December 9, 2022

Classification of Bank Transactions

Juan Barragan, Acme Corp

Abstract
In this note we explore how to establish categories within banking transactions.

Contents

1. Introduction

At Acme Corp, we finance digital companies, and we analyze the risk of such positions by looking at their data. Particularly their banking transactions. Our goal of this note is to classify banking statements on similar labels so to have a broad view on how a given client expends and earns its money. Particularly important are patterns showing that client pays taxes, salaries and has or not payment incidents

As said before (see [1]), there's a number of tools for studying text on the scientific literature. But our texts are composed by a complicated combination of keywords used by banks combined with some explaining text, dates and a lot of times, numbers. There is no semantic structure on this, not at least as we are used to in natural languages like plain English.

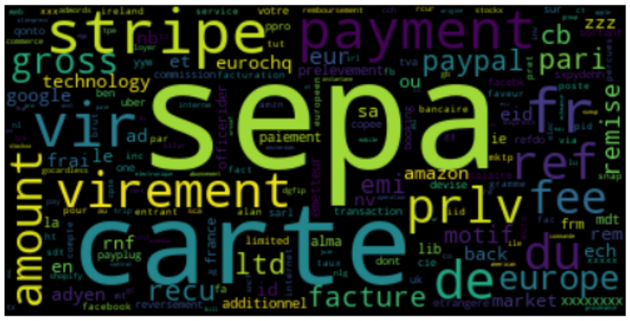

Figure 1. Word-cloud of bank transaction tokens

2. Neural Networks

As on [1] we are going to use *Neural Networks* for realizing embeddings into vector spaces. We use *skipgrams*, let us recall the followings: a vocabulary $V = \{w_i | i = 1, \dots, v\}$ is a suite of v words. Each word is simply an arbitrary concatenation of chars $w = a_1 a_2 \dots a_k$. We embed each word using for example the *One Hot Encoding*, namely each word w_i is mapped to a vector $w = (\delta_i)$ with the entries all equal to zero except the i-th one.

A bank description is a *context*. Write $\mathscr{C}$ for all the contexts. We would like to measure how frequent adjacent words appear. Say how often a word w_i appears with word w_k on a given context. More precisely, The idea behind of a skipgram is that for a given word w, what is the probability of having that word in the middle of a given context of length $2\kappa + 1$, say $C_\kappa(w) = [w_{-\kappa}, \dots, w, \dots w_\kappa]$?

For that case, we define the following Neural Network:

$$(\delta_i) \to (\delta_i) \sum_{\kappa=1}^{c} \frac{\text{count}_{C_\kappa \in \mathscr{C}}(w_i \in C_\kappa)}{\#\mathscr{C}}$$

As a recall, the embedding, given the evidence, is looking for the optimal factor allowing for the greatest probability of finding words on a given context. And that's exactly what we are looking for. This procedure has an important consequence: *word found frequently together are euclidian metrically closer*

Some final words concerning our particular setting. For our study case, we proceed as follows

Clean bank descriptions of accents, extraneous chars and numbers.

Tokenize the bank statement (split them by spaces) We keep only words having at least three chars.

Choose the longest context as the dimension of the embedding using skipgrams.

Embed a full context C as the mean of its embedding:

$$e(C) = \frac{1}{\#C} \sum_{w \in C} e(w)$$

where $e(w)$ is the individual word embedding.

We hence have now a mapping $i : \mathbb{B} \to \mathbb{R}^d$ from the set of banking transactions to a real vector space such that words which are *frequently* present on similar bank statement contexts are translated to vectors which are *metrically* closer. This affirmation is called *The Distributional Hypothesis*, you can have more details about this methodology on [3].

3. Decision Trees

Decision Trees are classification methods which partition the feature space into disjoint pieces. These disjoint pieces can be assimilated to labels. As the name suggest they look as trees, have a look to figure 2 for a classification tree on the popular *Wine Dataset*

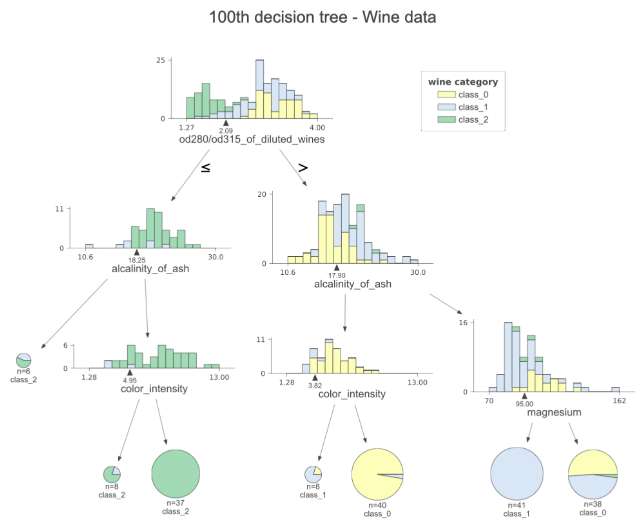

Figure 2. Simple tree for dataset wine

Lets analyze at how these threes are build. Say that we have only two labels $Y = \{0, 1\}$ and only one feature $X : \Omega \to \mathbb{R}$ from a probability space Ω to the real numbers. We chose a point θ dividing the real line into two subsets $A_0 = (-\infty, \theta]$ and $A_1 = (\theta, \infty)$, and we classify guys on A_i as belonging to category $i \in Y$. Count for each class the proportion of how members are placed:

$$e_{k,j} = \frac{\#\{X_i | X_i \in A_k, Y_i = j\}}{\#\{X_i | X_i \in A_k\}}$$

Let $\gamma_k = 1 - (e_{k,0}^2 + e_{k,1}^2)$. Of course $e_{k,0} + e_{k,1} = 1$ so $0 \leq \gamma_k \leq 1$. We define the ***impurity*** or ***Gini Index*** as:

$$G(\theta) = \gamma_0 + \gamma_1$$

The usual step here is the to optimize for θ, hence the optimal first split for the tree is given by:

$$\arg\min_{\theta} G(\theta)$$

In words, we look for the θ which splits the real line such that the obtained classes have the *less* impurity. We can move forward down the tree and consider either the same feature either a randomly chosen one if we have more features.

A *Random Forest* is just a set of trees where the classification task is agreed by voting, the so-called *ensemble learning*.

In our case we *cross validate* the following (Which by the way is a good rule of thumb). The number of trees is equal to the number of categories. The maximal depth of the tree is the number of features.

4. Algorithm

So here's the summary of the algorithm used:

Clean bank descriptions of accents, extraneous chars and numbers.

Tokenize the bank statement (split them by spaces) We keep only words having at least three chars.

Choose the longest context as the dimension of the embedding using skipgrams.

Embed a full context C as the mean of its individual word embedding:

$$e(C) = \frac{1}{\#C} \sum_{w \in C} e(w)$$

where $e(w)$ is the individual word embedding.

add to the embedding the sign of the transaction.

Use a Random Forest for classification.

The number of trees is equal to the number of categories.

The maximal depth of the tree is the number of features.

Minimize the Gini impurity.

5. Results

On the figure 3 we can appreciate the *Confusion Matrix* as a *heatmap* for a randomly chosen test set on a trained model. Recall that, by definition, the confusion matrix is composed by the values $C_{i,j}$ which equals to the number of observations known to be in group i and being predicted on group j.

The diagonal line on figure 3 is a good sign: the number of good predictions is bright. Just as a warning: this confusion matrix was plotted using *cubic normalizing*, meaning roughly that we rater plot $\sqrt[3]{C_{i,j}}$. This as some categories are overwhelmingly present.

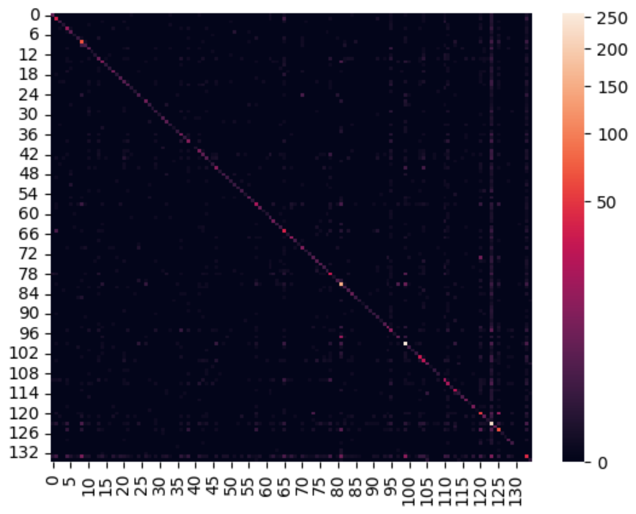

Figure 3. Confusion matrix

References

[1] J. Barragan Clustering of Banking Transactions at Acme Corp *Slvr*

[2] C. Bishop Pattern Recognition and Machine Learning *Springer*

[3] J. Eisenstein Introduction to Natural Language Processing *MIT press*

[4] T. Hastie, R. Tibshirani and J. Friedman The Elements of Statistical Learning *Springer*

[5] T. Mikolov, K. Chen, G. Corrado, J. Dean Efficient Estimation of Word Representations in Vector Space *arXiv:1301.3781*

[6] X. Rong word2vec Parameter Learning Explained *arXiv:1411.2738*

[7] L. Wasserman All of Statistics *Springer*