Categorization of B2B Service Offers: Lessons learnt from the Silex Use case
Molka Dhouib, Catherine Faron Zucker, Andrea Tettamanzi

To cite this version:
Molka Dhouib, Catherine Faron Zucker, Andrea Tettamanzi. Categorization of B2B Service Offers: Lessons learnt from the Silex Use case. 4ème conférence sur les Applications Pratiques de l’Intelligence Artificielle APIA2018, Jul 2018, Nancy, France. hal-01830905

HAL Id: hal-01830905
https://hal.archives-ouvertes.fr/hal-01830905
Submitted on 5 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Categorization of B2B Service Offers: Lessons learnt from the Silex Use case

Molka Tounsi Dhouib1,2 Catherine Faron Zucker1 Andrea Tettamanzi1

1 Université Côte d’Azur, Inria, CNRS, I3S, Sophia Antipolis, France
2 Silex France
{dhouib, faron, tettamanzi}@i3s.unice.fr

1 Introduction

The Silex France company offers a SaaS sourcing tool for identification of the service providers that are best suited to meet the service requests expressed by companies. The Silex platform is used by more than 5000 professionals to quickly identify and exchange with B2B service providers. Silex’s ultimate aim is to provide a network of qualified companies to service buyers combined with the best functionality to consult it.

In the framework of a collaborative project between Silex and the I3S research laboratory, we aim at introducing semantics into the B2B platform in order to enable automatic reasoning on service requests and offers and improve the recommendation of service providers.

As a first step, we are interested in automatically categorizing the textual description of companies, service requests and service offers. With data’s exponential growth, text categorization has become a crucial issue in Natural Language Processing (NLP). It consists in classifying texts according to their contents, into one or more predefined categories [2]. In recent years, the number of machine learning (ML) techniques that automatically generate text categorization has increased considerably [3].

In this paper we report some experiments we conducted to answer the Silex use case, using supervised ML techniques to classify Silex textual data into predefined categories. Meanwhile, we addressed the following questions:

(1) What is the best representation of Silex textual descriptions of service offers and requests to categorize it?
(2) What is the best ML algorithm to categorize Silex textual descriptions of service offers and requests?

This paper is organized as follows: Section 2 presents state-of-the-art text categorization methods. Section 3 introduces our approach of text categorization to answer the SILEX use case. Section 4 reports and discusses the results of our experiments of categorization of NL descriptions of service requests and offers. Section 5 draws some conclusions and directions for future work.

2 Related Works

Text categorization is the task of classifying data into a predefined set of categories. In other words, given a set of categories and a set of textual documents, text categorization is the process of automatically finding the correct category for each document [4].

2.1 Feature Vector Models

In the text categorization process, the first step consists in preprocessing textual documents, to convert them into feature vectors, a representation that can be automatically interpreted by machine. This step includes tokenization,
stemming and removing of stop words before the creation of feature vectors.

**Bag-Of-Words (BOW).** is the most common feature vector model. In this model, the features are the frequencies of each word in the textual document. The feature space’s dimension is the number of all different words in all documents [4]. The limitation of this model is that it ignores the semantic relations between words.

**Word Embedding (WE).** In order to overcome the weakness of BOW, a new model called Word Embedding (WE) has been successfully used in several NLP tasks. Word embeddings are projections in a continuous space of words that preserve the semantic and syntactic similarities between them [6]. There are many models that can produce a word embedding. In the following, we introduce the most important ones.

[7] proposes Word2Vec, a popular tool that produces word embeddings based on two models: CBOW and Skip-gram. CBOW is a Neural Network and log-linear model. It removes the non-linear hidden layer, and projects the contextual words on the same position. Word’s prediction is obtained according to its past and future contexts. The process consists in computing the average of the contextual word vectors and running a log-linear classifier on the averaged vector.

Skip-gram is similar to CBOW and is also a Neural Network and log-linear model. Contrary to CBOW, Skip-gram predicts the contextual words given the current word [6]. Another model was introduced by [8] and is called GloVE. This model is based on global matrix factorization that calculates the co-occurrence of words in the corpus [6]. Finally, FastText ¹ is a library for learning word representations and sentence classification. FastText published pre-trained word vectors for 294 languages, trained on Wikipedia. These vectors with dimension 300 were obtained using the skip-gram model [13].

### 2.2 Binary classification

After the construction of feature vectors, the second step of text categorization is the learning step. We present here some of the many learning algorithms for binary classification and multi-label classification.

**Naïve Bayes (NB) [9].** is a supervised, probabilistic algorithm based on Bayes theorem and the hypothesis of the independence between features. This algorithm is powerful due to the independence between features that ignores features’ order, and consequently, the presence of a feature does not affect other features in classification tasks [14].

**Support Vector Machine (SVM) [10].** is one of the most common and successful supervised classification algorithm used for text classification tasks [14]. This algorithm transforms training data into higher dimensions and searches for linear optimal separating hyperplane [1].

**Neural Networks (NN).** The NN is composed by many layers to perform text categorization. The perceptron is the simplest kind of a neural network and it has only two layers: the input nodes receive the feature values, the output nodes produce the categorization status values, and the link weights represent dependence relations. The NN text categorization process starts by loading feature weights into the input nodes; the categorization’s final result is present in the output nodes after the propagation of the activation of the nodes forward through the network. The neural networks are trained by back propagation in order to minimize the error. In case of misclassification, the error is propagated back through the network and modifies the link weights [15].

### 2.3 Multi-label classification

Multi-label classification is an approach to classification problems that allows each data point to be assigned to more than one class at the same time. There are two multi-label classification methods: (i) problem transformation and (ii) algorithm adaptation [11]. Here we describe the most popular algorithms of each category.

**Problem transformation.** A first approach consists in transforming the multi-label problem into one or more single label classification problem. Among the methods adopting this approach, we can highlight:

- **The Binary Relevance method (BR)** is based on one-vs-all ensemble approach. BR transforms any multi-label problem into binary problems to predict the relevance of each label to a data point. All binary classifiers are then aggregated to form a set of relevant labels [11].

- **The label PowerSet method (LP)** considers the multi-label problem as a single multi-class classification problem. In order to create a transformed multi-class dataset, each combination of relevant labels is mapped to a class [11].

- **The Random K-label set (RAKEL)** aims at finding a better balance between BR and LP. RAKEL generates a series of label subsets and builds a label powerset model for each of them [11].

**Algorithm adaptation.** Algorithm adaptation extends specific algorithms to carry out multi-label classification.

- **Multi-label lazy algorithm (ML-KNN)** is one of the most famous adaptation algorithms. MLKNN uses the maximum a posteriori principle in addition to the K-nearest neighbour algorithm [11].

- **Multi-label decision tree (ML-DT)** is an extension of C4.5 decision tree algorithm. ML-DT allows the creation of multiple labels in the leaves, and chooses node splits based on a pre-defined multi-label entropy function [11].

¹[https://github.com/facebookresearch/fastText](https://github.com/facebookresearch/fastText)
• AdaBoost is based on the addition of simple classifiers to a pool and the use of their weights to define the final classification [16].

3 Experimental methodology
To answer the Silex use case, we conducted some experiments to compare the performance of the above described state-of-the-art feature vector models and learning algorithms for the categorization of textual descriptions of service offers or requests.

3.1 Dataset
Silex distributes a service to purchasing departments that support the sourcing system within the company. Silex users can describe their company (company description), their needs (service request description) and the description of the offers provided by the company (service offer description) in Silex web application. All descriptions are written in French.

The dataset considered in our experiments comprises 3188 company descriptions, 580 service offers descriptions and 155 service requests descriptions.

Six main categories are defined by Silex to classify these descriptions. One description can be categorized in one or more classes. The categorization phase is done manually by experts in the current version of the Silex platform. Table 1 shows the distribution of all the descriptions into the different categories. We observe that the distribution of our dataset is unbalanced, for instance the Informatique (Information Technology) class has 915 more companies descriptions than the Services_industriels (Industrial Services) class.

<table>
<thead>
<tr>
<th>Categories</th>
<th>Companies descriptions</th>
<th>service offer descriptions</th>
<th>service request descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informatique</td>
<td>1043</td>
<td>147</td>
<td>42</td>
</tr>
<tr>
<td>Finance</td>
<td>393</td>
<td>40</td>
<td>13</td>
</tr>
<tr>
<td>Services généraux</td>
<td>929</td>
<td>149</td>
<td>65</td>
</tr>
<tr>
<td>Marketing</td>
<td>926</td>
<td>160</td>
<td>18</td>
</tr>
<tr>
<td>Ressources humaines</td>
<td>297</td>
<td>57</td>
<td>7</td>
</tr>
<tr>
<td>Services industriels</td>
<td>128</td>
<td>32</td>
<td>19</td>
</tr>
</tbody>
</table>

Table 1: Distribution of the textual descriptions in the corpus.

3.2 Implementation
In our implementation, we focused first on two methods to construct our feature vectors:

• The BOW representation is achieved by applying a traditional process of tokenization, stop words removal and lemmatization. Then we use the TFIDF to construct the matrix. This matrix will be used in different algorithms.

• There are two methods to obtain Word embeddings representation: (i) build word embeddings from scratch using a model like word2vec or FastText. This approach needs a very large corpus to train these word embeddings; and (ii) use a pre-trained word embeddings in any desired language. For our case, we used the latter approach to obtain word embeddings because we do not have a large enough dataset. After studying the different pre-trained word embeddings, we decided to use the FastText model because it is the only model that provides French word embeddings pre-trained on a Wikipedia dump. This word embedding model 2 contains 1,152,449 tokens, which are mapped to vectors in a space with 300 dimensions.

Our approach to produce the feature vectors representing the SILEX textual descriptions is inspired by [12] and [17]: As illustrated in Figure 1, the first step is to convert textual descriptions to lower case and remove all French stop words. Then the words are checked against the pretrained French vector. If the word exists in the dictionary, the vector representation of description is constructed by averaging the word embeddings vectors along each dimension for all the words in the description. If the dictionary does not contain the word, the zero vector is returned:

\[
\text{DescriptionWordEmbeddings} = \sum_{i=1}^{k} \frac{w_i}{k}
\]

where \( k \) is the number of words in a description and \( w_i \in \mathbb{R}^n \) denotes the word embeddings vector of the \( i \)th word. The vector representation of each description has exactly the same dimensions as the word embeddings vector \( w_i \) [17]. The vector representations of each description are then used as features and input for each classifier.

In order to test our feature vectors, we trained two methods of categorization: (i) three binary classification algorithms (NB, SVC, RNN) and (ii) three multi-label classification algorithms: two of them are problem transformation methods (Binary Relevance, Label PowerSet) and three of them are algorithm adaptation methods (Multi-label lazy algorithm (ML-KNN)). All these algorithms are implemented using scikit-learn\(^3\), and executed on a machine with 8 GB of memory and an Intel Core i7-7500U CPU.

To deal with the unbalanced data, there are two main state-of-the-art approaches to adjust the distribution of dataset: (i) over-sampling adds copies of instances from the under-represented class. The most common technique of over-sampling is called Synthetic Minority Over-sampling Technique (SMOTE), (ii) under-sampling deletes instances from the over-represented class. In our implementation, we tested first a very large parameter with a grid function in scikit-learn to identify the best parameters for each classifier and the best method to deal with unbalanced data. We obtained the best results with SMOTE method for SVM algorithm, and RandomOverSampler method for NB. Table 2 shows all parameters used in binary classification and Table 3 shows all parameters used in multi-label classification.

### Binary algorithms.
To evaluate binary classification algorithms, we used the F1 score measure, defined as the harmonic mean between precision and recall:

\[
F_{1}\text{score} = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \times |h(x_i) \cap y_i|}{|h(x_i)| + |y_i|}
\]

### Multi-label algorithms.
To evaluate multi_labels algorithms we used the following metrics:

- **Accuracy** allows to compute the percentage of correctly predicted labels among all predicted and true labels. It is defined as follows:

\[
\text{Accuracy}(h) = \frac{1}{N} \sum_{i=1}^{N} \frac{|h(x_i) \cap y_i|}{|h(x_i)| + |y_i|}
\]

- **Micro-precision** is defined as the precision averaged over all the example/label pairs:

\[
\text{Micro-precision} = \frac{\sum_{j=1}^{Q} \text{tp}_j}{\sum_{j=1}^{Q} \text{tp}_j + \sum_{j=1}^{Q} \text{fp}_j}
\]

where \(\text{tp}_j, \text{fp}_j\) are the number of true positives and false positive for label \(\lambda_j\).

- **Micro-recall** is defined as recall averaged over all the example/label pairs:

\[
\text{Micro-recall} = \frac{\sum_{j=1}^{Q} \text{tp}_j}{\sum_{j=1}^{Q} \text{tp}_j + \sum_{j=1}^{Q} \text{fn}_j}
\]

where \(\text{fn}_j\) is the number of false negatives for the label \(\lambda_j\).

- **Micro-F1** is the harmonic mean between micro-precision and micro-recall and is defined as:

\[
\text{Micro-F1} = \frac{2 \times \text{micro-precision} \times \text{micro-recall}}{\text{micro-precision} + \text{micro-recall}}
\]

- **Hamming loss** allows to evaluate how many times an example-label pair is misclassified. The performance is perfect when the value of this metric is equal to 0. Hamming loss is defined as:

\[
\text{Hamming loss}(h) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{Q} |h(x_i) \Delta y_i|
\]

where \(\Delta\) is the symmetric difference between two sets, \(N\) is the number of examples and \(Q\) is the total number of possible class labels.

---


### Table 2: Parameters used for binary classification

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Unbalanced data</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>RandomOverSampler</td>
<td>penalty=12, alpha=0.0001</td>
</tr>
<tr>
<td>SGDClassifier</td>
<td>SMOTE</td>
<td>C=0.1, penalty='l2', loss='squared hinge'</td>
</tr>
<tr>
<td>LinearSVC</td>
<td>SMOTE</td>
<td>C=1000.0, gamma=0.0001</td>
</tr>
<tr>
<td>SVC</td>
<td>SMOTE</td>
<td>kernel='rbf', C=1000.0, gamma=0.0001</td>
</tr>
<tr>
<td>RNN</td>
<td>balanced</td>
<td>hidden size=32, Activation('relu'), Activation('sigmoid')</td>
</tr>
</tbody>
</table>

### Table 3: Parameters used for multi-label classification

<table>
<thead>
<tr>
<th>Classifier</th>
<th>multi-label methods</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSVC</td>
<td>Binary Relevance</td>
<td>C=1, penalty='l2'</td>
</tr>
<tr>
<td>LSVC</td>
<td>Label Powerset</td>
<td>C=1, penalty='l2'</td>
</tr>
<tr>
<td>MLkNN</td>
<td>Algorithm adaptation</td>
<td>-</td>
</tr>
</tbody>
</table>

---

## 4 Experiments and discussion

### 4.1 Evaluation procedure

The experiments were conducted using the 5-fold cross-validation methodology. To evaluate our experiments, we calculated the classification duration and various evaluation measures inspired by [18] and described in the following.
4.2 Results and Analysis

The results of binary classification algorithms reported in Table 4, Table 5 and Table 6 show that there is not a significant difference between the results with a BOW representation and the results with a word embeddings representation. For example, for category "Informatique", we obtain the same precision value that equals to 0.82. The recall value equals to 0.82 for BOW representation, and is a bit better compared to WE recall value that equals to 0.79. The classification processing time is much better for WE (0.48 seconds) than for BOW (1572 seconds). This is due to the BOW expansive phases of lemmatization and matrix building compared to the vector representation of word embeddings.

Table 4: Best classifier of binary classification of companies descriptions

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informatique</td>
<td>LSVC + WE SGD+BOW</td>
<td>0.82</td>
<td>0.79</td>
<td>0.80</td>
<td>0.48</td>
</tr>
<tr>
<td>Finance</td>
<td>LSVC+WE NB+BOW</td>
<td>0.82</td>
<td>0.82</td>
<td>0.82</td>
<td>1572</td>
</tr>
<tr>
<td>Services généraux</td>
<td>LSVC + WE NB + BOW</td>
<td>0.87</td>
<td>0.84</td>
<td>0.85</td>
<td>1925</td>
</tr>
<tr>
<td>Marketing</td>
<td>LSVC + WE SGD + BOW</td>
<td>0.82</td>
<td>0.80</td>
<td>0.81</td>
<td>1517</td>
</tr>
<tr>
<td>Ressources humaines</td>
<td>LSVC + WE NB + BOW</td>
<td>0.91</td>
<td>0.83</td>
<td>0.86</td>
<td>1904</td>
</tr>
<tr>
<td>Services industriels</td>
<td>LSVC + WE NB+BOW</td>
<td>0.94</td>
<td>0.89</td>
<td>0.91</td>
<td>1914</td>
</tr>
</tbody>
</table>

We can draw similar conclusions when comparing multi_label classification algorithms, with a slight advantage to BOW compared to word embeddings in terms of accuracy and hamming_loss but not in term of time.

If we compare binary classification and multi_label classification algorithms, we can see that the results of the first algorithm are better than the second one. These results are as expected because first we don’t have a very large corpus, and second when we use a binary classification we can play as expected because first we don’t have a very large corpus, algorithm are better than the second one. These results are significant difference between the results with a BOW representation. For example, for category "Informatique" we obtain the same precision value that equals to 0.82. The recall value equals to 0.82 for BOW representation, and is a bit better compared to WE recall value that equals to 0.79. The classification processing time is much better for WE (0.48 seconds) than for BOW (1572 seconds). This is due to the BOW expansive phases of lemmatization and matrix building compared to the vector representation of word embeddings.

on the classifier parameters for each category to enhance its performance, which is not the case for multi_label classification. The different binary classification algorithms perform more or less well in categorizing our dataset using the same representation. For example, only the LSVC algorithm categorized correctly this text as "informatique" "Ascot accompagne les dirigeants et cadres de PME dans la définition, la maitrise et le suivi de leur transformation numérique. Plus aucune organisation ne peut ignorer l’impact global sur son métier, ses outils et ses processus. Notre expérience, de plus de 20 ans, en Conseil, Assistance technique, Développement, Ingénierie, ... est entièrement dédié à nos clients. Sous contrat ou à la carte, nos prestations leur apportent valeur et sérénité."

We obtained also different results when comparing the BOW and WE representations. For example, using the same LinearSVC algorithm, this text "Agence WebMarketing 360 de nouvelle génération exploitant de nombreux leviers marketing pour optimiser vos campagnes digitales & booster votre chiffre d’affaires Kalipseo développe pour vous les stratégies digitales les plus pertinentes et s’emploie à promouvoir l’image de votre marque." was wrongly categorized as "Informatique" when using the WE representation, but the categorization was right when using BOW representation.

As a conclusion, to best answer the Silex use case, we decided to use a binary classification algorithm with a word embedding representation. Table 8 shows the algorithms that are selected to classify each category of Silex data.
Table 8: Selected algorithms for Silex Company

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Representation</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informatique</td>
<td>Companies</td>
<td>WE</td>
<td>NB</td>
</tr>
<tr>
<td></td>
<td>Service offer</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Service provider</td>
<td>WE</td>
<td>NB</td>
</tr>
<tr>
<td>Finance</td>
<td>Companies</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Service offer</td>
<td>WE</td>
<td>SVC-RBF</td>
</tr>
<tr>
<td></td>
<td>Service provider</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td>Services</td>
<td>Companies</td>
<td>WE</td>
<td>NB</td>
</tr>
<tr>
<td>généraux</td>
<td>Service offer</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Service provider</td>
<td>WE</td>
<td>NB</td>
</tr>
<tr>
<td>Marketing</td>
<td>Companies</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Service offer</td>
<td>WE</td>
<td>SGD</td>
</tr>
<tr>
<td></td>
<td>Service provider</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td>Ressources</td>
<td>Companies</td>
<td>WE</td>
<td>NB</td>
</tr>
<tr>
<td>Humaines</td>
<td>Service offer</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Service provider</td>
<td>WE</td>
<td>NB</td>
</tr>
<tr>
<td>Services</td>
<td>Companies</td>
<td>WE</td>
<td>RNN</td>
</tr>
<tr>
<td>industriels</td>
<td>Service offer</td>
<td>WE</td>
<td>SVC-RBF</td>
</tr>
<tr>
<td></td>
<td>Service provider</td>
<td>WE</td>
<td>RNN</td>
</tr>
</tbody>
</table>

5 Conclusion

In this paper we reported the results of the experimental evaluations of various vector feature models and machine learning algorithms conducted to answer the real-world use case of the Silex company: how to categorize textual descriptions of service offers and requests with the ultimate goal of recommending service providers that better answer services requests. We compared the two main state-of-the-art feature vector models. A BOW representation is a bit better than a WE representation regarding the evaluation measures. This is due to the use of a generic pre-trained vector for WE descriptions, that is based on Wikipedia dump and that does not cover well our data set. On the other hand, a WE representation has the advantage of being less time consuming because of the additional lemmatization and matrix building phases specific to the BOW representation. Based on these results, we choose to use classifiers with word embeddings data for Silex company as shown in Table 8.

As future work, we aim to define a specific B2B pre-trained vector that best covers the Silex dataset instead of using a general reference like Wikipedia. We also aim to study a combined approach based on both word embeddings and knowledge engineering approaches. The objective is to improve the representation of texts by introducing semantics to increase the performance of algorithms. To achieve this, we are going to use our built ontologies to represent the sourcing domain, and we are going to enrich the original texts with additional information based on semantics relations between concepts such as generalization or specification.

6 Acknowledgement

This work is supported by the ANRT, CIFRE I3S-SILEX 2016/0947

References


