# Enforcing vector space stability in embeddings with evolving vocabularies: a web-app behavior perspective

Asier Rodríguez-González ✉, Ignacio Sisamón Serrano, Ignacio Esplugues Conca, Daniel Sánchez Santolaya, and Joel Medina Sánchez

BBVA - Behavioral Representation, BBVA AI Factory, Madrid, Spain
{asier.rodriguez.gonzalez, ignacio.sisamon, ignacio.espligues.contractor, daniel.sanchez.santolaya, joel.medina.sanchez}@bbva.com

**Abstract.** Embeddings generated from navigation data unlock valuable insights and provide strong baselines for a wide range of applications. However, the dynamic and evolving nature of financial applications presents significant challenges for the stability and adaptability of embedding models, particularly when these embeddings are used as inputs to downstream analytical models. In this paper, an alternative approach for a real-world constraints environment is proposed to address constantly changing vocabularies and dependencies across downstream systems. In order to ensure seamless integration of new elements into an established vector space, and using data from BBVA's application as a case study, a methodology is developed by combining Word2Vec-based embeddings with a two-step pipeline: Embedding Matcher and Space Mirroring. The former is an alignment mechanism that assigns new pages to existing embeddings using Levenshtein distance and cosine similarity. The latter is a technique for embedding projection into the original vector space in which multiple transformation techniques, including SVD, dense layers, ResNet, and GRU-based models, have been compared. The results obtained highlight the effectiveness of preserving semantic integrity and reducing the impact of updates on downstream models while minimizing computational overhead. The proposed approach is applicable to any context that involves dynamic vocabulary data.

**Keywords:** vector space, stability, embeddings, vocabularies, behavior, navigation, word2vec, client, clustering, aggregation, projection, nlp

## 1 Introduction

Since the introduction of the embedding concept [14], its use has become widespread due to its simplicity of application and its potential to be adapted for subsequent tasks [24]. However, the representation of non-textual entities has proven to be a key part of the business of many large companies. For example, in music recommendation, Spotify employs embeddings for the representation of its customers [12]. Uber[1] also uses embeddings to represent both its customers and establishments, and then offers appropriate recommendations. Netflix[2] represents its titles as embeddings based on different features of the movie so that other models can then feed on this useful information. These examples showcase the utility of embeddings to link products, regardless of typology and customers [3].

Embeddings have also found application in banking and finance: stock representation [10], transactions [20], using news for recommendation systems; [23] even to represent customers [4] [8], where these representations are presented as an opportunity to better understand the context around the customer [27]. Within a financial institution, a large amount of data is generated with a structure that is difficult to represent using traditional machine learning methodologies, such as transactions or user navigation in the application. Specifically in a bank like BBVA, with an App that is a fundamental pillar for the customer, navigation is of great importance [3].

Among the navigation we find key data [29], such as the pages the customers visit, the way they navigate and the time spent on each page, among others. When represented as an embedding, this data allows to carry out quantitative analysis such as identifying browsing patterns or performing clustering to identify similar customers[4] [7] [19]. In other words, in addition to providing direct value, embeddings can be considered as baselines and new input features for many other tasks, such as product recommendation systems, pricing or personalization of the users' own browsing experience. By using embeddings, we could reduce feature generation and data mining tasks, which can be very costly with complex and large volumes of data.

However, using embeddings as input to downstream models generates a direct dependency. Although continuously retraining the embedding models can ensure that the representation is updated, this process can alter the structure of the vector space and affect the performance of downstream models and applications that rely on those embeddings. This presents a unique challenge due to the highly dynamic characteristics of the web navigation data, where new pages could constantly emerge, user-interaction patterns evolve thus embeddings should be adjusted accordingly.

## 2 Problem Definition

Typically, embeddings are learned for a specific purpose, such as the representation of banking products [5]. This approach ensures alignment between the embedding and its purpose, simplifying updates, as embedding retraining occurs without extra dependencies. However, such embeddings are not easily generalizable to other tasks.

[1] Uber - Two tower embeddings
[2] Netflix - Supporting content decision makers
[3] BBVA - Spanish banking app with the best reviews on Google
[4] Instacart - Embeddings to improve search relevance

In contrast, foundational models often learn embeddings for generic purposes, such as understanding semantic language relationships. These embeddings can be convenient baselines for various use cases, allowing to take advantage of potential features without worrying about the high computational demand required for their development. The embeddings can also be adapted for specific tasks [11] [30], leveraging prior learning. Yet, they depend on some sort of foundational model's invariability, because updates in the embedding can degrade the results of specific tasks, which may lead to a need to retrain or readjust the specific use case's models.

This paper presents an approach to foundational models using BBVA navigation data, addressing challenges related to dynamic navigation patterns and the introduction of new app sections. Navigation data in this context includes user sessions, which can be defined as a sequence of visited pages in the App hierarchically organized (i.e., `operative:functionality:detail`). For example, a session that includes a query to the home page, where a money transfer is made, could be coded as follows:

```
global position
operations:index
payments:index
payments:operations payment:index
payments:operations payment:index:savedcontacts
payments:operations payment:conditions
payments:operations payment:confirmation
payments:index
global position:logout
```

Despite a well-defined taxonomy, campaigns, which follow market trends, and custom offers frequently expand the number of pages, introducing challenges in embedding stability, as adding these pages to the vocabulary would require constant retraining of the model. Figure 1 illustrates monthly trends in new pages and campaign-related additions. While new campaign pages constitute a minor percentage of total new pages (∼5%), significant changes in operations or page structures can alter user behavior, impacting the page embeddings. Detecting such changes requires studying embedding drift, which causes updates to the foundational model that involve cascading modifications to downstream models.

## 3 Dependencies and Restrictions

BBVA's app produces more than 5 million daily navigation sessions, with more than 20 thousand different pages. Preprocessing in state of the art natural language processing models implies many different steps, such as data cleaning, tokenization and padding. Given the large volume of data, these analytical solutions to generate embeddings from scratch are generally very computationally demanding and difficult to be applied when memory and execution times are limited and security and regulatory filters exist. Alternatively, using data subsets

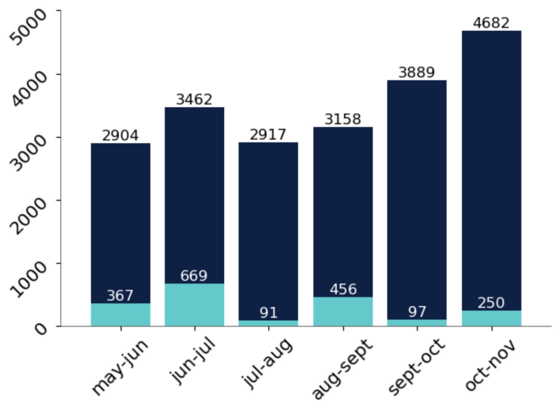

Fig. 1: [Dark Blue] Number of pages appearing for the first time compared to the previous month. [Cyan] Number of pages that are new campaigns each month.

is an option, but they are hard to make representative and would underutilize BBVA's vast data resources.

For this reason, we chose PySpark's Word2Vec as the framework to develop the embedding models presented in the following sections, encoding each page as a vocabulary token. Spark is a technology that allows us to perform distributed cluster computing to speed up the process and handle the entire data volume without having to use subsets. One significant drawback of PySpark's Word2Vec is its inability to resume training from a previous state, which hinders its flexibility in scenarios requiring incremental learning or continuous model updates.

There is an additional alternative of extending the embeddings matrix within the model and freezing the weights corresponding to the pages previously trained, resuming the training from there and preserving the original structure of the vector space [25]. However, within the bank platform, we face the limitations discussed previously, which prevent us from implementing this solution directly. On the other hand, from the embeddings we could try to self-implement the Word2Vec model (keeping the parameters, the hyperparameters and the architecture fixed) and take control of the whole training. Nevertheless, it is a tedious and computationally very expensive process, so it is not feasible in the current situation (but it is not ruled out in the future in higher-resource scenarios).

All these limitations and restrictions make it necessary to create a method that can keep page embedding stable over a period of time, ensuring that information from new pages entering the app can be incorporated in a useful and secure way for embedding-dependent systems.

## 4 Methodology

The first step of our methodology consists of generating an embedding through the model $e$ for each of the unique pages visited by users at the moment, $P_1 =$

$\{p_{1,1}, \dots p_{1,b}\}; P_1 \xrightarrow{e} V_1 \subseteq \mathbb{R}^n$, being $V_1$ the initial vector space. This process, as mentioned in Section 3, uses a Word2Vec model implemented in PySpark to handle the large volume of data processed. To build this vector space, we take as a reference the pages that appear at least ten times in the sessions recorded during a one-year interval, avoiding the use of excessively residual pages to the algorithm and with the aim of capturing representative interaction patterns while minimizing possible seasonal effects within the same year.

A second model, $g$, is then trained using the same parameters as the initial model but incorporating all newly added pages from the elapsed time since the initial training. This model will generate a new vector space, $V_2$, where the entire set of pages is denoted as $P_2 = \{p_{2,1}, \dots p_{2,k}\}; P_2 \xrightarrow{g} V_2 \subseteq \mathbb{R}^n$. For example, if we have detected that the number of pages increases substantially every 3 months, we will train again a model of page embeddings with a time window of one year shifted by 3 months from the time of the initial model training. This model ($g$) captures all new pages $q \in P_2 \setminus P_1; P_2 \setminus P_1 \neq \varnothing$ that have appeared in this time window, as well as many pages that do not change over time, $r \in P_1 \cap P_2; P_1 \cap P_2 \neq \varnothing$.

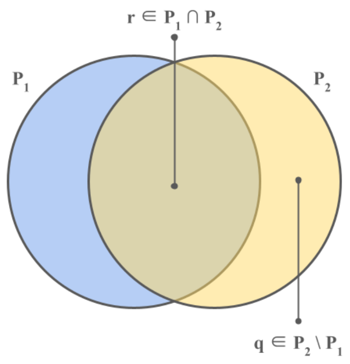

Fig. 2: Set representation of pages in $P_1$ and $P_2$.

The next steps aim to integrate the new pages within the previously defined vector space through a function $f : V_2 \to V_1$. This will allow us to update the model with new information and enrich the number of embeddings for all dependent models, avoiding the cascade update of all processes that use embeddings as input, by keeping the original vector space. To achieve this, the process is divided into two complementary stages. Each of them implements its own function:

1. **Embedding Matcher** ($f_{EM}$): Each new page is compared to existing ones using the Levenshtein distance to identify similarities in page names. Subsequently, the cosine similarity between the corresponding vectors is calculated, allowing us to match pages based on semantic similarity.

2. **Space Mirroring** ($f_{SM}$): To incorporate the new pages into the original vector space, a projection of their vectors into the existing space is performed. This process ensures the semantic coherence of the space by allowing the new embeddings to align with the already established representations.

Once completed, the new page embeddings are projected and adjusted to the original vector space. The projection of new embeddings into the original vector space is performed periodically, dynamically incorporating new interactions, and preserving the overall semantic structure of the vector space over time.

## 4.1 Embedding Matcher

The first step in the flow to keep the vector space as stable as possible is the Embedding Matcher, a component developed within Mercury [6], an inner-course library used in the development of BBVA's analytical processes. As mentioned above, there are pages or campaigns that change monthly, such as the offer of a loan that each month you are granted a different amount, with this amount, or the month of the offer, included in the name of the page for design convenience (i.e., `campaignclick:00000001-this-month-will-be-1600-euros`). This is the reason why all monthly campaigns of this nature could be represented with the same embedding, as they are similar entities, representing the same concept. This process allows for this association of new campaigns and pages that do not yet have a representation but can be matched to an existing embedding in the existing vector space. This is done in a two-step process.

First, the normalized Levenshtein distance ($lev$) of a new page ($q_m$) is calculated with the names of pages already existing ($P_1$), saving the set of pages whose distance is lower than one threshold ($\theta_1$).

$$\begin{gathered} N_{q_m} = \{n_i \in P_1 : lev(q_m, n_i) < \theta_1\} \\ N_{q_1} \cap N_{q_2} \cap \ldots \cap N_{q_m} \neq \varnothing \end{gathered} \tag{1}$$

Next, we calculate the cosine similarity ($Sc$) between the embeddings of all pages in $N_{q_m}$.

$$S_{q_m} = \{Sc(v_i, v_j) : (v_i, v_j) \in e(N_{q_m}) \times e(N_{q_m}), i < j\} \tag{2}$$

If the average cosine similarity exceeds another threshold ($\overline{S_{q_m}} > \theta_2$), we assign the mean embedding of all pages in $N_{q_m}$ to the new page $q_m$.

$$f_{EM}(q_m) = \frac{\sum\limits_{v \in e(N_{q_m})} v}{|N_{q_m}|} \tag{3}$$

To select the threshold values, $\theta_1$ is adjusted by taking the maximum threshold ($\theta_1 = 0$) and increasing it until some false positive samples are seen. Then, choosing $\theta_2$ becomes more straightforward, as most of the pages with high Levenshtein distance have a close cosine similarity. This statement can be seen in Figure 3, where as the value of $lev$ (X-axis) increases, the value of $Sc$ (Y-axis) decreases, concentrating the pages that we will take as a reference for our inference in the upper left corner.

Based on this analysis, the thresholds selected for our use case are $\theta_1 = 0.1$ and $\theta_2 = 0.75$.

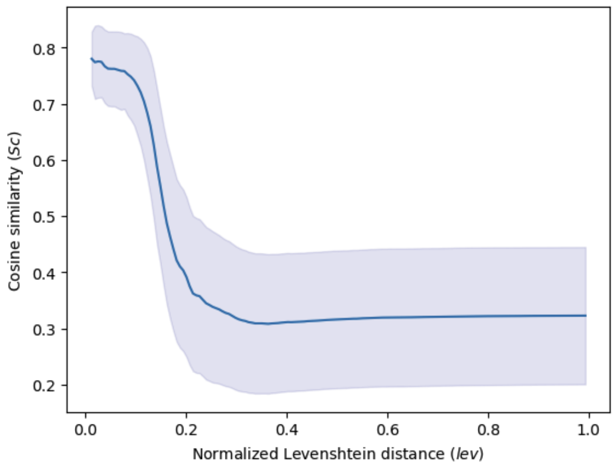

Fig. 3: The X-axis shows the normalization of the discretized Levenshtein distance. The Y-axis shows the average cosine similarity for each discretized value of the Levenshtein distance. The interval around the line refers to the standard deviation for each X-axis value.

Despite the strong correlation between $lev$ and $Sc$, both thresholds are necessary because there are outliers with low values for both $lev$ and $Sc$. Thus, by using $\theta_2$, we discard these pages and provide a more precious inference through the Embedding Matcher. Considering that BBVA's app includes multiple frequently asked questions, many of them exhibit high syntactic similarity—mainly due to standardized string formats such as 'faq:1', 'faq:2', 'faq:3', etc. However, this similarity does not necessarily reflect semantic relatedness. Therefore, the use of $\theta_2$ becomes essential to capture deeper content distinctions.

As an example of how the pipeline works (shown in Table 1), consider a new page introduced in November 2023: `campaignclick-loan-month-nov-2023`. First, the Levenshtein distance of the page name is performed with all the names already existing in the vector space of the model. Then, the pages falling behind the threshold $\theta_1 = 0.1$ are chosen.

Next, the cosine similarity between the embedding of `campaignclick-loan-month-nov-2023` and the other pages is calculated. Then, if the average of all the cosine similarities is above the second threshold

| Page name | Levenshtein | Cosine similarity |
|---|---|---|
| `campaignclick-loan-month-jul-2023` | **0.0909** | **0.92356** |
| `campaignclick-loan-month-sep-2023` | **0.0909** | **0.91534** |
| `campaignclick-loan-month-oct-2023` | **0.0607** | **0.89583** |
| `campaignclick-mortgagge-month-oct-2023` | 0.1831 | 0.69375 |

Table 1: Levenshtein and cosine similarity examples for `campaignclick-loan-month-nov-2023`.

$\theta_2 = 0.75$, the page is fully matched assigning the embedding of the highest cosine similarity or the mean embedding of all the previous pages.

After completing this process, some pages will be successfully matched with previously existing pages, while others will remain unmatched moving on to the next step: Space Mirroring.

## 4.2 Space Mirroring

The objective of this stage is to map the unmatched new pages to the original vector space. We have two different vector spaces: the original vector space $V_1$ composed of the initial pages, and a second vector space $V_2$ composed of the combination of pages present in both spaces, $r$, and new pages, $q$. Since vector spaces are learned using different models, $r$ is represented by different vectors.

With the previously defined functions $e$ and $g$ as $P_1 = \{p_{1,1}, \dots p_{1,b}\}; P_1 \xrightarrow{\text{e}} V_1 \subseteq \mathbb{R}^n$ and $P_2 = \{p_{2,1}, \dots p_{2,k}\}; P_2 \xrightarrow{\text{g}} V_2 \subseteq \mathbb{R}^n$, our goal is to learn a function $f_{SM}$ that maps $q$ into the original vector space $V_1$. We will take advantage of $r$ in order to minimize a specific loss $L$ trying to learn the following function:

$$f_{SM} = \arg\min_{l} L[e(r), l(g(r))] \tag{4}$$

Combining the Embedding Matcher and the Space Mirroring, the final function would be:

$$f : q \rightarrow \begin{cases} f_{EM}(q) & \text{for } \overline{S_{q_m}} > \theta_2 \\ f_{SM}(g(q)) & \text{o.w.} \end{cases} \tag{5}$$

## 4.3 Linear approach

We initially explored the possibility of using a Singular Value Decomposition (SVD) approach [18] to map the embeddings from the vector space $V_2$ to the original vector space $V_1$. The objective was to identify a linear transformation that projects the embeddings from the updated space to the original space. However, for interspace reconstruction and transformation, this approach only captures linear relationships between the dimensions of the matrices.

As expected, this approach proved to be somewhat ineffective due to the non-linear nature of the relationships between the embeddings of the two spaces.

The projected SVD transformation did not achieve the expected results and, therefore, led us to opt for more sophisticated approaches.

## 4.4 Non-linear approach

To explore non-linear solutions, we tested various architectures and loss functions on a neural network, with the aim of finding the best mapping from one space to another.

In the case of the loss function, the choices were between Mean Square Error (MSE) [16] and cosine distance. The former because our goal was to have $f(P_2)$ vectors close to $V_1$; the latter because it may be desirable to maintain the orientation or angle between the output and target vectors.

At the architectural level, and with the assumption that there were no linear relationships between the two spaces, the following configurations are tested and their results are shown in Section 5: architectures (**ResNet** [13], **Self-attention** [28] and **GRU** [9]), regularization (**Dropout** [15]), activation (**ReLU** [1]) and optimizer (**Adam** [21]).

## 4.5 Validation

Measuring the effectiveness of the Embedding Matcher, and thus the value of each threshold, is critical. During calibration, multiple values of each threshold were evaluated through an empirical analysis based on representative examples from different pages and domains. This process allows identifying a balance point where the system minimizes false positives (mismatches) and false negatives (unmatched words), optimizing the performance of the matcher in real scenarios. Although it is possible to automate this selection using optimization algorithms or supervised techniques, the manual approach provides the flexibility to adapt to specific contexts, taking advantage of expert knowledge during development and evaluation iterations.

The Space Mirroring evaluation is carried out in two different ways. First, using only the pages that are common in the initial and updated vector space, a K-Means [2] is trained with the embeddings of the original vector space. Then, an inference is performed with the embeddings of the pages mapped through the mirror function. If the pages with the transformed embedding fall in the same cluster as the original embedding, the transformation is assumed to be accurate since they are very close, and the decision boundaries of the K-Means are maintained in this transformed space (Figure 4). It effectively measures the consistency of cluster mappings between the two spaces, providing insight into how well the transformations preserve the clustering structure across different vector representations. To make the process more reliable, an iterative process is performed in which the number of clusters (K) increases. The higher the value of K, the more challenging the validation.

Unfortunately, this method is not perfect, as pages that are quite close to a K-Means cluster boundary could have the embedding transformed very close to the original and belong to different clusters, generating a false negative in the

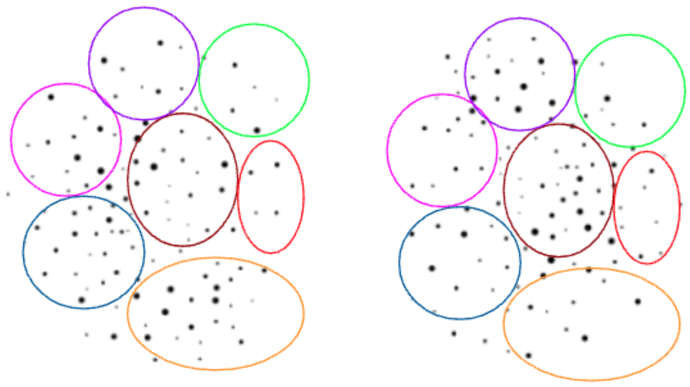
Fig. 4: [Left] K-Means on the original vector space. [Right] K-Means inference on the transformed updated vector space.

metrics. This is why we use a second type of manual validation to complement this first method. It consists of searching by cosine similarity close pages to one taken as a reference in the original vector space and in the transformed vector space. Thus, pages that were previously close should remain close and not have moved away, or at least the uppermost of close pages should be the same, maintaining the isometry. That is, bounding the subset of common pages ($r$): the vectors in $V_1$ and the representations of their vectors of $V_2$ in $V_1$ ($f(g(r))$) should be close. The results of these validations are presented in Section 5.

## 5 Results

In this section, the theoretical information previously presented will be validated through the results obtained in the experiments carried out. To do so, we can start by comparing different methodologies applicable to the transformation function from one dimensional space to another.

As shown in Table 2, the existence of non-linear relationships between the two vector spaces makes the SVD results one of the worst. Although it does seem to keep the points close since for K=2 the results are quite good, when we take higher K, the results deteriorate. This is why the best architecture consists of a GRU maintaining the previously learned residuals (GRU + Residuals) which manages to maintain a high accuracy for numerous clusters, and even keeps getting good results with higher K values (for K=30 it remains at 0.8449).

| Architecture | Min. MSE | K-Means Metric | | | |
|---|---|---|---|---|---|
| | | K=2 | K=5 | K=10 | K=15 |
| SVD | 0.24971 | 0.93187 | 0.86273 | 0.78838 | 0.76032 |
| Dense Layer | 0.06450 | 0.94735 | 0.89819 | 0.85135 | 0.83822 |
| ResNet | 0.06000 | 0.94249 | 0.89320 | 0.85146 | 0.84115 |
| Self-attention | 0.16750 | 0.78529 | 0.39715 | 0.14127 | 0.14070 |
| GRU + Residuals | 0.05060 | 0.94631 | 0.91128 | 0.87517 | 0.87517 |

Table 2: Results of different architectures based on the training of a large set of common pages between the vector spaces $V$ and $W$.

On the other hand, in manual validation, if we look at an example[5] that is in both vector spaces (Table 3), we find that the distance between pages remains fairly constant, although a new page sneaks into the top of the close ones.

| Reference page | Vector space | Most similar pages | Cosine similarity |
|---|---|---|---|
| `expenses:index` | `Original` | `expenses:index:subcategory` | 0.99516 |
| | | `expenses:index:category2` | 0.93397 |
| | | `expenses:index:daily` | 0.87933 |
| | `New` | `expenses:index:subcategory` | 0.99566 |
| | | `expenses:index:category2` | 0.94884 |
| | | `expenses:index:modalfeedback` | 0.92237 |
| | | `expenses:index:daily` | 0.90876 |

Table 3: Most similar pages to `expenses:index` in the original and new vector space embedding.

It is important to remember that the mirror function, despite being trained with the common pages in the original and the updated vector spaces, will only be applied to those new pages that appear in the vocabulary, so the representation of the previous pages would maintain the original embedding.

The example presented in Table 3 compares the page `expenses:index` with the most similar pages in both the original ($V_1$) and transformed ($V_2$) vector space embeddings. Here, `expenses:index` already exists in the original vector space, and we observe how its nearest neighbors are mapped in both spaces.

In contrast, Table 4 focus on the page `financial health rules:index`, which is a new page being projected into the original vector space. These tables compare `financial health rules:index`'s most similar pages in both the original and transformed vector spaces. Despite the change in the vector space projection, we find that the closest pages to both `expenses:index` and `financial`

[5] Both examples maintain the structure discussed in the introduction, in which `operative:functionality:detail` is established as the basis for the pages of the app. Thus, the closest pages within an operative will be functionalities of it, with different details.

| Reference page | Vector space | Most similar pages | Cosine similarity |
|---|---|---|---|
| `financial health rules:index` | `Original` | `financial health rules:index: modalmodalSuccess` | 0.91669 |
| | | `financial health rules:index:modalList` | 0.91379 |
| | | `financial health rules:index:modelCreate` | 0.89919 |
| | `New` | `financial health rules:index:modalSuccess` | 0.92165 |
| | | `financial health rules:index:modalCreate` | 0.83532 |
| | | `financial health rules:index:modalList` | 0.80786 |

Table 4: Most similar pages to `financial health rules:index` in the original and new vector space embedding.

`health rules:index` remain the same, demonstrating that the relative proximity of the nearest neighbors is preserved, even though their spatial positions within the vector spaces have changed.

### 5.1 Downstream Validation

In order to select candidate models and to ensure that the output embeddings can be used in other downstream tasks, it is necessary to continuously evaluate the procedures performed. For this purpose, an embeddings benchmark module has been developed in the aforementioned BBVA inner-source library, Mercury [6]. This benchmark, based on the MTEB [22], seeks to test them in different specific tasks, assuming that a better result in all tasks globally implies a better quality of the basic embedding. Although the embeddings trained in this document are page embeddings, as discussed above, most of the tasks for which these will be used are directly related to clients, so the benchmark will look at that level of granularity. To achieve the client embeddings, different aggregation methods have been applied to the page embeddings: **Mean Pooling**, **Time Weighted Mean Pooling** and **VLAD** [17].

For the evaluation, a representative sample of customers is used. An embedding of the client data is generated for each customer and a classifier or regressor is trained (depending on the benchmark problem) receiving the embedding as the only input. Through this approach, the variation in the results will correspond uniquely to the embedding model and the chosen aggregation method.

The estimator must be able to successfully predict different variables, such as customer movements, engagement levels, or financial behavior, from this embedding. If the estimator performs well, we can be confident that the embedding encodes the information accurately. The studied evaluators assess diverse aspects of customer data, so if a pooling method consistently outperforms others across different evaluators, it will be selected as the preferred aggregation approach. Some of the used evaluators are:

- **Employed**: the objective is to discern between the employed and those who are not actively working.
- **Engagement**: multi-class problem where a label is identified to represent the transactionality and connection a customer has with BBVA Bank.
- **Expertise**: multi-class problem in which the taxonomy of labels refers to the level of usage that a customer has when navigating the app.

| Aggregation Type | Dimension | Employed | Engagement | Expertise |
|---|---|---|---|---|
| | | AUC | F1 | F1 |
| Random | 40 | 0.50038 | 0.22230 | 0.14750 |
| Mean Pooling | 40 | 0.73951 | 0.65945 | 0.41465 |
| Time Weighted | 40 | 0.73340 | 0.65198 | 0.39851 |
| VLAD (K=3) | 120 | 0.77404 | 0.66929 | 0.47368 |
| VLAD (K=5) | 200 | 0.78943 | 0.69396 | 0.48591 |
| VLAD (K=10) | 400 | 0.79829 | 0.71038 | 0.52093 |

Table 5: Benchmark evaluators applied to different types of aggregation to generate client embedding. All for a 365-day window.

According to Table 5, the random baseline exhibits the lowest performance across all metrics, as expected, with an AUC of 0.5004 for Employed and F1 scores of 0.2223 and 0.1475 for Engagement and Expertise, respectively. Mean Pooling and Time Weighted techniques, both with a dimension of 40, significantly improve the results, achieving AUC values of 0.7395 and 0.7334 for Employed, and higher F1 scores for Engagement (0.6594 and 0.6520) and Expertise (0.4147 and 0.3985). Among the VLAD approaches, performance improves as the number of clusters (K) and dimension increase. VLAD (K=10) achieves the highest results, with an AUC of 0.7983 for Employed, and F1 scores of 0.7104 and 0.5209 for Engagement and Expertise, respectively, demonstrating the effectiveness of larger, more expressive embeddings.

The previously presented results include pages that have been projected back into the original space using the analytical methodology proposed in this work. We observed no variation in the metrics across the different evaluators, indicating that no degradation occurs when employing the pipeline in downstream tasks.

## 6 Future Work

The restrictions that revolve around the platform, the implementation time, and costs affect this work significantly. Once its effectiveness has been demonstrated and the informative capacity of embeddings as input for other processes has also been proved, the model could be improved. Currently, using a Word2Vec allows us to generate a functional, fast and simple solution. In order to improve it, different alternatives have been identified. Each page is treated independently of the others, but we know that there are operational relationships between them. For example, the hierarchy of pages, where the operation is listed first, followed by details of the operation, can provide a lot of information as they have a significant relationship. Alternatively to Word2Vec, to account for sequence order, any auto-regressive model could be explored.

An interesting technique to take advantage of the capabilities of existing LLMs is the one mentioned by Tan [26]. At its core, navigation could be seen as a graph of interconnected pages. Applying the above methodology, we could use connectors to convert the graph into a set of texts and then represent each of the sessions through an LLM. The complexity of the taxonomy of the pages themselves may require a fine-tuning of the model.

Another potential direction for future work could explore implementing continuous drift detection mechanisms within the BBVA's application to track changes in data distribution over time. This would enable a more targeted response to pipeline obsolescence, reducing reliance on costly full embedding retraining.

## 7 Conclusions

On several occasions, real problems bring with them restrictions that make it impossible to apply state-of-the-art solutions, so it is necessary to be able to find alternatives. This project presents an architecture that allows updating the embeddings of a constantly changing vocabulary, without altering the original vector space, as it is used as a foundational model that generates many dependencies with subsequent analytical processes.

In the performed tests, it was shown that the proposed method works effectively when dealing with dynamic vocabularies. Firstly, the use of the Embedding Matcher provides a key value by allowing the rapid identification and clustering of similar words based on a threshold that can be parameterized according to the use case. Subsequently, the implementation of Space Mirroring ensures that new words are not only integrated into the existing vector space, but also that they do so while respecting its coherence and minimizing the impact on subsequent analytical dependencies. Finally, automated validation using techniques such as K-Means allow to consistently measure the quality of the fit of the new vectors, ensuring a smooth transition without compromising the stability of the model. Consequently, this approach has been adopted within BBVA's production-ready environment to ensure scalable and reliable embedding updates.

This approach is not only useful in natural language processing contexts with highly dynamic vocabularies, such as the one discussed in this paper, but could also be extended to other domains. For example, applications in recommendation systems that need to constantly adapt to new products or categories, or in the generation of custom embeddings for specific technical domains, such as constantly evolving medical or legal terms. This presents a versatile framework that balances adaptability and stability, offering practical solutions to the challenges of changing vocabularies.

## 8 Acknowledgements

The authors extend their sincere gratitude to BBVA for providing the foundation and resources necessary for this research. We thank all our colleagues, with special attention to Mauricio Ciprián and Leonardo Baldassini for the supervision and council, and Álvaro Ibraín for the initial brainstorming from which we ended up with the final proposals.

## References

1. Agarap, A.F.: Deep learning using rectified linear units (relu). ArXiv **abs/1803.08375** (2018), https://api.semanticscholar.org/CorpusID:4090379
2. Ahmed, M., Seraj, R., Islam, S.M.S.: The k-means algorithm: A comprehensive survey and performance evaluation. Electronics **9**(8) (2020). https://doi.org/10.3390/electronics9081295, https://www.mdpi.com/2079-9292/9/8/1295
3. Ai, Q., Zhang, Y., Bi, K., Chen, X., Croft, W.B.: Learning a hierarchical embedding model for personalized product search. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 645-654. SIGIR '17, Association for Computing Machinery, New York, NY, USA (2017)
4. Baldassini, L., Serrano, J.A.R.: client2vec: Towards systematic baselines for banking applications. ArXiv **abs/1802.04198** (2018), https://api.semanticscholar.org/CorpusID:3609178
5. Boulenger, A., Liu, D., Farajalla, G.P.: Sequential banking products recommendation and user profiling in one go. In: Proceedings of the Third ACM International Conference on AI in Finance. p. 317-324. ICAIF '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3533271.3561697, https://doi.org/10.1145/3533271.3561697
6. Chaquet, J., Ibrain, A., Santolaya Sánchez, D., Galletero, M., Basaldua, S., Delgado, A.: Mercury: Reusable and efficient mlworkflows in finance. In: ACM ICAIF 2024: From Prototype to Production: Deploying Real-World AI / ML Models in the Financial Industry (11 2024), https://openreview.net/forum?id=NBeBMJIWFY
7. Chen, Y., Huzhang, G., Zeng, A., Yu, Q., Sun, H., Li, H.Y., Li, J., Ni, Y., Yu, H., Zhou, Z.: Clustered embedding learning for recommender systems. In: Proceedings of the ACM Web Conference 2023. p. 1074-1084. WWW '23, Association for Computing Machinery, New York, NY, USA (2023)
8. Chitsazan, N., Sharpe, S., Katariya, D., Cheng, Q., Rajasethupathy, K.: Dynamic customer embeddings for financial service applications (2021), https://arxiv.org/abs/2106.11880
9. Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing (2014), https://api.semanticscholar.org/CorpusID:5590763

10. Dolphin, R., Smyth, B., Dong, R.: Stock embeddings: Learning distributed representations for financial assets (2022), https://arxiv.org/abs/2202.08968
11. Dušek, R., Galias, C., Wojciechowska, L., Wawer, A.: Improving domain-specific retrieval by nli fine-tuning. Annals of Computer Science and Information Systems **Vol. 35**, 949-953 (2023), http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-1ba6acb7-cd08-4a70-8fde-b28b7cde7a05
12. Hansen, C., Hansen, C., Maystre, L., Mehrotra, R., Brost, B., Tomasi, F., Lalmas, M.: Contextual and sequential user embeddings for large-scale music recommendation. In: Proceedings of the 14th ACM Conference on Recommender Systems. p. 53-62. RecSys '20, Association for Computing Machinery, New York, NY, USA (2020)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770-778 (2016). https://doi.org/10.1109/CVPR.2016.90
14. Hinton, G.E.: Learning distributed representations of concepts. Proceedings of the Annual Meeting of the Cognitive Science Society **8**(1), 1-12 (1986), https://escholarship.org/uc/item/79w838g1
15. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR **abs/1207.0580** (2012), http://arxiv.org/abs/1207.0580, cite arxiv:1207.0580
16. Hodson, T.O., Over, T.M., Foks, S.S.: Mean squared error, deconstructed. Journal of Advances in Modeling Earth Systems **13**(12), (2021). https://doi.org/https://doi.org/10.1029/2021MS002681, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002681
17. Ionescu, R.T., Butnaru, A.: Vector of locally-aggregated word embeddings (VLAWE): A novel document-level representation. In: Burstein, J., Doran, C., Solorio, T. (eds.) 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). pp. 363-369. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1033, https://aclanthology.org/N19-1033/
18. Kalman, D.: A singularly valuable decomposition: The svd of a matrix. The College Mathematics Journal **27**(1), 2-23 (1996). https://doi.org/10.1080/07468342.1996.11973744, https://doi.org/10.1080/07468342.1996.11973744
19. Katić, T., Milićević, N.: Comparing sentiment analysis and document representation methods of amazon reviews. In: 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY). pp. 000283-000286 (2018). https://doi.org/10.1109/SISY.2018.8524814
20. Khazane, A., Rider, J., Serpe, M., Gogoglou, A., Hines, K., Bruss, C.B., Serpe, R.: Deeptrax: Embedding graphs of financial transactions. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). pp. 126-133 (2019)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR). p.13. Ithaca, NY: ArXiv (2015), https://arxiv.org/abs/1412.6980
22. Muennighoff, N., Tazi, N., Magne, L., Reimers, N.: MTEB: Massive text embedding benchmark. In: Vlachos, A., Augenstein, I. (eds.) Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics. pp. 2014-2037. Association for Computational Linguistics, Dubrovnik, Croatia (May 2023). https://doi.org/10.18653/v1/2023.eacl-main.148, https://aclanthology.org/2023.eacl-main.148/

23. Ren, J., Long, J., Xu, Z.: Financial news recommendation based on graph embeddings. Decision Support Systems **125**, 113115 (2019). https://doi.org/https://doi.org/10.1016/j.dss.2019.113115, https://www.sciencedirect.com/science/article/pii/S0167923619301447
24. Su, H., Shi, W., Kasai, J., Wang, Y., Hu, Y., Ostendorf, M., tau Yih, W., Smith, N.A., Zettlemoyer, L., Yu, T.: One embedder, any task: Instruction-finetuned text embeddings (2023), https://arxiv.org/abs/2212.09741
25. Tai, W., Kung, H.T., Dong, X., Comiter, M., Kuo, C.F.: exBERT: Extending pre-trained models with domain-specific vocabulary under constrained training resources. In: Cohn, T., He, Y., Liu, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 1433-1439. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.findings-emnlp.129, https://aclanthology.org/2020.findings-emnlp.129/
26. Tan, Y., Zhou, Z., Lv, H., Liu, W., Yang, C.: Walklm: A uniform language model fine-tuning framework for attributed graph embedding. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol.36, pp. 13308-13325. Curran Associates, Inc. (2023)
27. Tungjitnob, S., Pasupa, K., Suntisrivaraporn, B.: Identifying sme customers from click feedback on mobile banking apps: Supervised and semi-supervised approaches. Heliyon **7**, e07761 (08 2021)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol.30. Curran Associates, Inc. (2017)
29. Zhang, J., Bai, B., Lin, Y., Liang, J., Bai, K., Wang, F.: General-purpose user embeddings based on mobile app usage. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 2831-2840. KDD '20, Association for Computing Machinery, New York, NY, USA (2020)
30. Zhao, X., Wang, M., Zhao, X., Li, J., Zhou, S., Yin, D., Li, Q., Tang, J., Guo, R.: Embedding in recommender systems: A survey (2023), https://arxiv.org/abs/2310.18608