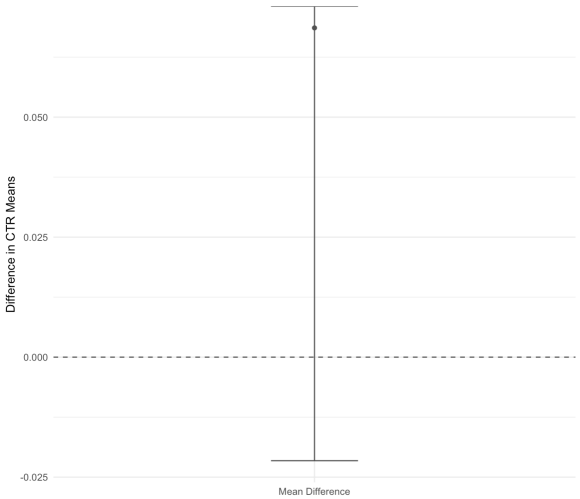

Figure 13: Confidence Interval for the Difference in Means for CTR

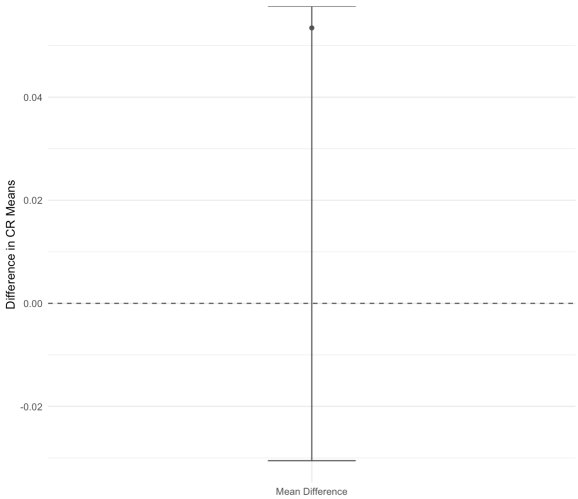

Figure 14: Confidence Interval for the Difference in Means for CR

5 Discussion and 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. In both the Click-Through Rate (CTR) and Click Rate (CR) comparisons, the TF-IDF algorithm displayed slightly higher mean values than GPT. However, the overlapping confidence intervals and non-significant t-test results suggest that the difference is not statistically significant. 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 CTR and CR in recommendation systems.

Our study has a few limitations. First, 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.

Secondly, we may not have taken enough steps in ensuring effective randomization. There were 32 dropouts for the GPT algorithm and only 24 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.

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.

Faul, F., Erdfelder, E., Buchner, A., and Lang, A.-G. (2009). Statistical power analyses using g*power 3.1: Tests for correlation and regression analyses. *Behavior Research Methods*, 41(4):1149–1160.

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.

Huang, D. and Luo, L. (2016). Consumer preference elicitation of complex products using fuzzy support vector machine active learning. *Marketing Science*, 35(3):445–464.

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.

Marcińczuk, M., Gniewkowski, M., Walkowiak, T., and Bedkowski, M. (2021). Text document clustering: Wordnet vs. tf-idf vs. word embeddings.

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.

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.

Tirunillai, S. and Tellis, G. J. (2012). Does chatter really matter? dynamics of user-generated content and stock performance. *Marketing Science*, 31(2):198–215.

Appendix

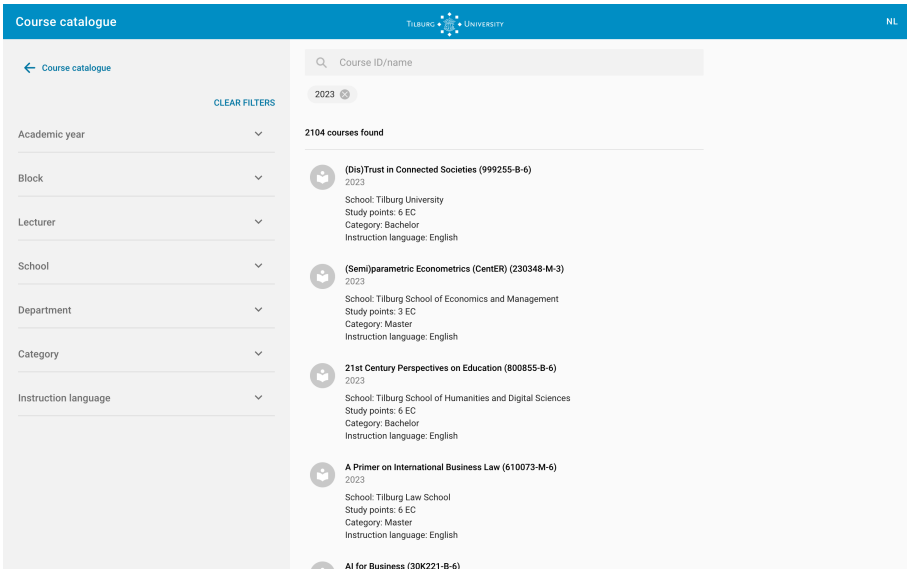

Figure A.1: The current course catalogue of Tilburg University, Osiris

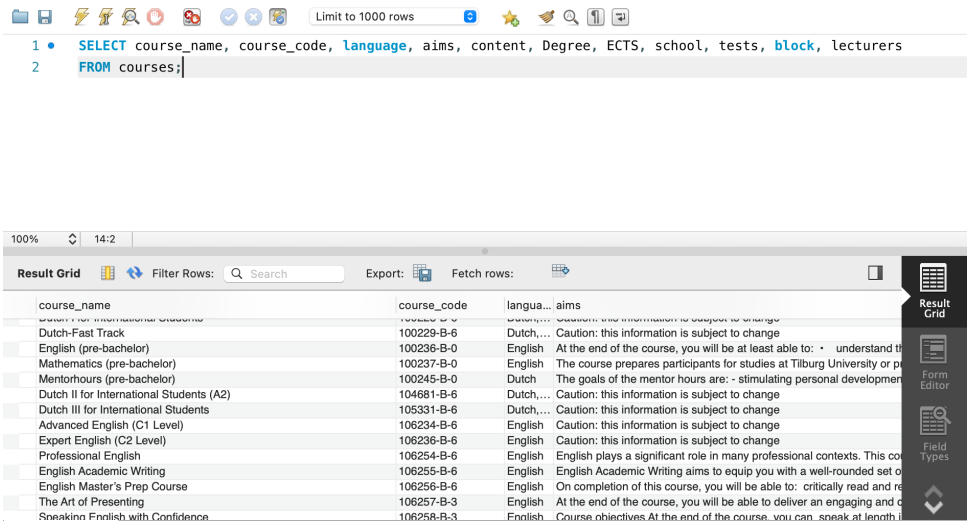

Figure A.2: SQL database query selecting the list of courses scraped from the Tilburg University course catalogue

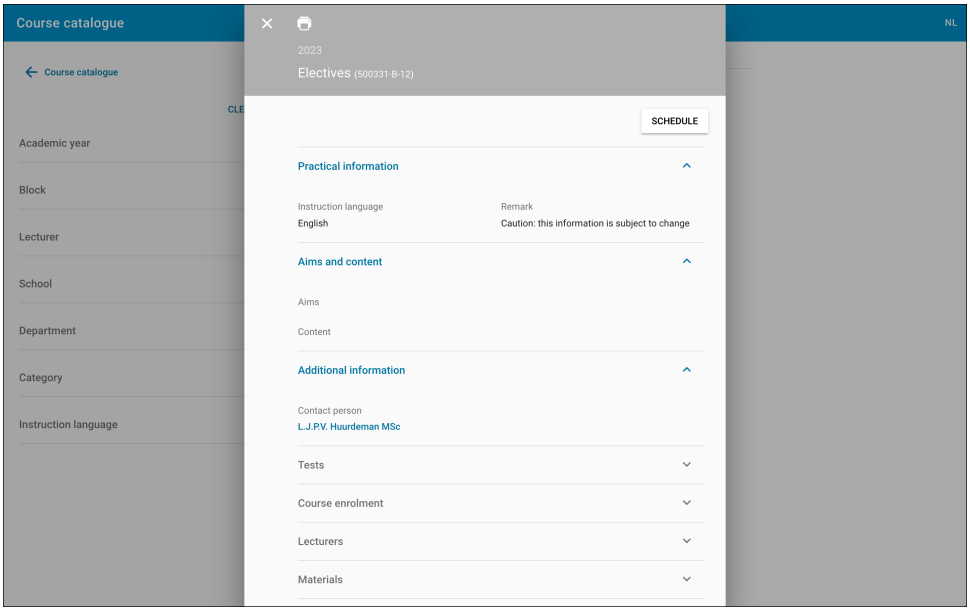

Figure A.3: Example of a useless course for our research, since its content and aim section are empty

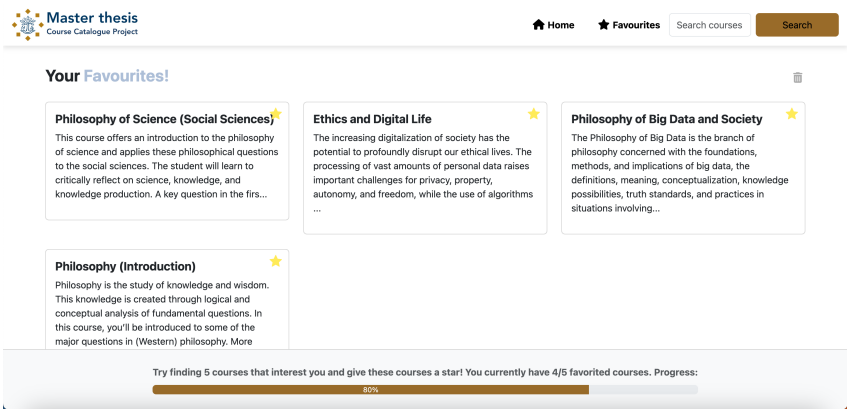

Figure A.4: Favorites page with four out of five (minimum) courses stared

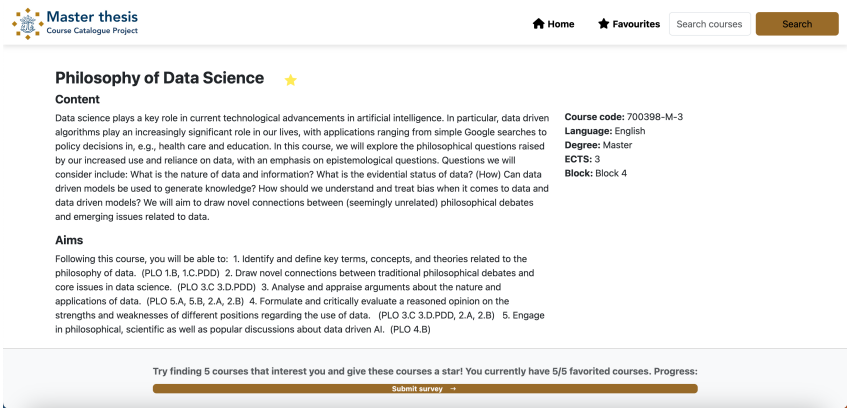

Figure A.5: A submit button appears after at least five courses are stared

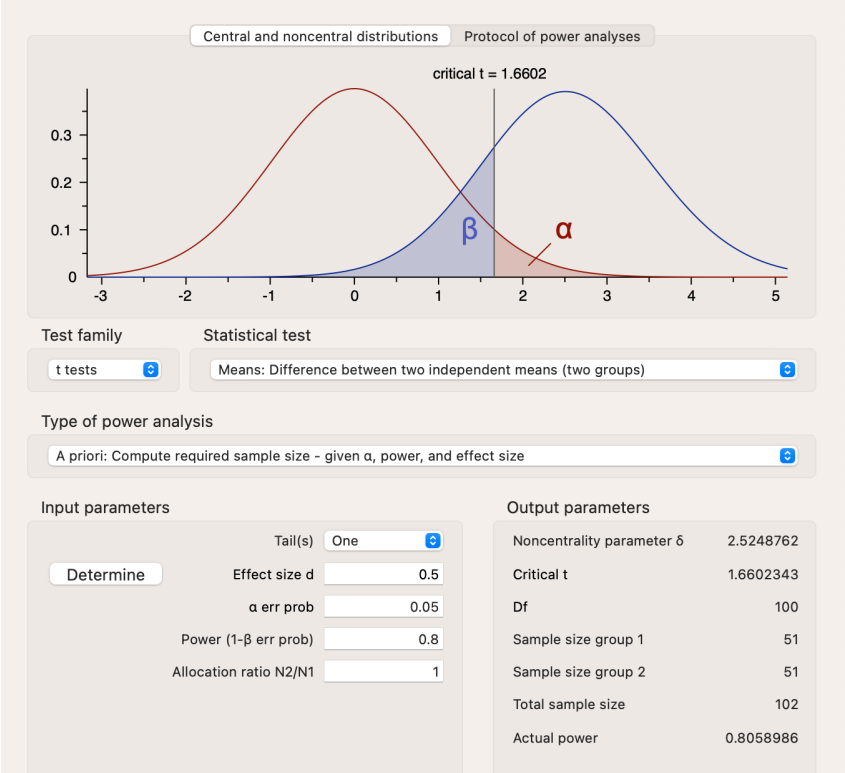

Figure A.6: Power analysis with G*Power (Faul et al., 2009)

$$CI = \bar{x} \pm t_{(0.975,n-1)} \times SEM \tag{10}$$

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 ($\sqrt{n}$)