Text Similarity Techniques for Services Categorization and Search

Louisa Camadini
ENSAE Paris
louisa.camadini@ensae.fr
https://github.com/louisacamadini/NLP_Project

Abstract

Accessing local services in a community can be a challenging task. Our proposed solution leverages machine learning algorithms to streamline the process of finding and connecting users with local service providers. We created an interactive search engine that utilizes a shared document in the city of Bastia to provide accurate information on available services. Users can input their specific needs, and the system returns the names and contact information of individuals offering those services in the area.

The goal of the search engine is to find the best matches between keywords and one or more services available in a database. To achieve better matches, we will test two vectorization methods (TF-IDF and BERT) and several similarity measures (cosine similarity, Euclidean distance, Pearson correlation).

Finally, to further improve search quality, we will use an unsupervised learning algorithm to classify the services into categories, here is an example of a category:

coach
développement
personnel
vie
communication
professionnels
sportif
relations
personnes
confiance
nouveaux
amoureuses
créativité
objectifs
soi
d'entreprise
jeunes
universitaires
professionnel
artistes
entrepreneuriat
adolescents
forme
créatifs
étudiants
chefs
atteindre
nutrition
personnels

1 Problem Framing

The goal of this initiative is to create a useful resource for the community: an interactive document was shared for individuals to input the services they can provide. In order to make our database reach a sufficient size (over a thousand lines), we have generated some sample services using ChatGPT, a language model developed by OpenAI. To ensure anonymity, random names and phone numbers were generated. The database is available in our Github repository[1].

In this data frame, we have added a column where you will find the text once processed.

Indeed, data preprocessing is a crucial step in preparing text data for use with embedding methods. In our case, it includes:

1. Text cleaning: removing punctuation, unwanted and special characters,

2. Tokenization: breaking the text into individual words (tokens),

3. Stop-word removal: removing common words such as "the", which do not make much sense and can affect model performance,

4. Lemmatization: Reducing words to their base form, taking into account the context.

In a first attempt, we had encountered problems because when searching for a specific keyword, the search engine did not put us in contact with a nearby service, hence the importance of applying lemmatization (reducing words to their base form) or stemming (root form). We then tested both techniques and chose lemmatization because it takes into account the context and gives us better results. Given the size of our database, this process is computationnally unexpensive. Stemming, on the other hand, simply removes the suffix from the word to obtain its root form, which is less accurate in our case.

The code that preprocesses the data can be found on Github[2]. Note that since it is about ser-

[1] https://github.com/louisacamadini/NLP_Project/df.csv
[2] https://github.com/louisacamadini/NLP_Project/data_cleaning.ipynb

vices rendered in Bastia, the dataset is in French.

Related Work Several previous studies have explored the use of NLP algorithms to improve access to local services. For example, in their paper "Using Natural Language Processing to Improve Access to Social Services", Smith 2019 developed a chatbot that could help users find social services in their area based on their specific needs. Similarly, Hussain 2018 used clustering techniques to categorize local service providers in their paper. In our study, we take advantage of this previous work to show that it is possible to create a search engine from scratch using accessible NLP methods.

2 Experimental settings

TF-IDF method We started by using the TF-IDF method which calculates the importance of each keyword in a service, by taking into account both the frequency of the keyword within the service and the frequency of the word across all services proposed in the database. TF-IDF assigns a weight to each keyword that reflects how important it is to the overall meaning of the service, in order to identify relevant service based on their content.

Several remarks about the TF-IDF method:

It assigns a relative weight to words and phrases that are redundant in a database, such as *"Aide à..."*. This is because these words and phrases are often used to describe the services offered, and thus are frequently present in the descriptions of services in the database.

Generally, since the description of the offered services is short, each sentence will contain the keyword only once, then each service will have the same weight for this keyword when calculated using this method.

Similarity measures After applying the TF-IDF vectorization method, we compared different similarity measures :

1. Cosine similarity (cosine of the angle between two vectors),

2. Euclidean distance,

3. Pearson correlation (linear relationship between two vectors).

We are looking for the similarity measure that best captures the semantic similarity between the keyword and the services in the database, which allowed it to match the keyword with the most relevant individuals. To compare the performance of these similarity measures, we first thought of using classical automatic measures such as BLEU and ROUGE (developed in C. Chhun 2022), which turn out to be completely irrelevant for matching keywords with phrases, as it uses n-gram similarity.

Finally, human evaluation of these three similarity measures revealed some obvious insights, especially focusing on certain clusters and applying the different similarity measures on keywords in specific clusters.

BERT We will now compare the TF-IDF approach with a BERT embedding method. This is a NLP technique based on a pre-trained deep neural network, which involves generating contextualized embedding of words. This allows BERT to capture the meaning and context of words in a service, rather than processing each word in isolation, and we will see if BERT can improve the performance of our search engine.

Classification The following unsupervised method is based on TF- IDF vectorization, which worked very well for the search engine. We generate word embeddings and perform K-means on these embeddings (a service being a data point). The idea is to hope that by dividing the data into a well-chosen number of clusters, these will correspond to the different categories we can think of (for example: *"jardinage", "cours particuliers", "artisanat" etc.*)

What is the most appropriate number of clusters to divide our services into different categories? Note that for our purposes, we would like a reasonable number of groups to be able to study them individually (15 at most). To answer this question, we first tried to implement an Elbow curve, which does not give a clear result for less than 20 clusters (Figure 3). We also implemented the DBSCAN clustering algorithm which groups together points that are close to each other based on a density criterion. This algorithm is quite sensitive to the parameters, but after several tests we found satisfactory results. You will find the code on Github[3].

[3] https://github.com/louisacamadini/NLP_Project/classification.ipynb

3 Results

3.1 Vectorization

As already mentioned and shown in the output 1, the TF-IDF vectorization method worked very well due to the structure of our dataset. However, we tried to use an embedding method. We replaced the vectorization step with BERT embeddings and modify the similarity calculation accordingly[4].

We used a pre-trained BERT model *("distilbert-base-nli-mean-tokens")* to encode the text into vectors, and then calculated the cosine similarity between these vectors and a keyword also encoded with BERT. This method gave us very convincing results. We needed to increase the cosine similarity threshold to limit the number of outputs, but the latter are consistent. The TF-IDF method keeps consistent recommendations even when decreasing the threshold, while BERT proposes inconsistent services more rapidly. As we expected, TF-IDF method performs very well for keyword search in a service database.

To compare the performance of the pre-trained BERT model with the TF-IDF method, we would need to perform an evaluation of our search engine using a metric such as precision, recall or F-measure. To do this, we would have had to split the dataset into two parts, one for training and the other for evaluation. Then, train the search engine on the training set using BERT for example, and evaluate the performance of the system on the test set using the chosen metric.

In our case, this is hardly feasible because of the size of our data. But it is within our reach to visually evaluate the performance of these systems, and conclude that the BERT method is less robust. We see this by moving the threshold slightly: for a high threshold, the results are few but quite consistent. However, by lowering the threshold, the outputs differ much more quickly than with the TF-IDF method.

In addition, we also tried to train a Word2Vec model on our data, using the Gensim library. While TF-IDF considers each word as an independent entity, Word2Vec takes into account the contextual relations between words. TF-IDF produces word vectors that are sparse, while Word2Vec produces word vectors that are dense. This means that Word2Vec word vectors contain more information per dimension, which can be advantageous for certain tasks. On paper, Word2Vec therefore seems to be a good candidate. In practice, our results with Word2Vec are very poor, probably due to the size of our dataset: if it is too small, the Word2Vec method may not be able to capture the semantic relations between words reliably.

TF-IDF	✓✓
BERT	✓
Word2Vec	✗

3.2 Similarity measures

We found that the "cosine similarity" measure performed very well in matching the entered keyword with the services offered in the database. Indeed, it measures the similarity between two vectors by computing the angle between them in a vector space. In this context, each word of the proposed service and each keyword are represented by a vector of real numbers, where each dimension represents an aspect of the word (for example, the frequency of its appearance).

The appendix 1 gives an output of the use of our search engine for NLP courses. Note that the search engine returns all contacts that have a cosine similarity measure greater than 0.2 (the threshold for which we have selected a consistent number of services). Our code compares these results with those obtained for the Euclidean distance, where we look at all vectors that are at a distance less than 1, and we get excellent results. The Pearson correlation however, even when adjusting the threshold, does not give us consistent results. In fact, the Pearson correlation can be used as a similarity measure, but it assumes a linear relationship between the variables, which may not be appropriate for our type of data.

Cosine similarity	✓
Euclidean distance	✓
Pearson correlation	✗

3.3 Classification

After considering the results of the Elbow curve and the DBSCAN algorithm (see 2 and 3), we decided to use a K-means clustering approach with $k = 10$ clusters as it provided a reasonable number of groups to study. However, these are not

[4] https://github.com/louisacamadini/NLP_Project/search_engine.ipynb

the only possible parameters we could have chosen, but we believe that the $k = 10$ solution will allow us to gain insights into the similarities and differences between our services, and to develop targeted strategies for improvement.

As a matter of fact, we obtained very coherent clusters, by grouping the services by categories explained in Table 1.

These clusters allowed us to obtain coherent word clouds (see 4), which we could use to improve our search engine, offering a pre-selection by category.

4 Discussion

This work allows us to affirm that it is quite possible to use simple NLP techniques to perform everyday tasks. Because of the collaborative construction of the database and the differences between its rows, we were able to do a sufficiently satisfactory data cleaning, which was not obvious either. Indeed, we evaluated techniques to match keywords and services on a medium size data set (over a thousand respondents).

We then select the techniques that work best for this kind of data: a TF-IDF vectorization to calculate the frequency of a word in a sentence, and a cosine similarity measure to match the keyword with the most relevant services (or the Euclidean distance).

Note that the use of the pre-trained BERT embedding method should not be discounted, thanks to the quality of the data. Overall, BERT embeddings tend to be more powerful than TF-IDF for more complex NLP tasks, especially when the data is larger and more varied. For our case study (relatively simple and small data), we preferred TF-IDF. This method is a simple but very efficient technique for text analysis and is particularly useful for information retrieval tasks.

It is possible, however, that we may need to use other methods if we extend our idea to cities larger than Bastia and obtain several thousand respondents. The answers would be somewhat diversified, although not so much, because of the relative redundancy of the services offered by people.

The K-means algorithm allowed us to efficiently organize the services into categories and obtain very interesting visuals for a potential website. As a next step, we are considering using BERT Fine-tuning for the classification of our data. For this, we would add a classification layer to the output of BERT, which would take the embedding representations of BERT and transform them into a classification probability for each category. During training, the model would adjust the weights of its layers to better represent the classification task. After training, we could evaluate the performance of the model on the validation set and use it for prediction.

To enable this, respondents would need access to a legend on the categories. Thus, when filling in the proposed service, they could provide a label for their proposal themselves. Although we doubt there was enough data for this experiment, it is an accessible extension to our work. With a decent amount of data, this model could allow us to predict the category for future respondents.

References

[AS19] A. Agrawal and A. A. Singh. “A Comprehensive Survey of Text Similarity Techniques” (2019).

[al18] F. A. Nascimento et al. “A Comparative Study of Text Similarity Measures for Content-Based Recommender Systems” (2018).

[al17a] Q. Li et al. “Service Recommendation Using Collaborative Filtering and Text Similarity Analysis” (2017).

[al17b] S. H. Kim et al. “A Hybrid Approach for Service Categorization based on Text Similarity and Semantic Relations” (2017).

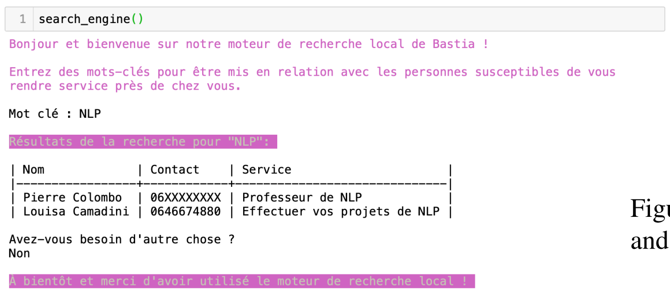
[C C22] F. Suchanek C. Chhun P. Colombo. “Of Human Criteria and Automatic Metrics: A Benchmark of the Evaluation of Story Generation”. *COLING* (2022).

[Hus18] M. H. Hussain. “Improving Service Categorization using Hierarchical Text Classification and Contextual Information” (2018).

[P C22] P. Piantanida P. Colombo C. Clavel. “Info LM: A New Metric to Evaluate Summarization and Data2Text Generation. Student Outstanding Paper Award”. *AAAI* (2022).

[Smi19] Smith. “Using Natural Language Processing to Improve Access to Social Services” (2019).

Appendix



```
search_engine()
Bonjour et bienvenue sur notre moteur de recherche local de Bastia !

Entrez des mots-clés pour être mis en relation avec les personnes susceptibles de vous
rendre service près de chez vous.

Mot clé : NLP

Résultats de la recherche pour "NLP":

| Nom              | Contact    | Service                        |
|------------------+------------+--------------------------------|
| Pierre Colombo   | 06XXXXXXXX | Professeur de NLP              |
| Louisa Camadini  | 0646674880 | Effectuer vos projets de NLP   |

Avez-vous besoin d'autre chose ?
Non

À bientôt et merci d'avoir utilisé le moteur de recherche local !
```

Figure 1: Output for the search engine *(TF-IDF and cosine similarity measure)*

```
Number of clusters: 9
Number of noise points: 1003
Noise points: 1003
Cluster 0 : 18
Cluster 1 : 30
Cluster 2 : 7
Cluster 3 : 7
Cluster 4 : 18
Cluster 5 : 15
Cluster 6 : 15
Cluster 7 : 9
Cluster 8 : 5
```

Figure 2: Output for the DBSCAN algorithm ($\epsilon = 1$ and $min\_samples = 7$)

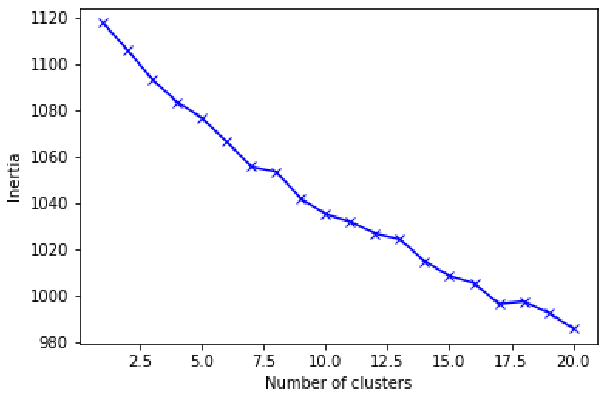


Figure 3: Elbow method

Table 1: List of clusters and associated services

Cluster 0	Coaching
Cluster 1	Art and Design
Cluster 2	Personal and Business Assistance
Cluster 3	Stay at home services!
Cluster 4	Management and Support
Cluster 5	Courses in all disciplines
Cluster 6	Fully customized services
Cluster 7	Gardening
Cluster 8	All local agents
Cluster 9	Care services

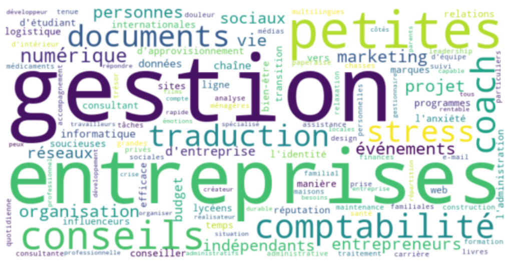


Figure 4: Word cloud for the 3rd cluster

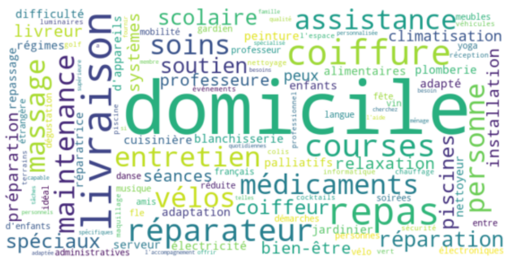


Figure 5: Word cloud for the 4th cluster

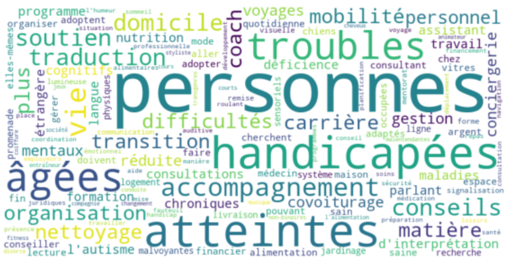


Figure 6: Word cloud for the 5th cluster

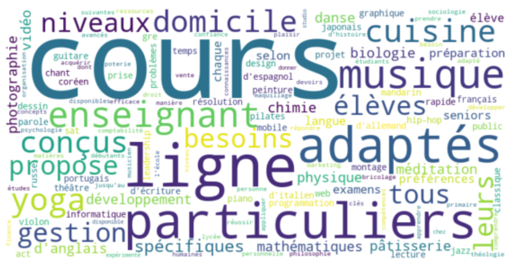


Figure 7: Word cloud for the 6th cluster