Semantic address matching using Keras for R

Paula Cruz *,** | Leonardo Vanneschi ** | Marco Painho **| Filipa Ribeiro *

*Statistics Portugal

**NOVA IMS

uRos2024, 27-29 November 2024

Agenda

1. Introduction
2. Proposed approach
3. Preliminary results
4. Conclusions and future work

1. Introduction: Background

National Buildings and Dwellings Database with 6 million registers (Census-based)

Current parsing and matching procedures need improvement

Potential for regular updates with administrative data

Acceptance of probabilistic, partial matches

Data privacy

1. Introduction: Standard approach (text similarity)

(ORACLE) methods based on string similarity (Levenshtein, Jaro-Winkler), both at the level of the full address and at the level of address components.

Preprocessing: capitalization, removal of extra blank spaces, special symbols, and normalization.

Partitioning into components: street type, street name, building type and name, door number, floor, and side.

Initially, the partitioning was done using specific software (QualityStage), but it was later replaced by the use of regular expressions in Oracle (REGEXP) as well as the Libpostal software (NLP + open data).

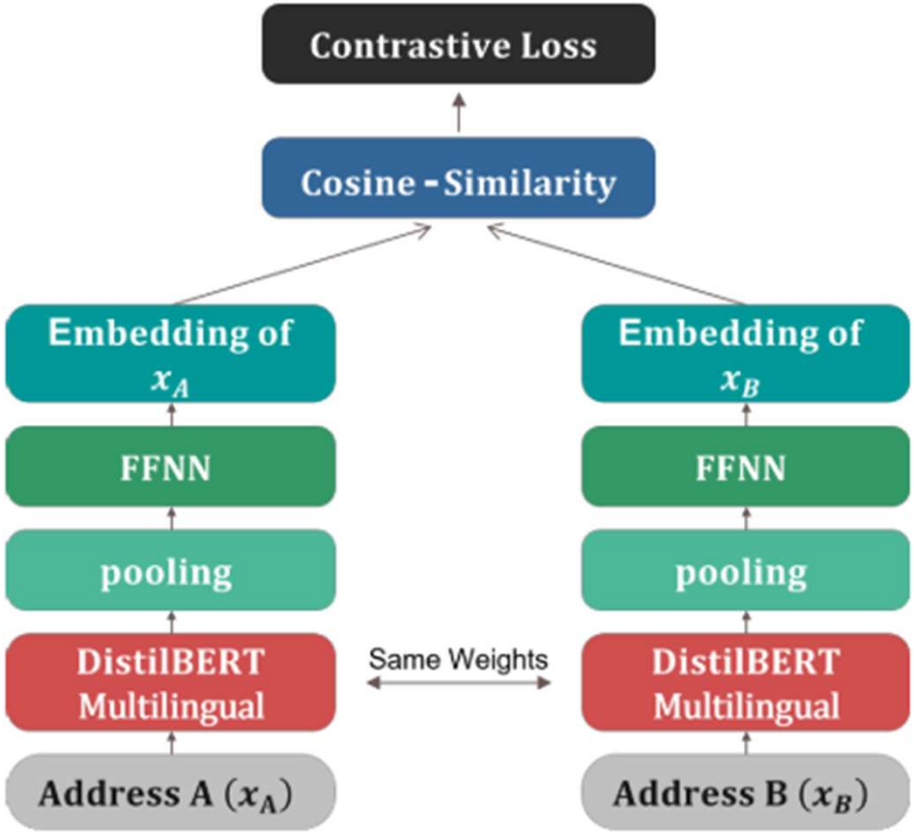
Minimum threshold of 79% to classify a record association as a match.

Limitations:

- Below the defined threshold, addresses with significant variations in spelling could still be matches
- Above the threshold, matching addresses with differences at the component level (e.g., door number, unit level) are difficult to capture

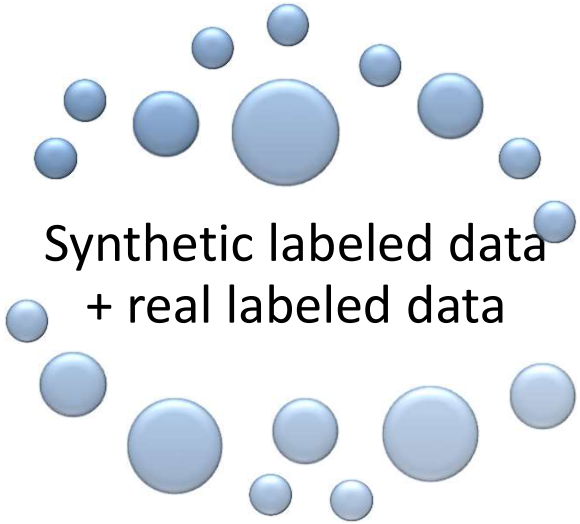
2. Proposed approach: Natural language processing (NLP)

- In general, deep learning methods achieve a higher matching accuracy than traditional text-matching models (Cruz et al. 2021).
- Until 2017, bidirectional RNNs (in particular, bidirectional LSTMs) were widely used for sequence modeling, due to their ability to capture context in both directions.
- Today, transformers, relying on self-attention mechanisms, are the dominant choice for sequence modeling, namely pretrained compact language models such as **DistilBERT**.



Source: Duarte, A. V., & Oliveira, A. L. (2023)

2. Proposed approach: Training dataset generation



Synthetic labeled data
+ real labeled data

25,000 Matches +

25,000 Non Matches

(Lin et al, 2020)

Final
Sample

50,000 registers

2. Proposed approach: Implementation in R

Main libraries used

```
# reticulate: R Interface for Python
            reticulate::virtualenv_create("r-tensorflow", python =  "python3.9")
# tensorflow: Machine learning platform
            tensorflow::install_tensorflow(envname = "r-tensorflow")
# keras: Deep learning API
            keras::install_keras(envname = "r-tensorflow")
# tfdatasets: TensorFlow data pipelines
# caret: Machine learning toolkit
# transformers: NLP transformer models
```

2. Proposed approach: Implementation in R

(Down)loading of distilBERT (Hugging Face's)

```
# Download DistilBERT model files for local use and set up paths
pretrained_path <- "URos2024/NLP/distilbert"
config_path <- file.path(pretrained_path, "config.json")
weights_path <- file.path(pretrained_path, "pytorch_model.bin")
vocab_path <- file.path(pretrained_path, "tokenizer")

tokenizer <- transformers$DistilBertTokenizerFast$from_pretrained(vocab_path)
distilbert_model <- transformers$TFAutoModel$from_pretrained(pretrained_path)

# Or Load DistilBERT's tokenizer and model
tokenizer <- transformers$DistilBertTokenizerFast$from_pretrained("distilbert-
base-uncased")
distilbert_model <- transformers$TFAutoModel$from_pretrained("distilbert-base-
uncased")
```

2. Proposed approach: Implementation in R

Load and prepare sample data

```
#Load data
data <- read.csv("URos2024/NLP/datasets/sample_50k.csv", sep=";", header=TRUE,
encoding = "UTF-8")

# Shuffle and clean data
data <- data[sample(1:nrow(data)), ]
data <- data %>%
          mutate(
          str_replace_all(address1, "[^\\w\\s]", "") %>%  # Special characters
          str_replace_all("\\s{2,}", " "),                # Extra spaces
          str_replace_all(address2, "[^\\w\\s]", "") %>%
          str_replace_all("\\s{2,}", " ")
          )
```

2. Proposed approach: Implementation in R

Tokenize data

```
sequence_length <- 50  # Sequence length for tokenization
set.seed(123)  # For reproducibility

preprocess_dataset_with_bert <- function(df) {

address1_encodings <- tokenizer(
    as.character(df[['address1']]),
    truncation = TRUE,
    padding = "max_length",
    max_length = as.integer(sequence_length),
    return_tensors = "tf"
  )
```

2. Proposed approach: Implementation in R

Tokenize data

```
address2_encodings <- tokenizer(
    as.character(df[['address2']]),
    truncation = TRUE,
    padding = "max_length",
    max_length = as.integer(sequence_length),
    return_tensors = "tf"
  )
  labels <- df[['match']]
  list(
    list(address1_encodings$input_ids, address1_encodings$attention_mask,
         address2_encodings$input_ids, address2_encodings$attention_mask),
    labels
  )
}
```

2. Proposed approach: Implementation in R

Split data into train, validation, and test sets

```
train_indices <- sample(1:nrow(df), size = floor(0.8 * nrow(df)))
remaining_indices <- setdiff(1:nrow(df), train_indices)
valid_indices <- sample(remaining_indices, size = floor(0.5 *
length(remaining_indices)))
test_indices <- setdiff(remaining_indices, valid_indices)

train_df <- df[train_indices, ]
valid_df <- df[valid_indices, ]
test_df <- df[test_indices, ]

# Preprocess datasets
train_data <- preprocess_dataset_with_bert(train_df)
valid_data <- preprocess_dataset_with_bert(valid_df)
test_data <- preprocess_dataset_with_bert(test_df)
```

2. Proposed approach: Implementation in R

Define the Siamese network

```
# First tower
inputs_1 <- layer_input(shape = c(sequence_length), dtype = "int32", name =
"address1_input_ids")
attention_mask_1 <- layer_input(shape = c(sequence_length), dtype = "int32", name
= "address1_attention_mask")
distilbert_output_1 <- distilbert_model(list(inputs_1,
attention_mask_1))$last_hidden_state %>%   layer_global_average_pooling_1d()

# Second tower
inputs_2 <- layer_input(shape = c(sequence_length), dtype = "int32", name =
"address2_input_ids")
attention_mask_2 <- layer_input(shape = c(sequence_length), dtype = "int32", name
= "address2_attention_mask")
distilbert_output_2 <- distilbert_model(list(inputs_2,
attention_mask_2))$last_hidden_state %>%   layer_global_average_pooling_1d()
```

2. Proposed approach: Implementation in R

Compute similarity measure and define classification layer

```
# Compute similarity measure (absolute distance)
layer_similarity <- layer_lambda(
    f = function(tensors) k_abs(tensors[[1]] - tensors[[2]]),
    output_shape = function(input_shape) input_shape[[1]])
similarity_output <- layer_similarity(list(distilbert_output_1,
distilbert_output_2))

# Final layer for classification
outputs <- similarity_output %>%
layer_dense(units = 128, activation = "relu", kernel_regularizer =
regularizer_l2(0.01)) %>% # Hidden dense layer
layer_dense(1, activation = "sigmoid") # Final output layer
```

2. Proposed approach: Implementation in R

Compile the model

```
# Compile the model
model <- keras_model(
inputs = list(inputs_1, attention_mask_1, inputs_2, attention_mask_2),
outputs = outputs
)%>%
compile(
optimizer = optimizer_adam(learning_rate = 1e-4),loss = "binary_crossentropy",
metrics = "accuracy“)

# Learning rate reduction and early stopping
learning_rate_reduction <- callback_reduce_lr_on_plateau(
monitor = "val_loss",  patience = 3,  verbose = 1,  factor = 0.3,  min_lr = 1e-8)
callback_list <- list(callback_early_stopping(restore_best_weights = TRUE,
patience = 2, min_delta = 0.001, verbose = 1),
learning_rate_reduction)
```

2. Proposed approach: Implementation in R

Train and evaluate the model

```
summary(model)#Total params: 66,461,441 (253.53 MB)

# Train the model
history <- model %>% fit(x = train_data[[1]], y = train_data[[2]], validation_data
= list(valid_data[[1]], valid_data[[2]]),  epochs = 5,  batch_size = 32,
callbacks = callback_list,  use_multiprocessing = TRUE, workers = 4)

# Plot training history
plot(history)

# Predict on test data
predictions_probs <- model %>% predict(test_data[[1]])
predictions <- ifelse(predictions_probs > 0.5, 1, 0)
true_labels <- test_data[[2]]
conf_matrix <- table(Predicted = predictions, Actual = true_labels)
```

3. Preliminary results

Available, preliminary results in terms of accuracy (~0.98) are comparable to those presented in the literature:

Model	*Accuracy*	*Main parameters*
***ESIM model + word2vec** (Lin et al., 2020) – Chinese addresses*	**0.97**	Epochs= 50; Batch Size= 50; Optim. = Adam; LR = 10^{-4}; Loss = Binary crossentropy
***ESIM model** + pre-trained Glove (Ramani, K., & Borrajo, D., 2023) – English addresses*	**0.95**	Epochs= 50; Batch Size= 8; Optim. = Adam; LR = 10^{-4}; Loss = Binary crossentropy
Siamese BERT (Duarte, A. V., & Oliveira, A. L., 2023) – Portuguese addresses	**0.96**	Epochs= 20; Batch Size= 16; Optim. = AdamW; LR = 10^{-5}; Loss = Contrastive Loss

However, generalization capability of the model to out-of-distribution examples needs to be improved (hard training samples).

4. Conclusions and future work

Main conclusions:

- Preliminary results comparable to those presented in the literature as well as to those obtained in a Python implementation.

Future work:

- Improve generalization capability;
- Run the model in a GPU environment for better performance;
- Include geographical coordinates as geohashes appended to address texts (Guermazi et al. 2023) to cope with missing information;
- Adapt the model to multiclass classification to deal with partial matches.

References

- Cruz, P., Vanneschi, L., Painho, M., & Rita, P. (2021). *Automatic identification of addresses: A systematic literature review*. ISPRS International Journal of Geo-Information, 11(1), 11
- Duarte, A. V., & Oliveira, A. L. (2023, September). *Improving Address Matching Using Siamese Transformer Networks*. In EPIA *Conference on Artificial Intelligence* (pp. 413-425). Cham: Springer Nature Switzerland
- Guermazi, Y., Sellami, S., & Boucelma, O. (2023, March). *GeoRoBERTa: A Transformer-based Approach for Semantic Address Matching*. In 7th International workshop on Data Analytics solutions for Real-LIfe APplications (DARLI-AP)
- Lin, Y., Kang, M., Wu, Y., Du, Q., & Liu, T. (2020). *A deep learning architecture for semantic address matching*. International Journal of Geographical Information Science, 34(3), 559-576.
- Ramani, K., & Borrajo, D. (2023). *Methods for matching English language addresses*. Transactions in GIS, 27(2), 347-363

» Semantic address matching using Keras for R

https://hour.ine.pt/

Obrigada | Thank You «

uRos2024, 27-29 November 2024