Instituto Tecnológico Autónomo de México

Tesis de Farid

TESIS

QUE PARA OBTENER EL GRADO DE

Maestro en Ciencias en la Computación

PRESENTA

ANGEL FARID FAJARDO OROZ

ASESOR

Víctor Manuel González y González

CIUDAD DE MÉXICO 2017

"Con fundamento en los artículos 21 y 27 de la Ley Federal de Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada "**Tesis de Farid**", otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la biblioteca Raúl Baillères Jr., autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por tal divulgación una prestación"

Angel Farid Fajardo Oroz

_____

Fecha

_____

Firma

A mi gente.

Agradecimientos

A la otra gente.

Resumen

Una tesis muy bonita.
wiiiii!!!

Abstract

The number of proposals of business model has been increasing in the last years, in that way that some organizations that give support to those *start-ups* need to select the “”best ideas“” with high chances of success, among a wide variety of applications. For this reason, we propose a methodology for ranking business model applications using text mining techniques and a fuzzy rule based system. We evaluated this proposed methodology using real data applications from a company that evaluates those business proposals.

The applications we ranked consisted in questions and answers, but the business model description is written as the response to some open questions as ”What is the need or problem you want to solve with your business idea?.ºr ”What population are you benefiting/helping with your business idea?”. For this reason, Natural Language Processing techniques had been applied in order to extract meaningful semantic features from texts, mapping some key-words from applications into a vectorial space. Then, we built an expert system whose rules were created taking into account the technological, social and economical impact of the business idea. Finally, the applications were sorted according to a given score that was assigned by our expert system.

On the other hand, it was trained a system in order to classify “good applications” and “not so good” aplications. We tested different models and selected those with best results; the labels were assigned according to some score given by expert human judges that assigned an overall score to each one application.

. . .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Business models can be understood as *stories that explain how enterprises work* [1], this histories have to take into account customers, what are the needs of customers and how this model is going to make money. But, there is also an extra factor that helps to improve the probability of successful of a business, and this factor is known as *Design Innovation*.

Design Innovation, or *Experience Innovation*, are the ideas and methodologies for improving business models, encouraging innovation and growth [**liedtka2015perspective**]. This approach suggest that an idea or business can achieve innovation if it is feasible (technology component), viable (can make money) and it is usable or desirable (human value of the idea).

On the other hand, the number of proposal of business models has been increasing in the last years, and most of this ideas can be supported by *startup founders*. Most of these startup founders are looking for great business ideas, providing them support and encouraging the development of new companies.

For this reason, we developed this work in order to create a methodology to identify interesting business models with high chances of success. This paper describes a proposed methodology that ranks business models applications, based on some *Design Innovation* concepts [**liedtka2015perspective**]. The rest of the paper

is organized as follows. In Section 2 we briefly review some concepts of Natural Language Processing (NLP), Text Mining techniques and Fuzzy Inference Systems (FIS). In section 3, we describe the steps of our proposed methodology for ranking Business Models applications. In section 4 we present experimental results using real data. Finally, conclusions are discussed in Section 5.

Chapter 2

Related work

lalala

2.0.1. Text mining

Text mining (*"knowledge discovery from textual databases"* [**tan1999text**]) is the process and techniques used to extract relevant and non trivial information from unstructured text data. In general, text data can be analyzed using lexical, sintactic or semantic approaches (for example, *Part-of-Speech tagging*) or a vector space model). In order to extract meaningful information from text, they are usually represented according to their semantic rather than using string-based approaches [**aggarwal2012mining**].

Some fields related to text mining involves information retrieval, text analysis, information extraction, clustering, categorization, visualization, database technology, machine learning and data mining [**tan1999text**].

2.0.2. Empirical Laws in Linguistics

There are some empirical laws in linguistics that try to explain some features of words in NLP context, specially in text documents.

Zipf's law establishes an approximate mathematical relation between the frequency of occurrence of each word and its rank in the list of all words used in a text ordered by decreasing frequency [**montemurro2001beyond**]. The Zipf's law states the following relation:

$$f(s) \propto \frac{A}{s^{\alpha}} \tag{2.1}$$

where α is slightly grater that 1, A is a normalizing constant, s is an index of a given word, and *f(s)* is the frequency of a word. This law can at most account for the statistical behavior of words frequencies in a limited zone (middle-low to low range of the rank variable) [**montemurro2001beyond**].

Mandelbrotś law introduces a modification to the Zip's law, by using arguments on the fractal structure of lexical trees. The only improvement over the original Zip's law is that it fits better to regions corresponding to lowest ranks (s $\leq$ 100), dominated by function words [**montemurro2001beyond**]:

$$f(s) \propto \frac{A}{(1+Cs)^{\alpha}} \tag{2.2}$$

where *(*C) is a second parameter that needs to be adjusted to fit the data.

It has been argued that using these laws it is possible to discriminate between human writings and stochastic version of texts, just analyzing the statistical properties of words that fall beyond the scope where (Eq. 2.2) holds [**cohen1997numerical**].

Using these empirical laws, we can generate models in order to weight the importance of certain words, based on the frequency of words itself in a given text. Actually, *stop-words* are considered with low relevance, because they have nothing semantically important about a document [**wilbur1992stopwords**].

2.0.3. Word Embeddings

Semantic word representation is based on the assumption that words with similar meanings are placed in similar contexts. In order to use a vector space model, it has been proposed to represent words as dense vectors derived by training methods inspired in neural-network language modeling [**compositionality2013Mikolov**] . These kind of representations are referred to as *"Neural Embeddings"* or *"Word Embeddings"*.

The main advantage of representing texts using word embeddings is that they are easy to work with (they are efficiently computational throw a low-dimensional matrix operations). On the other hand, word embeddings are considered opaque, in the sense that it is hard to assign meanings to the dimensions of the induced representation [**compositionality2013Mikolov**].

I. Skip-Gram Model

The skip-gram neural model introduced by Mikolov et al. was trained using the negative-sampling procedure. In this model, each word $w \epsilon W$ is associated with a vector $v_w \epsilon R^d$ and similarly each context $c \epsilon C$ is represented as a vector $v_c \epsilon R^d$, where W are the words in vocabulary, C is the context vocabulary, d is the embedding dimensionality. The entries in vectors are latent, and treated as parameters to be learned. So, we are expecting parameters values such that the dot product between vectors $v_w \cdot v_c$ produce an associated value, where a "good" word-context pair is maximized.

The objective function in this algorithm can be summarized as:

$$argmax_{v_w,v_c}(\sum_{(w,c)\epsilon D} log\sigma(v_c \cdot v_w) + \sum_{(w,c)\epsilon D'} log\sigma(-v_c \cdot v_w)) \tag{2.3}$$

So, optimizing this objective function, where $\sigma(x) = 1/(1+e^x)$, let us observe word-context pairs that have similar embeddings, while scattering unobserved pairs. Also, words that appear in similar context should have similar embeddings.

An alternative of the Skip-Gram model proposed by [**compositionality2013Mikolov**], is to replace the bag-of-words (linear contexts) with arbitrary context [**levy2014dependency**].

2.0.4. Compositional Distributional Semantics

Semantics of words can be approximated by the patterns of co-occurrence of words in a corpus (statistical semantics as *GloVe* proposes [**pennington2014GloVe**]), or through formal semantics, where the idea of compositionality can be reached in terms of a syntax-driven calculus [**baroni2014frege**].

The term of *Semantic compositionality* is about a complex expression, which is a function of the meaning of its constituent parts and how they are combined. For example, most of the phrases in NLP are composed by a noun and a verb, or an adjective and a noun, such that when they are joined they acquire a special value.

Mikolov et al. explored this approach in a multidimensional vector space where semantics of words are represented. He found that using neural networks to represent words, they encode linguistic regularities and patterns [**compositionality2013Mikolov**]. He also found (using an empirical approach) that those patterns have a linear translation, in this way, using simple vector arithmetics on word vectors, some pattern preserve a linear structure. For example, word("Berlin")-word("Germany") + word("France") is closest to word("Paris"), using the cosine distance.

I. Composition by vector mixtures

The most intuitive way of composing two word represented by vectors is by adding them (*additive approach*) or multiplying them (*multiplicative approach*). The

weighted additive model, is when vectors are modified by a scalar value before summing them [**mitchell2010composition**], this approach works well in tasks of predicting human similarity judgments about adjective-noun, noun-noun, verb-noun and noun-verb phrases.

The components of additive vectors inherit the cumulative score mass from the input components. Multiplication, captures the interaction between the values in the input components.

2.0.5. Singular Value Decomposition (SVD)

Singular Value Decomposition is a theorem from lineal algebra that says that any m x n matrix M whose entries are real numbers, can be decomposed into three matrices $(U_{m,m}, S_{m,n}, V^*_{n,n})$, such that:

$$M = U_{m,m}, S_{m,n}, V^*_{n,n} \tag{2.4}$$

The matrix $V_{n,n}$ is diagonal, and its values in the main diagonal after applying the SVD are called *singular values of M*, and they are ordered from greatest to least along the main diagonal of D.

The singular values represent the "dimensions of meaning" for words and passages, when words are represented in a *semantic vectorial space*. So, SVD is the base of the LSA (Latent Semantic Analysis) [**landauer1998introduction**], used for indexation and information retrieval.

It is common to use the cosine distance (Eq. 2.5) between words mapped into a vector space (A,B are two vectors), in order to measure the similarity between them [**manningh**]. one string into the other.

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.5}$$

2.0.6. Spelling correction of words

Typos and misspelling words are commonly done when someone is filling out an application. For this reason, an algorithm for correcting misspelling should be used, in order to acquire the complete semantic of a sentence without ignoring "inexistent" words that could be important. Otherwise, uncommon words (like those that are misspelled) should be discarded (according to Zipf's law, words with low frequency are not semantically important).

Using Peter Norving's algorithm for spelling corrections let us discard some words from applications that seem to be meaningless, changing them for most appropriate words. In essence, this algorithm computes a set of *similar* words, and proposes a candidate that maximizes the probability that this propose is the intended correction, given the original word.

$$argmax_{cecandidates} P(c|w) \tag{2.6}$$

Using the Bayes's Theorem:

$$argmax_{cecandidates} P(c)P(w|c) \tag{2.7}$$

In this case, the factor P(w) in the denominator was omitted, because it is the same for every possible candidate *(*c).

The expression 2.7 has four parts:

1. Argmax.- It is used to select the best candidate, which is the one with highest probability computed.
2. c ϵ candidates .- It indicates a candidate correction to consider from a set.
3. P(c) .- It can be seen as the language model. This indicates the probability that a word c appears as a word in the language.

4. P(w | c).- This is the error model, which indicates the probability that *w* would be typed in a text when it should meant *c*.

The set of candidate words for this algorithm are computed using a *Levenshtein Distance* of 1 and 2, and the proposed set of words are checked if they appear in the vocabulary of the language.

I. Levenshteing Distance

. The Levenshtein distance (LD), also known as *Edit distance*, is a measure of similarity between two strings. This distance is computed based on *simple edition* to a word (it can be a **deletion**,**transposition**, **replacement** or **insertion**) required to transform one string into the other.

The set of words created using a Levenshteing distance of 1 is huge. For example, for a word of length *n*, we have *n* deletions, *n - 1* transpositions, *26n* alterations and *26(n + 1)* insertions, giving a total of *54n + 25* candidate words [**NorvingSpelling**].

2.0.7. Fuzzy Rule Based System

Fuzzy logic emerged since 1965 [**Zadeh1965fuzzy**], and it is commonly used in many controlling processes, electronics devices, diagnosis systems and expert systems. This kind of logic is not restricted to zeros and ones (0,1) (it can be considered as an extension of multivalued logic). This kind of systems are useful when we want to make qualitative knowledge useful.

Usually, this qualitative knowledge is expressed in the form of rules like *If(statement) then (conclusion)*, where the statement and the conclusion use *linguistic variables*. These linguistic variables describe certain quantities from data, for example, 40 °C can be transform into the linguistic value *"hot"*.

In this case, the Fuzzy Rule Based System should approximate to a human being in a decision making process. In this way, the system has to take in to account

the combination of observations and the weighing up between the inputs to the system.

In order to express inputs and ouptus as linguistic forms, the Theory of Fuzzy Sets and Fuzzy Logic help.

I. Membership Functions

Membership functions works with linguistic variables. A linguistic variable is the one that express a quantity using linguistic values. For example, the linguistic variable "weight" has the linguistic value "heavy" for object with mass grater than 100 kg. In this way, linguistic values are interpreted establishing *membership functions* ($\mu_A(x)$).

This functions can be seen as a curve with unsharp boundaries defined for all elements x of the fundamental set X. So, the input variables into the Fuzzy System are mapped to a membership value between 0 and 1 (*degree of matching*). These ideas are similar to the theory of fuzzy sets [**zade1968**].

$$A = (x, \mu_A(x))|x\epsilon X \tag{2.8}$$

For example, the next diagram [**DIAGRAMA1**] gives an example for linguistic value "high" of the linguistic variable "salary".

Using memberships functions, it is possible to indicate in which degree a linguistic statement is regarded as a fulfilled for a certain real value of the considered quantity.

II. Fuzzy Operations

Usually, Fuzzy Rules in Fuzzy Inference Systems are composed by two or more linguistic statements using operators like:

· **or**.- Maximum or algebraic sum (eg. temperature = high ∨ pressure = low).

$$\mu_C = max(\mu_A, \mu_B) \tag{2.9}$$

$$\mu_C = \mu_A + \mu_B - \mu_A \bullet \mu_B \tag{2.10}$$

· **and**.- Minimum or algebraic product (eg. temperature = low ∧ humidity = high).

$$\mu_C = min(\mu_A, \mu_B) \tag{2.11}$$

$$\mu_C = min(\mu_A, \mu_B) \tag{2.12}$$

· **not**.- Complement of a fuzzy set (eg. ¬ low = not(low)).

$$\bar{\mu_a} = 1 - \mu_a \tag{2.13}$$

2.0.8. Latent Dirichlet Allocations

Description of the Latent Dirichlet Allocation (LDA) algorithm, that find n topics in a document, and assigns a certain probability to each word to be part of a given topic.

Once this algorithm was "trained", it is possible to make a query asking for "proportions" of probabilities that a query belongs to those n topics.

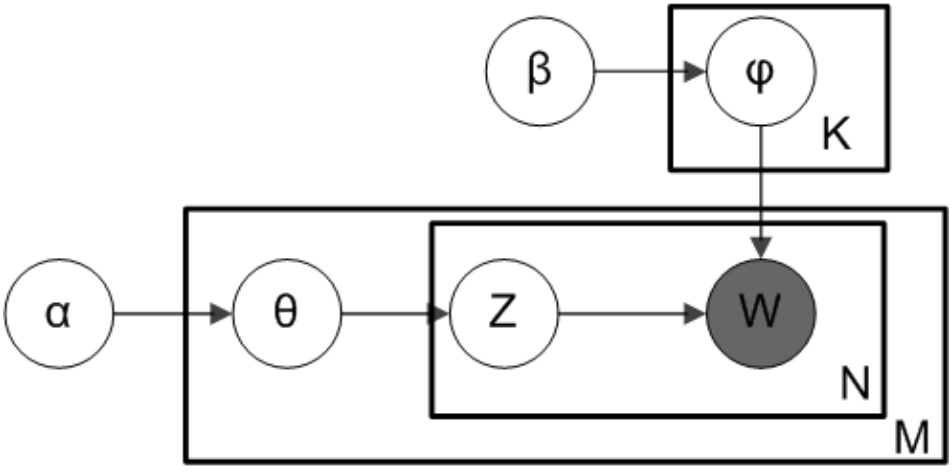

Figure 2.1: LDA diagram from wikipedia.

2.0.9. Term Frequency- Inverse Document Frequency

This form is used to proyect text data into a vectorial space. This algorithm use as input the well-known *BagOfWords* (BOW) text representations, that builds a matrix where each row represent a document, and each column represent a word. In this way, a text corpus is represented as a matrix that captures the frecuency of each words of each document.

Term -frequency establishes that most frequent words (terms t) are the most important in a document d (if a word is repeated many times, is because it should be important and relevant) (Eq. 2.14).

$$tf(t,d) = \frac{f(t,d)}{\max f(w,d) : w \in d} \tag{2.14}$$

Inverse-document-frequency establishes that if a word is repited many times in a corpus (in many documents), so, that word is not that important at all (stop words is an example of this idea). (Eq. 2.15)

$$idf(t, D) = \log \frac{|D|}{|d \in D : t \in d|} \tag{2.15}$$

where **D** the cardinality of the corpus (number of documents), and t denotes the terms (or words) in a document.

Combining equations 2.14 and 2.15, it is possible to compute the value of *Term frequency-Inverse document frequency* (Eq. 2.16) for a given word. In this way, high values are reached by words with high frequencies in some documents *tf*, but words that appear in many documents are affected by the *idf* term.

$$tfidf(t, d, D) = tf(t, d) * idf(t, D) \tag{2.16}$$

2.0.10. Multinomial Naive Bayes Classifier

The difference between *Multinomial Naive Bayes Classifier* (MNBC)and *Naive Bayes Classifier* (NBC) is that NBC assumes conditional independence in each feature of the model, while MNBC uses a multinomial distribution for each feature (is a particular instance of NBC [paginaWeb]).

NBC:

$$p(f_1, ..., f_n|c) = \prod_{i=1}^{n} p(f_i|c) \tag{2.17}$$

so, the posterior prability is given by:

$$p(c|f_i, ..., f_n) \propto p(c)p(f_1|c)...p(f_n|c) \tag{2.18}$$

2.0.11. Random Forest (an ensemble method)

These kind of models (ensembled) are used to improve generalizability and robustness over a single estimator, combining prediction from many estimators.

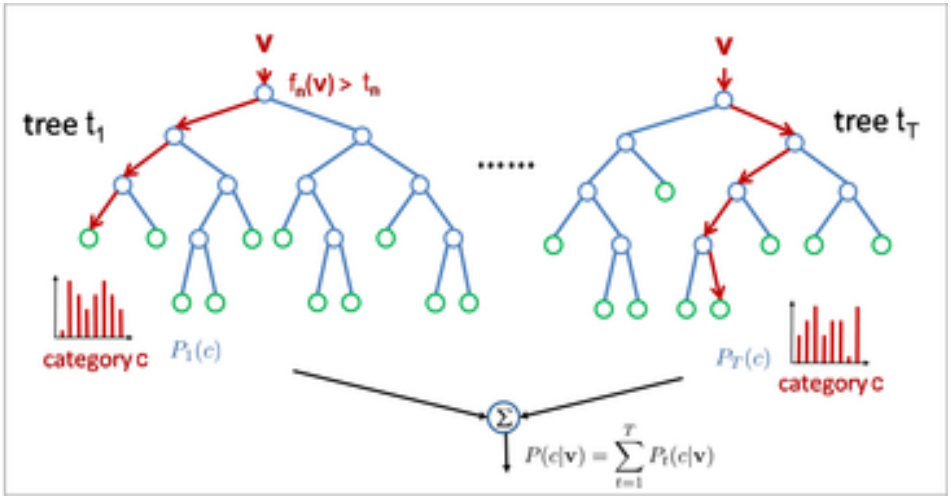

Figure 2.2: Random Forest (ensemble method).

2.0.12. Neural Networks

They are cool...

Chapter 3

Data and methodology

3.1. Proposed Methodology for Ranking Business Models

We propose a pipeline (Fig. 3.1) for ranking business models, which is based on a Fuzzy Rule Inference System whose rules were created using previous data from the company (applications with an assigned score).
The inputs for system created were the text applications itself, those who had to be parsed in order to extract the answers of questions. Those answers were preprocessed, and the words were projected into a vectorial space of 500 dimensions (this vectorial space were created to represent an Spanish language model).
Finally, a compositional operation were applied to applications, in order to measure the "proximity" between the *compositional text* and some *key concepts* regarding an *Design innovation* approach. After that, the "degree" of similarity between the *compositional text* and *key concepts* were the inputs to the Fuzz Rule Inference System, whose output is the score used to rank business applications.

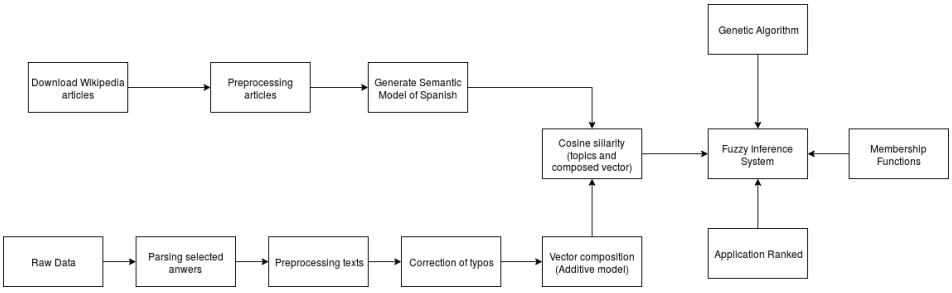

Figure 3.1: Proposed pipeline for ranking business models.

3.1.1. Text applications

Text data were *scraped* from the company website. For this task, the **GNU Wget** package and **Beautifulsoup** python library were used in order to get data from HTML files.
At the end of this part, and PSV file (*"pipe-separated values"*) was generated, where the answers for each question, the ID and the final score (given by judges) of applications were stored in different columns.

On the other hand, our "secondary" source of text data were Wikipedia. So, we download a database of articles in Spanish [**wiki:Download**] in order to create a model of the Spanish language.

For both sources of text data, a list of common stop-words [**ranks:stopwords**], symbols, accents, numbers and links were removed using the **GNU sed** and **GNU tr** command-line text editors. At the end of this step, most of text data consisted in words with relevant semantic meaning.

A second pre-processing step consisted in implement a spelling-corrector algorithm, in order to avoid "words without meaning" due to typos.

Some statistics about length aplications are show below:

```
.
count 8815.000000
mean 2152.153488
```

```
std 921.940859
min 213.000000
25% 1491.000000
50% 1970.000000
75% 2601.000000
max 7340.000000
Name:  length, dtype:  float64
```

In the next histogram (Fig. 3.1.1), it is showed the histogram of length of applications, counting the number of words in each one. All answers from data applications were merged into one field, and the number of word of that field was assigned as a new "length" feature in the dataset.

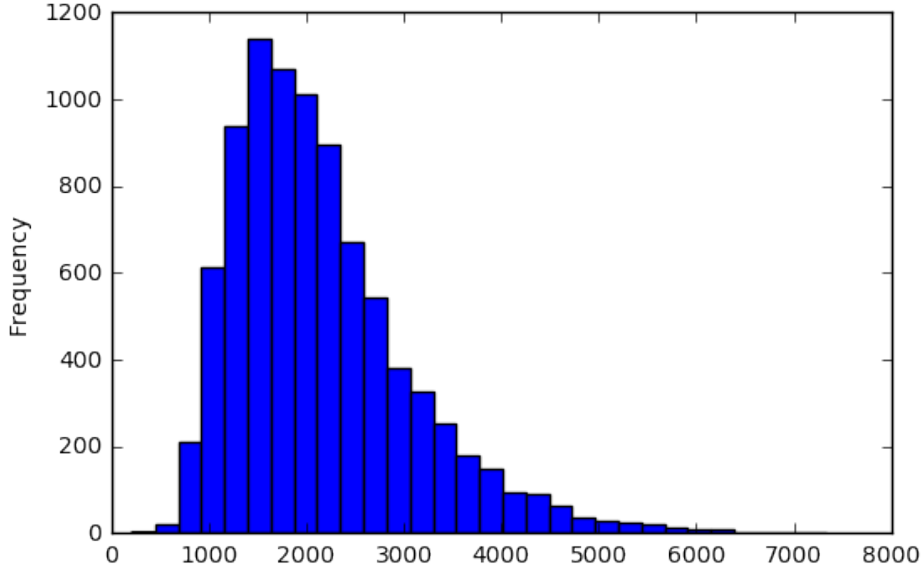

Figure 3.2: Histogra of length of all applications.

Two more histograms (Fig. 3.1.1) were created, in order to visualiza the lenght of applications according to the category they were assigned. The label "-1" was for

those applications with overall scores less than 0 (assigned by *expert judges*), and label "+1" with overall scores more than 0.

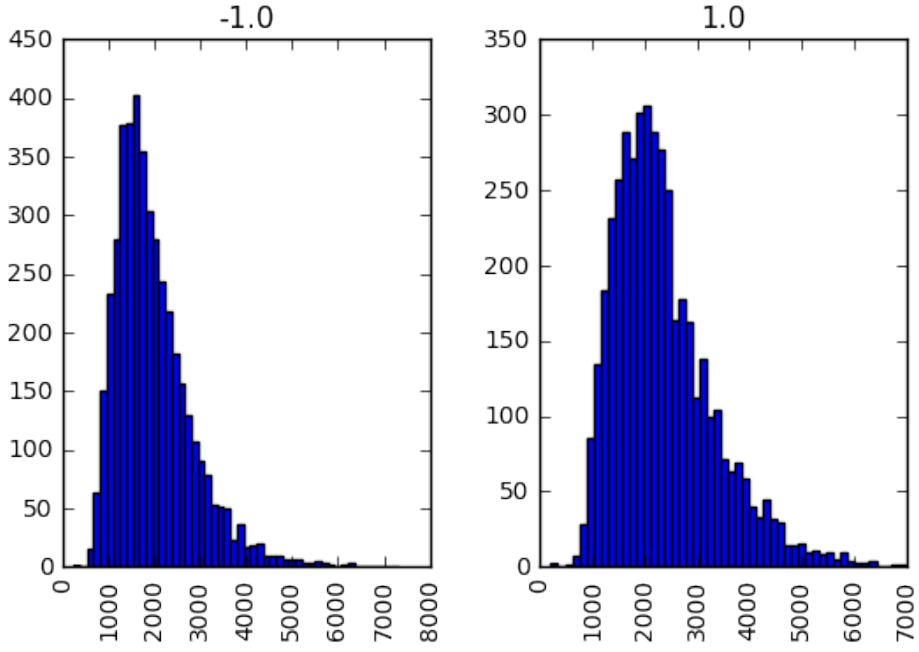

Figure 3.3: Histogram of length of applications, where Clase=-1 if they have a bad score, 1 otherwise.

I. Spelling correction algorithm

The spelling correction algorithm implemented in this work was developed by Peter Norving [**norvig2007write**].
How many words were corrected? spellCorrectiongNorving

II. Correlations between words and scores

Bimodal distribution across scores, length of aplications is skewed to right.

	score	noWords	uniqWords	nonStopWords	noPunct
score	1.0000000	0.2214465	0.2270032	0.2227057	0.1128915
noWords	0.2214465	1.0000000	0.9747430	0.9996054	0.4693714
uniqWords	0.2270032	0.9747430	1.0000000	0.9747562	0.4084167
nonStopWords	0.2227057	0.9996054	0.9747562	1.0000000	0.4770340
noPunct	0.1128915	0.4693714	0.4084167	0.4770340	1.0000000

Figure 3.4: There is not correlation between number of words, number of words without stop words or punctuations and score, in applications.

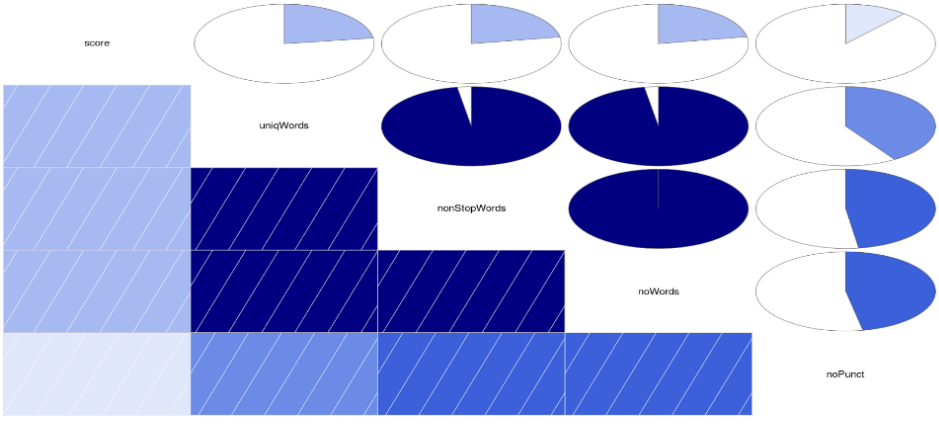

Figure 3.5: There is not correlation between number of words, number of words without stop words or punctuations and score, in applications.

3.1.2. Semantic Representation of Spanish Language

We generated a kind of Spanish "dictionary", preserving semantic properties of language, were each word is mapped into a vectorial space of 500 dimensions, using the Word Embedding algorithm implemented in *Gensim* [**rehurek'lrec**] python package. This algorithm took as input the articles downloaded from Wikipedia, and the output was a file were each word was represented with a vector of 500 features.

Singular Value Decomposition (SVD) was applied to a set of "key words", those with semantic meaning were related to *Design Innovation* concepts, in order to

visualize a proper relationship between our Spanish model.**insertar figura de SVD**.

3.1.3. Accumulative Semantic Value by Topic

Cosine distance

3.1.4. Fuzzy Inference System

3.1.5. Discovering Rules for Ranking Application

An Genetic approach

3.1.6. Ranking Applications

3.1.7. Dataset Description

Chapter 4

Results

4.0.1. Models Predicting Categories

I. Multinomial Naive Bayes Classifier

Some scores obtained: .

```
precision recall f1-score support

-1.0 0.69 0.49 0.57 859
1.0 0.62 0.79 0.69 904

avg / total 0.65 0.64 0.63 1763
```

Next, we plot the precision vs training samples:

II. Support Vector Machine

Some scores obtained: .

```
precision recall f1-score support
```

```
-1.0 0.62 0.62 0.62 859
1.0 0.64 0.64 0.64 904

avg / total 0.63 0.63 0.63 1763
```

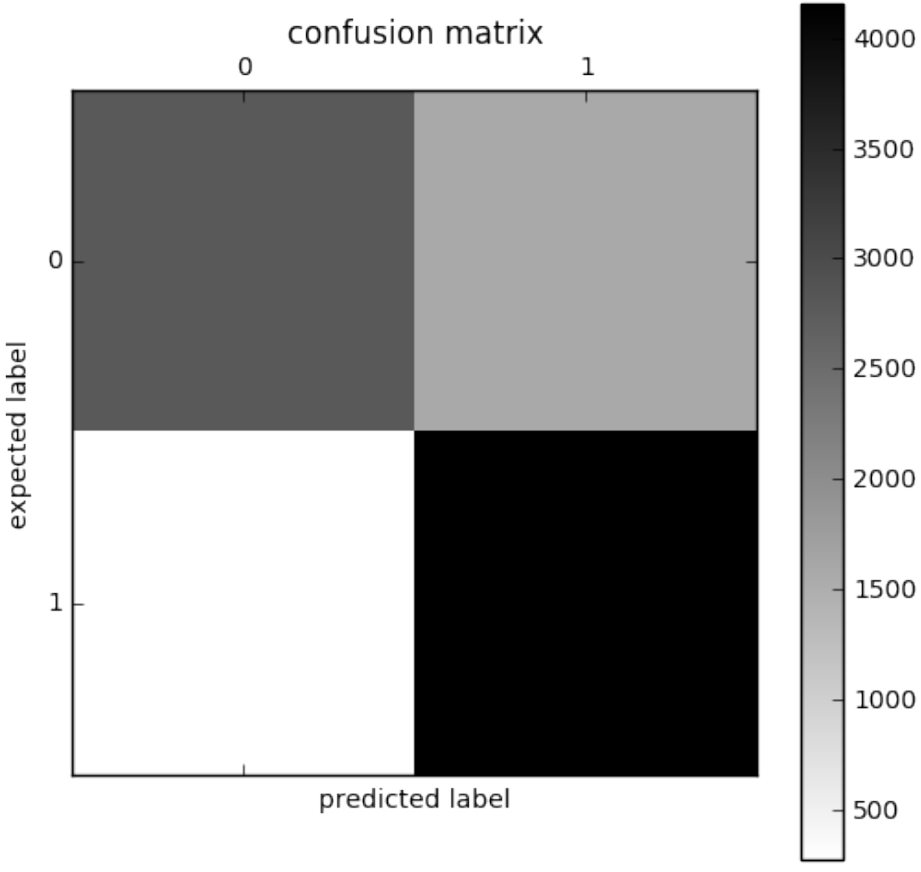

Figure 4.1: Confusion Matrix for Multinomial Naive Bayes Classifier.

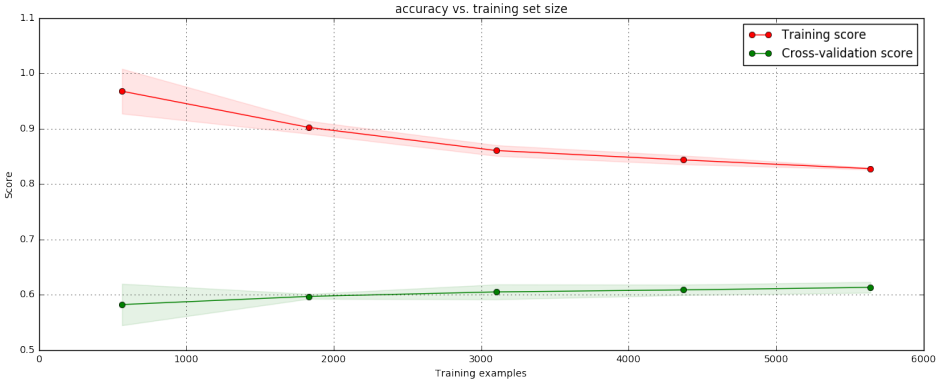

Figure 4.2: Accuracy Vs Training set size.

Chapter 5

Conclusions and future work

lalala **??**

Bibliography

[1] Joan Magretta. «Why business models matter». In: (2002).