Auditing Purchases without SKU using Text Description and Similarity Search

Ana Paula Appel  
IBM  
Brazil  
apappel@br.ibm.com

Anderson Luis de Paula Silva  
IBM  
Brazil  
anderson.silva@br.ibm.com

Adriana Reigota Silva  
IBM  
Brazil  
adrianeigota@ibm.com

Caique Dutra Santos  
IBM  
Brazil  
caique.santos@ibm.com

Thiago Lobo da Silva  
IBM  
Brazil  
thiagosilva@ibm.com

Rafael Poggi de Araujo  
IBM  
Brazil  
Rafael.Poggi@ibm.com

Luiz Carlos Faray de Aquino  
IBM  
Brazil  
luiz.carlos.faray@ibm.com

ABSTRACT

In this paper, we focus on the problem of auditing MRO (maintenance, repair, and operating) purchases which do not have SKU (Stock-Keeping Unit). Those specific purchases not only lack SKU but also contain short text description, making the audit processes even more difficult. Our goal is comparing recently purchased items with older ones using only the description provided by the purchase process. Therefore, evaluating price differences can uncover possible flaws during the acquiring phase. However, the text of the items that we were working on was very small, with numbers due to the nature of the products and we have a limited amount of time to develop the solution, which was 8 weeks. As a result, we showed that working using a well-oriented methodology we were able to deliver a successful MVP and achieve the results expected for the client, extended its search database by 30%, and allowed it to have a recovery of up to 20 million dollars.

KEYWORDS

similarity search, index, MRO suppliers

1 INTRODUCTION

Item matching [10] is a core function in several domains. In online marketplace where retailers compare new and updated product information against existing listings to optimize customer experience [13]. List comparison where retailers compare their listings with competitors to identify differences in price and inventory. It is also important in others domain as look for past similar purchases in order to evaluate price for example MRO Supply Chain Audit process which is the problem that we present in this work [14].

What is MRO? MRO (Maintenance, Repair, and Overhaul) is an important part of most manufacturing and service businesses and processes as well [15]. MRO includes everything the maintenance crew does to keep your facility (and the equipment inside it) in good operating condition. In other words, the main goal of MRO is to keep your business operations running smoothly. Some items could be represented as machinery and vehicle spare parts, lubricants, office materials, industrial equipment, consumables, computers or work equipment such as boots, uniforms, vests and helmets.

Companies that prioritize Maintenance, Repair, and Overhaul management will have better control over processes that affect their cash flow that’s what it’s all about. Our client is a large producer of long steel in the Americas, and one of the world’s largest suppliers of special steel. It is also one of the largest recyclers in the world. This means that it spent a large amount of money in purchases on MRO each month, around 300 millions of dollars. Part of this value are in purchases that do not have a SKU (Stock Keep Unit), which means that these purchases are bought using only a text description and do not fall inside the catalog. Usually these purchases represent 40% of all purchases monthly.

A regular purchase audit regime can find discrepancies early, cutting off the flow of easy cash to the thieves. The same regime can find ways to improve and streamline the purchase process, making it run faster and with fewer mistakes. Every bit of increased productivity, increased accuracy, and reduced cost is a net benefit for the company as a whole.
Audit of purchases is a significant part of independent financial audit because the purchases constitute the largest item of expense in the income summary of an organization. Purchase transaction might be resulted from different sources such as purchase of materials, merchandise, fixed assets of various types etc.

In our case, our client was able to compare only purchases that have a SKU, which leave out its audit process 40% of purchases which represent millions of dollars. Our goal was to build a solution in 8 weeks that enable to search and find similar purchases in the database allowing compare price and other information (as buyers, provider, contract, etc) and also compare if there is any purchase without SKU that has a similar SKU.

However our challenge is not only develop a solution that solves client problem but also we need to do this in a short amount of time and in a scalable way allowing our client be able to continue working in the solution adding new features. In order to do this we apply an approach where we co-create, co-execute and co-operate.

In more detail, during the co-creation we put together a diverse team of subject matter experts immersed in intensive design thinking and research activities to expose the true nature of a client’s opportunity and establish alignment on a “big idea” to address a specific pain point for a typical end user and create a vision for a minimum viable product (MVP).

After that, in co-execute a solution development cycle that uses Agile and DevOps practices to quickly launch and test an MVP, while capitalizing on business and technology expertise. In this phase is important also have technical team of client working closely. Our squad is usually composed by developer, data scientist, cloud architect and designer. The goal of this phase is validate and improve the MVP’s value in the marketplace through an iterative process of testing, measuring and re-launching.

In the end, we delivery the MVP but we co-operate with client and other business unit that will continue to harden and scale the solution, expanding DevOps practices to broaden feature sets, stress test code, strengthen security and resilience, deploy solutions widely, and expand capabilities, empowered with the confidence to continue to innovate and transform.

In the following sections we will detail our technical work where we built this solution in 8 weeks and allowed the client to retrieve and compare the purchased using only a poor text description. We were able to extend up to 20% the clients database (20% of 40% of no SKU purchases) and analysing three months of data we allow our client be able to recovery, only in the MVP, up to 22 million dollars. Also, with this solution it will be able to streamline in real time all the purchases with and without SKU.

1.1 Related Work

Search is a feature highly present in our life, since the start of relational database we are always want to search for things. Relational databases are with us for more than half a century and it works perfectly fine in search millions of objects. However, when we do not have a key in order to index our products it becomes a little more complicate. Similar search on documents have been with us around 30 years and drive the internet success, using efficient retrieval system and the right data structure to index billions of documents and allows to search in milliseconds.

In order to be able to index and search text we need to create a representation that allow us compare them efficiently, for example, use bag of words [16], TF-IDF [3], topic modeling [18], etc.

Despite the fact that these techniques work pretty well and are scalable, they do not carry meaning and context with them and are too shallow. However, these techniques are the base from the development of better methods.

The advent of embeddings, making use of small vectors to represent text - as item descriptions, allow us to build index that can help users find similar items as music, products, videos, recipes, etc.

As we know text can be represent by words, sentences and documents and deep learning is a powerful tool to work with these forms and lengths. One of the most popular embedding is Word2vec [4], it was one of the first and most applied to text. With it is possible correlate words with other words based on the meaning.

The evolution of word embedding are the transformers, that allows encode a whole sentence which makes possible use the relation between sentence word and the order among them. USE (Universal Sentence Encoding) [19] is one that works well and has a multilingual version. One special kind of transformer very popular is Bert [5] with also a multilingual version and the more recently GPT-3 [2] and OPT-175 [20] models allowing we go further with search and complete the whole sentence.

With the embeddings created we can finally index and search. We are able to do that using K-NN [6] (k nearest neighbor) query and Range query [1]. Basically, K-NN computes the distance between the query point with others in the search space to find the k similar points. This method suffer a lot evolution over the years in order to become efficiently as approximate k nearest neighbors algorithms [8].

In order to use this in real life problems we need the right implementation of K-NN index [21]. Using quantization techniques we are able to make real index and search billions of vectors of embeddings using only few gigabytes of memory. Usually, to choose we look into three aspects: latency - time consuming to index return results, Recall - how is the retrieve of the queries, Memory - how much memory it is consuming.

Depending of amount of data that one is trying to index and search we can choose to brute force but higher accuracy, in memory algorithms that are fast, quantization or disk index solutions.

Currently we have some libraries that implements K-NN and Range searches over index of embeddings:

- FAISS [9] - One of the best library with a very nice interface and several methods to index and search. Also, allows add more content after the index creation.
- Hnswlib [12] is the fastest implementation, highly specialized and optimized.
- Annoy [11] is another K-NN library implemented by Spotify.
- Scann [7] use anisotropic quantization and is one of the faster index outperformed HNSW in speed and recall.
- Catalyzer [17] use neural network to train quantizer.

Based on requirements as maturity level to deploy, efficiency, scalability, flexibility and possibility to add new items instead of re-build we decide to use FAISS in our solution.


2 PROBLEM DEFINITION

In audit purchases process, the ability of monitoring and correcting purchases are important to avoid liabilities and expenses. Thus, the auditor needs to compare the actual purchases with past ones and check not only if the product is the same but also other information, as provider, seasonality, contract number, etc.

This process is trivial when the product in question has a SKU, that is, the problem reduce to traditional query on a relational database. However, for products that do not have a SKU this become a more complex retrieval information problem that requires the use of a specialized data structure in order to perform nearest neighbour and range queries.

Despite the fact that our solution will work on MRO purchased audit we can define it as a item match problem. In a more formal way:

\[
\text{Definition 2.1. Given a set of vectors } x_i \text{ with dimension } d \text{ represent a product description we need find all the past purchases where } i = \text{argmin}_i \| x - x_i \| \text{ where } \| . \| \text{ is the Euclidean distance (L2)}. 
\]

For our case we define that we will consider a match if the distance between the item and the closest neighbour (\(k = 1\)) is less than 0.4. This value was elicited after we make an evaluation of a sample of items with a auditor that worked as a subject matter expert (SME) in the development of our solution (co-execute phase).

However, FAISS is not able to work with a tie list and we do not know how many items we have at the same distance of the \(k = 1\) neighbour. To solve this, we perform a K-NN and retrieve the closes point, if it fits our requirement of distance we perform a range query with the distance of this closest point plus a small number (0.00001) to retrieve all items that are at that distance.

A range search returns all vectors within a radius around the query point (as opposed to the k nearest ones).

3 DATA AND SOLUTION

Our client provided to us a dataset with around 301,204 purchased order from January 2021 to January 2022. In this dataset we had one third of orders (87.331) that were purchases without SKU and two thirds (213.873) purchases with SKU.

The dataset also contains other information as unit, price, provider, requester, date, description, SKU (0 was the value in case the product does not have a SKU), product category and so on.

As already said, the search based on product description was the focus of our solution. The text used to described the product was very short and poor, we present the distribution in Figure 1. Most of the text are between 4 to 6 words without remove any stop words. Another problem was that because of the nature of the product we have a lot of numbers in this text description, for example: screwdriver 1/8 x 4" and screwdriver 1/4 x 10", are different and the price can vary significantly.

In order to build our index we first generate the embeddings from text description using USE (universal sentence embedding), after that we create an index using FAISS, we use IndexFlatL2 that allows us to do exact search and since we are indexing less than half a million and it fits on memory this is the best choice for our solution. The rest of the data we store in a traditional database and the index of each tuple is the same of the embeddings on our FAISS index. Figure 2 present the pipeline of index creation.

After the index construction, we are able to use the index to query the products. One challenge is that we don’t know neither how many neighbours we should query or the distance we should use, because for each product we have different number of similar products with different distance. However we know that the similarity will be higher since the text is short and any variation may indicate a different product. Before set up our strategy we select a sample of products and query for \(k = 3\) and asked for our client to evaluate the answer. Based on this sample we realize the for distance higher than 0.4 the products were completely different.

So we come up with a strategy: First we check if the product has or not SKU, if has it goes to the normal pipeline, if not we need to generate the embeddings and them search the index with \(k = 1\) which is the closest neighbour, with that we will know the distance of this neighbour.

If the first neighbour is at most 0.4 from the query point we add a value 0.00001 to the distance and perform a range query. This step will retrieve the tie list will all similar products that are in the same distance, we show this in Figure 3.

After all candidates be retrieved we analyse the price difference. We compare with the most recent purchase that has price lower than the purchased that we are analysing, if this difference is higher or equal a 30% we flag the purchase to be audit. This value was...
defined by the client as the range of last year inflation and used in this current solution for products with SKU. With difference less than this we save the correlated purchase that we found but not flag it to be audit. For purchases that have distance higher than 0.4 we flag as unique purchases. With all the matches done we save our results to a database and them to our dashboard to be audit by the auditor. The query pipeline is presented in Figure 4. In our dashboard we show all the correlation for the ones flagged to be analysed not only the one that we compare the price, so the auditor could check other purchases.

This method was defined in collaboration with client teams and the information in our dashboard was defined after we did a shadow with one auditor to experience how the process occurs and practice and see how many widows and different information the auditor needs to look in order to evaluate if it is a deviation in the process.  

4 RESULTS
To test and validate our solution we run the analyses of three month of data, which corresponds to 72,175 thousand of purchases from October to December of 2021 (we previous running on January of 2022 but the volume was not enough for client validation). From these purchases 43,562 were purchases with SKU and 28,613. In order to be fair we remove these orders from our index.

From purchases without SKU that were our goal we had 4,991 that cost more than 30% than previous purchases, 10,875 that cost more but still less than 30% and 12,747 that were flagged as unique purchased. We can see the distribution on 1 with the purchases without SKU that we had a match is highlighted. Even the fact that purchases with no SKU that has price difference over than 30% represent around 22 million dollars and purchases with no SKU with difference less than 30% represents 5 million dollars. For the client this represent a potential recovery up to 20 million dollars looking only for purchases without SKU.

In order to show that the index was performing right and explain that we can only find if there is similar purchases we produce an experiment to explain how the index work and it is presented in Figures 5, 6, 7 and 8. The questioning came about the purchases flagged as unique that are the ones with distance higher than 0.4.

For Figures 5 and 6 we select on month from the dataset and compute the K-NN search with $K = 2$. Since we use data point that were indexed, the first neighbour should be the point itself and the distance should be zero. The second neighbour will be a different point and will show the distribution of distance of each neighbour.

For Figures 7 and 8 we select 2000 examples from a dataset with Portuguese sentences used to training translation models and that is not the one used to create the index. Thus, the domain and the content of queries are completely different from the one used to create the index. As we see in Figure 7 all the fist neighbours are more than 0.4 from distance of query point. Which shows that they are very different. This proves that we cannot find in the index purchases that are not there. The Figure 8 only reinforce the argument that the neighbours are very distant.

Both experiments also shows the the choice of using only results that are up to 0.4 from distance was a right choice.

To be able to optimize our solution we planned divide the index in several ones based on category of our items, however we found that a large number of items where more than one category so this strategy does not work. However for the MVP have only one index worked well, since the time to query $k = 1$ neighbour of 4,000 items was 800 seconds which means 0.2 seconds per item. As client is planning running a small set of purchases every week time will not be a problem. However, FAISS has other types of index that allow query be performed fast but for the initial solution we prefer accuracy instead of scalability.

5 CONCLUSION
This was a very challenging project, which was highly focus on data and we are able to delivery a MVP in 8 weeks that will be used right away what is exceptionally motivating and facilitates fast and successful business transformation for a data driven approach.

This solution seems simple and shows that even a simple approach can provide the client the goal of saving money using data smartly. Our approach of work team up with the client allow a more assertive to solve the right problem and don’t be lost in the way. Our user-centered methodology enable our solution assist the user perfectly.

As next steps we also delivery a plan to deploy the solution in the client infrastructure and a road-map for next features and steps,
for example label the matches that are right and not to be able to have a supervised model, also scale the index and connect the full supply chain of MRO and as soon as a purchase that has a high risk have a supervised model, also scale the index and connect the full supply chain of MRO and as soon as a purchase that has a high risk score of be a liability the auditor will be notified.

REFERENCES

[1] Pankaj K Agarwal, Jeff Erickson, et al. 1999. Geometric range searching and its relatives. Contemp. Math. 223 (1999), 1–56.
[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems 33 (2020), 1877–1901.
[3] Kenneth Church and William Gale. 1999. Inverse document frequency (idf): A measure of deviations from poisson. In Natural language processing using very large corpora. Springer, 283–295.
[4] Kenneth Ward Church. 2017. Word2Vec. Natural Language Engineering 23, 1 (2017), 155–162.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
[6] Evelyn Fix and Joseph Lawson Hodges. 1989. Discriminatory analysis. Non-parametric discrimination: Consistency properties. International Statistical Review/Revue Internationale de Statistique 57, 3 (1989), 238–247.
[7] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chen, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In International Conference on Machine Learning. https://arxiv.org/abs/1908.10396.
[8] Kianna Hajebi, Yasin Abbas-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In Twenty-Second International Joint Conference on Artificial Intelligence.
[9] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data 7, 3 (2019), 535–547.
[10] Yan Lei, Long Chen, and Ziyu Guan. 2019. Cloth Recommender System Based on Item Matching. In IOP Conference Series: Materials Science and Engineering, Vol. 533. IOP Publishing, 012044.
[11] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xueming Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. IEEE Transactions on Knowledge and Data Engineering 32, 8 (2019), 1475–1488.
[12] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence 42, 4 (2018), 824–836.
[13] Seyed Omid Mohammadi and Ahmad Kalhor. 2021. Smart Fashion: A Review of AI Applications in the Fashion & Apparel Industry. arXiv preprint arXiv:2111.00905 (2021).
[14] Hooi Ying Ng, Per Christen Trennes, and Leon Wong. 2017. Audit seasonality and pricing of audit services: Theory and evidence from a meta-analysis. Journal of Accounting Literature (2017).
[15] L Robert Pokorny and Harpreet Singh. 2004. WEAVE: An Automated System for Collating Unstructured Data from WEB and Legacy Sources to Enhance the MRO Supply Chain. In ICQ, 275–284.
[16] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. Journal of documentation (2004).
[17] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2018. Spreading vectors for similarity search. arXiv preprint arXiv:1806.00485 (2018).
[18] Hanna M Wallach. 2006. Topic modeling: beyond bag-of-words. In Proceedings of the 23rd international conference on Machine learning. 977–984.
[19] Yinfeng Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, et al. 2019. Multilingual universal sentence encoder for semantic retrieval. arXiv preprint arXiv:1907.04307 (2019).
[20] Suan Zhang, Stephen Roller, Naman Goyal, Mikael Arcteze, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diah, Xian Li, Xi Victoria Lin, et al. 2022. OPT: Open Pre-trained Transformer Language Models. arXiv preprint arXiv:2205.10108 (2022).
[21] Kang Zhao, Hongtao Lu, and Jincheng Mei. 2014. Locality preserving hashing. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 28.