Contextual kNN Ensemble Retrieval Approach for Semantic Postal Address Matching

El Moundir Faraoun[1,2], Nédra Mellouli[1,3], Stéphane Millot[2] and Myriam Lamolle[1]

[1]LIASD Paris 8 University, 2 rue de la liberté, Saint-Denis, France
[2]TEDIES, TALK solutions, 45 Av. de Paris, Monéteau, France
[3]ESILV DVRC, Léonard de Vinci group,12 Av. Léonard de Vinci Paris La défense, France.

Abstract

The biggest challenge today regarding courier services (delivery of small to medium-sized parcels) is the problem of *Address Matching*. With the expansion of geographical data and the diversity of formats in which it is received, traditional matching methods are becoming increasingly obsolete due to the lack of conformity of delivery information with postal address writing standards. These new constraints are affecting parcel delivery quality in terms of deliverables, cost and environmental impact. This research focuses on courier delivery data (i.e. postal addresses of recipients) in the context of matching French postal addresses. We introduce a new ensemble retrieval approach to the problem through a voting system leveraging multiple k-Nearest Neighbors search algorithms, called kNN-vote which effectively transform the *Address Matching* task to an *Address Retrieval* task. kNN-vote returns the top best normalized addresses similar to a given query (a non-normalized delivery address). The system takes advantage of several address representations, in particular Pre-trained Transformers-Based Sentence Embeddings. The system has been tested on a real database of French delivery addresses. The method meets high expectations, returning exactly matched addresses with a success rate of up to 96% in top 10 as well as 86% in top 1.

Keywords

Address matching or transport entity alignment, Recipients/consignees identification or pairing, Recovery of recipients, Address retrieval, Ensemble kNN retrieval models.

1. Introduction

The transport *Entity Alignment* problem, also known as the postal *Address Matching* (AM) problem is inherently an NLP task given that a postal address is mainly structured as a short sentence with a specific arrangement of Named Entities (i.e. attributes or features like Road Name or Door Number) which makes it fall within the scope of *Entity Matching* (EM). The task involves effectively processing and comparing structural components of a pair of addresses (a, b) for accurate matching (i.e. a and b refer to the same real world object).

Carriers identify delivery addresses received via EDI (Electronic Data Interchange) by matching them with recipient addresses already registered in their database. Nothing could be more simple at first glance, except that delivery addresses are increasingly received in non normalized forms. The addresses received are often incorrect and/or noisy, thus identifying a valid address from an invalid one becomes a very challenging task. The anomalies present in a delivery address can be: (1) Writing errors, including typographic ones, spelling mistakes, repetition, or the absence of specific address features ; (2) Address noise may involve personal information, such as names, phone numbers, or requests for appointments ; (3) Lastly, Semantic or contextual errors include the presence of features from unrelated addresses, feature replacements (e.g. "avenue" instead of "street"), feature aliases, like abbreviations or acronyms as well as polysemous[1] features and finally addresses represented by their semantic synonyms, specifically named zones or parks.

IAL@ECML-PKDD'24: 8th Intl. Worksh. & Tutorial on Interactive Adaptive Learning, Sep. 9th, 2024, Vilnius, Lithuania
el.moundir.faraoun@gmail.com (E. M. Faraoun); n.mellouli@iut.univ-paris8.fr (N. Mellouli); Stephane.millot@edies.fr (S. Millot); m.lamolle@iut.univ-paris8.fr (M. Lamolle)

[1]Polysemy in postal addresses: a single element with multiple related meanings, associated with different locations or recipients.

Let's take for example, the following real delivery address received by a French Carrier: **"avenue du g n ral leclerc centre commercial auchan 89200 avallon"**. Here the correct Road Type is **"rue"** instead of **"avenue"** and the typographic error in **"g n ral"** is intended as **"general"**. Not to mention the absence of a Door Number in the address. We finally note that **"centre commercial auchan"** is a semantic synonym for the address. These anomalies distort the structure of an address and prevent it from being paired with a valid address record.

The AM problem is traditionally solved with a binary "Match/No Match" classification of address pairs [1] mainly relying on neural network-based methods [2, 3, 4, 5, 1, 6]; yet, the task itself is imagined in a scenario of matching address records between two tables or deduplicating records in a data table. However, in the context of delivery, this correspondence is a search for information similar to a given request (address received). Thus, we are dealing with an unsupervised *Information Retrieval* (IR) problem in which each new address is treated as a query, which may be valid or incorrectly formatted, and for which we try to find valid "candidate" addresses in the database. This formalization is very relevant since it allows retrieved candidates for a delivery address to be sorted in terms of contextual similarity. Furthermore, the number of "candidate" addresses is relatively small, reducing the computation time if all reference address records have to be aligned.

Our objective in this research is to take advantage of the various possible representations of addresses, in particular Transformer-Based Sentence Embeddings in the context of Information Retrieval. We propose an ensemble multi-embeddings models approach based on the k-Nearest Neighbors algorithm (kNN) [7], with a voting process between multiple kNN search models.

The remainder of this paper is organized into 7 sections. Section 2 reviews the work carried out in relation to address matching. Section 3 formalizes the Address Retrieval problem. We describe our approach in Section 4 and present its experimental settings in Section 5. Results are detailed and discussed in Section 6. We conclude this work by considering its limitations and prospects for improvement in Section 7.

2. Related Work and State-of-the-Art

The existing solutions for *Address Matching* can be summed up in two approaches. An approach based on string similarity measures or matching rules [8, 9]. However, the problem remains that these methods are based mainly on structural comparisons between addresses, and they quickly become obsolete when faced with addresses that are written differently but retain the same semantic meaning [4]. In fact, textual similarity distances such as Levenshtein and others [10, 11, 12, 13, 14], are used for address matching. These distances depend on the choice of a similarity threshold, which is generally high. This score remains very approximate and eliminates the possibility of matching pairs below the chosen threshold. Other methods are based on decision tree matching rules [9]. These methods improve the matching performance but require systematic calibration of the rules by experts due to the diversity of address writing models.

A second approach based on machine learning (ML) or deep learning architecture (DL) aims to learn the semantic similarity between addresses [2, 3, 4, 5]. These methods mainly rely on vector representations of address elements such as Word2Vec [15] or FastText [16] to use them as input data of ML (e.g. Random Forest, XGBoost) or DL inference models (e.g. ESIM [4], ABLC [5]) for classification. However, in order to get those word embeddings, a parsing step is needed, which is the process of segmenting addresses into their essential features or elements (e.g. Road Number, Road Name or Postal Code). Various parsing techniques were used for this task. For instance, the latter studies used respectively CRFs [17], heuristic rules [3], Jieba[2] algorithm and the Trie syntax tree algorithm. Nonetheless, these methods often come short in the proper parsing of noisy erroneous addresses. Furthermore, the lack of context between words in an address due to the static nature of word embeddings suggests that these methods may fail to match certain ambiguous addresses, such as synonymous or polysemous ones [1], and addresses that are too distorted by noise and errors.

[2]https://github.com/fxsjy/jieba

Recently, the advent of pre-trained transformer encoders [18], like Roberta [19], has transformed various tasks by introducing hyper-contextualized word embeddings [1]. This breakthrough has enabled the achievement of state-of-the-art performances through fine-tuning these encoders for specific tasks, particularly in *Entity Matching* [20, 21]. In the context of *Address Matching*, a model named *GeoRoberta* [1] is proposed. It involves a generation of geographical knowledge for addresses by fine-tuning a Roberta encoder for the task of address features tags detection. It also allows to obtain a textual encoding of GoogleMaps API[3] geographical coordinates of addresses based on Geohash[4]. It is worth to know that *GeoRoberta*, is based on a pre-trained Roberta encoder as well. It generates augmented contextualized embeddings for an address pair by combining at input, elements of both addresses and their Geohash encodings. The output embeddings are fused afterwards with a second augmented pair of addresses, by combining the feature tags embeddings and their Geohash tag embeddings. This final fused representation is fed into a matching classification layer for the address matching task. The approach integrates textual and geographical data, leveraging the power of pre-trained transformers which allows the matching of polysemous and synonymous addresses more efficiently. However, the generation of Geohash coordinates is based on Google geocoding, which is likely to be wrong for certain ambiguous or excessively erroneous addresses.

We argue that the use of sentence embeddings to represent addresses in the context of similar information retrieval is much more adapted in terms of representation quality [6]. This type of representation uses the training of Trasformer-Based Bi-Encoders [22] for the *Semantic Textual Similarity* (STS) task. It succeeds in reducing the distance between two addresses in a latent space even though they have different expressions. Moreover it solves the problem of synonymous addresses and allows the resolution of *Address Matching* through *Information Retrieval* algorithms [7]. Such a solution was introduced in [6] by fine-tuning a DistilBert [23] Bi-Encoder on address pairs and utilize it for query top "Candidate" addresses retrieving after which a fine-tuned Cross-Encoder for address pair classification is used as a top candidates Re-ranker. To take this idea further, we propose several types of representations, vectors (sentence and word embeddings) and raw (textual address content). Giving rise to several lists of k normalized addresses candidates via the Ensemble kNN algorithms, we propose to finally re-rank them through a vote based on the maximum number of appearances of a given candidate (i.e. Term Frequency) among the ensemble kNN models.

3. Address Retrieval formalization

In this section, we introduce the address structure and we define its schema allowing us to formalize the *Address retrieval* (AR) problem. We are focusing on French reference addresses and considering only the French address features. Therefor, the correct structure of any address is the one that follows the official representation model[5] of French postal addresses, namely any address that contains the basic features of the latter which makes it possible to precisely identify the geographical point of the recipient. The features of a correct French address are described in Fig. 1.

3.1. Address model

Address structure definition: Let be a set of vocabulary V, which includes all permissible instances of possible features in a given address. For example, "avenue" might be an instance of the feature *RoadType*. We define D_{norm} as the set of all correctly structured, normalized address sentences. A normalized address would follow in this case the official model of a French address.

Within D_{norm}, there exits a reference set D_{ref} such that, $\forall\ a \in D_{ref}$, a is both normalized and corresponds to an actual real-world location. Thus D_{ref} is the set of all normalized valid addresses with a real geographical point.

[3]https://developers.google.com/maps/documentation/geocoding?hl=fr
[4]http://geohash.org/
[5]https://www.upu.int/

Feature	Example
DoorNumber	1
RoadType	rue
RoadName	marcel brault
PostalCode	89110
CityName	montholon

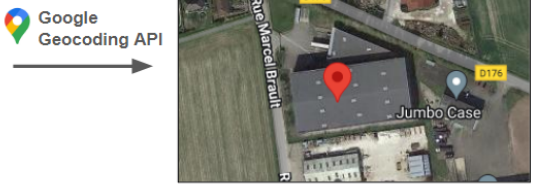


Figure 1: French Postal Address Features

An address model $\mathcal{M}$ is defined by a structure function $f_{\preceq}(\cdot)$. This function takes a sequence of elements from the vocabulary V and produces a normalized address in D_{norm}. Specifically: $f_{\preceq}(\cdot) : V^n \rightarrow D_{norm}$ where n can be 4 or 5, representing the number of components in a address and $n = 4$ being the special case where an address doesn't need a *RoadType*. The components $x_1, ..., x_n \in V$ must satisfy:

- x_1 is an instance of *DoorNumber*,
- x_n is an instance of *CityName*,
- $\preceq$ is a partial order relation defined on V, that we denote $(V, \preceq)$ such that for any $1 \leq i < j \leq n$, $\exists\ (x_i, x_j) \in V \times V$, and $x_i \preceq x_j$,
- $f_{\preceq}(x_1, ..., x_n) \mapsto a \in D_{norm}$ such that $a = x_1 x_2 ... x_n$.

With the following formalization framework, $f_{\preceq}(\cdot)$ could be assumed as a grammar allowing us to generate address sentences that are syntactically and semantically correct. Moreover, if $a \in D_{ref}$, a is a normalized address with a real-world location.

The latter definition allows us to consider any address that follows the address model $\mathcal{M}$ as normalized. That being said, an address can be normalized but nonexistent. The following examples of French addresses illustrate this point:

- (i) "16 avenue jean jaures 89000 auxerre" is a normalized existing address.
- (ii) "16 **rue** jean jaures **89300 joigny**" is a normalized address but nonexistent.

Although the second address is technically correct in its structure, a simple anomaly like features instance replacement of *RoadType*, *PostalCode* and *CityName* makes it not corresponding to a real location. In such a case, the ensemble kNN multi-embeddings models are interesting since the address semantic context is considered.

3.2. Address retrieval

Now that the address structure formalism is defined, through the lexicographic order relation defined on the feature instances of an address, we assume in the following of our work that an address is simply a structured sentence with a particular context (a.k.a an address sentence). We define the problem of Address Retrieval as a problem of semantic search of textual documents.

Address embedding definition: Let a be an address sentence. Given a textual encoders E, An address representation is defined as the output of E where a is the input. We define E_0 as id(·) (i.e. the identity function) and therefor, an address representation can be:

- raw (i.e. the textual content of the address itself) through E_0
- a vector embedding through a neural encoder E.

In the rest of the paper, and for the sake of simplicity, we refer to the ensemble of raw and vector model embeddings as multi-embeddings models.

Contextual kNN Address Retrieval task: We want to obtain for a given address query q and through an encoder E, a query representation $e_q \in \mathcal{X}_E$ for kNN retrieval. The Neighborhood of e_q is then constructed by fetching its k nearest neighbors from a set of reference address sentence representations $\mathcal{X}_{D_{ref}} \subset \mathcal{X}_E$ according to a distance function $d(\cdot) : \mathcal{X}^2_{D_{ref}} \to \mathbb{R}$.

More formally, the k nearest neighbors of e_q can be obtained by:

$$\mathcal{K} := \{i_1, i_2, \ldots, i_k \mid d(e_q, e_{i_j}) \text{ are the } k \text{ smallest distances}, \ i_j \in [|\mathcal{X}_{D_{ref}}|]\} \tag{1}$$

where $\mathcal{K}$ denotes the set of indices in $[|\mathcal{X}_{D_{ref}}|] = \{1, ..., |\mathcal{X}_{D_{ref}}|\}$, which points to k neighbors with the smallest distances close to 0.

Although a distance d depends on the fixed encoder used for an address representation, our kNN retrieval model is still generic. For example, if E is a Transformer-Based Bi-Encoder model then the distance d would be a $cosine$-like distance. Roughly speaking, our kNN model have three parameters: the k, the E representation and the distance d [24, 25, 26].

4. Our Approach

Ensemble voting for multi-embeddings kNN models is a robust technique that exploits the strengths of different embedding methods to improve prediction accuracy. By generating multiple embeddings for the same data and combining the predictions of multiple kNN models through voting, we can achieve better performance and more reliable results. This approach is particularly useful for our task in which different embeddings capture different aspects of the addresses. In order to perform the task of correct address retrieval, we had to undergo the subsequent steps: (1) Data pre-processing and deduplication for both delivery and reference addresses, (2) Offline fine-tuning of different Bi-Encoders on the STS task in order to construct multiple retrieval sets of normalized address embeddings, (3) kNN retrieval models construction (see Fig. 2) and (4) Online aggregating of the different search results through the design of a vote schema (see Fig. 3).

4.1. Data pre-processing

Before fine-tuning the Bi-Encoders, it was necessary to go through two word pre-processing steps then a deduplication step:

- The first step is the cleaning of both delivery and reference addresses and it involves removing accents and punctuation that might be present in data.
- The second step concerns the removal of interfering elements. This step is only applied to the delivery addresses given that all reference addresses are supposed to be correct and normalized. This step removes a set of unnecessary symbols that can be found in non-normalized addresses (e.g. '+', '*', '&', … etc.).
- The third step is the deduplication of delivery address records. By removing these exact duplicates, we ensured that our fine-tuning process was efficient and not biased by redundant data points.

The final step is dataset creation for the Bi-Encoders fine-tuning. This step includes another cleaning process we explained in details in 5.1.

4.2. Offline fine-tuning of Bi-Encoders

Here, we have as an input a set of (delivery, reference) address pairs. The aim of this step is to fine-tune multiple bi-encoders to generate the address sentence vector embeddings.

4.2.1. Bi-Encoder

Bi-Encoders are Siamese Transformers Networks generally fine-tuned on *Semantic Textual Similarity* tasks for the purpose of generating meaningful sentence embeddings. Typically, a pre-trained transformer model is first chosen as the training base of the Bi-Encoder. We use two types of pre-trained models:

- "Camembert-base" [27] , a model specific to the French language,
- "XLM-Roberta-base" [28], a multilingual model,

which we both adapted on a large corpus of French postal addresses by continuing their training on the Masked Language Modeling (MLM) task. We also used the MLM objective to train a third small Roberta-based model [19] from scratch on the same corpus.

Given an address sentence pair (a, b), a forward pass of the transformer over each tokenized address generates token embeddings for both a and b. Mean pooling is then applied on each address token representations resulting in two fixed length vectors which will be our address sentence embeddings. Considering a specific STS task, the best semantic address matching performance is found through the optimization of an objective function such as the "contrastive loss" [29] which is used mainly in neural networks for classification and matching tasks, such as similarity learning. It is often used in Siamese networks to train models to learn similar representations for pairs of similar samples and dissimilar representations for pairs of dissimilar samples. Readers interested in exploring the Bi-Encoder architecture can refer to [22].

In our case, our sentences are postal addresses that are no more than a few words long. In addition, all addresses have, more-or-less, the same vocabulary that repeats itself, such as road types or city names. All this reduces the diversity of context between dissimilar addresses. This constraint led us to believe that using a basic objective function would not succeed in creating a sufficient gap in terms of distance between dissimilar addresses. To overcome this, we decided to use the "Multiple Negative Ranking Loss" (MNLR) objective function [30], which is often used in the context of ranking and information retrieval tasks and therefore more suited to our similarity search task. This approach is supported by findings in [31] which highlights that including multiple negatives in each batch enhances the model's ability to distinguish between dissimilar examples without the need to specifically design hard negative pairs. Finding truly effective negative examples can be challenging and significantly impact the performance, making MNRL's ability to utilize multiple negatives in a straightforward manner highly advantageous which leads to better performance and more robust embeddings.

Multiple Negative Ranking Loss definition: For a given N address sentence embeddings pairs $[(e_{a_1}, e_{b_1}), ..., (e_{a_N}, e_{b_N})]$ between **query-reference** address sentences $(a_1, ..., a_N)$ and $(b_1, ..., b_N)$ where (a_i, b_i) are labeled as similar, and (a_i, b_j) where $i \neq j$ are labeled as not similar. The loss function is as follows:

$$-\frac{1}{N}\sum_{i=1}^{N}\left[S(e_{a_i}, e_{b_i}) - log\sum_{j=1}^{N} e^{S(e_{a_i}, e_{b_j})}\right] \tag{2}$$

This function allows the model to consider in a given batch of positive address pairs, for one sample (a_i, b_i), using all the normalized reference addresses b_j in the other positive pairs, $N-1$ negative pairs (a_i, b_j). This strategy helps the model to widen the distance between negative examples a_i, b_j where S is the score function (Generally $S(e_{a_i}, e_{b_i}) = cosine(e_{a_i}, e_{b_i})$). This loss function helps reducing the impact of the lack of context in the addresses.

4.2.2. Retrieval set creation

Having a dataset of normalized reference address sentences D_{ref} and a fine-tuned Bi-Encoder E, we can generate a retrieval sentence embedding set $\mathcal{X}_{D_{ref}}$ through a forward pass over all address instances of D_{ref}. This embedding set would be later used at inference time for the retrieval of a given query nearest neighbors.

4.3. kNN retrieval models

kNN-vote is an Ensemble Information Retrieval system based on the search results of multi kNN models, all similar in their operations but very different in their basis of representations of the searched addresses. In general, an individual kNN search model is a k-Nearest Neighbor algorithm which takes as a parameter a distance d specific to the type of representation of the searched points (e.g. a Levenshtein or Jaccard distance for a raw textual representation). The algorithm computes all the distances between a query and the search points previously pre-registered in the retrieval reference set $\mathcal{X}_{D_{ref}}$ ($\mathcal{X}_{D_{ref}} = D_{ref}$ for raw textual representation) and returns the list of the k most similar points having the smallest distance with the query. Table 1 shows the different combinations of (encodings, similarities) that can be used in a kNN search model (kNN Retriever) within the voting system. The table illustrate the possible types of address representation previously mentioned in Section 3.2, That is, the raw textual representation through which we will have different kNN search models each with a well-defined type of string distance (see Table 1); and (2) vector representation divided into two types:

- traditional embeddings built by way of mean pooling the static word embeddings of address elements such as Word2Vec,
- contextual sentence embeddings, fine-tuned for textual similarity, of postal addresses.

Table 1
Different Combinations used for kNN Retriever

Representation	Encoding	Similarity
Raw	Textual content	Jaro, Jaro-Winkle, Levenshtein, Jaccard, Damerau-Levenshtein, Ratio, Token set ratio, Token sort ratio, Partial ratio, Set ratio, Seq ratio
Vector	Csent (Camembert Bi-Encoder) XLMsent (Xlm Roberta Bi-Encoder) Rsent (Roberta custom Bi-Encoder) wvavg (Word2Vec word embeddings averaged) ftavg (fastText word embeddings averaged)	Cosinus Euclidean Correlation Cityblock

Without any a priori hypotheses about the origin of the errors, we have carried out an empirical search for the best address representation spaces with the appropriate similarity measures. We simply applied the various representations and similarity measures in the literature and compared eleven string similarity measures for the raw representations and four vector similarity measures for the static and dynamic embedding. representations (see Table 1). Some of the string similarity measures, such as "Ratio" or "Token_set_ratio," are taken from the fuzzywuzzy library[6] as they enable more robust and flexible comparisons by incorporating tokenization and sorting mechanisms. Unlike traditional metrics like Levenshtein and Jaro, which focus solely on character-level edits, fuzzywuzzy's methods account for word order and partial matches, making them more suitable for real-world text data. The chosen sentence embedding models are considered (see Section 4.2), hence we had a total of 31 kNN models. The advantage here is to allow us to have a maximum of individual candidate lists of retrieved addresses in order to compare, firstly, the performance of each kNN Retriever model and, secondly, use them to draw the candidates in common between the lists as the most similar candidates. Fig. 2 shows the architecture of a single kNN Retriever. We finally define the similarity search process as follows: (1) we convert a query q by the desired representation type to have e_q; (2) e_q is then passed into the kNN Retriever which will be responsible for computing the distances between e_q and all the representations

[6]https://github.com/seatgeek/fuzzywuzzy

in the retrieval set in order to return the k address indices most similar to q ranked according to the smallest distance.

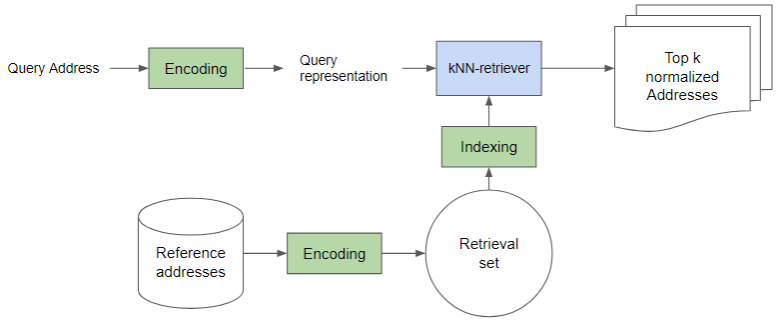


Figure 2: kNN Search Model Architecture.

4.4. Ensemble voting retrieval system

The system is termed "multi-embeddings models" due to its dual approach, leveraging both raw address representations and advanced deep learning (DL) techniques for vector text representations in address matching. The core functionality of the system involves returning a final list of m similar candidate addresses through a voting process. Among the kNN ensemble models, the voting process is based on the maximum number of occurrences of a candidate address for a given query. It should be noted that this system needs two types of important values: (1) the number of repetitions of each candidate address i in the different k-lists; (2) the different similarity scores of a pair (q,i) for which i appeared with different kNN models.

4.4.1. Retrieval Flow

1. Candidate address lists retrieval: The system begins by retrieving the k-lists of candidate addresses using the ensemble kNN retrieving pipeline. Each model in the ensemble provides a list of address indices for a given query.
2. Voting process:
    - *Repetition counting*: The first step in the voting process is to count the number of repetitions of each candidate address i across the different k-lists.
    - *Grouping and sorting*: Candidates indices are then grouped based on their repetition counts. This creates "bags" of indices, where each bag contains one or more indices pointing to associated addresses. Then the bags are sorted by the maximum number of repetitions.
    - *In-bag max pooling of similarity scores*: Within each bag, the system collects the similarity scores for each address from the different kNN models in which they appeared. Max pooling is then applied to these scores to determine the maximum similarity score for each address within the bag.
    - *In-bag Ranking*: The addresses are then sorted within each bag based on their maximum similarity scores.
3. Final address list retrieval:
    - *Final output*: All the bags are concatenated resulting in a sorted list of addresses where the top candidate address has been repeated the most times and possesses the highest similarity score.
    - *Cut-off value choice*: The system sets the value of m (the number of neighbors to return) and computes performance metrics to evaluate the effectiveness of the address matching process. The value of m is not necessarily the same as k since the voting process ultimately

is ranking all the candidates of the k-lists combined which would naturally produces k-plus candidates depending on how heterogeneous the k-lists are.

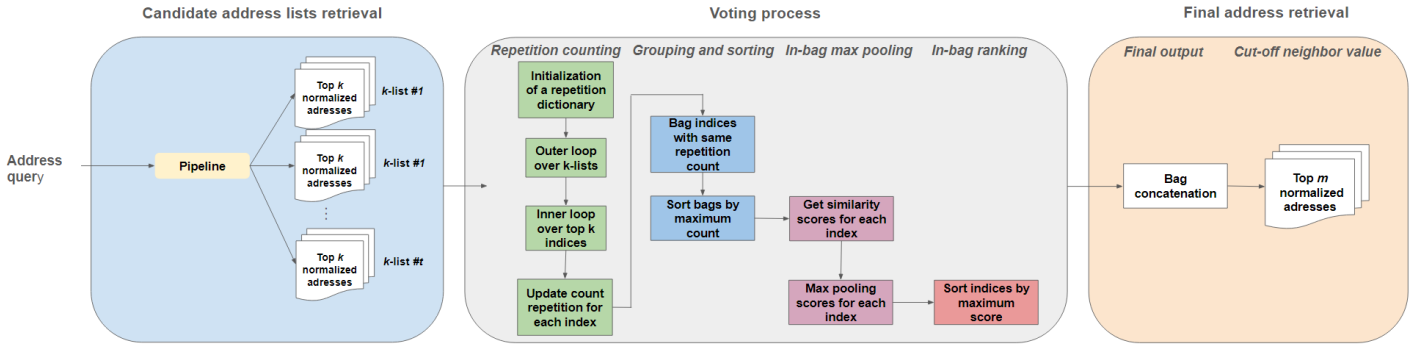


Figure 3: Ensemble Vote Process.

5. Experimental Settings

5.1. Data description

In our experiments, we use real private postal address data made available by a carrier in the region of Yonne, France. This data consists of two database tables, a table of approximately 1M non-normalized addresses of deliveries received via EDI and another table of registered recipients of more than 42K normalized postal addresses. After the de-duplication step mentioned above, and due to the presence of large number of identical delivery instances, just over 85% of all delivery address instances have been de-duplicated, mainly because most deliveries are business addresses. As a result, we are left with just over 147K distinct delivery addresses.

Dataset creation: We are in an offline training set up (i.e. our ensemble kNN retriever doesn't need training but rather takes advantage of the different representations, vector or raw, of postal addresses in order to search for the most similar addresses). That said, the creation of a dataset of address pairs (i.e. non-normalized query-address, normalized reference-address) is necessary for two reasons: (1) the offline fine-tuning of the different sentence representation models for the addresses and (2) using the dataset in the final performance test of the kNN-vote system. To do this, we use the recipient keys associated with the records in the two tables to create a dataset of over 147K address pairs. The dataset is then divided between training data and test data with respective proportions of 90% and 10%. The same test data will be used to evaluate kNN-vote. A second cleaning is carried out on the training dataset to eliminate certain non-normalized entry addresses likely to reduce the learning quality of the Bi-Encoder such as addresses having only the postal code and the city name. These type of addresses lacks completely the context linking them to their supposed normalized counterparts. Around 0.8% of the training data was impacted by this second cleaning. The Table 2 shows some examples of this kind of addresses.

5.2. Bi-encoders fine-tuning parameters

5.2.1. Fine-tuning Base

The Three chosen base transformers were trained on a corpus of approximately 950K official French postal addresses from the Yonne region, France and adjacent regions taken from the official governmental

Table 2
Examples of Delivery Address Deletion

Received address	Normalized address	Justification for deletion
trichey 89430 trichey	4 rue maillet 89430 trichey	This address only have the postal code and the name of the city
89160 89160 sambourg	11 rue d argenteuil 89160 sambourg	Here another example where door number and road name are missing
xxxx 89240 pourrain	30 route d aillant 89240 pourrain	In this example, 'xxxx' is used as a placeholder because the expediter only had the recipient's name and needed to fill in something for the incomplete address

website[7]. The complete training of the three encoders was carried out during 5 iterations and no parameter optimization was done. The aim here was simply to adapt the three language models to the postal addresses and have them as a basis for fine-tuning the Bi-Encoders. The "transformers" package from HuggingFace[8] was used to train these language models.

5.2.2. Bi-encoders fine-tuning

The three Bi-Encoders were fine-tuned according to the best combination of hyper-parameters presented in Table 3. Both Camembert-base and XLM-Roberta-base architectures used for the first two Bi-Encoders fine-tuning can be explored in details in [27, 28] as for the third one, a custom pre-trained Roberta-small architecture (6-layers, 128-hidden, 8-heads and 8 million parameters) is used. The three Bi-Encoders were adjusted on a local server with an NVIDIA Tesla A100 graphics card (20 GB) via the SBERT[9] "sentences-transformers" package.

Table 3
Best Found Fine-tuning Hyper-parameters for Bi-Encoders

	Bi-Encoder		
Parameter	Camemebert	XLM Roberta	Roberta from scratch
epochs	19	19	20
batch size	32		
Optimizer	AdamW		
Learning Rate Scheduler	2e-5	2e-5 WarmupLinear	2e-3
Warmup Steps	100		
Weight Decay	0.01		
Loss	MNR Loss		
Base Transformer	Camembert-base	XLM-roberta-base	Roberta-custom

5.3. Models evaluation

For the evaluation of the proposed voting approach, we compare it with our different individual kNN models in addition to the bi-encoder (BI_DistilBert) model proposed by Duarte et al. [6], where they

[7]https://adresse.data.gouv.fr/
[8]https://huggingface.co/docs/transformers/index
[9]https://www.sbert.net/

use DistilBert Multilingual as a basis for fine-tuning their model. To remain consistent with the cited research, we consider a value of k neighbors equal to 10 but we take the time to test other values of k with respect to our individual systems. The models were evaluated based on two metrics: (1) The existence ratio (ER), which is the proportion of correctly predicted positive pairs out of all pairs in the test data set, and (2) the MRR, i.e. the Mean Reciprocal Rank, which is a measure used to evaluate the quality of the appearance ranks of correct query responses via information retrieval systems. For a sample of queries Q and $rank_i$, i.e. the position of the correct searched address for a query $q_i \in Q$ with $i = 1, ..., |Q|$, the MRR formula can be defined as follows:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}, \tag{3}$$

The primary objective of the models is to achieve a maximum ER at the exact matching level (i.e. the predicted address is exactly the address sought for the query). In addition, two types of ER are computed: (1) The ER of the correct predictions in the first rank (top 1) and (2) the ER of the correct predictions among the k address candidates (top k). We are also interested in the matching ER at the road level (i.e. the predicted address is at least in the correct road of the searched address). This type of ER is all the more important since in practical cases, carriers will generally be able to successfully deliver parcels as long as they are in the same lane of the delivery address[6].

6. Results

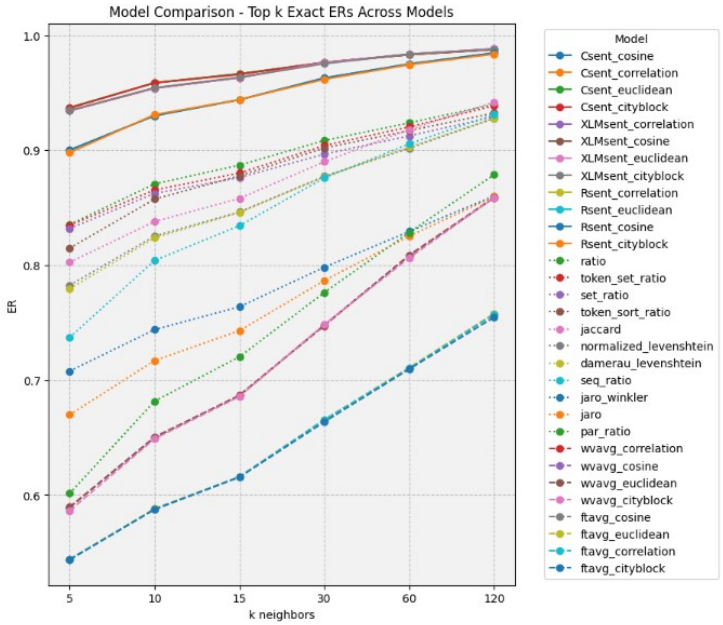
6.1. Comparison of individual kNN models

Our first intent was to compare individual kNN systems in order to identify the best performing model in terms of top k ER at the exact search level (top k exact). The results illustrated in Fig. 4a show the superiority of kNN models based on the different sentence representations and this comes down to the quality of the hyper contextualized embeddings in comparison for example with word embeddings like Word2Vec or FastText. We also note that models based on raw representations are generally more efficient than Word2Vec and FastText. The reason is probably because of the enormous loss of information in the static embeddings due to the mean pooling used to create the address vectors. Increasing the value of k positively impacts the existence ratio overall the models because the larger the list of neighbors, the greater the chance of more difficult addresses to be retrieved. However, the increasing levels of the existence ratio vary between 5% for sentence embedding, 11% for raw embedding and 26% for static embedding with a k value between 5 and 120, as shown in Figure 4a. This can be explained by the level of accuracy of the sentence embedding, as the majority of positive pairs are already identified within the first 5 candidate addresses. In contrast, the raw and static embedding models require a very high k value of up to 120. In terms of MRR, the results in Figure 4b are consistent with the existence ratios, as the best models should have the highest MRR at the lowest possible k value. The dynamic finetunner sentence kNN embedding models retrieve the searched addresses at the highest ranks compared to the other models. Furthermore, they remain stable as k increases, thus demonstrating their strong retrieval ability even with the earliest candidates thanks to their ability to capture address context. This was expected as well, as the very purpose of sentence transformers is to learn how to reduce the distance between vectors of positive address pairs, even if they are very different syntactically, whereas models based on string similarity distances only perform well when the addresses are relatively similar syntactically.

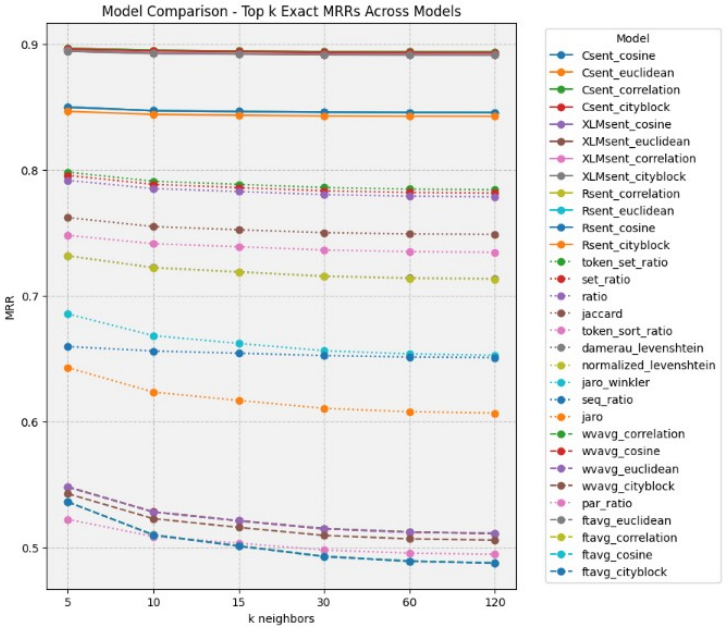
6.2. kNN multi-embeddings models experiment results

6.2.1. Multi-embeddings models instances

We wanted to test the performance of the voting system using the set of individual kNN models while having the flexibility to select different subsets to maximize the voting efficiency. Fig. 5 illustrates the



(a) kNN's top k exact ERs



(b) kNN's top k exact MRRs

Figure 4: Evaluation of Individual kNN models with regards to metrics: ER and MRR

results of the ERs top k exact of the chosen subsets that had the overall better performance. We observe that the subset of sentence only models performs generally better. If we further exclude from the latter the kNN models based on Roberta from scratch (camembert + XLM), we see a small increase in ERs at k values of 5 and 10. This increasing aspect is due to the original pre-training of Camembert and XLM_Roberta. It shows the extent that language models (pre-trained on large language corpora) have in terms of performance quality when used in other tasks such as the STS task. The voting system with all models is the least efficient and this can be explained by the large differences between the neighbor lists returned by the sentence models and the other models. In other words, it is natural that kNN retrivers with the most mistakes in predicting positive pairs impact the ability of the vote to systematically propose a high number of repetitions to the sought-after addresses. This explanation remains even more coherent when we remove the static vectors models from the vote (sentence + raw).

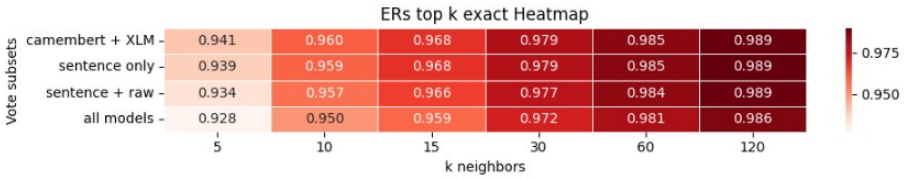


Figure 5: kNN-vote top k exact ERs

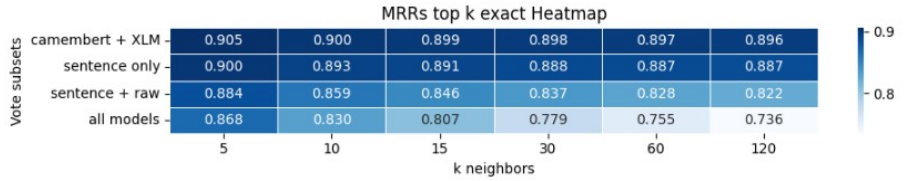


Figure 6: kNN-vote top k exact MRRs

We notice a clear improvement in ERs. As for the MRR results, we observe in Fig. 6 that in general, voting systems based on sentence models succeed in recovering more positive normalized addresses at the highest ranks.

6.2.2. Discussion

We put ourselves in comparison with BI_DistilBert. We take into account the addresses found at the road level and we also consider the top 1 results. We remain consistent regarding the value of $k = 10$. Table 4 illustrates the best individual kNN models and voting systems in comparison with BI_DistilBert. We find that BI_DistilBert performs better than the raw models. However, it remains below the ER results of the individual kNN sentence models and this is due to two main reasons. First, the base of the model, which is multilingual DistilBert, was not pre-trained on a corpus of postal addresses before fine-tuning its Bi_Encoder. However, we believe that it is important that basic language models learn the structure of a postal address independently of the similarity task. Second, the additional difficulty that our dataset brings. Indeed, our addresses are much more difficult in terms of the errors and noise likely to occur. kNN-vote systems are better overall, supporting our intuition that aggregating results from multiple sources significantly improves similarity search performance. We do, however, note exceptions to the rule. Some individual kNN models such as (A) and (B) come before the (G) and (H) vote systems. This decrease in ERs confirms to us that aggregation alone does not always guarantee better results and that a high and heterogeneous number of models used in the voting process negatively impacts the prediction quality. This is why the individual performance of the models used in the vote must also be taken into account. More specifically, the vote will be more likely to have superior results if it uses as its aggregation sources, search models that are the least wrong in their predictions. (I) manages to compete with the two best individual kNN sentence models but adds no improvement and in particular in the top 1 exact. It is undoubtedly the participation of Rsent models in the vote that prevents it from standing out from the other search systems, since Rsent is significantly less efficient than Csent and XLMsent. In conclusion, the best vote is the one that uses the Csent and XLMsent models with a ER top 1 exact of 86.2% and a ER top 10 exact of 96% thus demonstrating the ability of the voting system to retrieve more positive address pairs in the top 1.

Inference time: We measured the retrieval time for 100 address queries to compare the various solutions, as shown in Table 4. Retrieval times for voting systems (between 51s and 173s) are notably longer than individual kNN models. Despite being conducted without optimization in an experimental setup, we find these times acceptable for business applications.

Table 4
Existence Ratios of the Best Methods

System	Top 1 exact	Top 1	Top 10 exact	Top 10	MRR	Time (100 $queries$)
(A) Csent_cosine	0.857	0.916	0.959	0.970	0.895	12
(B) XLMsent_cosine	0.856	0.917	0.954	0.968	0.893	13
(C) Rsent_cosine	0.799	0.879	0.930	0.956	0.847	6
(D) Token_set_ratio	0.740	0.829	0.866	0.889	0.791	8
(E) Ratio	0.730	0.834	0.871	0.891	0.785	5
(F) BI_DisilBert	0.763	0.793	0.918	0.939	0.826	10
(G) all models	0.760	0.872	0.950	0.962	0.830	173
(H) sentence + raw	0.801	0.896	0.957	0.967	0.859	144
(I) sentence only	0.852	0.920	0.959	0.972	0.894	69
(J) camembert + XLM	**0.862**	**0.921**	**0.960**	**0.972**	**0.900**	51

7. Conclusions

In this work we focused on the problem of matching postal addresses. We first showed that this task can be simply formalised as an information retrieval problem where models such as kNN have been shown to be efficient in both computation time and accuracy. For this purpose, we have assumed that an address is a sentence described with a set of entities and, consequently, it could contain erroneous elements or noisy elements. However, the positions of the entities have an impact on address recognition. For these reasons, we have proposed using different address representation spaces, such as the word embedding space or the sentence embedding space with a pre-trained transformer. Each representation contributes in part to the search for the closest address in that space. In order to aggregate the contribution of the different spaces, we proposed a kNN ensemble models based on a voting system called kNN-vote. The experimental results show that our system performs very well, achieving an accuracy of around 96% in the top 10 and 86.2% in the top 1. This system shows its value for this type of task, even though the voting algorithm is still very naive for the time being. In fact, the algorithm favours addresses with a maximum number of repetitions and re-ranks them solely on the basis of the highest similarity score. Hence the impact of the number of voters on the number of appearances. In addition, the system's focus on the highest score of the address, without taking into account the overall quality of the scores, can lead to the dominance of a single score, even if other scores are more indicative. As a perspective, we are improving the voting process in order to consider and reinforce the potential effectiveness of a model with a lower but more significant score.

Acknowledgments

Thanks to the french ANRT (Association Nationale de la Recherche et de la Technologie) for funding this project under the "*Cifre convention for thesis funding*" https://www.anrt.asso.fr/ and to the developers of TEDIES, TALK solutions who assisted in this project https://site.tedies.eu/.

References

[1] Y. Guermazi, S. Sellami, O. Boucelma, Georoberta: A transformer-based approach for semantic address matching, in: HAL, 2023. URL: https://hal.science/hal-04465164.

[2] S. Comber, D. Arribas-Bel, Machine learning innovations in address matching: A practical comparison of word2vec and crfs, in: Transactions in GIS, volume 23, 2019, pp. 334–348. URL: https://doi.org/10.1111/tgis.12522.

[3] Y. Guermazi, S. Sellami, O. Boucelma, Address validation in transportation and logistics: A machine

learning based entity matching approach, in: Communications in Computer and Information Science, volume 1323, 2020, pp. 320–334. URL: https://doi.org/10.1007/978-3-030-65965-3_21.

[4] Y. Lin, M. Kang, Y. Wu, Q. Du, T. Liu, A deep learning architecture for semantic address matching, in: International Journal of Geographical Information Science, volume 34, 2019, pp. 559–576. URL: https://doi.org/10.1080/13658816.2019.1681431.

[5] J. Chen, J. Chen, X. She, J. Mao, G. Chen, Deep contrast learning approach for address semantic matching, in: Applied Sciences, volume 11, 2021, p. 7608. URL: https://doi.org/10.3390/app11167608.

[6] A. Duarte, A. Oliveira, Improving address matching using siamese transformer networks, in: Lecture Notes in Computer Science, volume 14116, 2023, pp. 413–425. URL: https://doi.org/10.1007/978-3-031-49011-8_33.

[7] H. M. Rakotondrasoa, et al., Quantitative comparison of nearest neighbor search algorithms, in: arXiv, 2023. URL: https://arxiv.org/abs/2307.05235.

[8] T. Gschwind, Fast record linkage for company entities, in: IEEE Conference Publication, 2020. URL: https://ieeexplore.ieee.org/document/9006095.

[9] K. Mengjun, D. Qingyun, W. Mingjun, A new method of chinese address extraction based on address tree model, in: Acta Geodaetica et Cartographica Sinica, volume 44, 2015, pp. 99–107.

[10] V. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, in: Soviet Phys. Doklady, volume 10, 1966, p. 707.

[11] F. J. Damerau, A technique for computer detection and correction of spelling errors, in: Communications of the ACM, volume 7, 1964, pp. 171–176.

[12] P. Jaccard, Distribution de la flore alpine dans le bassin des dranses et dans quelques regions voisines, in: Bulletin de La Société Vaudoise Des Sciences Naturelles, volume 37, 1901, pp. 241–272. URL: https://www.scirp.org.

[13] M. Jaro, Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida, in: Journal of the American Statistical Association, volume 84, 1989, pp. 414–414. URL: https://doi.org/10.2307/2289924.

[14] W. E. Winkler, String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage, in: ERIC, 1990. URL: https://eric.ed.gov/?id=ED325505.

[15] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: arXiv, 2013. URL: https://arxiv.org/abs/1301.3781.

[16] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, in: arXiv, volume 5, 2017. URL: https://arxiv.org/abs/1607.04606.

[17] C. Sutton, A. McCallum, An introduction to conditional random fields, in: arXiv, 2010. URL: https://arxiv.org/abs/1011.4088.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NeurIPS, 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, in: arXiv, 2019. URL: https://arxiv.org/abs/1907.11692.

[20] Y. Li, J. Li, Y. Suhara, A. Doan, W.-C. Tan, Deep entity matching with pre-trained language models, in: Proceedings of the VLDB Endowment, volume 14, 2020, pp. 50–60. URL: https://doi.org/10.14778/3421424.3421431.

[21] U. Brunner, K. Stockinger, Entity matching with transformer architectures - a step forward in data integration, in: EDBT, 2020. URL: https://doi.org/10.5441/002/edbt.2020.58.

[22] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019. URL: https://doi.org/10.18653/v1/d19-1410.

[23] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, in: arXiv, 2019. URL: https://arxiv.org/abs/1910.01108.

[24] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with gpus, arXiv preprint

arXiv:1702.08734 (2017). URL: https://doi.org/10.48550/arXiv.1702.08734, submitted on 28 Feb 2017.

[25] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (1975) 509–517. URL: https://doi.org/10.1145/361002.361007. doi:10.1145/361002.361007.

[26] S. M. Omohundro, Five Balltree Construction Algorithms, Technical Report, International Computer Science Institute, Berkeley, CA, 1989. URL: https://www.icsi.berkeley.edu/pubs/techreports/TR-89-063.pdf.

[27] L. Martin, B. Muller, P. Suárez, Y. Dupont, E. de la Clergerie, D. Seddah, B. Sagot, Camembert: a tasty french language model, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7203–7219. URL: https://doi.org/10.18653/v1/2020.acl-main.645.

[28] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, in: arXiv, 2020. URL: https://arxiv.org/abs/1911.02116.

[29] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005. URL: https://doi.org/10.1109/cvpr.2005.202.

[30] M. Henderson, R. Al-Rfou, B. Strope, Y. Sung, L. Lukacs, R. Guo, S. Kumar, B. Miklos, R. Kurzweil, Efficient natural language response suggestion for smart reply, in: arXiv, 2017. URL: https://arxiv.org/abs/1705.00652.

[31] A. Chernyavskiy, D. Ilvovsky, P. Kalinin, P. Nakov, Batch-softmax contrastive loss for pairwise sentence scoring tasks, Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (jul 2022) 116–126. doi:10.18653/v1/2022.naacl-main.9.