An Approach for Product Record Linkage Using Cross-Lingual Learning and Large Language Models

Andre Luiz Firmino Alves[1], Cláudio de Souza Baptista[2], José Itallo Martins Silva Diniz[2], Francisco Igor de Lima Mendes[2] and Mateus Queiroz Cunha[2]

[1]*Federal Institute of Paraíba, Brazil*

[2]*Federal University of Campina Grande, Brazil*

andre.alves@ifpb.edu.br, baptista@computacao.ufcg.edu.br, {jose.diniz, francisco.mendes, mateus.cunha}@ccc.ufcg.edu.br

Keywords: Cross-Lingual Learning, Record Linkage, Product Matching, Information Retrieval.

Abstract: Organizations increasingly rely on data for the decision-making process. Nevertheless, significant challenges arise from poor data quality, leading to incomplete, inconsistent, and redundant information. As dependency on data grows, it becomes essential to develop techniques that integrate information from various sources while dealing with these challenges in the context of product matching. Our work investigates information retrieval and entity resolution approaches to product matching problems related to short and varied product descriptions in commercial data, such as those found in electronic invoices. Our proposed approach, STEPMatch, employs deep learning models alongside cross-lingual learning techniques, enhancing adaptability in contexts with limited or incomplete data, effectively identifying products accurately and consistently.

1 INTRODUCTION

The internet has become a vast repository of information about real-world entities, such as products, people, and organizations, described heterogeneously across distinct platforms (Han et al., 2023). The rise of such unstructured data has made it essential to develop solutions that integrate this information effectively. The task of Entity Resolution (ER) emerges as an effective technique for identifying and linking these different representations, ensuring data consistency and quality, which are critical aspects in a myriad of applications, from business decision-making to government oversight (Christophides et al., 2020; Christen, 2012).

Product matching, a subset of Entity Resolution, aims to identify similar products even when described in varying ways. This task poses unique challenges in e-commerce and government procurement, where incomplete descriptions, spelling variations, and inconsistencies complicate the data linkage. Prior research on product matching has predominantly focused on e-commerce data with detailed and structured descriptions, primarily in English (Gözükara and Özel, 2021; Barlaug and Gulla, 2021; Christophides et al., 2020). However, this focus does not reflect the characteristics of sales records from electronic invoices. Furthermore, much of the research has been limited to pairwise product matching, neglecting record linkage approaches that could integrate products within broader and more diverse datasets. This restriction limits the application of the previous research in more complex data integration scenarios (Köpcke et al., 2010; Tracz et al., 2020; Peeters and Bizer, 2022; de Santana et al., 2023; Traeger et al., 2024).

Product data obtained from electronic invoices often includes brief and unclear descriptions and a lack of standardized information. Consequently, significant challenges arise for product matching approaches that aim to manage these documents. Additionally, the limited availability of annotated data in low-resource languages, such as Portuguese, hinders the effectiveness of traditional supervised entity resolution methods. This situation presents an opportunity for cross-lingual learning (CLL) approaches, which transfer knowledge from annotated corpora in other languages to contexts with limited annotations (Peeters and Bizer, 2022; Pikuliak et al., 2021; De Oliveira et al., 2024). This method offers a viable alternative for product matching in low-resource languages, mainly when dealing with short and low-quality descriptions.

In this work, we propose STEPMatch, derived from a methodology based on cross-lingual learning for product matching in short descriptions. We evaluate our model's effectiveness in retrieving and linking

Alves, A. L. F., Baptista, C. S., Diniz, J. I. M. S., Mendes, F. I. L. and Cunha, M. Q.
An Approach for Product Record Linkage Using Cross-Lingual Learning and Large Language Models.
DOI: 10.5220/0013285000003929

In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 1*, pages 63-74
ISBN: 978-989-758-749-8; ISSN: 2184-4992

products from textual descriptions, utilizing information retrieval techniques and semantic refinement to overcome the limitations of keyword-based methods, such as TF-IDF and BM25, which do not adequately capture the semantics of short descriptions (Rateria and Singh, 2024; Hambarde and Proença, 2023). Our proposal aims to enhance the performance in entity resolution for products and contribute a solution applicable to scenarios with scarce and noisy data typical of tax and e-commerce.

We highlight the following contributions of our work:

- An Approach for Product Record Linkage Using Cross-Lingual Learning and Large Language Models;
- Assessment of Cross-Lingual Learning for Product Matching;
- An Analysis of Lexical, Semantic, and Hybrid Methods for Searching Products with Short Descriptions; and
- A novel reranking approach for Information Retrieval Systems using Cross-Lingual Learning and Large Language Models to enhance the ranking of search results.

The remainder of this work is structured as follows: section 2 discusses the related work; section 3 details the designs of the algorithms that compose STEPMatch, as well as an overview of the steps utilized to achieve effective product matching; section 4 focuses on the experiments we conducted; section 5 discusses our findings and what those findings mean for the effectiveness of STEPMatch; and lastly, section 6 encompasses our conclusions, pointing out our contributions followed by future work to be undertaken.

2 RELATED WORK

Entity resolution, also known as record linkage, duplicate detection, or reference reconciliation, aims to identify different representations of the same real-world entity, promoting consistent data integration across various applications (Köpcke et al., 2010; Barlaug and Gulla, 2021; Christen, 2012). The entity resolution task typically comprises two main steps: 1) Blocking, which reduces the number of necessary comparisons, and 2) Matching, which determines whether a pair of entities refers to the same object.

Product matching is a particular application of Record Linkage that aims to identify equivalent products across different data sources. Various approaches, such as probabilistic models, rule-based algorithms, and machine learning techniques, are employed for this task. Deep learning and large language models are currently considered state-of-the-art product matching solutions (Barlaug and Gulla, 2021).

Researchers have extensively studied the optimization of entity-matching techniques for large data volumes. Xiao et al. (2011) developed a filter to avoid calculations between all possible pairs using token ordering. Ristoski et al. (2018) proposed a product matching approach based on Natural Language Processing (NLP) and deep learning, combining textual and visual features extracted with Conditional Random Field (CRF) and Convolutional Neural Network (CNN) for classification with traditional machine learning algorithms. Barbosa (2019) utilized diverse textual representations and a deep learning-based binary classifier to capture similarity patterns in product matching. To overcome the lack of annotated data in a specific language, leveraging available data in other languages to train and optimize machine learning models, Peeters and Bizer (2022) employ Cross-Lingual Learning in product matching classification.

Various Entity Resolution frameworks stand out for their diverse approaches. Christen (2008) and Bilenko and Mooney (2003) apply blocking and classification with algorithms such as Support Vector Machines (SVM) to identify duplicate records. Konda (2018) offers a comprehensive solution for ER, including pre-processing, data analysis, and machine learning-based blocking. Meanwhile, DeepER (Ebraheem et al., 2017) and DeepMatcher (Mudgal et al., 2018) utilize vector representations and embeddings to capture semantic similarities. Finally, Ditto (Li et al., 2020) employs pre-trained language models to perform contextualized classification of product pairs. At the time of this writing, Ditto currently represents the state-of-the-art in entity matching (Peeters and Bizer, 2022; Barlaug and Gulla, 2021).

This work distinguishes itself by addressing the challenge of matching product descriptions found in electronic invoices, which are typically shorter and less structured than those commonly used in e-commerce. While most existing entity recognition and product matching approaches have focused on structured data, our study proposes a comprehensive solution that includes blocking techniques and innovative re-ranking methods within information retrieval systems. To the best of our knowledge, this approach is novel as it explores cross-lingual learning and information retrieval methods as effective strategies to improve product linkage precision, especially in fragmented data and multiple languages.

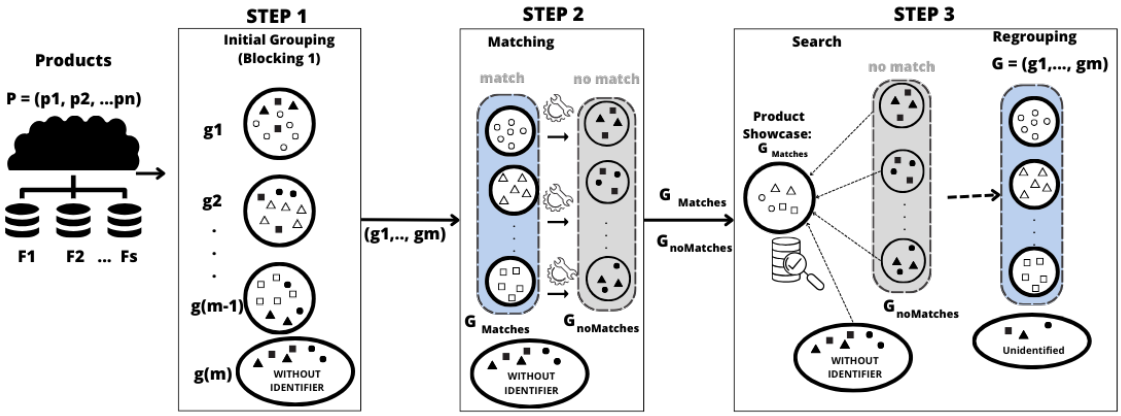

Figure 1: STEPMatch general overview.

3 STEPMatch: SHORT TEXT PRODUCT MATCHING

This section introduces the STEPMatch approach proposed in our work to perform record linkage on short texts. We present the key components and methods involved in addressing the product matching problem, particularly emphasizing the Information Retrieval (IR) mechanisms proposed to retrieve the associations of product identifiers.

3.1 Overview

In electronic invoices, a single product identifier may refer to multiple distinct descriptions of the same item, and errors can occur in the association between product codes and descriptions. We aim to correctly associate the product descriptions with their respective identifiers, resolving record inconsistency issues. Our approach includes discovering non-corresponding products with the same identifiers and correcting them with the most appropriate ones, especially for products with inconsistent records. Our work serves as a solution to address data inconsistency problems in this type of scenario.

Figure 1 provides an overview of STEPMatch. The process begins in step 1 with an initial clustering of products, denoted as the set $P = \{p_1, p_2, \ldots, p_n\}$, sourced from various data sources. This step groups products with similar attribute values into $G = \{g_1, g_2, \ldots, g_m\}$. Each group $g_i \in G$ contains a subset of similar products, defined as $g_i = \{\, p_j \in P \mid j = 1, \ldots, k\}$, where $1 \leq k \leq n$, with $g_i \subset P$.

In step 2, the product groups $g_i \in G$ undergo processing, and matching verification is carried out internally among the products within each group. This results in two types of groups: matching groups (G_{Matches}) and non-matching groups ($G_{\text{noMatches}}$).

Finally, in step 3, the focus is on identifying product matches that were not detected in the previous step, specifically targeting the products in $G_{\text{noMatches}}$.

Figure 2 illustrates an example of the operations performed in the steps of STEPMatch. The process begins given a set of products from various data sources: in Step 1 the algorithm identifies two groups of products right after analyzing the input data; in Step 2, products that do not belong to any of the groups are detected; these mismatched products are therefore forwarded to Step 3, which is responsible for correctly associating them with their respective groups. Products that remain unassociated with any group are set aside and, along with future data loads, will be reprocessed by STEPMatch.

Similarity functions are used to define product groups at different steps of STEPMatch. These functions analyze each pair of products by processing data based on the current step. Typically, the similarity between any two products, p_i and p_j, is determined by a function $F_{Sim}(p_i, p_j) \geq \theta$, where θ represents a similarity threshold.

The following subsections describe the steps of STEPMatch, emphasizing step 3, which focuses on this work's main contribution.

3.1.1 Step 1: Blocking

The blocking step adopted by STEPMatch involves dividing the product dataset into blocks or smaller groups based on specific criteria. These groups were designed to select products that could be potential candidates for comparison during the matching step. We seek to limit comparisons to entities within each block, avoiding the algorithmic complexity of $O(N^2)$

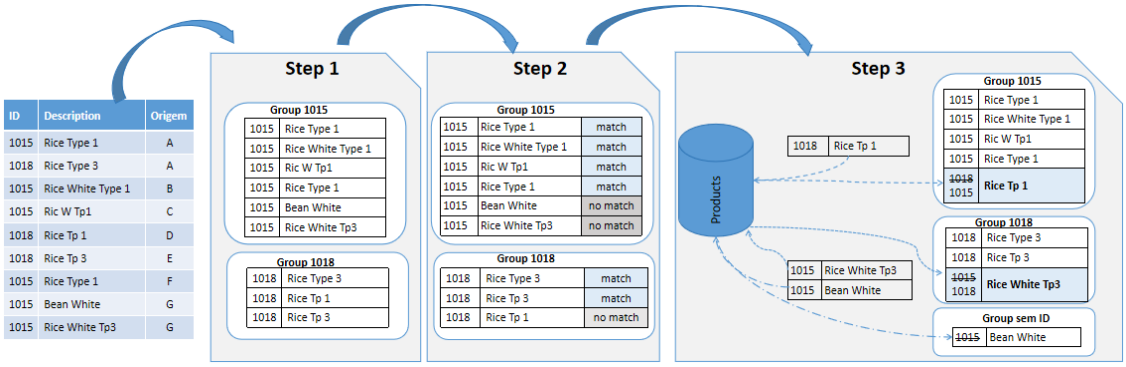

Figure 2: Illustrative example of the operation of STEPMatch.

during the matching phase (Papadakis et al., 2021; Christophides et al., 2020).

The Standard Blocking (SB) is a hash-based strategy for entity resolution. It generates blocking keys by concatenating parts of selected attributes, forming groups of entities with identical keys (Papadakis et al., 2021). The initial clustering of products uses the SB method, in which the attribute *product identifier*, present in the data, was used to represent the blocking key. At this stage, the similarity function $F_{Sim}(p_i, p_j) \geq \theta$ used to group two products p_i and $p_j \in P$ based on the unique identifier of each product, defined by $id(p_j)$. Thus, the similarity function for product clustering, F^{SB}_{Sim}, based on the SB method, can be defined as:

$$F^{SB}_{Sim}(p_i, p_j) = \begin{cases} 1 & \text{if } id(p_i) = id(p_j) \\ 0 & \text{if } id(p_i) \neq id(p_j) \end{cases}$$

In our experiment, the similarity defined by the threshold θ is equal to 1. Two products are considered similar if and only if their identifiers are equal.

3.1.2 Step 2: Match Verification

This step verifies matches between product descriptions and their respective identifiers. To carry out this task, we defined the Algorithm 1, implemented to check the matches of the products within the provided groups. The *GroupProducts* function receives the initial grouping of products $G = \{g_1, \ldots, g_m\}$ defined in step 1 as input.

Upon receiving the product groups as input, our algorithm checks the product matches for each received group and returns two sets with the same number of products per group. The first set represents the groups of intrinsically matched products, while the second set represents those that did not match the initial grouping.

Let $G = \{g_1, g_2, \ldots, g_m\}$ be the set of product groups, the function *GroupProducts* processes

Algorithm 1: Group Products Function.

Input : $G = \{g_1, .., g_m\}$;
Output: $G_{Matches} = \{g'_1, .., g'_m\}$,
$G_{noMatches} = \{g''_1, .., g''_m\}$;

1 **foreach** g *in* G **do**
2 $\quad g.canonDesc \leftarrow findCanonDesc(g)$;
3 **end**
4 $G_{noMatches} \leftarrow \emptyset$;
5 $G_{Matches} \leftarrow copy(G)$;
6 **foreach** g *in* $G_{Matches}$ **do**
7 $\quad P_{noMatches} \leftarrow \emptyset$;
8 $\quad$ **foreach** p *in* $g.products$ **do**
9 $\quad\quad$ **if** *not* $isMatch(p.desc, g.canonDesc)$ **then**
10 $\quad\quad\quad P_{noMatches}.add(p)$;
11 $\quad\quad\quad g.delete(p)$;
12 $\quad$ **end**
13 $\quad G_{noMatches}[g.id].add(P_{noMatches})$;
14 **end**
15 **return** $(G_{Matches}, G_{noMatches})$;

each group $g_i \in G$ and returns two sets $G_{\text{Matches}} = \{g'_1, \ldots, g'_m\}$ and $G_{\text{noMatches}} = \{g''_1, \ldots, g''_m\}$, where G_{Matches} represents the products of group g_i that have intrinsic matches, and $G_{\text{noMatches}}$ represents the products of group g_i that do not have matches in the initial grouping. Thus, for each group $g_i \in G$, it holds that $g_i = g'_i \cup g''_i$ and $g'_i \cap g''_i = \emptyset$, where $g'_i \in G_{\text{Matches}}$ and $g''_i \in G_{\text{noMatches}}$, ensuring that all products are exclusively categorized in one of the two sets, preserving the structure of the initial grouping G. In other words, $G_{\text{Matches}} \cup G_{\text{noMatches}} = G$ and $G_{\text{Matches}} \cap G_{\text{noMatches}} = \emptyset$.

To avoid the $O(N^2)$ complexity in the comparisons of all products in the formed groups, a valid description for each group is initially defined, referred to here as the canonical description (line 2 of Algorithm 1). The matching verification of the products in the group is performed only with this canonical

description, resulting in a complexity of $O(N)$ per grouping. The canonical group description was established through a majority voting approach, whereby we selected the description with the highest number of occurrences. In the case of a tie, when multiple descriptions have the same number of occurrences, a secondary criterion for breaking the tie is proposed, such as choosing the description with the most words or characters.

Once the canonical description of each product group is defined, our algorithm identifies and separates incorrect associations of products, maintaining groups whose products are indeed corresponding (G_{Matches}) and creating groups of non-corresponding products ($G_{\text{noMatches}}$) (lines 6 to 14). This identification of products is carried out through the similarity function $F_{Sim}(p_i, p_j) \geq \theta$, where $\{p_i, p_j\} \in g_i$, p_i is the product that contains the canonical description of group g_i, and θ represents the similarity threshold. Formally, we have:

$$g_i' = \{p_j \mid F_{Sim}(p_i, p_j) \geq \theta\}$$
$$g_i'' = \{p_j \mid F_{Sim}(p_i, p_j) < \theta\},$$

where $g_i' \in G_{\text{Matches}}$ and $g_i'' \in G_{\text{noMatches}}$. Then, these two sets of product groups, G_{Matches} and $G_{\text{noMatches}}$, are returned to the main algorithm (Algorithm 2).

The similarity function $F_{Sim}(p_i, p_j)$ is defined in the function *isMatch*() (line 9, Algorithm 1). The techniques for implementing the function *isMatch*() can explore lexical approaches (Christen, 2008; Konda, 2018) or advanced machine learning techniques (Li et al., 2020; Peeters et al., 2020; de Santana et al., 2023; Primpeli et al., 2019; Barlaug and Gulla, 2021), including Cross-Lingual Learning (Peeters and Bizer, 2022).

3.1.3 Search for Matching Products: Step 3

While step 2 identifies products with invalid matches ($G_{\text{noMatches}}$) in the initial grouping (G), our step 3 aims to associate the products identified as non-matching ($G_{\text{noMatches}}$) with other products that represent the same entity, establishing the matches correctly.

Initially, the products with valid matches (G_{Matches}) are used to create a showcase of indexed products in an Information Retrieval system. Subsequently, the products contained in $G_{\text{noMatches}}$ are used as search keys to find index matches. This search process enables the identification of the most suitable products to make the correct associations.

The process carried out in step 3 can be formally described as follows:

1. Indexing: products $p \in G_{\text{Matches}}$ are indexed in the IR system to enable more efficient retrieval;
2. Searching: for each product $p'' \in G_{\text{noMatches}}$, a search is conducted in the IR system using p'' as the key;
3. Matching: the IR system returns a set of products $\{p_i\}$ for each p'' searched, where $\{p_i\} = \text{findSimilarity}(p'')$; and
4. Linkage: the correct correspondence between p'' and $\{p_i\}$ is determined based on similarity, $F_{Sim}(p'', \{p_i\}) \geq \theta$. The linkage is carried out by considering the highest value of the similarity function F_{Sim}. That is, for each $p'' \in G_{\text{noMatches}}$, p^* is found such that:

   $$p^* = \arg_{\, p_i \in G_{\text{Matches}}} \max F_{Sim}(p'', \{p_i\}).$$

   In this case, the product p^* is the one that maximizes the similarity function F_{Sim} between p'' and the products G_{Matches}.

The function findSimilarity(p'') aims to locate the most suitable products for making the most appropriate associations. For that purpose, two mechanisms for retrieving and classifying relevant documents are used:

1. Search Algorithm:
   - Initially, the search algorithm is used to calculate the relevance of the indexed products ($p_i \in G_{\text{Matches}}$) in relation to the query product ($p'' \in G_{\text{noMatches}}$);
   - The similarity function $F_{Sim}^{find}(p'', p_i)$ is then used to rank the candidate products based on textual similarity.

   The initial search can be formally depicted as:

   $$\{p_i\} = \text{findSimilarity}(p''),$$

   where $p_i \in G_{\text{Matches}}$ e $F_{Sim}^{Search}(p'', p_i) > \theta$
2. Reordering with Cross-Encoder:
   - The reordering is carried out using a cross-encoder language model.
   - The language model evaluates the relevance of the pairs (p'', p_i) more accurately, generating a refined similarity score $F_{Sim}^{\text{Cross-Encoder}}(p'', p_i)$. We calculate the similarity score considering the semantics associated with the product name. Thus, the reordering of the pairs (p'', p_i) is carried out in such a way that $F_{Sim}^{\text{Cross-Encoder}}(p'', p_i)$ is greater than or equal to $F_{Sim}^{\text{Search}}(p'', p_i)$.
   - Formally, this reordering can be represented as:

     $$\{p_i\}_{\text{final}} = \text{Reorder}_{Cross-Encoder}(\{p_i\}, p''),$$

     where $F_{Sim}^{\text{Cross-Encoder}}(p'', p_i) = \text{Cross-Encoder}(p'', p_i)$ and Cross-Encoder represents a language model trained to calculate the similarity between two products.

For each product $p'' \in G_{\text{noMatches}}$, the function findSimilarity(p'') performs an initial search and a subsequent reordering with *cross-encoder*, returning the most relevant products $\{p_i\}_{\text{final}}$ for each product p''.

Figure 3 illustrates this process of searching for corresponding products implemented in *STEPMatch*.

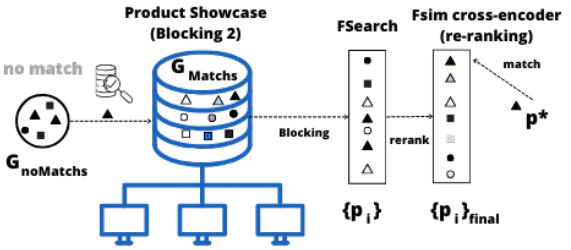

Figure 3: Search for product matches.

We indexed the products in *STEPMatch* using a blocking strategy to avoid the complexity $O(N^2)$ in reordering with *Cross-Encoder*.

The design of step 3 is detailed in the Algorithm 2, where it takes the following parameters as input:

- $G_{\text{noMatches}}$: a set of product groups without matches, identified in the Matching Stage (step 2) using the Algorithm 1; and
- G_{Matches}: a showcase of products matching set by the Algorithm 1. This showcase represents the products that are indexed in the IR system.

Two empty sets are instantiated at line 2 of the Algorithm 2: G_{newMatch} and G_{unknown}. The set G_{newMatch} represents the groups of products for which new matches with products from the showcase were possible, while G_{unknown} represents a set of products for which matches could not be determined. These sets constitute the final result of the algorithm.

The Algorithm 2 iterates over each product in every group of $G_{\text{noMatches}}$ to perform searches within the showcase G_{Matches} (lines 3-18). The function *findSimilarity* (line 6) is responsible for returning a list ordered by relevance, considering the degree of similarity of the searched item $p_j \in G_{\text{noMatches}}$ with the products $p_i \in G_{\text{Matches}}$. The first element of the list, p^*, represents the product p_i with the highest degree of similarity, matching it with the searched product p_j (line 9). The function *isMatch()* (Algorithm 1) is used again to verify if there is indeed a match between p_j and the first element of the list p^* (line 10). Once the matching of the items is confirmed, the product p_j is added to the same group as the p^* element at the top of the search results (lines 11-13). If the function *isMatch*() does not confirm the match, the item p_j is added to the set $Products_{\text{unknown}}$ of unmatched products (line 15). Finally, the sets G_{newMatch}, which include groups of matching products, and G_{unknown}, with unmatched products (line 17), represent the final result of the processing of Algorithm 2 and are returned.

Algorithm 2: Matching Locator.

```
   Input  : G_Matches = {g'_1, .., g'_m},
            G_noMatches = {g''_1, .., g''_m};
   Output: G_newMatch = {g_1, g_2, ..., g_n},
            G_unknown = {g_unknown};
1  G_newMatch ← ∅ ;
2  G_unknown ← ∅ ;
3  foreach g_aux in G_noMatches do
4      Products_unknown ← ∅ ;
5      foreach p_j in g_aux.products do
6          products_result ←
             findSimilarity(p_j, G_Matches);
7          flag_match ← False ;
8          if products_result.size() > 0 then
9              p* = products_result[0];
10             if isMatch(p*.desc, p_j.desc) then
11                 p_j.id ← p*.id;
12                 flag_match = True;
13                 G_newMatch[p_j.id].add(p_j) ;
14         if (not flag_match then
15             Products_unknown.add(p_j)
16     end
17     G_unknown['unknown'].add(Products_unknown);
18 end
19 return (G_newMatch, G_unknown);
```

3.2 Cross-Encoder Model with Cross-Lingual Learning

The STEPMatch uses the similarity function F_{Sim} to perform product matching. We employed this function in both of the presented algorithms. Our approach aims to apply transfer learning, enhancing our model by fine-tuning it with task-specific inputs. For product matching, the model receives two product descriptions P_i and P_j as input, classifying them as Matched ($y = 1$) or Not Matched ($y = 0$). The classification is achieved through a probability $P_m(y_{ij} = 1 \mid (p_i, p_j))$ that indicates the confidence of a match occurring between p_i and p_j. Formally, the output of our model is represented by:

$$\hat{P}_m = f_m(M_{\theta*}(p_i, p_j))$$

$$\hat{y} = \begin{cases} 1 & \text{se } M_{\theta*}(p_i, p_j) \geq \tau \\ 0 & \text{se } M_{\theta*}(p_i, p_j) < \tau, \end{cases}$$

where $\hat{P}_m$ represents the similarity index, quantifying the degree of correspondence or similarity between products p_i and p_j. This index is calculated from the similarity function f_m, which receives the value returned from the softmax activation function used in the output layer of the LLM $M_{\theta*}$. Finally, $\hat{y}$ represents the binary classification (0 or 1) predicted through a threshold τ. By default, the threshold τ is set to 0.5 and may be adjusted according to the desired optimization.

In the context of product matching, our work also contributes to the state of the art by evaluating transfer learning techniques between languages by exploring distinctive CLL strategies. In this approach, several LLMs are evaluated, including both monolingual and multilingual models. Our goal with CLL is to enhance the performance of LLMs by using annotated corpora from a high-resource specific language to build classification models applicable to a different low-resource language through transfer learning. This process revolves around training classification models from a source language and fine-tuning with a smaller portion of data from the target language. This approach allows for using learning models in languages with limited resources, maximizing efficiency and accuracy in the product matching task.

The use of CLL strategies (Pikuliak et al., 2021; De Oliveira et al., 2024) assumes the use of at least two distinct language corpora to develop, with a transfer learning method, where the classification model uses data from a source language D_s to improve product matching classification in a target language D_t. The trained model M_θ^{CLL} is fine-tuned using a combination of data from the corpora D_s and D_t, controlled by the parameters α and β, which adjust the proportion of data from each language in the fine-tuning process. Formally, we have:

$$D = \alpha D_s + \beta D_t,$$

where α and β control the amount of data from the source and target languages, respectively.

Our study used data in English as the source language (D_s) and Brazilian Portuguese as the target language (D_t). Inspired by the works of Alves et al. (2024); De Oliveira et al. (2024), this research explored the combined strategy of Joint-Learning (JL) and Cascade-Learning (CL) in refining the model. The JL technique uses corpora from specific languages during training, including a subset of data from the target language as part of the training corpus. On the other hand, in the CL technique, the model undergoes fine-tuning exclusively using the training language corpora and then further fine-tuning utilizing a subset of the target language data. In the combined strategy, referred to as JL/CL, the trained model underwent two refinements, wherein the first phase, a fraction of 50% of the source language data ($\alpha = 0.5$), as well as 50% of the target language data ($\beta = 0.5$), and then in the second adjustment, we only used the remaining 50% of the target language data ($\beta = 0.5$).

4 EXPERIMENTAL SETUP

This section provides an overview of the dataset and LLMs comprising STEPMatch and details of our experiments.

4.1 Dataset

This work utilizes CLL approaches that adopt a labeled product corpus from a source language to train models capable of evaluating products in a target language. For the source language, we used the WDC Product corpus (Web Data Commons Training and Test Sets for Large-Scale Product Matching), which contains paired product annotations in English and has been used in other product matching studies.Peeters et al. (2020); Primpeli et al. (2019)

For the target language, we used data in Brazilian Portuguese from products derived from Electronic Fiscal Invoices (NFe-BR) issued in a Brazilian state. The data was collected over a three-month period, from May to July 2023, totaling approximately 6.6 million records. The database includes information such as the identification code (GTIN), a short description of the product, and the price. This dataset encompasses many products, accounting for 578,640 distinct barcodes (GTIN) and 942,447 unique descriptions.

To construct the NFe-BR corpus containing pairs of products labeled as “match” and “no match” we adopted a contrastive approach to obtain a diverse and representative set of product pairs, similar to the methodology in Peeters et al. (2020); Embar et al. (2020); de Santana et al. (2023). Positive pairs were formed by grouping products with identical GTINs. We employed the BM25 algorithm via ElasticSearch[1] for negative pairs to find similar product descriptions. For each positive pair, k negative pairs were generated, resulting in a *1:k* ratio. In our experiments, we set *k=5* to create a dataset with a higher proportion of negative instances. Furthermore, a subset of categories was selected, prioritizing those with the highest representation in terms of product quantity.

The WDC Products corpus includes product pairs

[1]https://www.elastic.co/elasticsearch

Table 1: Product Corpora.

	Train		Valid		Test	
Corpora	**Match**	**No Match**	**Match**	**No Match**	**Match**	**No Match**
WDC	1410	5065	352	1267	300	800
NF BR	1419	6.946	281	1511	298	1493

designated for training, validation, and testing. For the NFe-BR, we randomly divided the annotated product pairs into 70%, 15%, and 15% for training, validation, and testing, respectively. Table 1 presents the quantitative details for each corpus.

4.2 Information Retrieval

Our trained models are not limited to classification tasks but can also return the probabilities of match between pairs of products. These probabilities are used as criteria to determine the relevance of a search result in the reordering process. In other words, the higher the probability of a match between a searched product and the retrieved items, the greater the result's relevance for the model.

In our experiments, the function *findSimilarity*(p") of the algorithm 2 was implemented through various approaches, including lexical, semantic, and hybrid search methods. Initially, these techniques were evaluated without our re-ranking method, establishing baselines for comparison with the STEPMatch approach, which presents the reordering using a cross-encoder language model.

For conducting the searches, we indexed the distinct descriptions of the products from the electronic invoices dataset in ElasticSearch, including both the textual descriptions and their vector representations, generated from pre-trained models from the *Sentence-Transformers*[2], which we used to generate semantic embeddings.

In our experiments, the methods were applied to retrieve the top-k products most similar to the item of interest. Next, we describe the search methods used in our work.

4.2.1 Search Methods Without Re-Ranking

To implement the function $F_{Sim}^{Search}(p'', p_i) > \theta$, we explored lexical, semantic, and hybrid approaches. In the lexical search, we used the BM25 algorithm, implemented in Elasticsearch[3]. In the semantic search, the vector search algorithm Approximate Nearest Neighbor (ANN) was applied to identify the products with the highest similarity. This approach effectively navigates the high-dimensional space of document embeddings, identifying the subset of documents most similar to the query based on their cosine distance. We evaluated three embedding models based on SBERT (Reimers and Gurevych, 2019):

- all-MiniLM-L6-v2: Offers high performance and compact embeddings in a dense vector space of 384 dimensions, making it suitable for large-scale query processing;
- LaBSE: With 768-dimensional embeddings, this language-agnostic cross-encoder model supports various languages;
- quora-distilbert-multilingual: With 768-dimensional embeddings, it is designed to work with multiple languages.

The hybrid search approach we adopted in this work was based on the Reciprocal Rank Fusion (RRF) technique (Cormack et al., 2009), which allows for combining the results of different types of queries, such as those retrieved by lexical and semantic approaches, into a single ranking as shown in the following equation:

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

where D is the set of documents to be classified, R is the set of rankings from different information retrieval systems, and $r(d)$ represents the position of document d in ranking r.

4.2.2 Search Methods with Re-Raking

To implement the reordering with the adopted Cross-Encoder, we trained our models to perform the function $F_{Sim}^{\text{Cross-Encoder}}(p'', p_i)$ which evaluates the relevance of the products p_i compared to the product p''. For this, we selected the best methods from the search without re-ranking in section 4.2.1 and carried out the reordering using the BERT-multilingual model trained precisely for product matching.

To evaluate the effectiveness of the Cross-lingual Learning technique, we trained a model based on the LLM BERT-Multilingual. We compared its performance with a baseline, in which the same LLM was trained without the CLL approach, meaning the model was adjusted exclusively with product descriptions in Portuguese.

4.3 Evaluation Metrics

We classified relevant documents as positive, while non-relevant ones were considered negative. Based on this classification, it was possible to calculate the percentage of relevant documents retrieved using the

[2]https://www.sbert.net/

[3]https://www.elastic.co/pt/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables

recall metric. Additionally, the NDCG (Normalized Discounted Cumulative Gain) metric, widely used in information retrieval (IR), was employed to evaluate the effectiveness of search algorithms by considering the relevance of documents and applying a discount factor according to the ranking. This factor reflects user behavior, prioritizing documents with higher rankings, making NDCG an essential quantitative measure for assessing algorithm performance.

5 RESULTS AND DISCUSSION

The experiments in this section aim to evaluate the search mechanism implemented by the function *findSimilarity()*, presented in step 3. The main focus is to analyze the relevance of search results in identifying corresponding products in an information retrieval environment.

Our objective with this analysis is to evaluate the search methods that retrieve relevant items for a specific product from the test corpus of electronic invoices. An item returned in a search is seen as relevant when it has the same GTIN as the searched item, even if it has alternative descriptions.

The order of the relevant retrieved items is not crucial in searching for corresponding products, as all descriptions refer to the same product, represented by the same GTIN. What is most important is that all variations of the product description are present at the beginning of the search results, regardless of their ranking position.

For example, if the product ”Skim Milk XYZ” is registered in the system with three distinct descriptions: ”Skim Milk XYZ 1L”, ”XYZ Skim Milk 1000ml”, and ”Skim Milk 1L”, we consider all these descriptions relevant. Thus, it does not matter if ”XYZ Skim Milk 1000ml” appears in the first position and ”Skim Milk XYZ 1L” in the third; the main point is that both descriptions are retrieved as variations of the same product.

This way, the information retrieval system searched for each item in the test set, computing the evaluation metrics for the Top 500 items returned by the search methods. These results were analyzed considering the average of all the queries made.

We evaluated the results using recall and Normalized Discounted Cumulative Gain (NDCG) metrics, which were considered appropriate for the product matching context. Recall assesses the system’s ability to retrieve all relevant matches for a given query, where, in this context, a high recall value indicates that the model was able to recover most of the relevant products from the dataset. In contrast, NDCG measures the quality of the ranking of the retrieved documents, assigning higher scores to the most relevant documents located at the top of the results list.

The Figures 4 and 5 show the results of the experiments conducted with lexical and semantic searches for the function $F_{Sim}^{Search}(p'', p_i) > \theta$. The lexical search is labeled as ”bm25,” while the semantic searches, based on the vectors generated by the models all-MiniLM-L6-v2, LaBSE, and quora-distilbert-multilingual, are labeled as ”semantic_all_minilm,” ”semantic_labase,” and ”semantic_quora,” respectively. Among the approaches tested, ”bm25” and ”semantic_all_minilm” stood out with the best recall and NDCG metrics, indicating that these methods retrieved more relevant documents and positioned them more accurately in the top ranks. These results encouraged the development of a hybrid search, combining the features of both approaches.

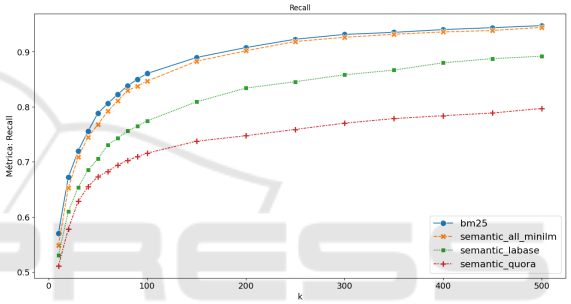

Figure 4: Recall for Lexical and Semantic search methods.

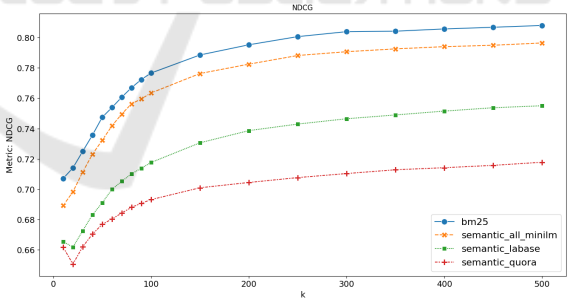

Figure 5: NDCG for Lexical and Semantic search methods.

The Figures 6 and 7 show the results after the introduction of hybrid search (“bm25_all_minilm”), through the combination of lexical (“bm25”) and semantic (“semantic_all_minilm”) methods using the RRF technique. We observed that the recall achieved by the hybrid method is comparable to that of the best lexical and semantic methods, demonstrating the effectiveness of combining the approaches for retrieving relevant items. However, the NDCG was lower, indicating that the relevance of the retrieved items was inferior to that of the other methods.

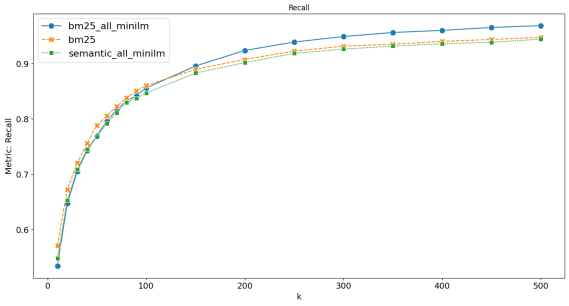

Figure 6: Recall including a hybrid approach.

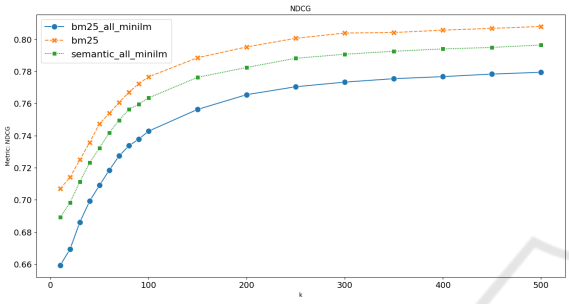

Figure 7: NDCG including a hybrid approach.

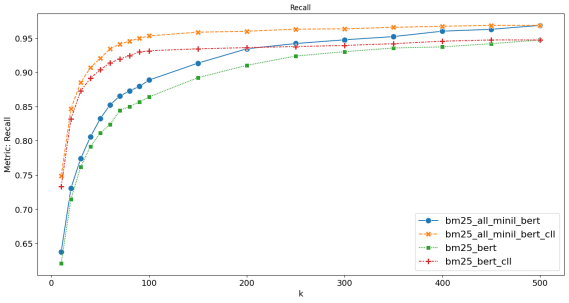

Figure 8: Recall with re-ranking using BERT.

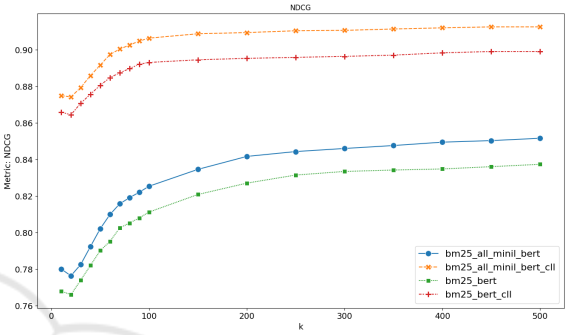

Figure 9: NDCG with re-ranking using BERT.

For the implementation of the function $F_{Sim}^{\text{Cross-Encoder}}(p'', p_i)$ we trained a model from the *BERT* family, specifically the *BERT-Multilingual*, which calculates a similarity value between the product p'' and p_i, where p_i represents each element retrieved in the initial search (lexical or semantic). For comparison, we used a baseline model trained exclusively with data from Brazilian electronic invoices and a second model trained using the CLL approach. Table 2 presents the metrics *f1-score*, *recall*, and *precision* of the models for the similarity classification task. We used a bootstrapping strategy, training and evaluating the model over ten repetitions to estimate its uncertainty. The results in the table indicate that the model trained with CLL outperformed the baseline.

Table 2: Baseline vs. CLL - Scores for BERT-Multilingual Models Trained Mean Value and Standardized Error (95% Confidence Level) Calculated from 10 Samples.

Strategy	F1	Recall	Precision
baseline (without CLL)	94.3 ±0.0053	94.2 ±0.0049	94.3 ±0.0052
CLL JL 50%+CL 50%	**98.6** ±0.0064	**98.2** ±0.0060	**98.9** ±0.0062

With the trained models and evaluated search methods, we implemented the *findSimilarity*(p") function with reordering applied by the models. The results, presented in Figures 8 and 9, demonstrate an improvement both in the number of retrieved items and the quality of the ranking when using reordering with CLL, compared to reordering with the baseline model.

The results show that the reordering strategy using a cross-encoder model trained with CLL demonstrated superior performance compared to traditional information retrieval (IR) approaches, such as lexical, semantic, and hybrid searches. This advantage arises from using the cross-encoder, which compares pairs of descriptions more accurately, enabling a more contextualized and detailed assessment of the similarity between products. Furthermore, transfer learning, made possible by data from products annotated in another language, improved the performance of our adopted model.

From Figure 10, it is possible to compare the NDCG obtained by the different search approaches evaluated in this study. The strategies that use reordering with cross-encoder models, adjusted with product data, stand out from the others, retrieving relevant items more accurately. These results mainly highlight the potential of the reordering strategy with Cross-lingual Learning to improve the retrieval of relevant products in product matching applications.

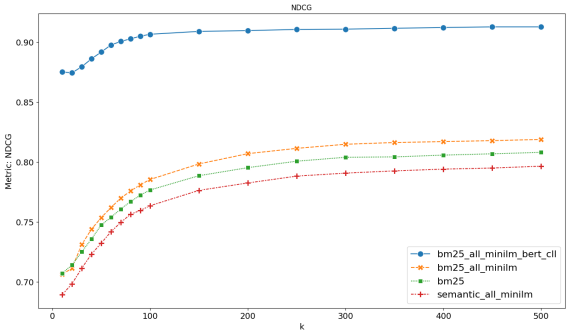

Figure 10: NDCG comparison of the best approaches.

6 CONCLUSION

Our work proposed an approach to product matching in short descriptions, focusing on electronic invoices issued in Brazil. The scenario is characterized by short, unstructured, and often inconsistent descriptions, making product matching challenging. To tackle these challenges, we developed the STEPMatch approach (Short Text Product Matching), integrating information retrieval techniques and supervised machine learning, aiming for effective matching of products in the context of invoice product data. The proposed approach promotes integrating and enriching product data from diverse sources to provide consistent information to support management processes that depend on accurate product data.

We used machine learning techniques in an Information Retrieval (IR) environment to search for matching products. Initially, we apply lexical search techniques, such as the BM25 algorithm, in conjunction with semantic searches to retrieve a set of candidate products. Subsequently, a cross-encoder language model, trained specifically for the product matching task, reorders these candidates, prioritizing the matching products at the top of the list.

The main contribution of this work was the use of Large Language Models with Cross-lingual Learning strategies, which improved the relevance of the items retrieved in the search for corresponding products. The research demonstrated the effectiveness of model adjustment in scenarios with scarce annotated data. The experiments revealed that a model trained with the JL/CLL strategy, initially with 50% of the training data from products in English and Portuguese and then adjusted with the remaining 50% of the data in Portuguese, outperformed the reference model, which was trained exclusively with data in Portuguese. This experiment confirmed the capability of CLL to promote model generalization across different languages and domains, optimizing classification and retrieval methods for corresponding products. Furthermore, the techniques applied for reordering search results surpassed traditional approaches for this application.

We intend to evaluate other CLL strategies for model adjustment for future work, exploring different LLMs, languages, and product categories. Additionally, we aim to explore hybrid search techniques further to enhance accuracy and effectiveness in product matching. Lastly, we plan to make a direct comparison of STEPMatch with state-of-the-art approaches in the field of Entity Resolution.

ACKNOWLEDGEMENTS

The authors would like to thank the Brazilian National Council for Scientific and Technological Development (CNPq) for partially funding this research.

REFERENCES

Alves, A. L. F., Baptista, C. d. S., Barbosa, L., and Araujo, C. B. M. (2024). Cross-lingual learning strategies for improving product matching quality. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, page 313–320, New York, NY, USA. Association for Computing Machinery.

Barbosa, L. (2019). Learning representations of Web entities for entity resolution. *International Journal of Web Information Systems*, 15(3):346–358.

Barlaug, N. and Gulla, J. A. (2021). Neural networks for entity matching: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(3):1–37.

Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 39–48, New York, NY, USA. Association for Computing Machinery.

Christen, P. (2008). Febrl: A freely available record linkage system with a graphical user interface. In *Proceedings of the Second Australasian Workshop on Health Data and Knowledge Management - Volume 80*, HDKM '08, page 17–25, AUS. Australian Computer Society, Inc.

Christen, P. (2012). Data matching systems. In *Data Matching*, pages 229–242. Springer Berlin Heidelberg.

Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., and Stefanidis, K. (2020). An overview of end-to-end entity resolution for big data. *ACM Computing Surveys*, 53(6):1–42.

Cormack, G. V., Clarke, C. L. A., and Buettcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of*

the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

De Oliveira, A. B., Baptista, C. d. S., Firmino, A. A., and De Paiva, A. C. (2024). A large language model approach to detect hate speech in political discourse using multiple language corpora. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, page 1461–1468, New York, NY, USA. Association for Computing Machinery.

de Santana, M. A., de Souza Baptista, C., Alves, A. L. F., Firmino, A. A., da Silva Januário, G., and da Silva Caldera, R. W. (2023). Using machine learning and NLP for the product matching problem. In *Intelligent Sustainable Systems*, pages 439–448. Springer Nature Singapore.

Ebraheem, M., Thirumuruganathan, S., Joty, S. R., Ouzzani, M., and Tang, N. (2017). Deeper - deep entity resolution. *CoRR*, abs/1710.00597.

Embar, V., Sisman, B., Wei, H., Dong, X. L., Faloutsos, C., and Getoor, L. (2020). Contrastive entity linkage: Mining variational attributes from large catalogs for entity linkage. In *Automated Knowledge Base Construction*.

Gözükara, F. and Özel, S. A. (2021). An incremental hierarchical clustering based system for record linkage in e-commerce domain. *The Computer Journal*, 66(3):581–602.

Hambarde, K. A. and Proença, H. (2023). Information retrieval: Recent advances and beyond. *IEEE Access*, 11:76581–76604.

Han, J., Pei, J., and Tong, H., editors (2023). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, fourth edition edition.

Konda, P. V. (2018). *Magellan: Toward building entity matching management systems*. The University of Wisconsin-Madison.

Köpcke, H., Thor, A., and Rahm, E. (2010). Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493.

Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1):50–60.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep Learning for Entity Matching. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34, New York, NY, USA. ACM.

Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. (2021). Blocking and Filtering Techniques for Entity Resolution. *ACM Computing Surveys*, 53(2):1–42.

Peeters, R. and Bizer, C. (2022). Supervised contrastive learning for product matching. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 248–251, New York, NY, USA. Association for Computing Machinery.

Peeters, R., Bizer, C., and Glavaš, G. (2020). Intermediate training of bert for product matching. *small*, 745(722):2–112.

Pikuliak, M., Šimko, M., and Bielikova, M. (2021). Cross-lingual learning for text processing: A survey. *Expert Systems with Applications*, 165:113765.

Primpeli, A., Peeters, R., and Bizer, C. (2019). The WDC training dataset and gold standard for large-scale product matching. In *Companion Proceedings of The 2019 World Wide Web Conference*. ACM.

Rateria, S. and Singh, S. (2024). Transparent, low resource, and context-aware information retrieval from a closed domain knowledge base. *IEEE Access*, 12:44233–44243.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

Ristoski, P., Petrovski, P., Mika, P., and Paulheim, H. (2018). A machine learning approach for product matching and categorization. *Semantic web*, 9(5):707–728.

Tracz, J., Wójcik, P. I., Jasinska-Kobus, K., Belluzzo, R., Mroczkowski, R., and Gawlik, I. (2020). BERT-based similarity learning for product matching. *Proceedings of Workshop on Natural Language Processing in E-Commerce*, pages 66–75.

Traeger, L., Behrend, A., and Karabatis, G. (2024). Scoping: Towards streamlined entity collections for multi-sourced entity resolution with self-supervised agents. In *Proceedings of the 26th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, pages 107–115. INSTICC, SciTePress.

Xiao, C., Wang, W., Lin, X., Yu, J. X., and Wang, G. (2011). Efficient similarity joins for near-duplicate detection. *ACM Trans. Database Syst.*, 36(3).