Decision Support System for SAP Products' Alternatives using Information Retrieval

Saloni Verma
Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
saloni.verma@ovgu.de

Gaurav Singhal
Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
gaurav.singhal@ovgu.de

Matthias Volk
Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
matthias.volk@ovgu.de

Klaus Turowski
Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
klaus.turowski@ovgu.de

***Abstract*—This document is a comprehensive paper for the activities, research and results of the project Decision Support System for finding market alternatives of SAP products. In this project, we propose a solution to finding alternatives of SAP products that are available in the market. It is always an obstacle to decide which product a company should implement in order to make their process easier and faster. We are proposing a tool developed using Python based on information retrieval which bridges the gap between the Business, user and various product options that are available. We also outline other approaches that we tried to create an ontology like structure for grouping the products. We also highlight the future scope and possible extensions to this work.**

***Index Terms*—machine learning, decision support system, ERP, HRM, Analytics, SAP, information retrieval, ontologies**

I. Introduction

In today's contemporary scenario the use of software for data management is trivial for all businesses and institutions. And having the appropriate software for all the requirements becomes increasingly difficult seeing the range of products available in the market before. Each software has its own features, hardware specifications, implementation ease and maintenance effort. And different users seek to use the software in different ways and finding the best option available in the market, according to your budget becomes the most Important driver for software selection.

SAP is the industry giant in serving large and small businesses for all their user and data management tools. The German software company is known for their software that can be used to track customer and business interactions. Its Enterprise Resource Planning (ERP) and data management programs are also widely used in commercial and non-commercial places. It is quite aptly named Systems, Applications and Products (SAP).

As a motivation, we can assume that some small scale companies would also like to opt for cheaper alternatives that don't cost a fortune to develop and maintain. Since they are just starting, they cannot afford to purchase or license the use of SAP products, or cannot create an entire ecosystem where each SAP product can be deployed in the company, it becomes imperative to find out which products exist in the open market that are worthy and appropriate alternatives for such products.

II. Background

SAP produces and sells software that is apt, easy to use and that can be moulded to the size and requirements of the company. But for some companies it would be preferable to opt for other tools available in the market. The reasons could include the price factor, deployment process, customisability etc. So this is a real need that the alternatives of SAP products are known and classified in a database where referencing and usage could be extracted easily. Since such a simple query and response system does not exist, we would focus our attention on it. Also since there is no universal database for SAP that classifies and arranges them in clusters/classes for generating knowledge that will form the basis of our project repository.

III. Problem Statement

In this project, we aim to create a tool where the user can find out alternatives to the SAP products available in the market today. The goal is to find comparative products at a more affordable price or just to provide the options to the users to be able to choose competitive products in the market.

And we propose to create a system with a repository of SAP products that are classified according to their key features and uses. And then we create a repository of SAP-like products in the market today. We match the features like price, user interface, etc and display them to the user when the query is executed.

IV. Motivation and Benefits of this Research

The motivation behind working on this project was manifold. We wanted to apply our knowledge of machine learning,

	Unnamed	name	description	category	license	pricing	users
0	0	Microsoft Project & PPM	['Microsoft Project & Portfolio Ma	Project Management Software	Proprietary	Subscription	Small (1001 employees)
1	1	Celoxis	['Celoxis advanced features addre	Project Portfolio Management (PPM)	Proprietary	Subscription	Small (1001 employees)
2	2	SAP Portfolio and Project Man	['SAP Portfolio and Project Manag	Project Portfolio Management (PPM)	Proprietary	Subscription	Small (1001 employees)
3	3	HP PPM	["HP PPM software provides the r	Project Portfolio Management (PPM)	Proprietary	Subscription	Small (1001 employees)
4	4	Dassault SystÃ¨mes 3DEXPERIE	['Dassault SystÃ¨mes 3DEXPERIEN	Project Portfolio Management (PPM)	Proprietary	Subscription	Small (1001 employees)
5	5	Portfolio for Jira	['Portfolio for Jira is an agile portf	Project Portfolio Management (PPM)	Proprietary	Subscription	Small (1001 employees)
6	6	VersionOne	['VersionOne is an enterprise soft	Product Management Software	Proprietary	Subscription	Small (1001 employees)
7	7	Planisware	['Planisware is a software solutior	Project Portfolio Management (PPM)	Proprietary	Subscription	Small (1001 employees)
8	8	Clarizen	['Clarizen is an industry-leading cc	Task Management Software	Proprietary	Subscription	Small (1001 employees)
9	9	Sopheon	['Sopheon Accolade project portfc	Project Portfolio Management (PPM)	Proprietary	Subscription	Small (1001 employees)
10	10	QuickBooks Desktop Enterpris	['QuickBooks Desktop Enterprise :	Inventory Management Software	Proprietary	Subscription	Small (1001 employees)

Fig. 1. Snapshot of the data file with the list of SAP and alternate products gathered from PredictiveAnalytics.com [7]

data science to a real world application in the domain of how very large businesses are run. Creating a product that could bring multi-purpose software and products under one umbrella of an ontology was the vision of this project. The other reasons why this research could be beneficial are outlined below.

- **Competitive Price:** The SAP products are proprietary and expensive in effect [15]. Sometimes the deployment and maintenance costs over the years exceeds the budgets specified by any company for such expenses. In other cases, some companies cannot associate with a direct competitor or associate with some products because of company policies could benefit from using the substitutes available in the market.
- **Benefit for Small Companies:** Certain companies that are starting out might not be able to licence or buy the SAP products. Or they might need customisation according to size and company's needs could do better with the other possible software that our tool could recommend them.
- **Hassle-free Deployment, Support and Maintenance:** The process of deployment of a particular SAP product requires the hardware and infrastructure that not all companies or institutes would be able to arrange. The maintenance costs and man-power is also sometimes a constraint that the unconventional products may be able to provide.
- **Licensed as Open Source:** It is very important for some customers to opt for open source software that can be moulded according to the specific demands of the company and the job at hand. The freedom, and security and customisation ability that an open source software provides can seldom be compared with the features of a proprietary product.
- **Match for Suitable Requirements:** SAP has had products in the market for a very long time that have received updates and mods, but can still be tailored according to different requirements of the user/company.
- **Unfavourable UI and long training times:** A lot of organisations have reported that the SAP User interface and navigation is not ideal [16]. It is a flexible but complex solution. Users reported that there is some confusion with the SAP products and a lack of simplicity. Moreover, this results in extra time and cost for training and adapting the personnel to new software. The motivation to find alternative products will be late st user friendly interface and report enhanced user experience.

Further benefits can be to extend this project to other products that companies offer and a self learning model which can classify products on the basis of the inherent features of the product. Such autonomous decision support systems can be trusted by the user online who can make an informed decision for the choice of products based on different filters.

To capture semantics in a computable way is desirable for many domains like as information retrieval, document clustering/classification etc. The common first step is embedding words or documents in a vector space. All types of embedding techniques have their own unique characteristics which means it is difficult to narrow down the best one for our application. This paper [17] argues that it is imperative to take into account the requirements from the applications to choose the most suitable embedding method. Moreover, to achieve better retrieval performance,a combination of bibliometric measures and semantic embedding can be tried to improve ranking. An advantageous property of embeddings obtained by all methods will be the computable similarity: similarity between two words (or documents) correlates with the cosine of angle between their vectors. For example, one can use Word2Vec to provide hypernymy(hypo) recommendations i.e. for words that belong to same lexical class (parts of speech. On the other hand, if the application gives an overview of a word, Ariadne [17] could be a better choice. If efficiency is key, a basic weighted average performs well in terms of document retrieval.

Ontologies are a pictorial representations of gathered knowledge but only for a particular domain [4]. An important characteristic is to enhance the information exchange sytactic and semantic information. They are a good way to enhance existing technologies from ML and information retrieval (extraction) [18]. The knowledge contained in ontologies is used to annotate web pages, generalize/specialise concepts, can drive intelligent search engine by using relations between concepts [20]. They are are one of the key elements of present day Semantic Web. Their usage in information sharing and management requires using effective and very efficient approaches to ontology development and maintenance [19] [21]. Moreover, by providing a shared schema, they facilitate

```
query_string = "sap business one"

# Check if Query has exact match with database
name_query_vector = name_vectorizer.transform([preprocess(query_string, False)])
q_sim_index, q_sim_score = get_query_related_tool(name_features, name_query_vector)
# If cosine similarity score is high then change query to that document
if q_sim_score > 0.8:
    query = df.loc[q_sim_index]['description']
else:
    query = query_string

print("Query match with tool: '{}', with score: {}".format(df.loc[q_sim_index]['name'], q_sim_score))

embed_q_vec, tf_q_vec = query_synthesizer(query, tfidf_vectorizer, embed_vectorizer)

# Finding Cosine similarity between Documents and Query
cos_score = get_cosine_nearest_documents(embed_q_vec, tf_q_vec, tfidf_features, embed_features, TOTAL_DOCS)
result = df.join(cos_score)
result = result.sort_values(by=['embedd_match', 'string_match'], ascending=False)

# Printing Similar documents. Top 20
total_results = 20
# Filter out SAP products
result = result[result.name.str.contains("SAP") == False]
result = result[result.name != df.loc[q_sim_index]['name']]
result['total_score'] = result['embedd_match'] + result['string_match']
result[['name', 'embedd_match', 'string_match', 'total_score', 'pricing', 'users','license']][:total_results]\
    .sort_values(by=['total_score'], ascending=[False])
```

```
Query match with tool: 'SAP Business One', with score: 1.0000000000000002
```

Fig. 2. Code Snippet for Executing the Query Term for SAP product or business requirement

query answering and reasoning over different and domain specific data sources Although some progress has been made over the last few years, it is not still possible to fully automate the ontology development process. In some approaches they try to build the ontology only [22] [23], or concepts with their hierarchy [18] [22]. It is highly important to personalise the ontology creation process to the use case [24] [25]. This problem has yet to be fully resolved.

It is postulated activities supporting automatic ontology construction and Ontology Learning (OL) may reduce some research problems [26]. Automated generation will be fundamentally different than manual construction by a designer. Since the aim of ontology automatic construction is to provide a scope of creating ontologies from different input sources and limit the human intervention in this process [2] [20] [27]. Some research refers to the automatic generation of ontologies. Examples include ripple down rules, logic programming, word sense clustering, and information extraction and more. Other researches concentrated on the use of inductive logic programming for learning logical theories which also aided in reducing human-introduced biases and inconsistencies [22].

V. Approach Chosen

We decided to approach this problem using Natural Language Processing techniques of machine learning and information retrieval. We gathered data about SAP products from their official website. We learnt that they classify their products under different categories such as CRP and finance, CRM and Customer Experience, Digital and Supply chain, Human Resource Management and Employee Motivation, and many more. In the initial exploratory data analysis phase, we had collected the basic data to start modelling the SAP products and their alternatives. The early models had low accuracy but showed promising results. The pre-processed data through Information retrieval methods was used in the basic machine learning models resulting in different observations. Some challenges with the data and its adaptability to the model means that data collection and updating was a constant process as no one model was a perfect fit for the type and range of data we gathered.

We then proceeded to create a comprehensive repository of SAP products and respective alternatives, classify them on the basis of common features using Machine Learning techniques. Meanwhile, for querying, we would use Information Retrieval methodologies to parse the user query, break it to keywords and match the keywords in our system to give the best match result. The process we followed can be listed as the 4 following steps:

- *Data Gathering:* Imperative to find all the products of SAP that exist in the market for commercial and non-commercial use.
- *Feature Representation:* After the software is identified, identify the features of that software that we would be able to build the classifier on.
- *Information Retrieval:* Building on the basics of IR, find the requested features from our repository and query according to best match.
- *Query Handling:* Using NLP, and basic information retrieval techniques to display the ranked search results of the query given.

A. Data Gathering and Understanding

The initial and most crucial challenge was the varied forms of data that existed on the official websites and comparison

sites which have different databases and criteria for comparison. We initially learned more about the SAP products from the official documentation available [9]. Then we saw as to how they contrast using tools like Capterra [8] which has a huge database of softwares and is simple to use. We could differentiate on the basis of pricing, number of users, reviews and product features. But we could not have manually collected the data from this site for hundreds of SAP and alternate products. Due to the immense volume of data available, we opted to ethically scrap the website for creating our data file. After trying our engine and preemptive code on multiple types of data, we decided to opt for the data from the website PredictiveAnalytics.com [7]. The last crucial step was to identify the best products i.e the most feature-rich products that we could utilise for our comparison that would give fruitful results. We finalised some products based on the reviews from TrustRadius, just to cement our understanding of the different demands in the market. The final result after running some scripts for data collection was an Excel file where we gathered the name of the product, its description, Category, License (Open Source, Proprietary Free, Proprietary), Pricing and the number of users. Fig 1 shows a small subset of the file.

Our file consists of 144 such products including SAP products and alternate products. The representation and retrieval engine is run on this data file for now. The results are also evaluated based on this set of data.

B. Developing the Appropriate Representation Vectors

Preprocessing of raw data plays a vital role in classification task as it directly affects the success rate of the model. In terms of preparing the raw data,

The main idea is to first extract features from the dataset given to us in the problem. The elements of fiction - character, plot, point of view, setting, style and themes are a good starting place. We took inspiration from the paper SIMFIC [2] to identify the most useful features for our task.

In a Bag of Words approach (BoW), it creates a sparse matrix which is very long and may be computationally expensive. It is simple to understand but we miss out on the semantic dependencies and influence of the text documents. So we opt for other approaches that include the semantic meaning of the dataset into consideration for the classification. In this regard, we have represented our features in 3 distinct ways:

- **TF-IDF** : Term Frequency - Inverse Document Frequency shows how important a word is to the document in the corpus.
- **Document to Vector** : creates a numeric representation of the document, before running the similarity queries.
- **Word to Vector**: creates a numeric representation of each word in the document in a dimensional space, for similarity comparison.

We use these feature representation for the next step namely the model learning and testing. We opted for Naive Bayes, Logistic Regression, Support Vector Machines and Neural Networks for our classification. We chose these models because they are simple to implement and their results can be interpreted well. On the other hand, if we choose various Deep Learning models, some approaches give good results but they are not explainable. It works as a black box which becomes complex to explain and interpret for simple problems.

C. Retrieval Engine and Query Parsing

The data gathered was read in the form of a pickle file. Pickling converts a data hierarchy into a stream of bytes in python. The CSV is read into a dataframe, pre-processing is done by separating the different fields like Description, Licencing etc. First we generated the representation from our data file using TF-IDF, word2Vec and Doc2Vec. We then proceeded to calculate the similarity score for our document with respect to the query that the user would input. The total score is a sum of the TF score and the embed score i.e a max similarity achieved could be of 2 (max score 1 of each similarity measure), which would mean a perfect match for the user's query. Through the Query Synthesizer, we parsed the query, created its feature vector similar to the process for the documents. The user has to enter his query in the double quotes for the query cell in the python notebook and run the cell. The query is matched with the vectors for the documents and the ranked results based on highest matching scores are displayed to the user. The number of results and the matching score can be changed according to the requirements.

Python Libraries and Explanation :

- *numpy*: NumPy is Numerical Python, which consists of multidimensional array objects and a collection of routines for processing those arrays for mathematical and logical operations.
- *pandas*: Python library for data manipulation and analysis that offers data structures and operations for manipulating numerical tables and time series.
- *nltk*: Important library for human language data, working with corpora, linguistic structure, categorizing text, etc.
- *sklearn* - Scikit learn contains algorithms for implementing basic machine learning and data mining tasks like classification, regression, dimensionality reduction, clustering, and model selection.
- *operator*: Library for object comparisons, logical operations, mathematical operations, sequence operations. These functions are useful for all objects.
- *scipy*: SciPy is open source scientific and numerical tools like special functions, gradient optimization, integration, ordinary differential equation (ODE) solvers, tools for parallel programming, etc.
- *re*: Library for Regular Expression that lets us check if a particular string will match a given regular expression.
- *collections*: Containers used to store collections of data, like list, dict, set, tuple etc. These are built-in options of collections.

```
Query match with tool: 'ECount ERP', with score: 0.7071067811865475
```

	name	embedd_match	string_match	total_score	pricing	users	license
95	Jeeves ERP	0.482246	0.999876	1.482122	Subscription	Small (1001 employees)	Proprietary
93	Orion ERP	0.305936	0.999672	1.305608	Subscription	Small (1001 employees)	Proprietary
92	NetSuite ERP	0.256210	0.999594	1.255804	Subscription	Small (1001 employees)	Proprietary
91	Brightpearl	0.142586	0.999774	1.142360	Subscription	Small (1001 employees)	Proprietary
98	Oracle ERP Cloud	0.110647	0.999752	1.110399	Subscription	Small (1001 employees)	Proprietary
84	Zycus Procure to Pay	0.093071	0.999723	1.092795	Subscription	Small (1001 employees)	Proprietary
141	AIMMS	0.078926	0.999884	1.078810	Subscription	Small (1001 employees)	Proprietary Software
12	RF-SMART	0.061200	0.999808	1.061007	Subscription	Small (1001 employees)	Proprietary

Fig. 3. Engine Results for Query Term ”ERP”. The Name of the alternative products along with Match Scores, Pricing, Number of Current Users, and Licensing Information (Proprietary or Open Source) is displayed.

D. Results

When the user inputs the query, we first check if there is an exact match for it. If not, we proceed to find the closest match for the query. If the similarity score is more than 0.8 (developer chosen value for a good score), the results are displayed along with the statement showing the score of the match found from our data file. By matching the cosing similarity, the top results are sorted in a decreasing order and then displayed. Currently the top 20 results are displayed in the output for the query cell. We can change this number to 30 or 40 or any arbitrary number depending on what we are looking for in the results. The Name of the alternative, alongwith its embed match score, string match, and total score are displayed. The pricing is also crucial for all users making a decision on the basis of business process. The number of users and licencing (Open source, Proprietary, etc) are also showed. Fig 4 shows the results for the query term ”ERP”.

TABLE I
SAP PRODUCT CATEGORIES

Table	**No. of Users**		
Product Name	***Category***	***Alternatives***	***Subcategory***
SAP Business One	ERP	Orion-ERP [11]	Vox
	ERP	Epromis-erp [12]	
	ERP	Sigmamrp [13]	
	ERP	Salesbabu-erp [14]	
SAP BW/4HANA	CRM	Novosales	
	CRM	Mrpeasy	
	CRM	Sigmamrp	

[a]SAP Products based on Categories and some alternatives

Making data better

In the data file, there are SAP products as well which were initially displayed as the top results since they were the perfect match. We created a flag for this particular condition that if the term ’SAP’ is contained in the result, that it would not be displayed as a result since it is going against the basic requirement of finding alternatives to SAP products. We also handled common terms like specifying that CRM category and CRM software and Customer Relationship Management would point to the same thing. This particular learning given to the model gave us better results since the vectors created must have had better similarity scores since they were semantically same. For example, ’project’ would mean project management in descriptions. These intricacies also model user behaviour and tendencies since each person is unique when they are searching for something.

VI. User Interface and Evaluation Measures

The final assessment of the efficiency of the tool will be based on the query results for finding alternatives. When the search query is input, it will move through the information retrieval system and search for the key terms in our list of features. The alternative with the most matching features will be displayed to the user. The details of the alternative tool will also be displayed for ease of understanding and user satisfaction. In case that the search terms do not match entirely, the closest search result and features will be displayed.

The evaluation will be done on the basis of the best match scenarios and test cases for different variants of SAP products. We will try to test the limits of the system by querying using key terms, natural language and similar key terms for finding its robustness. That way we will be able to ensure that users with different levels of technical and business knowledge can use the software well.

VII. Why This Approach was Chosen

This project will be an extension of the techniques that our data science degree consists of. We can implement in a hands-on manner to create something that can possibly have a *commercial application*. When executed ideally, this tool could work as a holy grail of searching and finding various replacements for the SAP products. The idea of the project can be likened to creating ontologies that are classically easily interpret-able. But ontologies combine the concepts, attributes, and knowledge sets of the given data and show implicit

relationships between objects under study. The approach that we have chosen builds on the concept of an ontology but uses the basics of *machine learning* algorithms to find relationships within our data, does the *classification* based on the most important features and will retrieve results based on the user's query.

Since SAP products are varied and meant for a wide range of tasks, companies and purposes, it becomes even more complex to find the best choice of software as its replacement. Our initial knowledge about SAP products needs to be apt and wholesome so that even distinctive features can be matched from the very helpful and *comprehensive Alternatives' repository* for the best match and sustained user satisfaction. It will be a mix of exact match and best match search results based on the extracted features.

In *very large business applications*, there are multiple tasks that come under Human Resources (HR), Accounting, Marketing and Logistics etc. which each require a dedicated software. Then these applications need to work in an integrated manner for smooth operations of a company. The alternatives would be beneficial for the general public and common people including students and freelancers etc, who would prefer using substitutes that our tool would offer. Sustainability of the software is also a string driver for finding the most suitable software for each person and institution.

Why choose Word Embeddings? A normal Bag of words representation of text in Information Retrieval suffers from two issues - it does not focus on the order of words in a sentence, and it ignores the semantics of a sentence as well. Which is why we believe that opting for a semantically sound representation of text from our raw data would give better results that can be explained better than a simple word-word representation.

VIII. Deliverables

- Finding the best match results for other queries: From the current database, we can successfully report that the queries find good and acceptable alternatives to SAP products.
- Natural language processing techniques for user queries: The user queries tested have been distinct and very varied and the engine has been able to model the semantic similarities well and deliver appropriate results.
- Details of SAP substitute and its features: In the initial data file, we have included quite a few SAP products, alternative products searched manually and through websites online to create a comprehensive initial database like structure. Their most important features and pricing have been included to make it quite useful for an initial user.
- Apt classification and accurate results: We believe that the semantic characteristics that word embeddings are capturing are quite good. The similarity match are apt, as observed from manual annotation of files and confirmation with stand-alone websites that offer alternative suggestions.
- Scalable model for classifying new training examples: This particular engine is highly scalable since it is complete in itself. The data file can be made even more extensive and it will be handled by the pickle format of Python. Any customisations and UI developed on top of this engine would be scalable and robust, in our opinion.

IX. Other Approaches Tried

- Hierarchical Clustering: Hierarchical clustering groups similar objects into clusters. This was the first and most basic idea that we tried to implement. The final result is a umbrella cluster under which each item is a cluster in itself. This yieled very simple results and could not have worked well for complex requirements and categorisations.
- Wards Method: Ward's method is a criteria applied for hierarchical cluster analysis. Ward's minimum variance is a special case where the criterion for choosing the pair of clusters to merge at each step is based on the optimal value of a given objective function. Fig 3 shows the Wards clusters developed for an initial data file. We can see that it modelled some similarities and was able to cluster on the basis of that.
  Another option was the nearest-neighbor chain algorithm used to find the same clustering defined by Ward's method, Wardp could have been another approach that introduces the use of cluster specific feature weights meaning each feature has its own relevance score.
- We also attempted topic modeling as well using LDA (Latent Dirichlet Allocation), but the results were not optimum and they are usually non-explainable. So we have not used it for the final comparison document.
- Whoosh: Whoosh allows to index free-form/structured text to then quickly find matching documents, based on very simple or complex search criteria. Whoosh is a very flexible pure-Python search engine that was also an initial idea that was explored for implementing this project. It is fast, but is used on pure Python. It works with Python without the need of a compiler. By default, it uses the Okapi BM25F ranking function, but that could have been customised to our requirements. Whoosh creates fairly small indexes as compared to other search libraries. It requires all indexed text to be unicode. Unlike Lucene, Whoosh is not quite a search engine, but a programmer library for creating that desired search engine. From indexing of text, information stored for each term of a field, parsing of search queries, types of queries allowed, and the scoring algorithms, etc. everything is customizable and extensible. We tried to build this turnkey search engine on top of Whoosh, using Lucene as an alternative but our current approach seemed simpler and provided excellent results so the Whoosh route was not explored extensively.

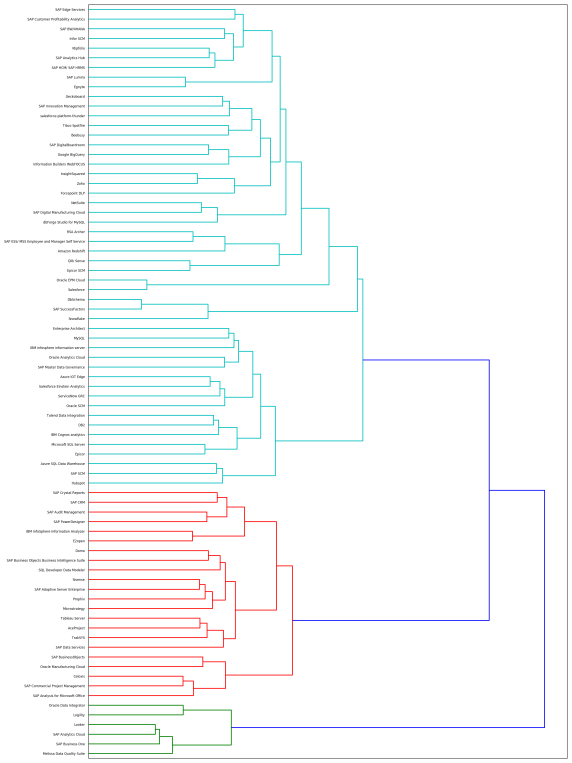

Fig. 4. Ward Clusters Created for Different Categories of the SAP products

X. Evaluation Measures

We have selected multiple evaluation measures for this query engine. We are evaluating the performance of our engine on the all the representations using doc2vec and TF-IDF. The evaluation measures used here for model comparison are precision, recall and accuracy. However, we did not include accuracy, as the data set at hand could report confusing results for accuracy. Accuracy is good for exact match, and fails in imbalanced problems. This problem might lead to high accuracy score since the results would be, which might not always be the case. We manually annotated the software from the data file and then compared the results from the query engine to the manual results obtained. The results were annotated on the basis of domain knowledge, Google search and various classification sites that suggested alternatives well. These results were then matched from the results given by the engine. At each query result, the precision and recall is calculated/ We then plotted these values to see the range of improvement from the number of results that are shown. In Fig 5, we show the Precision for the query "Project Management". Precision shows the number of selected documents that are relevant.

$$Precision = \frac{true\ positives}{true\ positives\ +\ false\ positives}$$

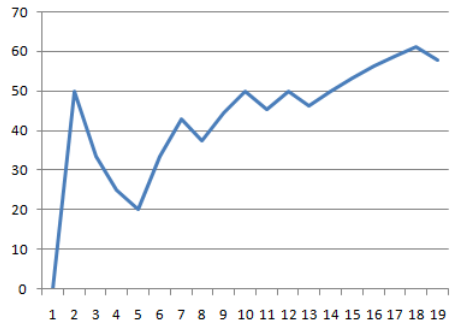

Fig. 5. Precision for Query "Project Management"

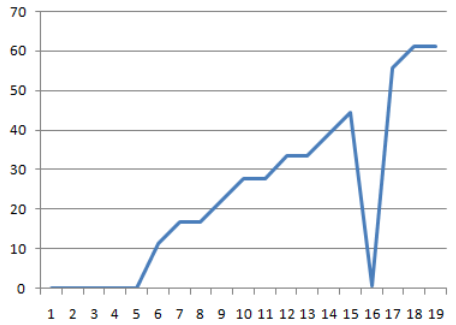

Fig. 6. Recall for Query "Project Management"

In Fig 6, we show the Recall for the query "Project Management". Recall shows the number of relevant documents that are retrieved.

$$Recall = \frac{true\ positives}{true\ positives\ +\ false\ negatives}$$

In Fig 7 we show the Precision for the query "Customer Relationship Management". In Fig 8, we show the Recall for the query "Customer Relationship Management". We can see that the recall value is increasing steadily. The number of relevant search results that are obtained are improving which is a good evaluation measure for the engine.

We want to reduce the impact of false positives and hence we chose Precision as our evaluation metric for choosing the

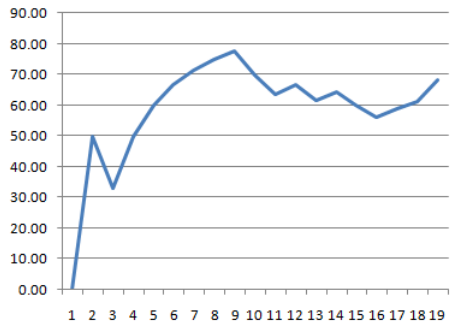

Fig. 7. Precision for Query "CRM" ie. Customer Relationship Management

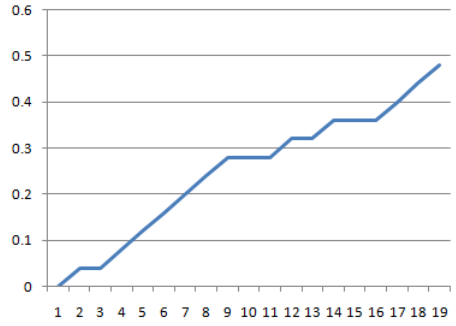

Fig. 8. Precision for Query "CRM". Engine handles the abbreviated term CRM for its full meaning too

best model. Because one could have 100 percent recall yet have a useless model: if the model always outputs a positive prediction, it would have 100 percent recall but be completely uninformative. The models largely perform as expected on the given dataset. A cent by cent metric cannot be achieved when it comes to retrievers due to different assumptions, sampling criteria and other user errors that may result in less than perfect results. Even so, we have achieved reasonable results after development.

XI. Conclusion and Future Work

This project was an initial attempt to create a basic version of a decision support system (DSS) for a user searching for alternatives to SAP products. It works as a Python notebook on a system that has Python installed. In future, we can develop into a fully functional DSS with the help of user interface for a simpler and clean experience. Fig 9 shows a first draft of the possible UI where the user will just enter the query term and the results can be displayed. We can add the filters for price, Open source, or number of users etc which can be chosen or removed at the user's preference.

As an extension, we can create an ontology for a graphical representation of how different products connect to each other. An OWL file can be integrated with the system for even better representation of the textual data. As an aid, the similarities can be shown on the connection arrows for better acceptance by the user. The main goal of any tool these days is to be able to convince the user that the results are trustworthy and the logic behind them is valid. Since vectors in a vector space are difficult to explain, the similarities can be represented through these tree-like structures for ac comprehensive understanding.

We can also augment this whole process with the help of Topic Modelling. It is a machine learning technique i.e. a certain type of statistical model that can discover the abstract "topics" that occur in a collection of documents. It is used as a text-mining tool for discovery of apparently hidden semantic structures in a given text body. Moreover, one huge leap would be if the model could self-learn i.e. with each addition and on the basis of manual annotation the model would keep learning, penalising for wrong results and rewarding for correct retrieval. This can be done on the basis of user feedback and data gathered from each query. Such an engine would be visionary because it would self-train, or reinforce. This would also mean that any new software or product added could be predicted on the basis of similar older software present in the data file, or query results. A self-learning ontology could then be applied to industry software in general and not just SAP products. The database that we used was comparatively very small as compared to the number of products that are available in the market today. As an extension, we can make the data file even larger covering more SAP products and more alternatives. We can also increase the number of features that we are filtering on. The more extensive the data file, the more appropriate results could be obtained for specific user queries. Some niche requirements or businesses could benefit from an extensive database and query engine. One more optimisation that could be possible would be to utilise KD-tree to make the search process even quicker. k-d trees are a special case of binary space partitioning trees, used for searches involving a multidimensional search key.

Acknowledgments

We would like to thank Mr. Matthias Volk for his constant guidance and mentor-ship during the course of the project. Apart from ensuring the goals and the timeline of the project, he was a source of constant motivation while being very objective. His calming presence and leadership throughout the project ensured the smooth functioning of all stages of the project. His focus on the research process and validation of results also ensured the process was very analytical always with reasons for each decision, and awareness of why we chose a particular approach or how it would have to be justified in the scientific process. We are also thankful to Prof. Klaus Turowski for allowing us the opportunity to be working under the support of his research and department. This project has been executed under the aegis of the MRCC (Magdeburg Research and Competency Center) department and the supervision of Mr. Matthias Volk.

References

[1] Mabert, Vincent A and Soni, Ashok and Venkataramanan, MA "Enterprise resource planning survey of US manufacturing firms" Production and Inventory Management Journal, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] Hamed Zamani and W. Bruce Croft, "Relevance-based Word Embedding", 2017, Center for Intelligent Information Retrieval, University of Massachusetts Amherst.
[4] Agnieszka Konys, "Knowledge systematization for ontology learning methods,", 2018, West-Pomeranian University of Technology in Szczecin, Faculty of Computer Science and Information Technology, Szczecin, Poland.
[5] T.Velmurugan and C. Anuradha, "Performance Evaluation of Feature Selection Algorithms in Educational Data Mining,", 2009.
[6] Qiannan Zhu, Xinyi Yu, Yuankang Zhao, Deyi Li, "Customer churn prediction based on LASSO and Random Forest models", 2020.
[7] "https://www.predictiveanalyticstoday.com/sap-business-bydesign/"
[8] "https://www.capterra.com/p/92075/SAP-BusinessObjects/comparisons".
[9] "https://www.sap.com/germany/products/s4hana-erp.html".
[10] "https://www.trustradius.com/products/oracle-financials-cloud/reviews"
[11] "https://epromis.com/"
[12] "https://www.orion-soft.com/en/products/erp/"

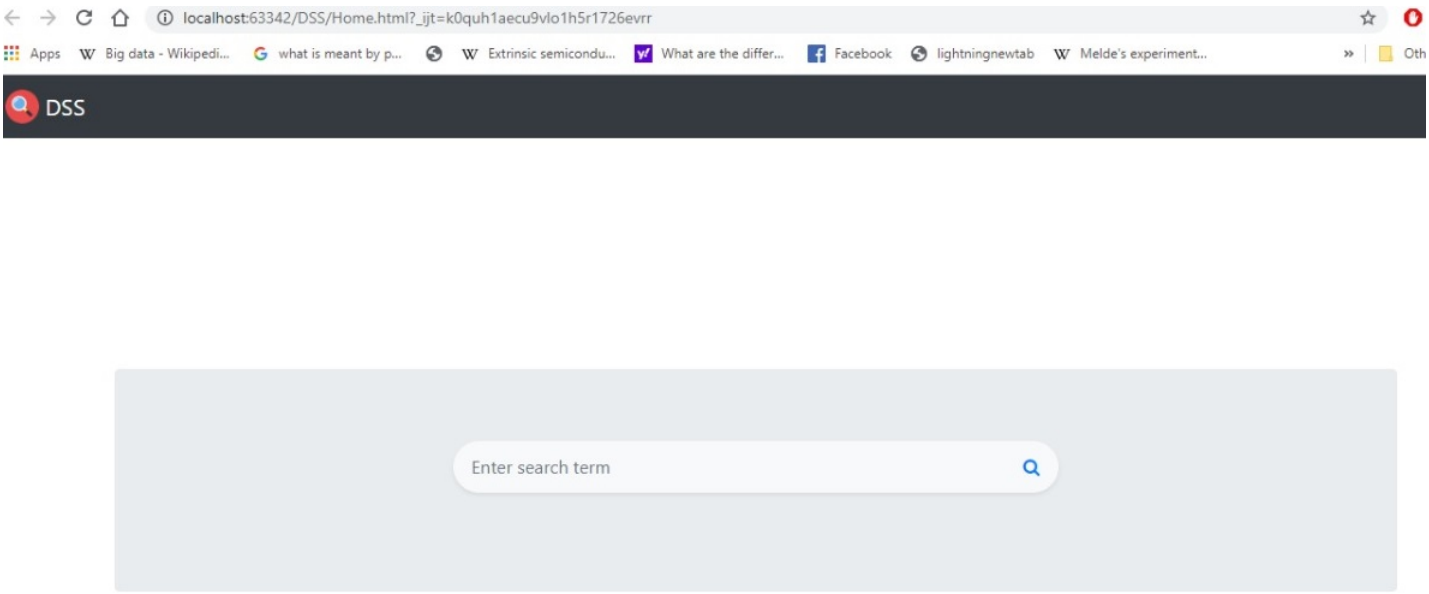

Fig. 9. Proposed UI. Could add filters and other options according to requirements.

[13] ”https://www.sigmanest.com/products/sigmamrp/”
[14] ”https://www.salesbabu.com/small-business-software/small-business-erp-software/”
[15] ”https://www.quora.com/For-a-large-organization-looking-at-SAP-or-Oracle-what-is-a-good-SaaS-cloud-based-alternative”
[16] ”http://www.centriqs.com/smallbusiness/sap-alternative.php: :text=SAP
[17] Shenghui Wang and Rob Koopman, ”Semantic embedding for information retrieval”, OCLC Research, Leiden, The Netherlands
[18] Cimiano P, Völker J. Text2Onto. In: Montoyo A, Muńoz R, Métais E, editors. Natural Language Processing and Information Systems, vol. 3513, Berlin, Heidelberg: Springer Berlin Heidelberg; 2005, p. 227–38.
[19] Wong W, Liu W, Bennamoun M. Ontology learning from text: A look back and into the future. ACM Computing Surveys 2012;44:1–36.
[20] Drumond L, Girardi R. Survey of Ontology Learning Procedures. CEUR Workshop Proceedings 2008;427.
[21] Karoui L, Aufaure M-A, Bennacer N. Ontology Discovery from Web Pages: Application to Tourism. Workshop on Knowledge Discovery and Ontologies 2004:115–20.
[22] Sánchez D, Batet M, Isern D, Valls A. Ontology-based semantic similarity: A new feature-based approach. Expert Systems with Applications 2012;39:7718–28.
[23] Sabou M, Wroe C, Goble C, Mishne G. Learning domain ontologies for Web service descriptions: an experiment in bioinformatics, ACM Press; 2005, p. 190.
[24] Maedche A, Staab S. Ontology learning for the Semantic Web. IEEE Intelligent Systems 2001;16:72–9.
[25] Buitelaar P, Magnini B. Ontology Learning from Text: An Overview. In: Buitelaar P, Cimiano P, Magnini B, editors. Ontology Learning from Text: Methods, Applications and Evaluation, IOS Press; 2005, p. 3–12.
[26] Shamsfard M, Abdollahzadeh Barforoush A. The state of the art in ontology learning: a framework for comparison. The Knowledge Engineering Review 2003;18:293–316
[27] Gomez-Perez A, Manzano-Macho D. OntoWeb Deliverable 1.5: A Survey of Ontology Learning Methods and Techniques, Universidad, Politecnica de Madrid; 2003.