Universidad del País Vasco Euskal Herriko Unibertsitatea

Máster Universitario en
Ingeniería Computacional y Sistemas Inteligentes

K I S A I C S I

Konputazio Zientziak eta Adimen Artifiziala Saila –
Departamento de Ciencias de la Computación e Inteligencia Artificial

Tesis de Máster

Título: Building a predictive model for Kaggle's "Home Depot Product Search Relevance" Competition

Luis Roberto Jácome Galarza

Tutor(a/es)

Aitor Soroa
Departamento de Ciencias de la Computación e Inteligencia Artificial
Facultad de Informática

Arantxa Otegi
Departamento de Lenguajes y Sistemas Informáticos
Facultad de Informática

informatika fakultatea facultad de informática

KZAA /CCIA

Septiembre 2016

Abstract

The present work analyses different techniques in order to build a predictive model that could be able to solve the Kaggle’s competition called “Home Depot Product Search Relevance”. Several NLP methods were used for data pre-processing like tokenization, lemmatization, extracting stop words, etc. Word overlap and Mikolov word embeddings were used for feature extraction, Random Forest algorithm was used for applying regression. Finally the statistical open-source R language was used for building the scripts.

The results indicate that distributed word representations are a very useful technique for many NLP applications. Word embeddings helped to improve the accuracy of the predictive model; having this experience it can be realized the power of this technique and its ease of use.

A big concern of the project was the long processing time of processing the word embeddings in regular desktop/laptop computers. In order to reduce the processing time, it was necessary to extract the words embeddings only of the words found in the datasets. Moreover, some of the datasets were split and processed in different machines. Other possible solutions to this problem are renting cloud computing, grid computing, parallel computing, servers, etc.

Index

1. Introduction

1.1 Objectives of the thesis

The main objective of the thesis is to build a predictive model that can solve the problem found in the Kaggle's competition called "Home Depot Product Search Relevance".

For the accomplishment of the main objective, it is necessary to fulfil some steps like data pre-processing, which is a very important step for cleaning the data. The feature generation step is necessary for transforming the text data into numerical data that will be used in the next step which is the model building with a regression algorithm.

Finally, it is also important for this project to measure the accuracy of the model; the same metrics used to evaluate the submissions for the competition must been applied to the project's model.

1.2 Field of study

Data Science is a powerful yet new field of Artificial Intelligence, in which large amounts of data are transformed into valuable information. This is possible because of the explosion of Internet sites that gather information from millions of users all over the world, companies' databases and the increase of data storage capacity and the improvement of computing processing.

Nowadays, data scientists are the most required IT professionals due to the necessity of companies to extract knowledge that would bring competitive advantages and solve problems. Growing constantly, the demand of data scientists surpasses the number of qualified available professionals; some of their most desired skills and knowledge are Computer Programming, Statistics Analysis, Database Management, Machine Learning, Data Mining, etc.

Kaggle **[1]** is an online platform where enterprises submit their data analysis problems and data scientists compete for finding the model that bring the best solution. Most of the competitions' results show improvement of the existing solutions which points out that, companies will get the desired results. Kaggle is also a way data scientists have to interact with other people with similar skills and profiles, team up to participate in the competitions and win rewards for their data analysis works.

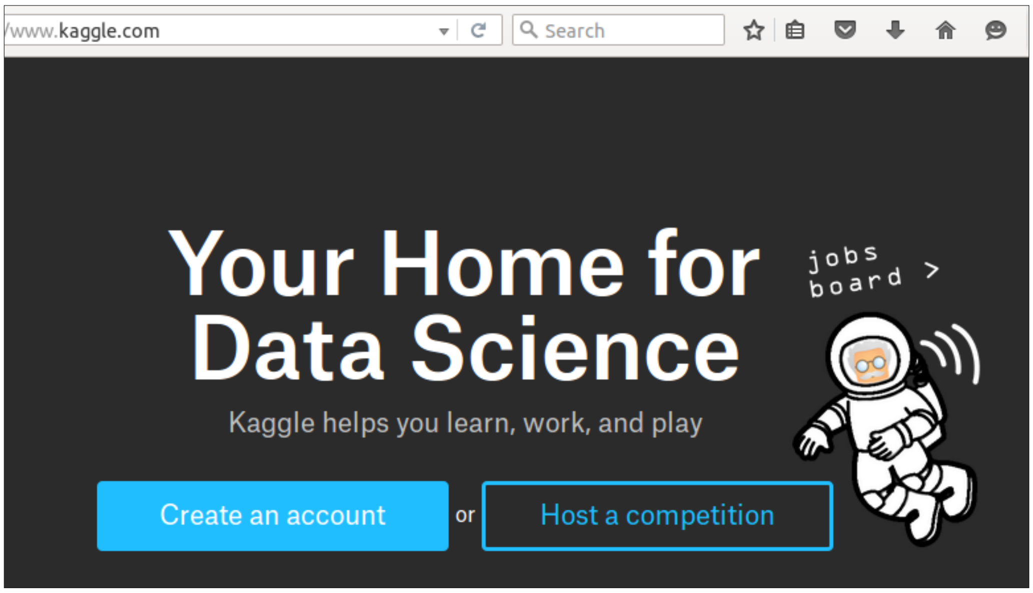


Figure 1. Kaggle **[1]**

The Kaggle's "Home Depot Product Search Relevance" competition is a contest in which data scientists are asked to create a predictive model that could give the relevance of query searches in Home Depot's web site **[2]** and the results provided by the algorithms, this information will give the programmers information of the best settings for the search algorithms to improve the user's experience.

This work presents a solution for the Kaggle's "Home Depot Product Search Relevance" Competition in which the developed model predicts the relevance of the query searches given products names, descriptions and attributes. The Random Forest algorithm is used for data regression.

A data pre-processing work was needed in which Natural Language Processing techniques like tokenization, lemmatizing, stemming, embedding were used. For data regression, the Random Forest algorithm was used, R language (open-source Statistical language) was also used.

2. Motivation

The Master in Computational Engineering and Intelligent Systems of the Basque Country gives its students the necessary skills to perform real-world data analysis works. Being Data Science a trendy discipline, proposing a predicting model is a challenging but gratifying task, in which all the knowledge learned will be used in a practical project.

Improving data science skills by gaining experience from a Kaggle competition is also rewarding, it is estimated that there is in the present and there will be in the next years a shortage of IT professionals, especially in the data science field, so building a data science career is a promising endeavour. In the other hand, each Kaggle's competition is very difficult and challenging because data scientists with very high education background and experience propose their predictive models and only the model which gives the best results wins. This fact is beneficial as well, because each competitor will improve his model over and over again to beat other's models.

Besides competitions, Kaggle platform allows data scientists to interact with more experienced colleagues, share code, learn from tutorials and data sets. It is paramount important for a data scientist to participate constantly in Kaggle for training, getting new skills, trying the latest techniques and tools, and winning monetary prizes. Companies that host competitions also have benefits because they get the best solution (made by high qualified data scientist from all over the world) for their data analysis problems which mean a huge increase of productivity.

3. Theoretical Framework

The exponential growth of technology allows us to improve the quality of people's life. More often, machines are taking over control and assisting human daily activities. The explosion of data in databases and the Internet had led to the necessity of new approaches to process and use those data. Many fields of knowledge have contributed to this continuous and remarkable progress. The next section describes some of the areas from which it has been extracted valuable knowledge for the present project.

3.1 Artificial Intelligence

It's the name given to the intelligence of the machines and the software. It's a scientific discipline that research the methods of creation or simulation of intelligence.

The term "Artificial Intelligence" was created by John McCarthy but Alan Turing first referred with his article "Computing Machinery and Intelligence" in 1950. Turing also proposed the Turing Test in which it could be established if a machine could be defined as conscious.

The application domains of the AI are:

- ✓ Banks: with Expert Systems that evaluate the risk of give a credit to a customer (credit-scoring)
- ✓ Military: with autonomous systems like drones, systems for decision support.
- ✓ Games
- ✓ Medicine: with Expert Systems for diagnosis.
- ✓ Logistics: with heuristics approaches for solving hard mathematical problems.
- ✓ Robotics

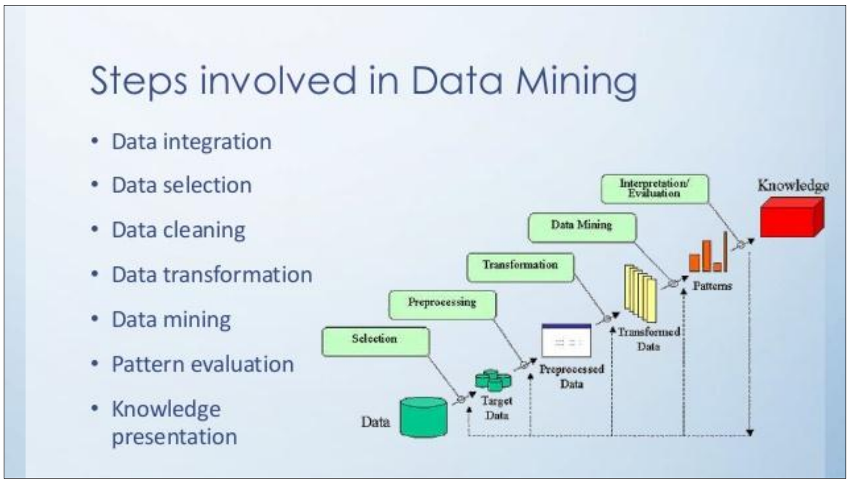
3.2 Data Science

Data Science is the extraction of knowledge from data. It applies the techniques of many fields like Math, Statistics, Information Theory and Information Technology, Probabilistic Models, Machine Learning, Computer Programming, Data Engineering, Forms Recognition, etc. The main goal of Data Science is to build methods for analysis of massive data with the objective to extract useful or potentially useful information.

The term "Data Science" was first used in 2001 by William Cleveland in his article "Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics". This discipline appears in response of the growing complexity found in databases and the Internet with huge amounts of data available.

3.3 Data Mining

Data Mining consists of extracting knowledge from large amounts of data using automatic or semi-automatic methods. Its goal is to use techniques of different disciplines like Statistics, Artificial Intelligence, and Computer Science, to build models from data that find structures of different criteria that bring valuable knowledge. Figure 2 describes the steps involved in Data Mining in which is remarkable to realize that the previous treatment of the data is very important for the success of the Data Mining project; the data pre-processing also takes an important amount of time an effort.



Steps involved in Data Mining

- Data integration
- Data selection
- Data cleaning
- Data transformation
- Data mining
- Pattern evaluation
- Knowledge presentation

Figure 2. Steps in Data Mining **[3]**

3.4 Machine Learning

Machine Learning is a field of Artificial Intelligence which Arthur Samuel defined in 1959 as “Field of study that gives computers the ability to learn without being explicitly programmed”; it also has been described as algorithms that learn from data.

Applications of Machine Learning include perception of their environment (vision, object recognition, natural languages), search engines, medical diagnosis, brain-machine interfaces, fraud detection, financial analysis, games, robot motion, predictive analysis, etc.

Machine Learning is divided into Supervised Learning, Unsupervised Learning and Semi-supervised Learning.

3.5 Supervised Learning

If classes are predetermined and the examples are known, the system learns to classify according to a classification model. An expert has to label the examples and the process is done in two phases. The first phase consists of creating a model of labelled examples (train the model), the second phase (test the model) consists in predicting the class of a new data samples using the model. This prediction can be used as feedback to refine the model. After the model has been trained and tested it will be able to predict the values for new instances.

3.6 Regression

Regression is a statistical process for estimating the relationships among variables. In regression, most of the cases, the target variable is continuous. To solve regression supervised learning algorithms are used. Regression is used to make predictions, for example, regression can be used to predict the sales of a store, given the sales of the past months. Figure 3 shows an example of regression.

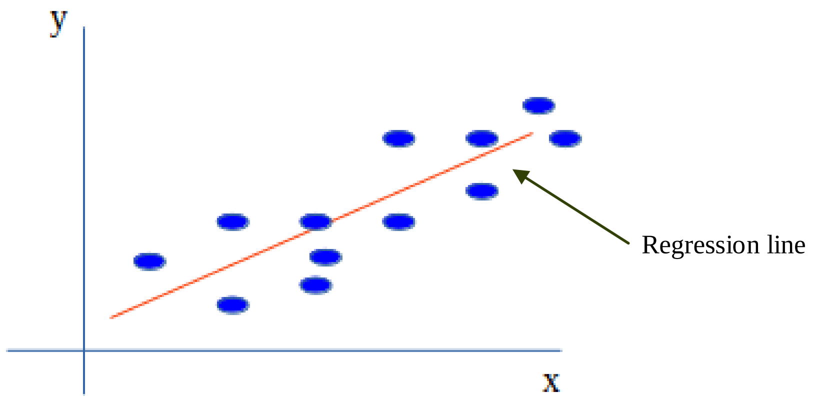


Figure 3. Regression line **[4]**

As mentioned before, regression treats continuous variables, while classification treats discrete values, which means it tries to predict what class an instance is part of. In the other hand, clustering has no predefined classes and tries to group together elements with the more similar features.

3.7 Feature Engineering

A feature is a piece of information which is potentially useful for prediction. Feature engineering is the process of using domain knowledge of the data to create features that make Machine Learning algorithms work.

Features are very important in predicting models, the quantity and quality of features will determine if the model is good or not. When it comes to relevance, features could be strongly relevant, relevant, weakly relevant or irrelevant.

3.8 Natural Language Processing

Natural Language Processing (NLP) is part of the human-computer interactions, it consists the study of interactions between computers and human natural languages.

NLP has many research areas like:

- ✓ Automatic summarization
- ✓ Discourse analysis
- ✓ Machine translation
- ✓ Morphological segmentation
- ✓ Named entity recognition

- ✓ Natural language generation
- ✓ Natural language understanding
- ✓ Optical character recognition
- ✓ Part-Of-Speech Tagging
- ✓ Parsing
- ✓ Sentiment analysis
- ✓ Speech recognition
- ✓ Speech segmentation
- ✓ Word segmentation
- ✓ Information retrieval
- ✓ Information extraction
- ✓ Speech processing

3.8.1 Tokenization

Tokenization is part of the lexical analysis which consists of the process of breaking a stream of text up into words, phrases, symbols or other meaningful elements called tokens.

3.8.2 Lemmatization

Lemmatization in linguistics consists of grouping together the different inflected forms of a word so they can be analysed as a single item.

Examples:

The word “worse” has “bad” as its lemma
“Play” is the base form word of “playing”

3.9 Machine Learning Algorithms

3.9.1 Decision Trees

Learning by decision trees is a method that uses a decision tree as permanent predictive model that tests the value of a feature of a system after the observation of other features of the same system.

Decision Trees have advantages like simplicity of understanding and interpretation, a few preparations of data, they can process a large amount of data and the model could be tested by statistical functions.

3.9.2 Random Forests

The Random Forests classifier is a technique of Machine Learning that was proposed by Leo Braiman and Adele Cutler, the algorithm combines random subspaces and bagging. It also does the learning using multiple decision trees trained by slightly different subsets of data as shown in the figure 4.

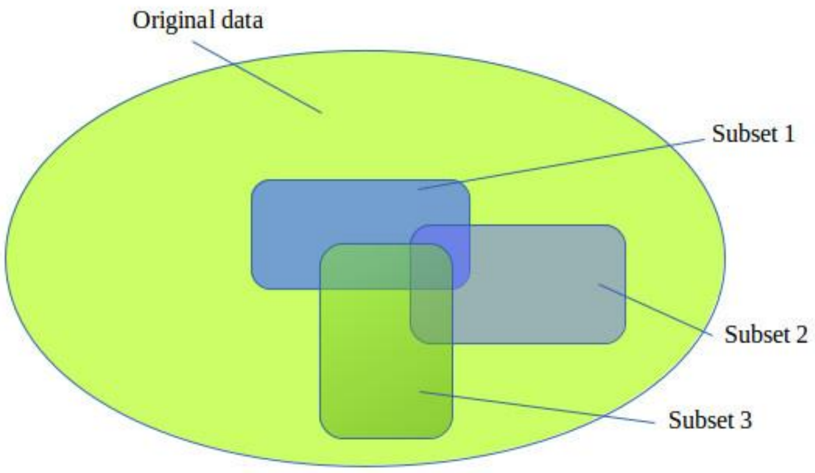


Figure 4. Random selection of data in Random Forest

The decision trees are also built by a subset of the original variables, it is shown in the figure 5.

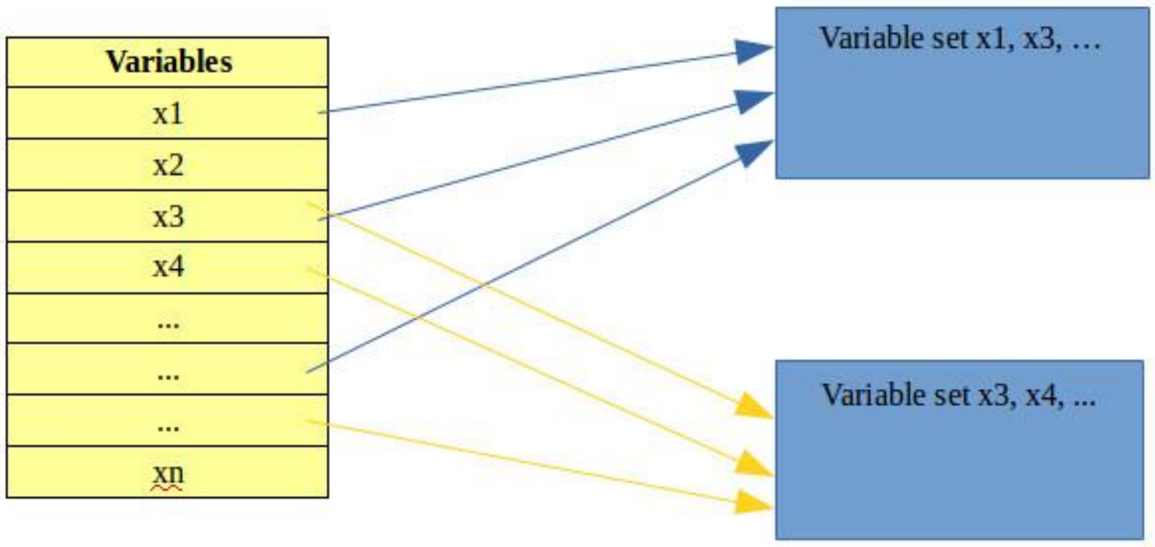


Figure 5- Random selection of variables in Random Forest

Having multiple decision trees with a subset of data and variables, the algorithm provides accurate prediction for most parts of the data, making mistakes at different places. Finally, the prediction is made by voting for each of the observations; this is expected to be closer to the right classification. The figure 6 describes the implementation of the Random Forest algorithm for the train and test process.

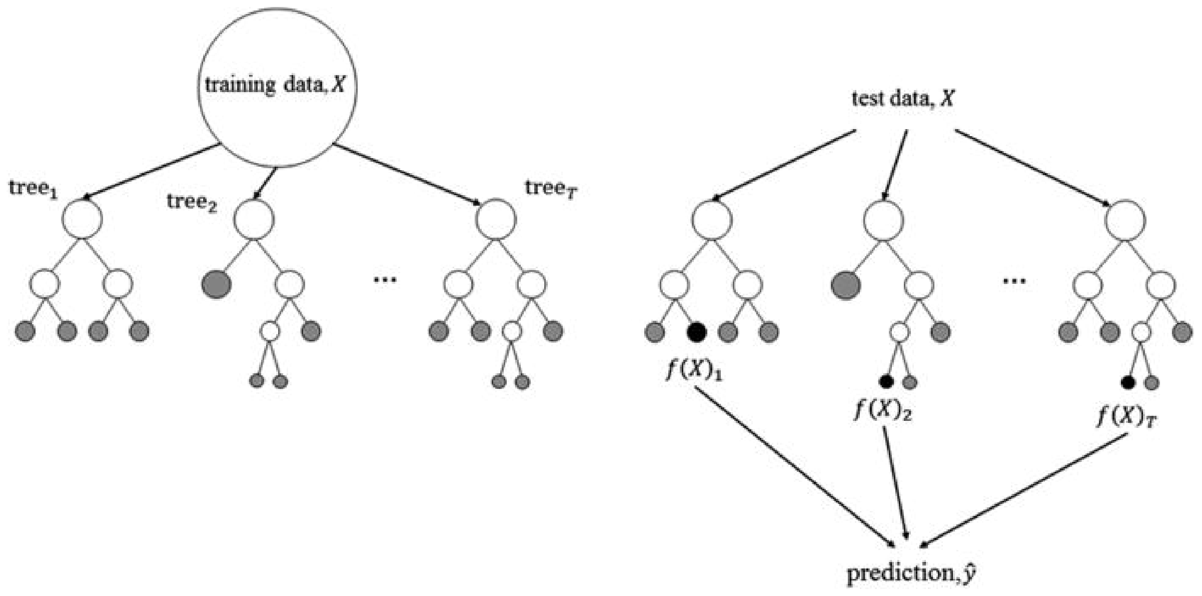


Figure 6. Random Forest: multiple decision trees

3.10 Distributed word representations

Many techniques have been developed for representing the human natural language in a way in which computers can work with. The next section describes some of them.

The **Bag of Words** model learns a vocabulary from all documents, after that it models each document by counting the number of times each word appears. For example, having these 2 sentences:

s1: "The cat sat on the hat"
s2: "The dog ate the cat and the hat"

With the 2 sentences, the following vocabulary is built {the, cat, sat, on, hat, dog, ate, and}

To obtain the Bag of Words, it is necessary to count each word in the sentences and register its occurrence.

S1: {2, 1, 1, 1, 1, 0, 0, 0}
s1: {3, 1, 0, 0, 1, 1, 1, 1}

An **N-gram** is a sequence of N items that could be letters, words, and phonemes. It is known that certain word or item pairs (triplets, quadruplets, etc.) are likely to occur more often than other pairs, for example the letter Q most of the times is found in words with an U as in "queen", "question", "frequent"

Having enough data, it can be computed the frequency distribution of all N-grams found in the data. In the other hand, the permutations increase greatly with N, for example, in English language, having 26 letters, it is possible to form 26^2 letter pairs, 26^3 letter triplets and so on, that's why N is often taken as small number. Google has calculated the word N-gram from its data **[5]**.

N-grams are very similar to the multi-order Markov model in which the Nth. Position depends on the previous N-1 items, and can be calculated from a data corpus. Having the N-gram data information it could be used for suggested auto-completion of words and phrases, spelling correction, speech recognition, word disambiguation.

A **word vector** is a vector of weights. In a simple 1 of N schema, every element in a vector is associated to a word in the vocabulary. Encoding a word is just filling the numbers of the vector, having a number 1 to the place of a word and 0 to all other places.

If having only 5 words in the vocabulary like: king, queen, man, woman and child, encoding the word 'queen' would be like:

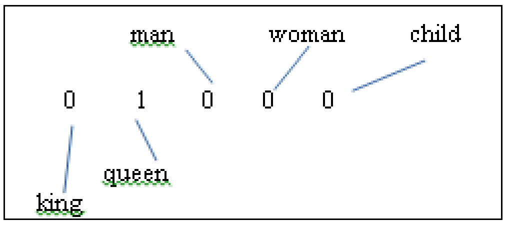
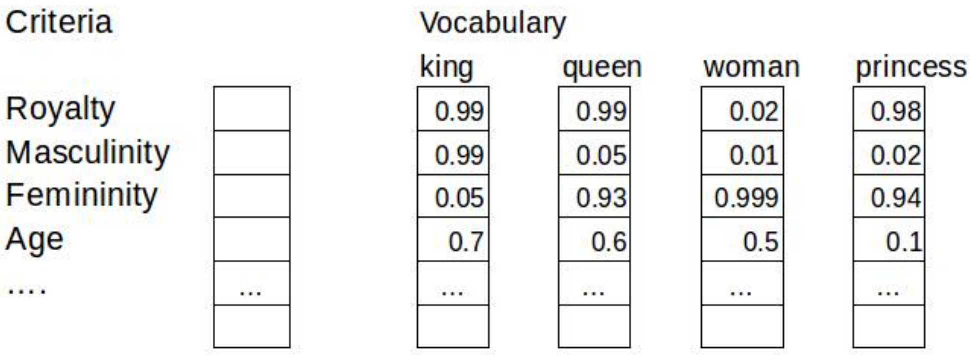


Figure 7. 1-of-N encoding **[6]**

Word embeddings is the representation of a word using numbers. Every word or phrase from a vocabulary is mapped in a vector of real numbers in a low-dimensional space.

In **word2vec** words are represented in a vector that has N dimensions. Each word is represented by weights in the vector, so instead of a 1 to 1 mapping between a value in the vector and a word, the word is represented by all the values in the vector, and each element of the vector contributes to the definition of the word **[6].** The following example describes how word vectors are created. In this case, a small domain is taken, columns represent the words of the vocabulary and rows are the different criteria used to describe those words. Each word has a numerical vector that corresponds to the weights of that word in specific criteria. The figure 8 describes this example.



Criteria		Vocabulary			
		king	queen	woman	princess
Royalty		0.99	0.99	0.02	0.98
Masculinity		0.99	0.05	0.01	0.02
Femininity		0.05	0.93	0.999	0.94
Age		0.7	0.6	0.5	0.1
....	...	...	...	...	...

Figure 8. Word2vec example **[6]**

By examining a large corpus, it's possible to build word vectors that are able to capture the relationships of the words in a very expressive way. These vectors can also be used as input of neural networks.

The figure 9 represents a 2-dimentinal word vectors.

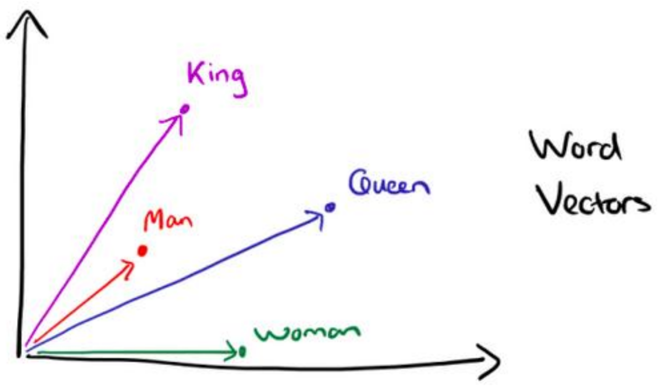


Figure 9. Word Vectors **[6]**

Vectors are also a very useful tool for answering analogy questions, using a simple vector offset method based on cosine distance. Some examples are:

man is to a *woman* as *uncle* is to? ….. *aunt*

Vector composition also finds relations like:

king – man + woman = ? .... *queen*

Figure 10 describes an example of word vector Composition.

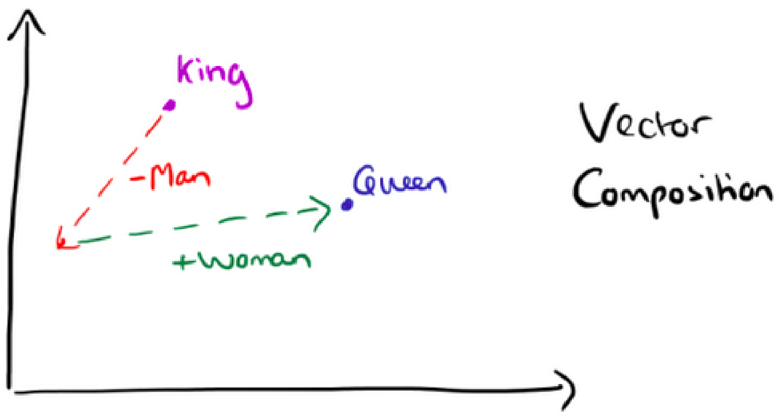


Figure 10. Word Vector Composition **[6]**

Word representations are very powerful because they can describe meaningful syntactic and semantic patterns in a very simple way. Regularities are found as constant vector offsets between pairs of words sharing a particular relationship. If a word vector is denoted by $\mathbf{x_i}$ for the i-word it could be obtained the following relationships:

$x_{apple} - x_{apples} \approx x_{car} - x_{cars}$

$x_{apple} - x_{apples} \approx x_{familiy} - x_{families}$

As it can be seen, the vectors can find the relationship between singular/plural words.

These examples show that word vectors with semantic relationships could improve many Natural Language applications like Machine Translation, Information Retrieval, Question Answering Systems, etc.

More examples of semantic and syntactic word relations are shown in table 1.

Type of relationship	**Word Pair 1**		**Word Pair 2**	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	Kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	Brother	Sister	Grandson	Granddaughter
Adjective to adverb	Apparent	Apparently	Rapid	Rapidly
Opposite	Possibly	Impossibly	Ethical	Unethical
Comparative	Great	Greater	Tough	Tougher
Superlative	Easy	Easiest	Lucky	Luckiest
Present Participle	Think	Thinking	Read	Reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	Walking	Walked	Swimming	Swam
Plural nouns	Mouse	Mice	Dollar	Dollars
Plural verbs	Work	Works	Speak	Speaks

Table 1..Examples of types of semantic and syntactic word relations **[7]**

Accuracy is quite good when using word vectors, more interesting outcomes can be found playing with the vectors like: Paris – France + Italy = Rome.

The table 2 shows more outstanding examples:

Relationship	**Example 1**	**Example 2**	**Example 3**
France – Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big – bigger	small: larger	cold: colder	quick: quicker
Miami – Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein – scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy – France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper – Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi – Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft – Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft – Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan – sushi	Germany: bratwurst	France: tapas	USA: pizza

Table 2. Examples of the word pair relationships **[7]**

Cosine similarity

In order to measure the properties between word embeddings, the cosine similarity metric is used.

The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them **[8]**. This metric is a measurement of orientation and not magnitude.

The formula 1 represents the dot product operation of vectors.

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

Formula 1. Dot product between 2 vectors **[8].**

Considering formula 2, the formula of the Cosine similarity can be obtained (formula 2).

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Formula 2. Cosine similarity between 2 vectors **[8].**

It can be seen in figure 11 that 2 vectors with a small angle of deviation are very similar between them; the cosine similarity score is near 1. Two vectors with an angle of 90 degrees are unrelated; the cosine similarity is 0. Two vectors with opposite directions (near 180 degrees) are opposite; the cosine similarity score is near -1.

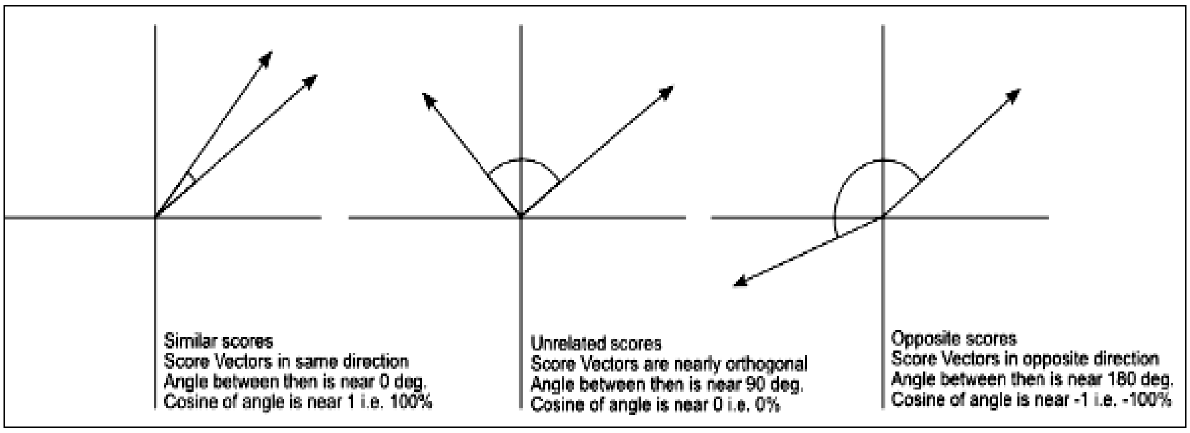

Similar scores
Score Vectors in same direction
Angle between then is near 0 deg.
Cosine of angle is near 1 i.e. 100%
Unrelated scores
Score Vectors are nearly orthogonal
Angle between then is near 90 deg.
Cosine of angle is near 0 i.e. 0%
Opposite scores
Score Vectors in opposite direction
Angle between then is near 180 deg.
Cosine of angle is near -1 i.e. -100%

Figure 11. Pairs of word vectors with different cosine similarity **[8].**

It can be demonstrated the properties of word embeddings with practical cases. The following example shows the implementation of word vectors in R language

In order to use the Cosine similarity operation, the **lsa** package was needed. The **cosine** function is used to calculate the cosine similarity. First, a basic example was executed. The cosine similarity was calculated between 2 numeric vectors.

```
> vec1 = c( 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 )
> vec2 = c( 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0 )
> cosine(vec1,vec2)
          [,1]
[1,] 0.2357023
```

For further experimentation, the vectors and values of the word2vec example of the figure 8 were taken into consideration. Five words were considered; their values were filled in the criteria “royalty”, “masculinity”, “femininity” and “age”.

```
> king = c(0.99, 0.99, 0.05, 0.7)
> queen = c(0.99, 0.05, 0.93, 0.6)
> woman = c(0.02, 0.01, 0.999, 0.5)
> princess = c(0.98, 0.02, 0.94, 0.1)
> prince = c(0.99, 0.99, 0.05, 0.1)
```

The following results were obtained for the cosine similarity between the words, were a value of zero means no similarity and a value of one means completely similar.

```
> cosine(king, queen)
          [,1]
[1,] 0.6429661
> cosine(king, woman)
          [,1]
[1,] 0.2455264
> cosine(queen, woman)
          [,1]
[1,] 0.7525689
```

Queen and princess have a lot of similarity:

```
> cosine(queen, princess)
          [,1]
[1,] 0.9417467
> cosine(king, princess)
          [,1]
[1,] 0.5190673
> cosine(princess, woman)
          [,1]
[1,] 0.6630324
```

As expected, the result of the following operation of vectors was true:

King – queen + princess = prince

```
> cosine(prince, (king-queen+princess))
          [,1]
[1,] 0.9972254
```

The cosine function gave a score of 0.9972254 which means that there is a very high similarity between the 2 vectors. More similar examples explained in the theory can be successfully obtained.

Deep Learning and word2vec

Deep Learning is a field of Machine Learning that attempts to model high-level abstractions in data by using a deep graph with multiple processing layers. Deep learning is considered a development of Neural Networks; they have more layers than traditional models and are trained in higher orders of magnitude than was available before **[9]**.

In Deep Learning systems, algorithms can automatically learn feature hierarchies. Some researchers consider word2vec as Deep Learning because it is a two layer neural network that processes text; while word2vec is not a deep neural network, it turns text into numerical form that deep nets can understand.

4. State of the art

Distributed word representations are a valuable resource for many Natural Language Processing applications. This section describes state to the art applications for word representations.

In **[7]**, Mikolov et al. studied the quality of vector representations of words using various models and applied to different syntactic and semantic tasks. It is explained that it is possible to train high quality word vectors using very simple model architectures having much lower computational complexity.

In **[10]**, bilingual word embeddings are introduced to improve Machine Translation; they help to match the semantic similarities between languages. Experiments are done with English and Chinese languages.

Semantic hierarchy constructions uses hypernym-hyponym relations (“is a”) to build structures of concepts. For example, the words “dog” and “canine” are connected directly, “canine” is a hypernym of “dog” and “dog” is a hyponym of “canine”. In **[11]**, a new method of hierarchy constructions using word embeddings is presented, which is used to measure the semantic relationships between words. The model identifies if a word pair has a hypernym-hyponym using semantic projection based on word embeddings.

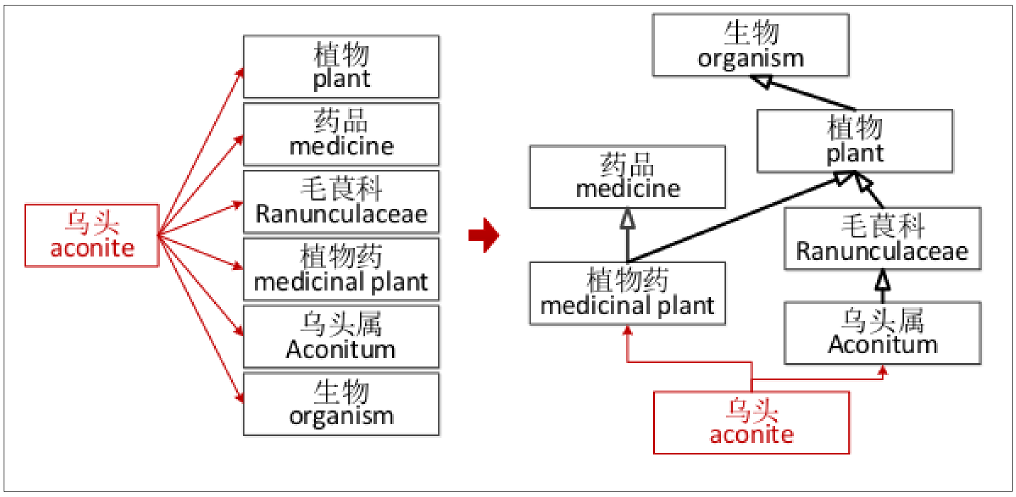


Figure 12. An example of semantic hierarchy construction **[11]**

In **[12]**, they show an analysis of the contribution of word embeddings to Natural Language Processing applications. It remarks its importance for semantic analysis, and studies the variation of the number of dimensions. The following tasks have been performed to evaluate different sets of embeddings:

Sentiment polarity: create sets of words that have positive or negative connotations.

Noun gender: To compile a list of masculine and negative words. Names that frequently appear with he/she words respectively are categorized as masculine/feminine.

Plurality: to extract nouns in the singular and plural forms.

Synonyms and antonyms: extract synonym and antonym pairs.

Regional spelling: collecting words that differ in spelling between UK English and American English.

In **[13]**, an analysis is presented in which supervised NLP systems' accuracy is improved by using unsupervised word representations. Moreover, combining different word representations can improve accuracy further. Word features can be learned in advance in an unsupervised task. Once learned, word features can be easily shared with other researchers and integrated into supervised NLP applications.

In **[14]**, a paper is presented in which word embeddings are trained for more than 100 languages using their corresponding Wikipedia; Moreover, it is explained that using word embeddings in conjunction with traditional NLP methods can improve NLP tasks; for this reason and many other applications, distributed word representations are a valuable resource for any language. It is claimed that the embeddings will be released for public use at www.cs.stonybrook.edu.

In **[15]**, it is highlighted that words with different meaning (polysemous), can be represented by different embedding vectors. However, due to the large amount of parameters to train, this method is not practical. Considering that fact, a probabilistic model is proposed in which the continuous Skip-gram model is integrated with the embedding vectors, as a result, there are less parameters to train and it leads to better results.

In **[16]**, a new model for Speech Recognition is proposed in which words are projected into continuous embedding space where words that sound alike are scored nearby. It is shown that embeddings allow words that are not in the training dictionary have also a score. The table 3 shows some examples of words and their nearest neighbour in the embedding space:

Words	**Neighbours**
Heart	hart, heart's, iheart, hearth, hearted, art
Please	Pleased, pleas, pleases, pleaser, plea
Plug	plugs, plugged, slug, pug, pluck
Chareety	sharity, hare, cheri, tyree, charice, charities

Table 3. Nearest neighbour examples in the acoustically similar embedding space

In **[17]**, the Skip-gram model of Mikolov et al. **[7]** is extended by taking visual information into consideration. The model builds word representation using vectors and also is exposed to visual representations of the objects it denotes, and must predict linguistic and visual features jointly. Having propagating visual information to words, it has been used to improve image labelling and retrieval.

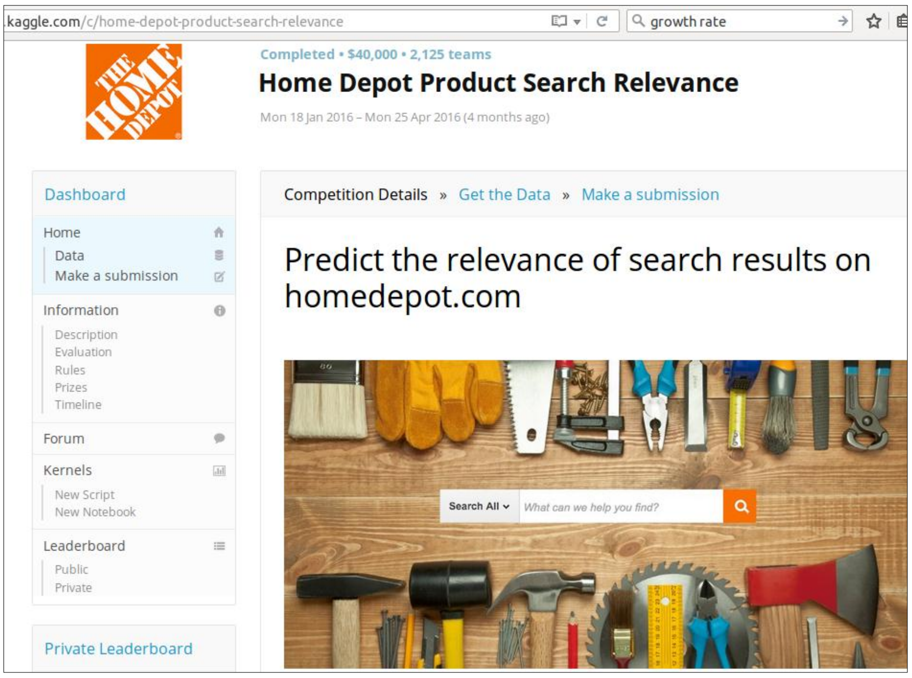
In **[18]**, it analyses the use of skip-grams when trying to model all possible combinations of words extracted from large text corpus. The results show that skip-grams are highly valuable to model context; moreover, the use of skip-grams is explained to be a better option than increasing the size of text corpus that would require more computing processing.

5. Experiments

5.1 Understanding the problem

In the Kaggle's "Home Depot Product Search Relevance" competition, it is asked to build a predictive model that could give the relevance of search queries of Home Depot's web site users and the results given by the search algorithm.

Understanding the context, in Home Depot's web site **[2]**, buyers go online to buy the products and find solutions to their necessities of home improvement. It is very important the accuracy and the speed of the searches in order to give customers the best user experience. Currently, in Home Depot, human evaluators analyse the impact of potential changes to parameters of the search algorithms, which is a slow and subjective process. By removing or minimizing human intervention, it is expected that it will be increased the number of interactions of changing the search algorithms parameters.


kaggle.com/c/home-depot-product-search-relevance

Completed • $40,000 • 2,125 teams

Home Depot Product Search Relevance

Mon 18 Jan 2016 – Mon 25 Apr 2016 (4 months ago)

Predict the relevance of search results on homedepot.com

Figure 13. Home Depot **[2]**

5.1.2 Datasets

The provided datasets have a considerable number of search terms of products and their correspondent results, product titles, descriptions and attributes.

In order to create "ground truth" labels (precision in the training dataset), Home Depot has given "search term/product" pairs to multiple human evaluators. Relevance is denoted with a real number between 1 (no relevant) and 3 (highly relevant). For example, a search query “AA battery”, would be considered highly relevant for the product “AA battery package” (relevance = 3), medially relevant for the product “battery for wireless drill” (relevance = 2) and no relevant for “snow shovels” (relevance = 1)

Each pair was evaluated at least 3 times; the final evaluation value is the average of all evaluations. The task is to predict the relevance of each pair of the test dataset.

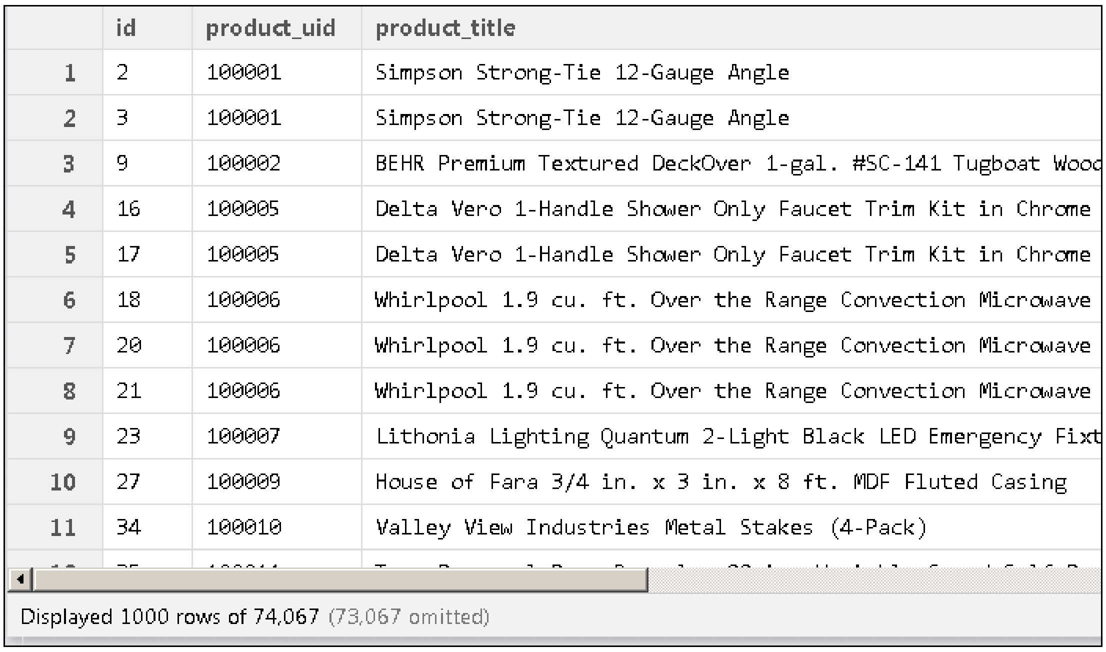
Datasets descriptions
- **train.csv:** The training dataset, it contains products ids, searches and relevance value.
- **test.csv:** It is the test dataset, it contains products ids, searches, but the relevance of search has to be predicted.
- **product_descriptions.csv:** It contains descriptive text for each product. Using the product_id it is possible to link this dataset with train.csv and test.csv.
- **attributes.csv:** It provides additional information about a subset of products (typically technical details), not every product has this attributes.
- **sample_submission.csv:** This file shows the right format for submission.

The fields found in the datasets are:
- **id:** A unique field identifier that represents a pair search_term/product_uid.
- **product_uid:** The identifier of the products.
- **product_title:** The title of the product.
- **product_description:** The description of the product, (it could contain HTML content).
- **search_term:** The search query typed by users.
- **relevance:** It is the average of relevance evaluation for a pair search_term/product.
- **name:** The name of the attribute. (attributes.csv)
- **value:** The value of the attribute. (attributes.csv)

The following R command shows the structure of the train dataset.

```
> str(train)
'data.frame':   74067 obs. of  5 variables:
 $ id            : int  2 3 9 16 17 18 20 21 23 27 …
 $ product_uid   : int  100001 100001 100002 100005 100005 100006 100006
100006 100007 100009 …
 $ product_title: Factor w/ 53489 levels "# 62 Sweetheart 14 in. Low
Angle Jack Plane",..: 44305 44305 5530 12404 12404 51748 51748 51748
30638 25364 …
 $ search_term   : Factor w/ 11795 levels "$ hole saw",". Exterior floor
stain",..: 1952 6411 3752 8652 9528 3499 7146 7148 4417 7026 …
 $ relevance     : num  3 2.5 3 2.33 2.67 3 2.67 3 2.67 3 …
```

The figure 14 displays some fields and instances of the **train** dataset.



	id	product_uid	product_title
1	2	100001	Simpson Strong-Tie 12-Gauge Angle
2	3	100001	Simpson Strong-Tie 12-Gauge Angle
3	9	100002	BEHR Premium Textured DeckOver 1-gal. #SC-141 Tugboat Wood
4	16	100005	Delta Vero 1-Handle Shower Only Faucet Trim Kit in Chrome
5	17	100005	Delta Vero 1-Handle Shower Only Faucet Trim Kit in Chrome
6	18	100006	Whirlpool 1.9 cu. ft. Over the Range Convection Microwave
7	20	100006	Whirlpool 1.9 cu. ft. Over the Range Convection Microwave
8	21	100006	Whirlpool 1.9 cu. ft. Over the Range Convection Microwave
9	23	100007	Lithonia Lighting Quantum 2-Light Black LED Emergency Fixt
10	27	100009	House of Fara 3/4 in. x 3 in. x 8 ft. MDF Fluted Casing
11	34	100010	Valley View Industries Metal Stakes (4-Pack)

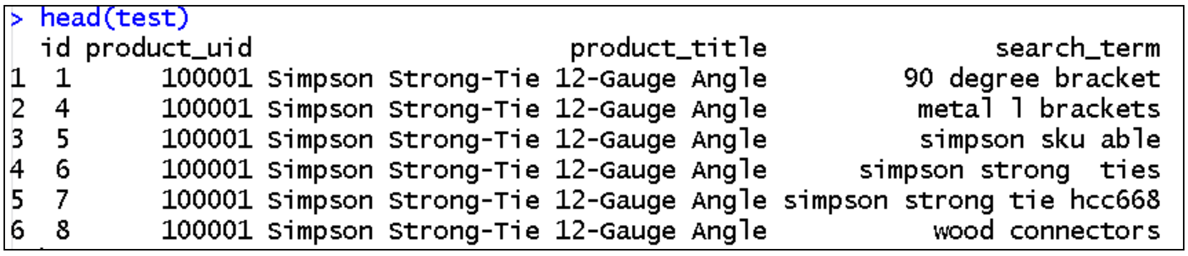
Displayed 1000 rows of 74,067 (73,067 omitted)

Figure 14. Train dataset 74067 rows.

The structure of the **test** dataset is shown below.

```
> str(test)
'data.frame':   166693 obs. of  4 variables:
 $ id           : int  1 4 5 6 7 8 10 11 12 13 ...
 $ product_uid  : int  100001 100001 100001 100001 100001 100001 100003
100003 100003 100004 ...
 $ product_title: Factor w/ 94731 levels "#10-24 Coarse Zinc-Plated Steel
Cap Nuts (4-Pack)",..: 78732 78732 78732 78732 78732 78732 82350 82350
82350 37014 ...
 $ search_term  : Factor w/ 22427 levels "'1-3/4' tap wrench",..: 3274
13720 18407 18409 18411 22101 4351 4357 14639 18817 ...
```

The figure 15 shows the firsts instances of the **test** dataset.



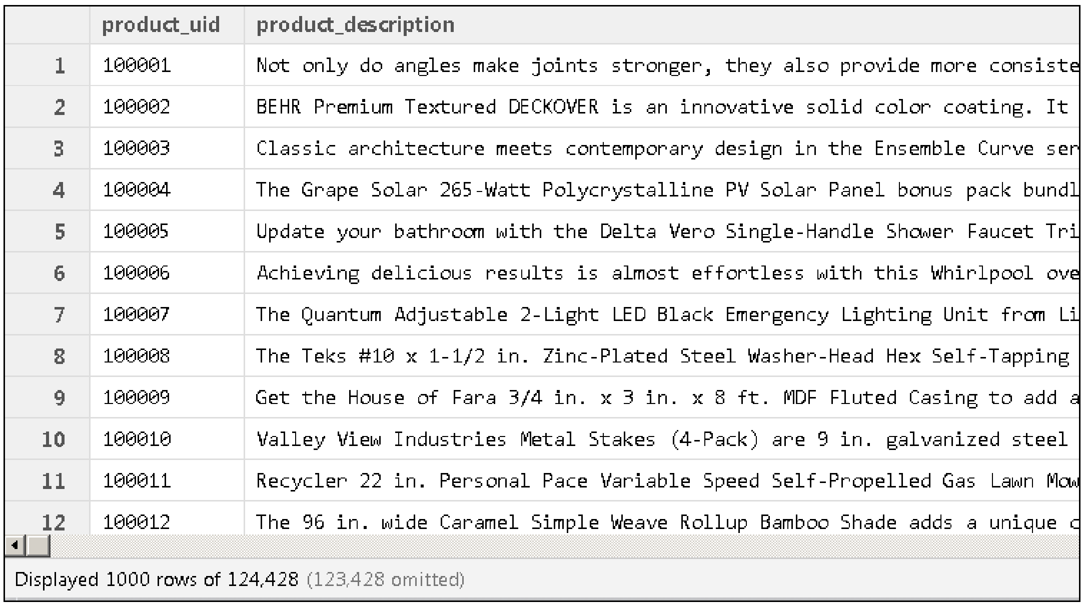
```
> head(test)
  id product_uid                     product_title                search_term
1  1      100001 Simpson Strong-Tie 12-Gauge Angle          90 degree bracket
2  4      100001 Simpson Strong-Tie 12-Gauge Angle           metal l brackets
3  5      100001 Simpson Strong-Tie 12-Gauge Angle            simpson sku able
4  6      100001 Simpson Strong-Tie 12-Gauge Angle        simpson strong  ties
5  7      100001 Simpson Strong-Tie 12-Gauge Angle simpson strong tie hcc668
6  8      100001 Simpson Strong-Tie 12-Gauge Angle            wood connectors
```

Figure 15. Test dataset 166693 rows.

The structure of the **product_description** dataset is shown below

```
> str(product_descriptions)
'data.frame':	124428 obs. of  2 variables:
 $ product_uid        : int  100001 100002 100003 100004 100005 100006 
100007 100008 100009 100010 ...
 $ product_description: Factor w/ 110128 levels "\"Building Outdoor 
Structures\" offers practical, easy-to-follow instructions on enhancing 
any home's front and backyard with t"| __truncated__,..: 37430 8184 11605 
66590 102940 3044 78451 84019 24867 104547 ...
```

The figure 16 shows the first instances of the **product_description** dataset.



	product_uid	product_description
1	100001	Not only do angles make joints stronger, they also provide more consiste
2	100002	BEHR Premium Textured DECKOVER is an innovative solid color coating. It
3	100003	Classic architecture meets contemporary design in the Ensemble Curve ser
4	100004	The Grape Solar 265-Watt Polycrystalline PV Solar Panel bonus pack bundl
5	100005	Update your bathroom with the Delta Vero Single-Handle Shower Faucet Tri
6	100006	Achieving delicious results is almost effortless with this Whirlpool ove
7	100007	The Quantum Adjustable 2-Light LED Black Emergency Lighting Unit from Li
8	100008	The Teks #10 x 1-1/2 in. Zinc-Plated Steel Washer-Head Hex Self-Tapping
9	100009	Get the House of Fara 3/4 in. x 3 in. x 8 ft. MDF Fluted Casing to add a
10	100010	Valley View Industries Metal Stakes (4-Pack) are 9 in. galvanized steel
11	100011	Recycler 22 in. Personal Pace Variable Speed Self-Propelled Gas Lawn Mow
12	100012	The 96 in. wide Caramel Simple Weave Rollup Bamboo Shade adds a unique c

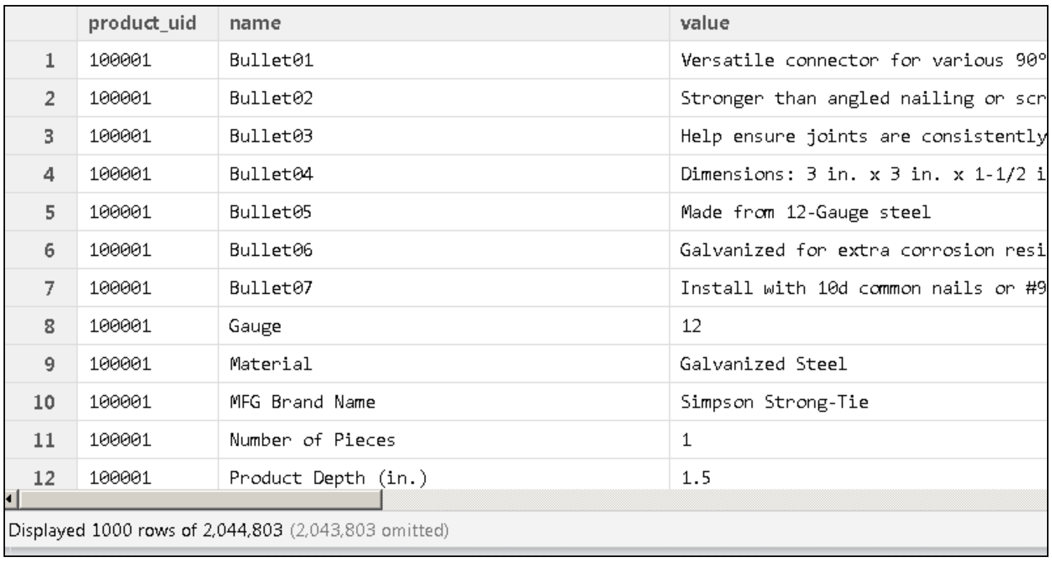
Displayed 1000 rows of 124,428 (123,428 omitted)

Figure 16. Product_description dataset 124428 rows.

The structure of the **attributes** dataset is show below.

```
> str(attributes)
'data.frame':	2044803 obs. of  3 variables:
 $ product_uid: int  100001 100001 100001 100001 100001 100001 100001 
100001 100001 100001 ...
 $ name       : Factor w/ 5411 levels "","# of Line Wires",..: 595 596 
597 598 599 600 601 1932 2684 2933 ...
 $ value      : Factor w/ 307591 levels "","'U.S Patented'",..: 296931 
272215 168106 120013 200857 157944 185144 15553 157982 260160 ...
```

Figure 17 shows the first instances of the **attributes** dataset.



	product_uid	name	value
1	100001	Bullet01	Versatile connector for various 90°
2	100001	Bullet02	Stronger than angled nailing or scr
3	100001	Bullet03	Help ensure joints are consistently
4	100001	Bullet04	Dimensions: 3 in. x 3 in. x 1-1/2 i
5	100001	Bullet05	Made from 12-Gauge steel
6	100001	Bullet06	Galvanized for extra corrosion resi
7	100001	Bullet07	Install with 10d common nails or #9
8	100001	Gauge	12
9	100001	Material	Galvanized Steel
10	100001	MFG Brand Name	Simpson Strong-Tie
11	100001	Number of Pieces	1
12	100001	Product Depth (in.)	1.5

Displayed 1000 rows of 2,044,803 (2,043,803 omitted)

Figure 17. Attributes dataset 2044803 rows.

In order to have an idea of the searches made by users, the R's **worldcloud** and **tm** packages were used to generate the word cloud image for the **search_term** field of the train dataset. Figure 18 shows the word cloud.

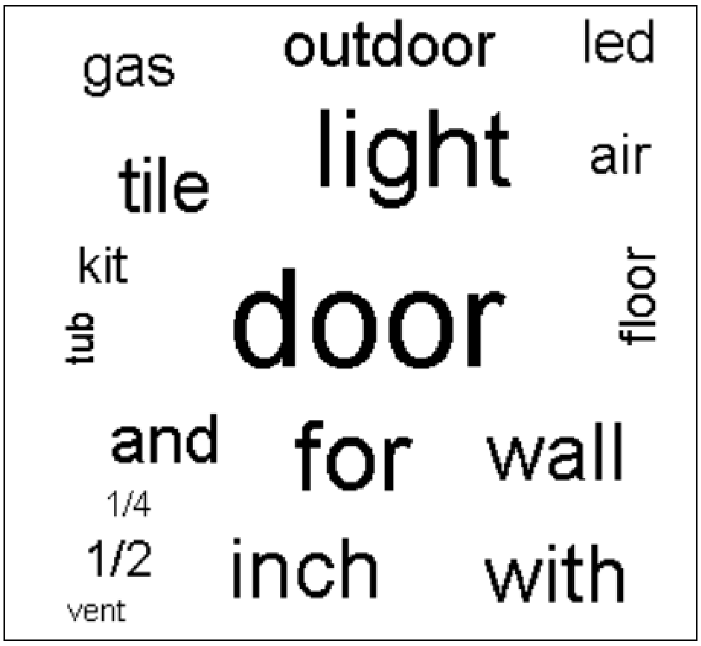


Figure 18. Word Cloud of search_term field of train dataset.

The figure 19 shows the resulting word cloud with the most frequent words found in **product_title** of the **train** dataset.

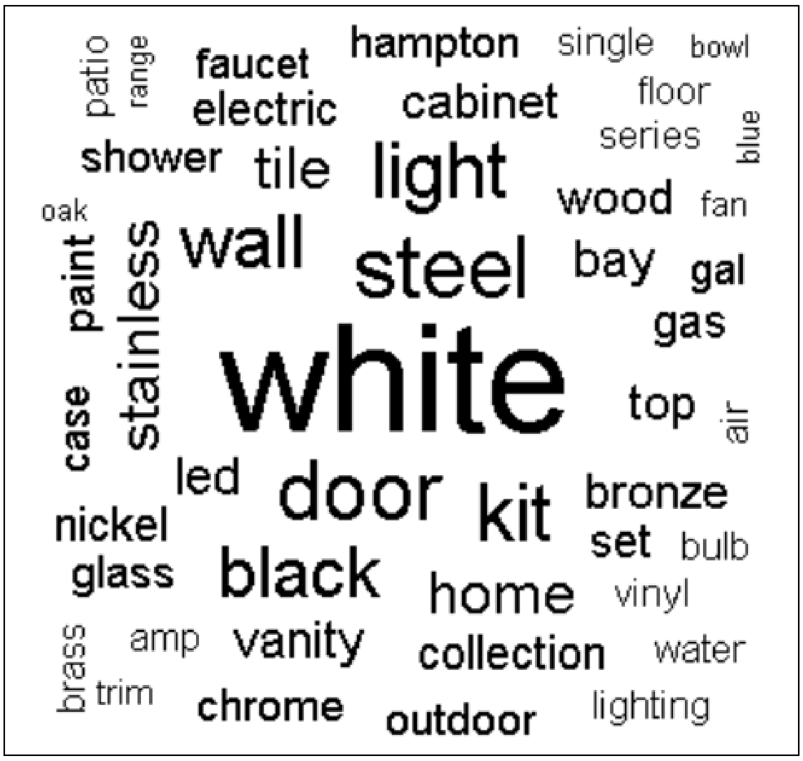


Figure 19. Word Cloud of product_title field of train dataset.

Putting an eye to the **relevance** field in the **train** dataset, the following histogram is obtained:

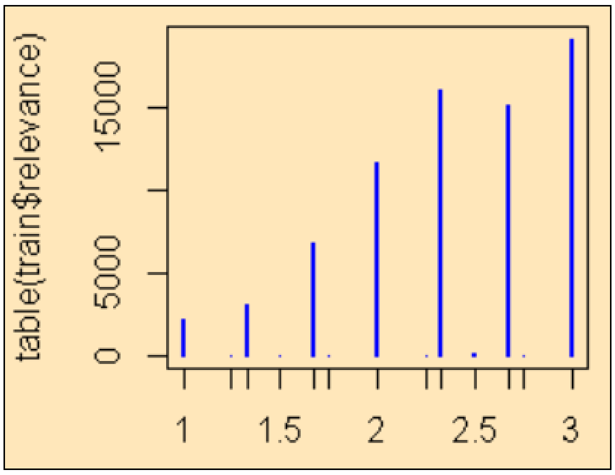


Figure 20. Histogram of relevance field of train dataset

Taking into account the number of instances of the **train** and **test** datasets, it can be seen that the **train** dataset is small compared to the predictive **test** dataset. The figure 21 shows this comparison:

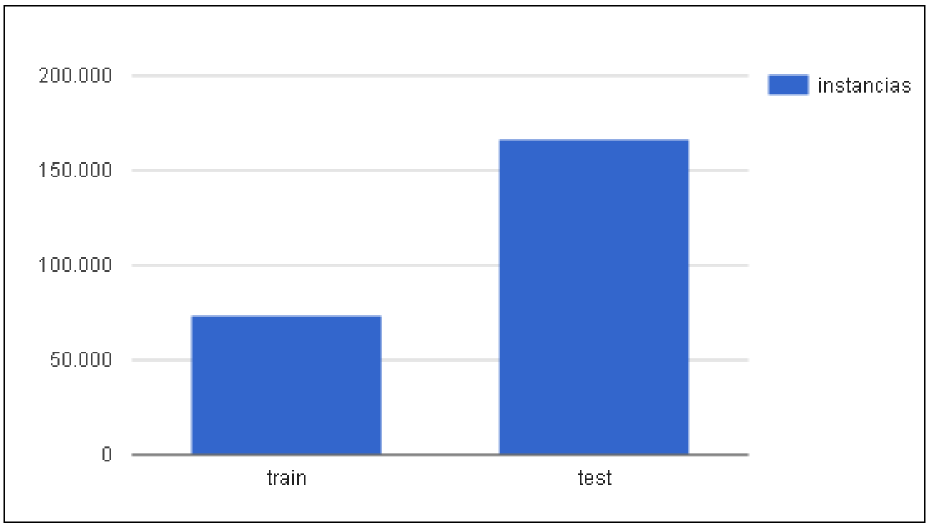


Figure 21. Comparison between the number of instances of train and test datasets

5.2 Software packages and algorithms

Many software tools can be used for building predictive models. Each of them has advantages and disadvantages; the most popular are Python, R, SAS, Matlab or even spreadsheets like Microsoft Excel. The figure 22 shows the use of different statistical programming languages in Kaggle's competitions:

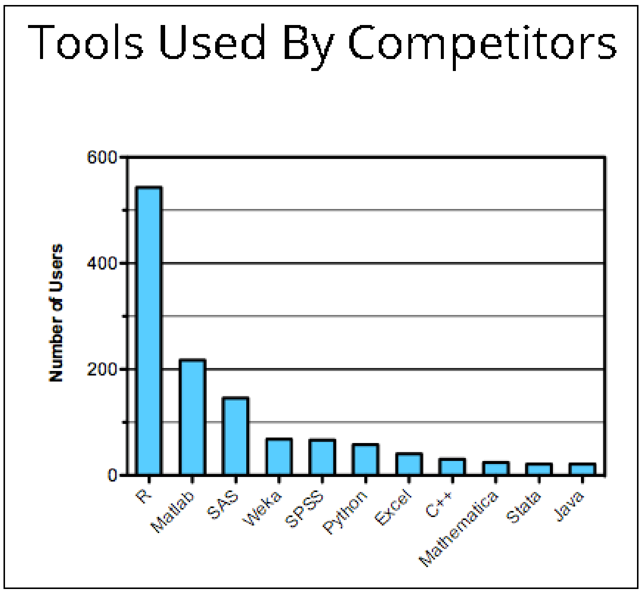


Figure 22. Tools used by Kaggle's competitors **[1]**

Among many options, the software chosen for this project is the R language which is a powerful, easy to use, open-source, statistical software. R’s large and active community develops packages for a variety of topics. Moreover, been a multi-platform software it allows working with machines running GNU Linux and Microsoft Windows operating systems.

Figure 23. R language logo

R packages

Many R packages were used in this project to perform specialized tasks; these packages helped to overcome complex problems. The table 4 describes the use of R packages in the present project.

Package	**Description**
NLP	It is a package for Natural Language Processing. It enables to do tasks as tokenizing.
data.table	It enables to have data structures and functions for storing large amounts of data.
Stringr	It is a package for working with strings in matrix and arrays.
randomForest	It is used for applying the Random Forest algorithm.
RmySQL	It is a package for connecting to the MySQL database; it allows reading and writing data to tables.
Lsa	It was used to calculate the cosine similarity operation.
Tm	It is used for building text corpus and NLP tasks.
snowballC	It is a base package for using with tm and wordcloud packages.
wordcloud	It was used for building the word cloud of relevant datasets
googleVis	It allows using the google API to generate graphic reports.

Table 4. Summary of R packages used in the project

Regression algorithms

Many algorithms and techniques can be applied for prediction; most of them are available in R packages. The figure 24 shows the comparison of use of algorithms by Data Scientists.

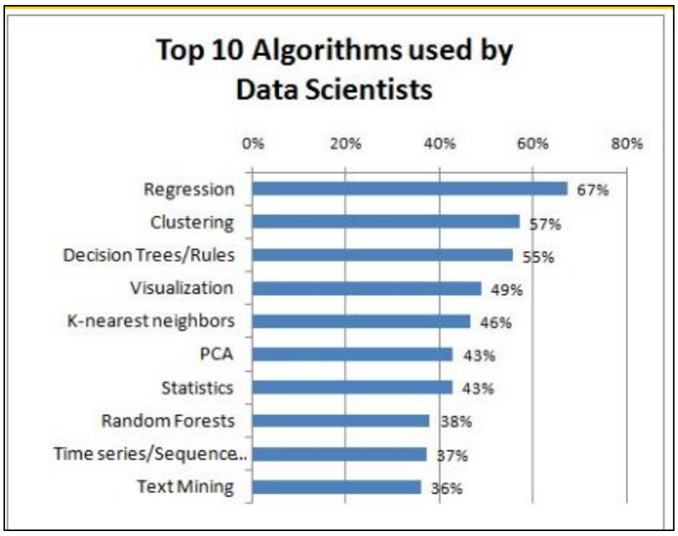


Figure 24. KDNuggets' poll : Top 10 Algorithms used by Data Scientists **[19]**

Besides the academic use of Data Science algorithms, it is important to consider the use of those algorithms in real world problems. The figure 25 shows the use of Data Science algorithms taking into consideration the industry vs. academic fields.

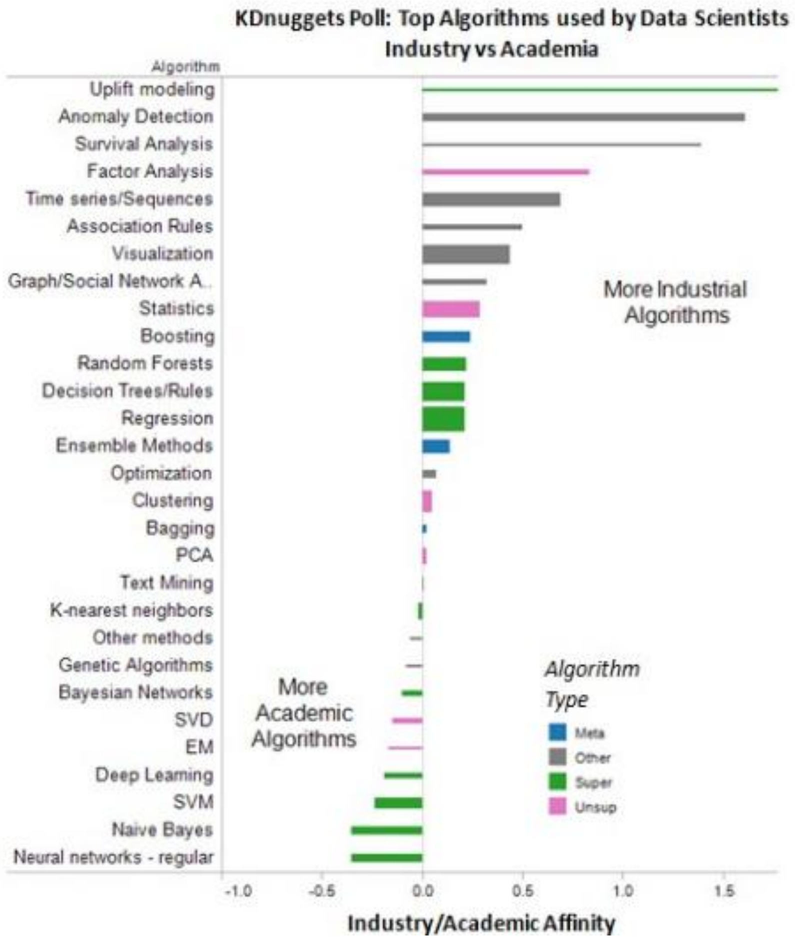


Figure 25. KDNuggets' poll: Top 10 Algorithms used by Data Scientists. Industry vs. Academy **[19]**

As it was mentioned before, a regression algorithm will be used for prediction in the present project because it has numerical continuous data. The algorithm considered for applying regression is Random Forest due to its good performance and precision.

5.3 Methodology for the solution of this project

This project is a Machine Learning problem in which the model will learn to predict values from thousands of known examples. It falls in the category of supervised learning, because the algorithm has to learn from a dataset (**train** dataset) that has examples of correct values. Once the model has learned by the implementation of a regression algorithm, the model will be tested with the instances stored in the **test** dataset. The accuracy of the model will be measured and after refining its parameters it will be ready for predicting new instances.

For building the predictive model some steps have to be taken. First of all, a data pre-processing step is necessary to prepare and clean the data. After that, a Feature Extraction step will be conducted in which it will be got the most relevant features that can be applied in a regression algorithm, since regression algorithms need numerical data, it is necessary to generate numerical features from the text attributes; moreover overlap and distributed word embeddings are calculated. The random forest algorithm is used for supervised learning and obtaining the prediction variable. Finally an error metric has to be applied to measure the accuracy of the predicting model. The figure 26 describes the methodology used in the present project:

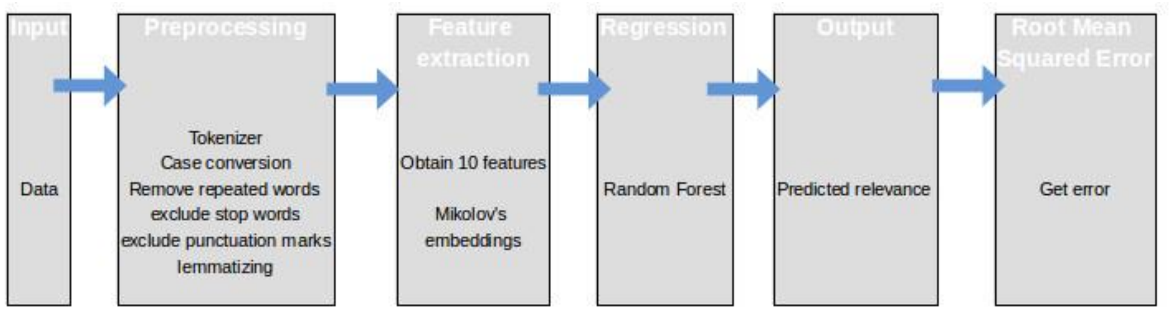


Figure 26. Prediction model overview

Experimenting with the train dataset

All the experiments use the **train** dataset because the **test** dataset will only be used at the end of the project for doing the final evaluation of the model. The **train** dataset is split randomly in 3 parts, so it will be possible to do the training and testing processes during the experimentation.

The algorithm considered for applying regression is Random Forest; to get the best results several experiments will be conducted to find the optimal Random Forest’s hyper-parameters.

The experiments will also use different features in order to get those which give the most accurate results. These features are combinations of the following forms:

Form of feature	Explanation	Operation
Original words	Original words	Overlap
Word lemmas	Canonical form, dictionary form of the words, for example: run, runs, running, ran have run as their lemma.	Overlap
Distributed word embeddings	Numerical vectors representing the similarity of the words	Cosine similarity

Table 5. Form of features used in the experiments

Once the optimal parameters have been found the model will be evaluated with the instances of the **test** dataset.

5.3.1 Data Pre-processing

Data pre-processing is a very important step for the success of the project. In real world problems, data is not found in a structured form; besides a lot of data cleaning work is necessary, there are incomplete, redundant, not relevant values. Data pre-processing may take most of the time and effort in a project but it will ease the process to a well-done work.

In the present project many steps of data pre-processing were done, the following section describes them in detail.

5.3.1.1 Case conversion

Case conversion in R is very easy, the **tolower** function converts text to lower case; the following code converts to lower case the third and fourth columns of the **train** data.frame.

```
train[,3] = tolower(train[,3])   #minusculas
train[,4] = tolower(train[,4])
```

5.3.1.2 Tokenization

In order to perform tokenization, it was necessary to install the R package NLP, this package has many Natural Language Processing algorithms; the following functions are used for tokenization:

blankline_tokenizer(): the separator are any blank lines.

whitespace_tokenizer(): the whitespace is the separator.

wordpunct_tokenizer(): tokenizes by finding patters of sequence of alphabetic characters and sequences of non-alphabetic characters (non-whitespaces).

5.3.1.3 Stop words

The stop words are the ones which are filtered out before or after processing of natural language data. There are lists of stop words in many languages.

In this project, it was necessary to download a list of stop words of the English language **[20],** and load them to the R environment, these words were loaded to a vector that was used to compare to the texts of the datasets. The following code was used to load the stop words into a vector and clean the data of the vector.

```
stop <- read.csv("stop.txt", header=FALSE)
dim(stop)

words <- NULL
for (i in 1:dim(stop)){
   texto=stop[[1]][i]
   texto=paste(texto, " ")
   index=str_locate(texto, " ")

   texto2=substr(texto, 1, index[[1]][1])
   t3=str_replace_all(texto2, " ", "")
   if (t3 !="") words <- c(words, t3)
}
```

Despite being found frequently in a language, the stop words are considered irrelevant for text analysis; the following R's command shows the stop words for the English language:

```
> stopwords('english')
  [1] "i"          "me"         "my"         "myself"     "we"
  [6] "our"        "ours"       "ourselves"  "you"        "your"
 [11] "yours"      "yourself"   "yourselves" "he"         "him"
 [16] "his"        "himself"    "she"        "her"        "hers"
 [21] "herself"    "it"         "its"        "itself"     "they"
 [26] "them"       "their"      "theirs"     "themselves" "what"
 [31] "which"      "who"        "whom"       "this"       "that"
 [36] "these"      "those"      "am"         "is"         "are"
 [41] "was"        "were"       "be"         "been"       "being"
 [46] "have"       "has"        "had"        "having"     "do"
 [51] "does"       "did"        "doing"      "would"      "should"
 [56] "could"      "ought"      "i'm"        "you're"     "he's"
 [61] "she's"      "it's"       "we're"      "they're"    "i've"
 [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
 [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
 [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
 [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
 [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
 [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
 [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
[101] "who's"      "what's"     "here's"     "there's"    "when's"
[106] "where's"    "why's"      "how's"      "a"          "an"
[111] "the"        "and"        "but"        "if"         "or"
[116] "because"    "as"         "until"      "while"      "of"
[121] "at"         "by"         "for"        "with"       "about"
[126] "against"    "between"    "into"       "through"    "during"
[131] "before"     "after"      "above"      "below"      "to"
[136] "from"       "up"         "down"       "in"         "out"
[141] "on"         "off"        "over"       "under"      "again"
[146] "further"    "then"       "once"       "here"       "there"
[151] "when"       "where"      "why"        "how"        "all"
[156] "any"        "both"       "each"       "few"        "more"
[161] "most"       "other"      "some"       "such"       "no"
[166] "nor"        "not"        "only"       "own"        "same"
[171] "so"         "than"       "too"        "very"
```

5.3.1.4 Exclude punctuation marks

A vector with the most frequent punctuation marks was defined in the pre-processing stage; these symbols were omitted from the text of the datasets. The following line of code shows the vector.

```
punctuation <- c(",", ".", "(", ")", "-", ";", "[", "]", "/", "+",
            "#", "%", "$", "*", "=", """, "?")
```

At the beginning, the question mark symbol “?” was not considered for exclusion with the punctuation marks, but when the scripts were run, there were errors caused by this symbol, so it was necessary to exclude it from the text.

5.3.1.5 Remove repeated words

In order to increase the accuracy of the model, the repeated words in an instance were omitted, to accomplish this, it was necessary to write a function that was used in the pre-processing stage for all the string fields of the datasets. Installing R packages like “stringr” was also necessary.

5.3.1.6 Lemmatizing

Lemmatizing the entire dataset is a key part of this project, for doing it, it was used the **ixa-pipe.pos** tool which is a Part of Speech tagger and Lemmatizer that support several languages.

After running the ixa-pipe.pos tools **[21],** new files were created, these files contain the dataset's lemmas. The **_pos** suffix was used to identify the .csv files.

"train_pos.csv"
"test_pos.csv"
"product_descriptions_pos.csv"
“attributes_pos.csv”

5.3.2 Splitting the train dataset

In order to test the model in early stages and before using the **test** dataset, it was considered to split randomly the **train** dataset in 3 parts as it follows:

- 50% for development
- 30% for training
- 20% for testing

The following code was used for splitting the **train** dataset in 3 parts. The **sample** function was used for generating randomness and the argument **replace=FALSE** guaranteed no repetition of values.

```
#extraer porcentajes
tamano <- dim(train)[1]

indices <- sample(tamano, tamano, replace=FALSE)
indices
indices50 <- indices[1:as.integer(tamano*.5)]
indices30 <- indices[as.integer(tamano*.5+1):as.integer(tamano*.8)]
indices20 <- indices[as.integer(tamano*.8+1):as.integer(tamano)]
```

The number of instances of each resulting dataset is shown in table 6.

Dataset	# of instances	Percentage
Train	37033	50%
Development	14814	20%
Test	22220	30%
Total	**74067**	**100%**

Table 6. Number of instances of the resulting datasets after splitting the train dataset

Figure 27 represents the percentage of the split of the **train** dataset.

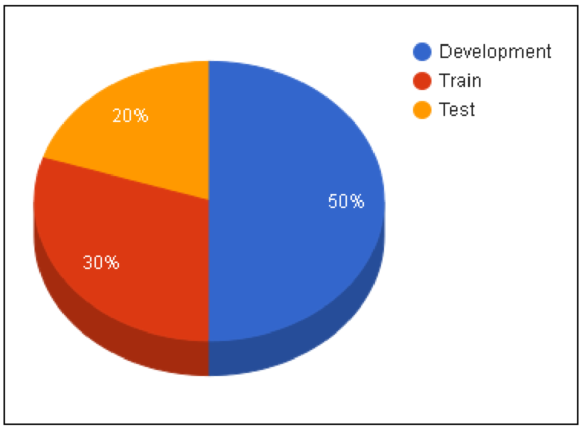


Figure 27. Splitting train dataset into 3 groups

5.3.3 Feature extraction

Since the data of the datasets are text values, it is necessary to convert text data into numeric data in order to be able to perform analysis of that data and building a predictive model. For generating numerical features it will be applied two operations: overlap and distributed word embeddings.

There are 2 main datasets the **train** and **test** datasets that contain a **search_term** field. The **train** dataset also contains the **relevance** field which will be used with the model for training. The **product_description** field of the **product_descriptions** dataset also will be used for feature generating.

The **attributes.csv** dataset contains additional information of the products, but not every product is in this dataset. The following code shows the **value** field of the **attributes**:

```
> attributes[,2]
 [9995] Bullet04
 [9996] Bullet05
 [9997] Bullet06
 [9998] Certifications and Listings
 [9999] Detection Range (ft.)
[10000] Dusk to Dawn
 [ reached getOption("max.print") -- omitted 2034803 entries ]
```

5411 Levels: 15 Gauge Finish Nailer Included ... Zone-specific Sounds

It can be seen that there is 5411 Levels for the **name** field of the **attributes** dataset, among them, 2 attribute names were mainly considered, “Material” and “Brand”, which were taken as features for prediction. The rest of the levels were group in a single prediction feature called “others”. The table 7 shows the presence of features in the datasets.

Attribute name	Frequency
Product title	Every instance
Product description	Every instance
Material	Some
Brand	Some
Others: Bullet01, Bullet02, Gauge, etc.	Some

Table 7. Comparison of the presence of features in the **train** and **test** datasets

The main idea is to predict the relevance of the search according to the similarity of the search query with the product's description and attributes. The following fields have been considered as predicting features:

- ✓ Product title
- ✓ Product description
- ✓ Material
- ✓ Brand
- ✓ Others: Bullet01, Bullet02, Gauge, “Deck use”, …

5.3.3.1 Overlap

Overlap is a technique is which it is calculated the percentage of words present in the query that are found in an attribute. The formula 3 represents the overlap.

$$\text{Overlap} = \frac{\text{Words in “search query” found in “product's attribute”}}{\text{Length of the search query}}$$

Formula 3. Overlap

Overlap will be used between the **search_query** field of the **train** and **test** datasets against the **product_title** (**train** and **test** datasets), **product_description** (**product_descriptions** dataset), **value** (**attributes** dataset) fields.

5.3.3.2 Mikolov word embeddings

The most difficult part of the project was the use of Mikolov word embedding **[22]** because its calculation requires large amount of computer processing and time. The embeddings are free to download from the Internet in different sources; In this project the Mikolov word embeddings are formed by 2 files. The table 8 describes the files containing the Mikolov word embeddings

File name	Size in bytes	Number of instances
Emb-CorCBgoog-S300W5SS1e-5HS0NG3.txt	8,564,012,821	3,000,000
VocabGoogle.txt	44,258,510	3,000,000

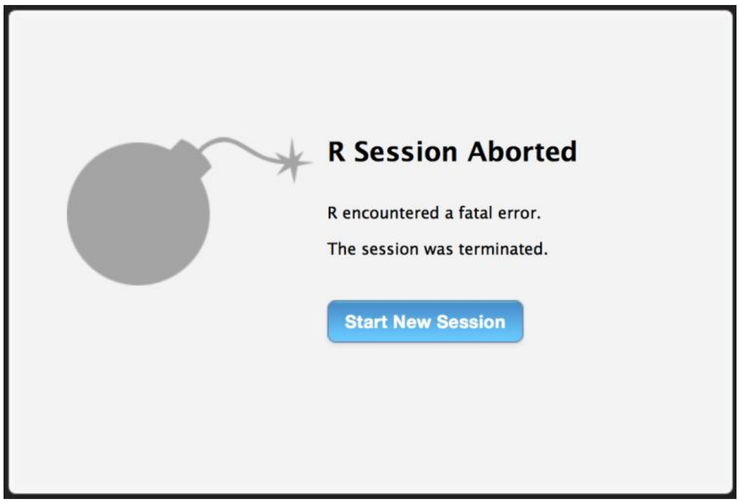
Table 8. Features of word embedding files

With the files provided in table 8, it was necessary to build a word vector for each instance of the datasets, after doing that it will be possible to build a predictive model using word embeddings.

It can be seen in the table 8 that the Emb-CorCBgoog-S300W5SS1e-5HS0NG3.txt file has 3 million instances and also has 300 numerical columns; each of them represents a word vector. The size of the file is near 9 GB. To perform the pre-processing of data, it was used a laptop computer with the following features:

Brand/model: Dell Inspiron 14.
Processor: Intel Core i5 1.80 GHz x 4.
Memory: 6 GB RAM.
O.S.: Linux Ubuntu 15.10.

After some attempts of building the word vector for the instances of the datasets with the machine described above, it started to lose processing speed because the machine started to run out of memory, getting crashes like the error shown in figure 28.


R Session Aborted

R encountered a fatal error.

The session was terminated.

Figure 28. R Studio error message got while calculating the word embeddings

In a second attempt of building the vector, the idea was to store the results in a MySQL database, for doing this, it was necessary to install the **RmySQL** package. This approach looked good but the processing still took a lot of time for all the datasets.

Finally, another improvement was made, in which the embeddings were reduced only to the words present in the datasets; the resulting files are described in the table 9.

File name	Size in bytes	Number of instances
EmbKKaggle.txt	82,467,194	28897
VocabKaggle.txt	234,417	28897

Table 9. Features of word embeddings files after applying the reduction to the words present in datasets

It is described in the table 9 that the new EmbKKaggle.txt file has only 28897 instances and its size is just over 82 MB. Generating these files enables working with a normal desktop or laptop computer; less computer processing is needed and less time will be taken to get results.

With the considerable reduction of words it was possible to start building the vector with the available hardware. The processing time needed for calculating the word vectors for all datasets is shown in the table 10.

Dataset file	Field	# of instances	Processing time
Train.csv	Product Title	74067	15 hours
Train.csv	Search query	74067	7 hours
Test.csv	Product Title	166693	1 day 5 hours
Test.csv	Search query	166693	20 hours
Product_descriptions	product_description	124429	5 days 2 hours
Attributes.csv	Value	2044804	7 days 18 hours

Table 10. Processing time for datasets applying Mikolov word embeddings

As it can be seen, the product_descriptions and attributes datasets took long time to be processed and had to be split in 2 parts in order to process them in 2 different computers.

5.3.3.3 Cosine similarity

To calculate the cosine similarity between the embeddings of the **search_term** field and the embeddings attributes of the products, the R's **lsa** package was used.

The following figures show the results of applying the cosine similarity to the datasets.

In figure 29 a box plot graphic represents the scores of cosine similarity applied to the embeddings of the **search_term** and **product_title** fields of the **train** dataset. It can be seen that the first quartile is above 0.4 which means most of the values have a significant cosine similarity score. Some values are between 0 and 0.2 having little value of cosine similarity. There is 1 outlier that has a 1 of score meaning complete similitude between instances.

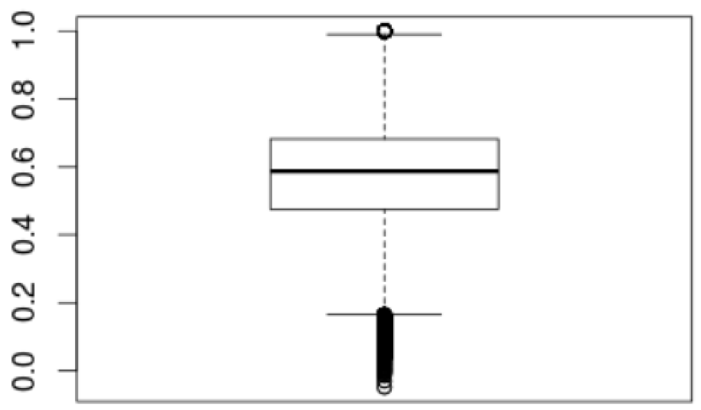


Figure 29. Boxplot of cosine similarity between **search_term** and **product_title** fields of the **train** dataset

Figure 30 shows the histogram of scores of cosine similarity between the **search_term** and **product_title** fields of the **train** dataset. The graphic looks pretty much like a normal distribution.

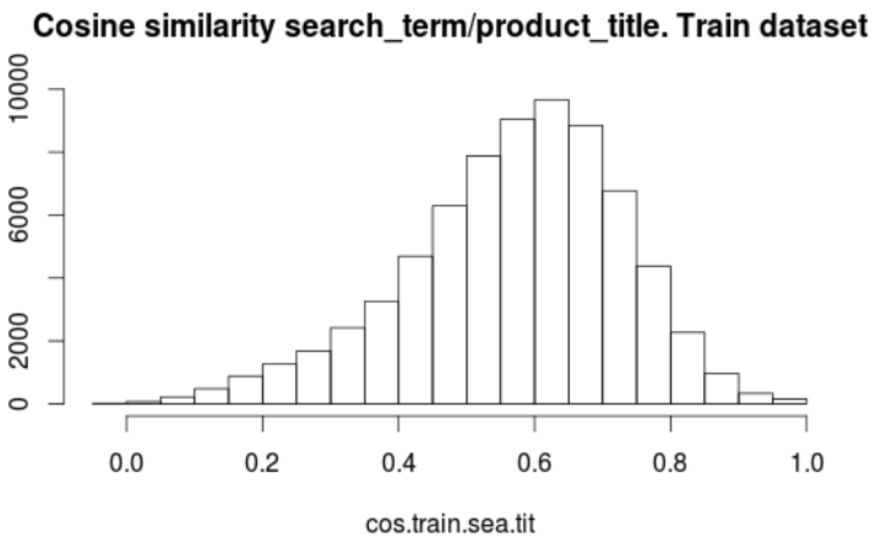


Figure 30. Histogram of cosine similarity between **search_term** and **product_title** fields of the **train** dataset

Figure 31 represents a box plot of scores of cosine similarity between the **search_term** and **product_desc** fields of the **train** dataset. It can be seen some values with a value near zero or below 0. There is an outlier with score of 1.

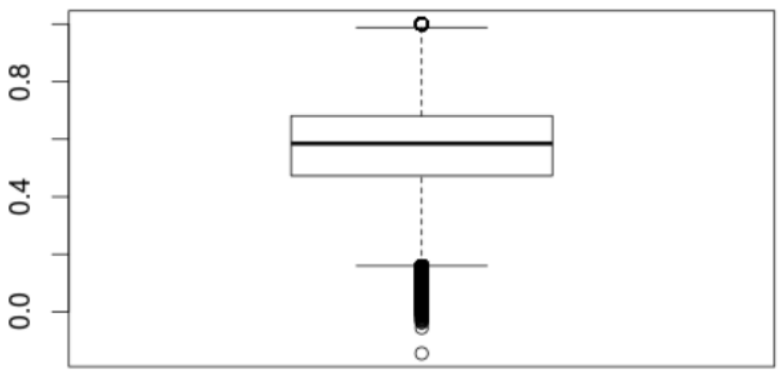


Figure 31. Boxplot of cosine similarity between **search_term** and **product_desc** fields of the **train** dataset

Figure 32 shows the histogram of scores of cosine similarity between the **search_term** and **product_desc** fields of the **train** dataset. The graphic looks also like a normal distribution.

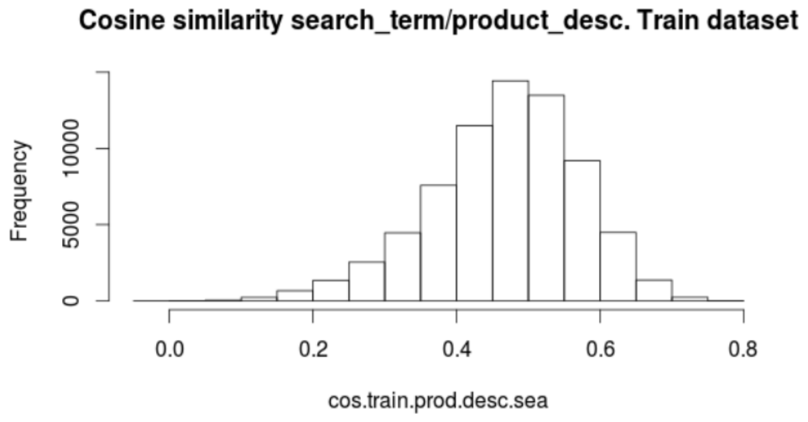


Figure 32. Histogram of cosine similarity between **search_term** and **product_desc** fields of the **train** dataset

Based in the information shown in the histograms, it can be said that the cosine similarity of the embeddings of **product_title** and **product_desc** fields are going to be very important for prediction.

Figure 33 shows the histogram of scores of cosine similarity between the **search_term** and **value** fields of the **train** and **attributes** dataset; it is considered the brand feature. The graphic show that many instances have a score of 0; this could be explained because many products don't have the brand attribute.

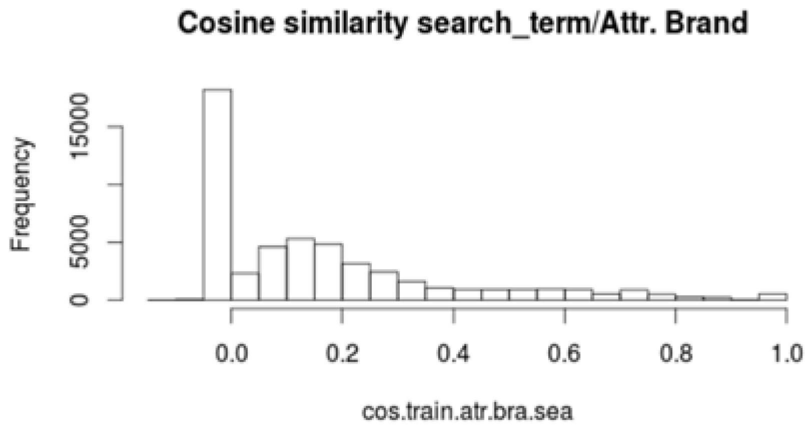


Figure 33. Histogram of cosine similarity between **search_term** and attribute: brand. **Train** dataset

Figure 34 shows the histogram of scores of cosine similarity between the **search_term** and **value** fields of the **train** and **attributes** dataset; it is considered the material feature. Likewise figure 33, the graphic show that many instances have a score of 0; as it was explained for the brand feature, many products don't have the material attribute.

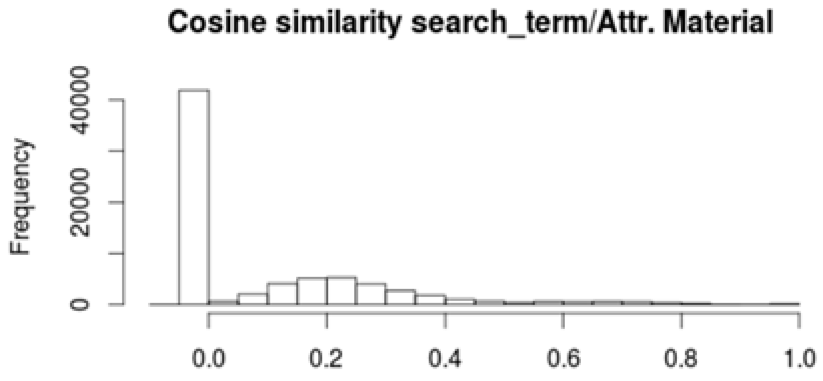


Figure 34. Histogram of cosine similarity between **search_term** and attribute: material. **Train** dataset

Figure 35 shows the histogram of scores of cosine similarity between the **search_term** and **value** fields of the **train** and **attributes** dataset; it is considered the others feature. The graphic looks like a normal distribution and it can be said that the "others" attribute has very high importance to the model because most of the instances have a good level of similarity when comparing with this feature. In the other hand, it can be seen that the attributes "brand" and "material" have much less influence and similarity.

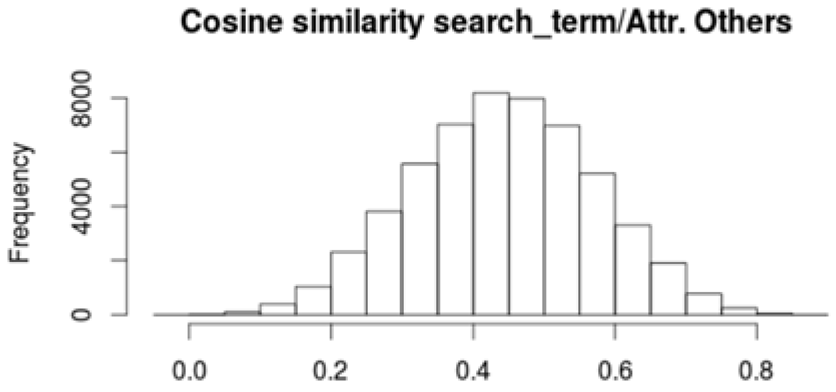


Figure 35. Histogram of cosine similarity between **search_term** and attribute: others. **Train** dataset

The following figures show the histograms of the cosine similarity found for the **test** dataset.

Figure 36 shows the histogram of scores of cosine similarity between the **search_term** and **product_desc** fields of the **test** dataset. It can be seen that many instances have a strong cosine similarity score.

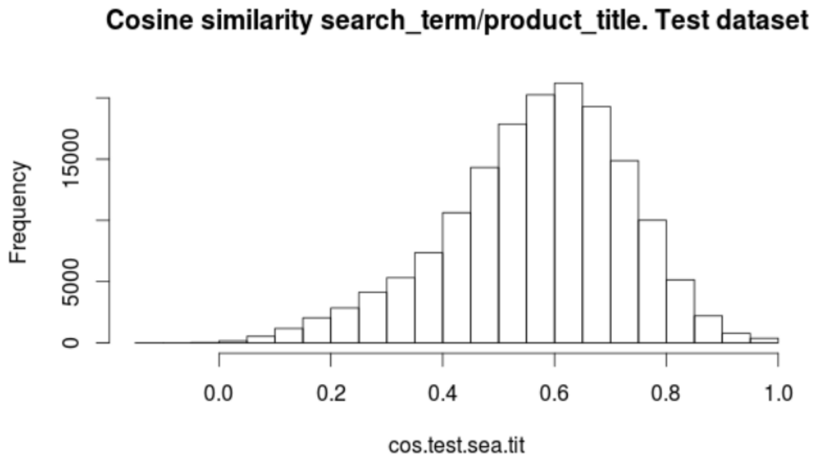


Figure 36. Histogram of cosine similarity between **search_term** and **product_title** fields of the **test** dataset

Figure 37 shows the histogram of scores of cosine similarity between the **search_term** and **product_desc** fields of the **test** dataset. The graphic looks also like a normal distribution.

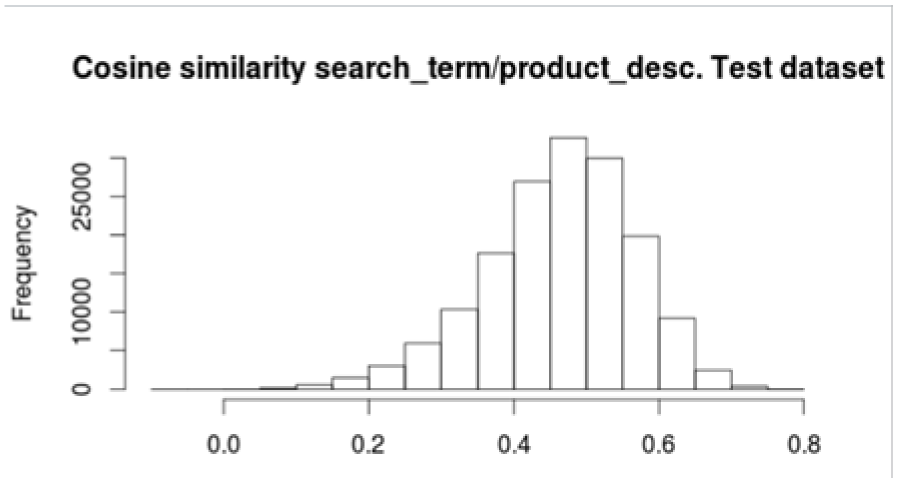


Figure 37. Histogram cosine similarity between **search_term** and **product_desc** fields of the **test** dataset

It can be seen that the effect of the cosine similarity operation is similar in **train** and **test** datasets and it will be relevant for the predictive model.

Due to the amount of data generated by the cosine similarity operation, it was necessary to store the values in the hard drive. The table 11 shows the files used to store the cosine similarity results.

File	Content
cos.test.atr.bra.sea.csv	Cosine similarity of search_term (test) – attribute brand
cos.test.atr.mat.sea.csv	Cosine similarity of search_term (test) – attribute material
cos.test.atr.oth.sea.csv	Cosine similarity of search_term (test) – attribute others
cos.test.prod.desc.sea.csv	Cosine similarity of search_term (test) – product description
cos.test.sea.tit.csv	Cosine similarity of search_term (test) – product title
cos.train.atr.bra.sea.csv	Cosine similarity of search_term (train) – attribute brand
cos.train.atr.mat.sea.csv	Cosine similarity of search_term (train) – attribute material
cos.train.atr.oth.sea.csv	Cosine similarity of search_term (train) – attribute others
cos.train.prod.desc.sea.csv	Cosine similarity of search_term (train) – product description
cos.train.sea.tit.csv	Cosine similarity of search_term (train) – product title

Table 11. Cosine similarity files.

5.3.3.4 Predictive Features

After combining all the feature extraction techniques, the resulting predictive features are obtained.

- ✓ F1: overlap(query, description):
- ✓ F2: overlap(query, title)
- ✓ F3: overlap(query, Brand)
- ✓ F4: overlap(query, Material)
- ✓ F5: overlap(query, Others)
- ✓ F6: overlap(query_lemma, description_lema):
- ✓ F7: overlap(query_lemma, title_lema)
- ✓ F8: overlap(query_lemma, Brand_lema)
- ✓ F9: overlap(query_lemma, Material_lemma)

- ✓ F10: overlap(query_lemma, Others_lemma)
- ✓ F11: cosine_similarity(embeddings_query, embeddings_desc)
- ✓ F12: cosine_similarity (embeddings_query, embeddings_title)
- ✓ F13: cosine_similarity (embeddings_query, embeddings_brand)
- ✓ F14: cosine_similarity (embeddings_query, embeddings_material)
- ✓ F15: cosine_similarity (embeddings_query, embeddings_others)

5.3.4 Regression analysis

5.3.4.1 Building a regression model

As it has been mentioned the dataset values have been converted to numerical and continuous values. That is the reason why the prediction model of the present project uses a regression analysis method.

The regression analysis will be conducted between the **relevance** field of the **train** dataset and the numerical features that have been generated from the products' attributes fields.

In order to generate the prediction results, using R language, the **randomForest** package was used. As its name says, it allows the use the Random Forest algorithm for prediction. Given the data, it generates a predicted relevance column.

5.3.4.2 Evaluating the model

To evaluate the results, the RMSE metric is used **[23]**. The Root Mean Squared Error is largely used for numerical predictions. The formula 4 describes RMSE.

$$\boxed{\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}$$

Formula 4. RMSE **[23]**

The following line of code gets the RMSE metric using the R language

```
RMSE <- sqrt(mean((y-y_pred)^2))
```

Where **y** is a vector that has the real values of the relevance field, and the **y_pred** is a vector that has the predicted values of the relevance. The variable **n** is the number of instances in the dataset.

6. Results

To evaluate the results using different scenarios, a script was written in which the hyper-parameters of the Random Forest algorithm were changed. Several experiments will be conducted in order to find the best hyper-parameters for the predicting model. The Random Forest algorithm has mainly 2 hyper-parameters to setup, the number of decision trees that will be used for training and the number of variables that also will be used for evaluating with those trees.

Another important factor is what features will be used, besides the overlap of the original attributes of the products, more features were generated; the overlap of lemmas of the attributes and the cosine similarity of word embeddings of the attributes. It is important to find the combination of features that give the best results.

All the experiments will be applied to the **train** dataset. As it was explained before, the **train** dataset was split into 3 subsets. The **train** subset will be used for training the Random Forest algorithm and the **development** subset will be used for testing during development. The combination of features with better results will be implemented in the **test** subset. (All these experiments are conducted in the train dataset).

Once the best combination of features and parameters have been found there will be a final experiment applied to the **test** dataset.

Experiment #1

For the experiment #1 the overlap of the words were used (F1 – F5); it was calculated the overlap between the **search_term** field (what the user typed) and the attributes of the product. The parameter **number of trees** takes the values 400, 500, 600, 700 and the parameter **number of variables** takes the values 2, 3, 4, 5. The figure 38 shows the values of the error metric RMSE in the experiment #1.

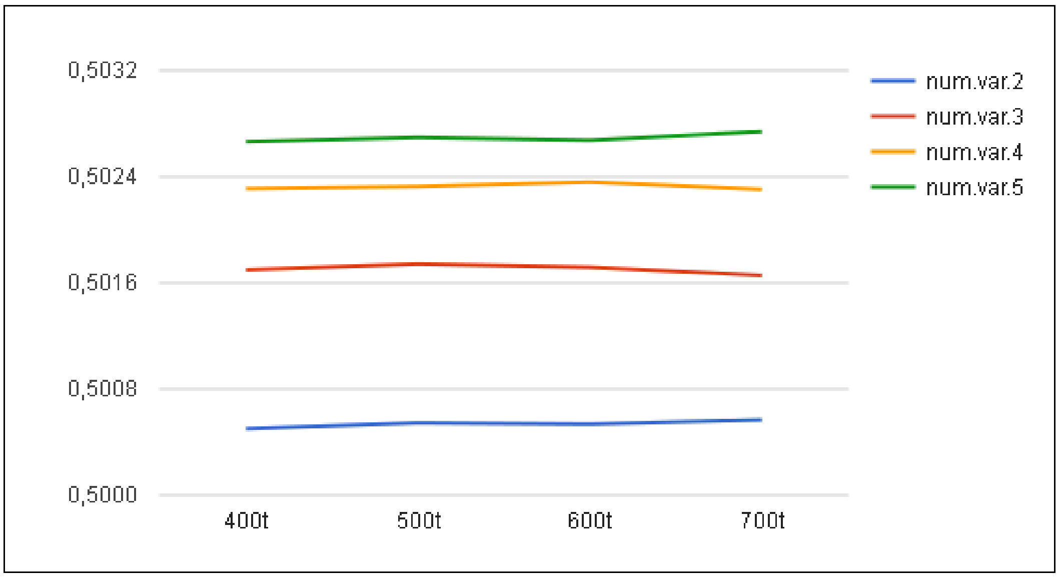


Figure 38. Graphic of RMSE in experiment #1.

Best result: 0.500501398386279
# of trees: 400
# of variables: 2

Experiment #2

The features considered for this experiment are F6 to F10 which are the overlap of the lemmas of the words of datasets. The overlap of the **search_term** and the attributes was calculated. The parameter **number of trees** takes the values 400, 500, 600, 700, 800, 900, 1000, 1100 and the parameter **number of variables** takes the values 2, 3, 4, 5. The figure 39 shows the values of the error metric RMSE in the experiment #2.

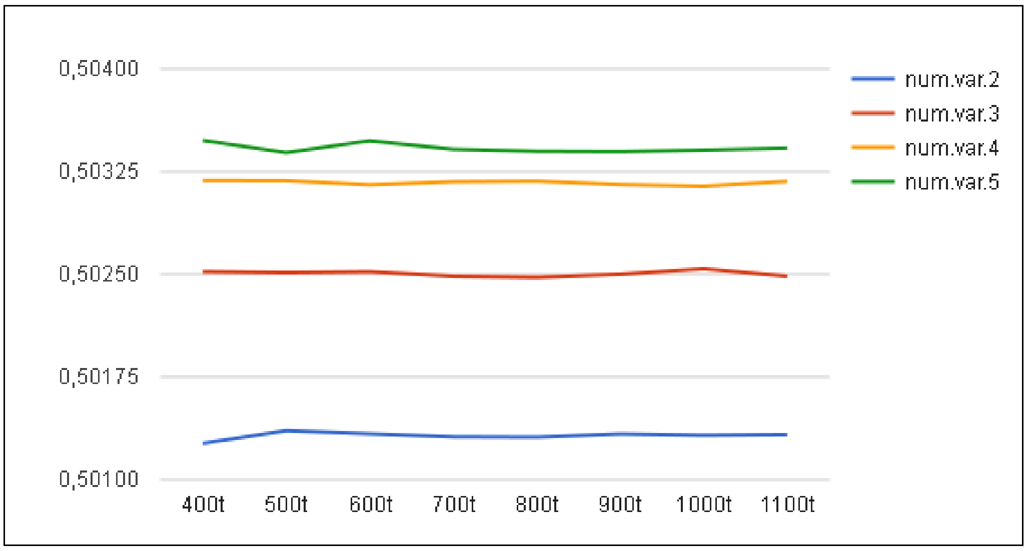


Figure 39. Graphic of RMSE in experiment #2

Best result: 0.501264829445384
# of trees: 400
# of variables: 2

Experiment #3

In the experiment #3 it is combined both the overlap of original words and the lemmas of the words (F1 – F10); the overlap of the **search_term** and attributes was calculated. The parameter **number of trees** takes the values 400, 500, 600, 700, 800, 900, 1000 and 1100. The parameter **number of variables** takes the values 2, 3, 4, 5, 6, 7, 8, 9, 10. The figure 40 shows the values of the error metric RMSE in the experiment #3.

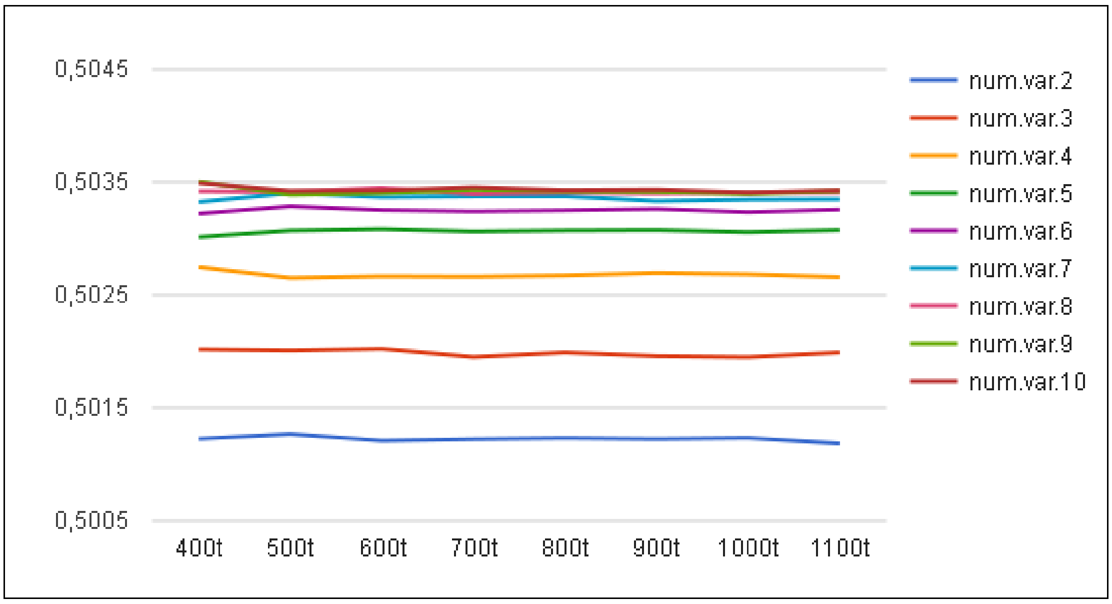


Figure 40. Graphic of RMSE in experiment #3

Best result: 0.501183172761222
# of trees: 1100
# of variables: 2

Experiment #4

The experiment # 4 uses the cosine similarity between the word embeddings of the datasets. The features are F11 – F15. The features represent the same the differences between the **search_term** field with the attributes of the product. The parameter **number of trees** takes the values 400, 500, 600, 700, 800, 900, 1000, and 1100. The parameter **number of variables** takes the values 2, 3, 4, 5. The figure 41 shows the values of the error metric RMSE in the experiment #4.

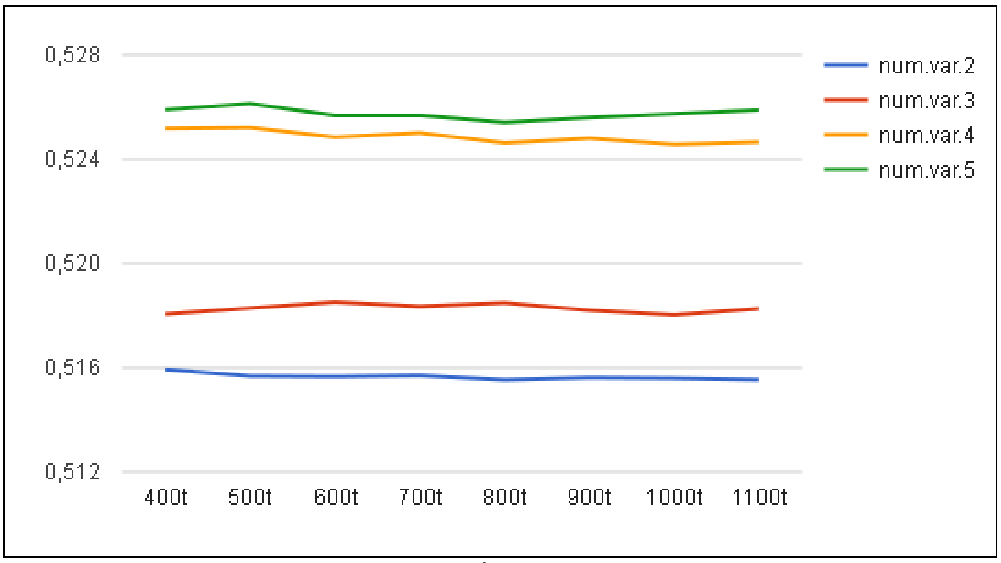


Figure 41. Graphic of RMSE in experiment #4

Best result: 0.515549996800693
# of trees: 800
# of variables: 2

Experiment #5

The experiment # 5 considers all the features (F1 – F15), it is the combination of overlap of words and lemmas of words, plus the cosine similarity of word embeddings of the datasets. All 3 categories of features represent the differences between the **search_term** field and the attributes of the products. The parameter **number of trees** takes the values 400, 500, 600, 700. The parameter **number of variables** takes the values 12, 13, 14, 15. The figure 42 shows the values of the error metric RMSE in the experiment #5.

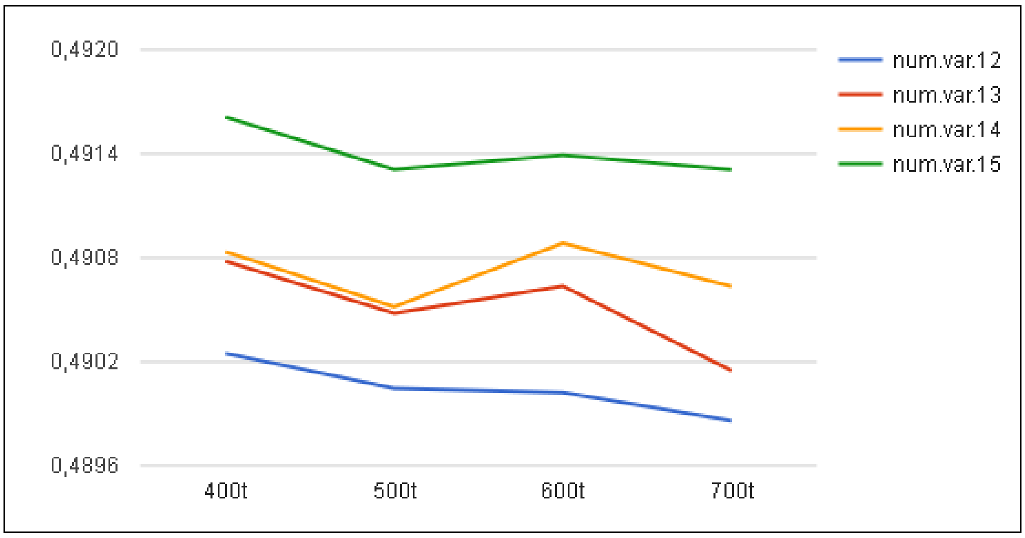


Figure 42. Graphic of RMSE in experiment #5

Best result: 0.489859605277241
# of trees: 700
# of variables: 12

The figure 43 shows the comparison of the results of the 5 experiments.

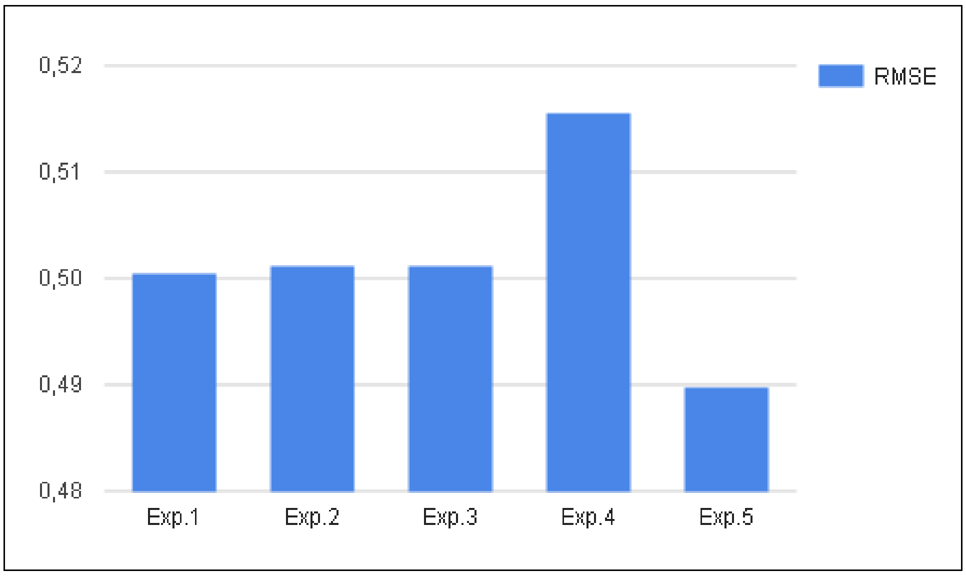


Figure 43. Comparison of the results of the 5 experiments.

Where

Exp.1 = Overlap of words
Exp.2 = Overlap of lemmas of words
Exp.3 = Overlap of words + Overlap of lemmas of words
Exp.4 = Cosine similarity of word embeddings
Exp.5 = Overlap of words + Overlap of lemmas of words + Cosine similarity of word embeddings

It can be seen that the use of NLP techniques like word lemmas (Exp.2) and the cosine similarity of word embeddings (Exp.4) gave a similar results (but not better) of prediction that using the overlap of words (Exp.1).

In the other hand, the best results were obtained by combining all the features: Overlap of words + Overlap of lemmas of words + Cosine similarity of word embeddings (Exp.5) which means that the word lemmas and the word embeddings helped to improve the predictive model.

Working with the test dataset

After finding the best hyper-parameters for the 5 experiments, those optimal values are used to repeat the experiments in the **test** subset; it will help to analyze the results in different data. The figure 44 shows the results for the **test** subset.

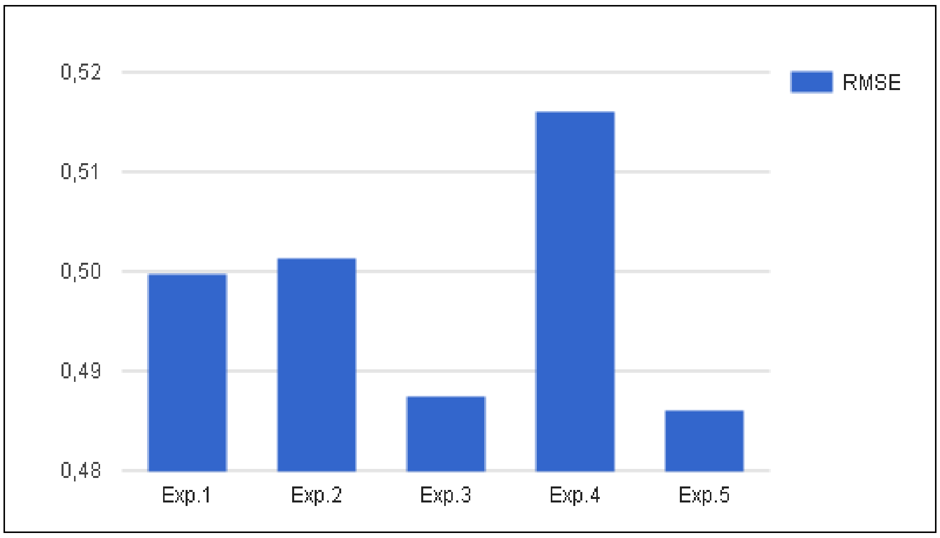


Figure 44. Comparison of the results of the optimal hyper-parameters for the 5 experiments in the **test** subset.

It can be seen that the word embeddings alone gave the worst result, but in combination with the overlap of words and the overlap of lemmas of words, they gave the best results improving the accuracy in 2.72%. The table 12 shows the optimal hyper-parameters of the predictive model.

Predictive Features:	F1 – F15
Number of decision trees:	900
Number of variables:	12

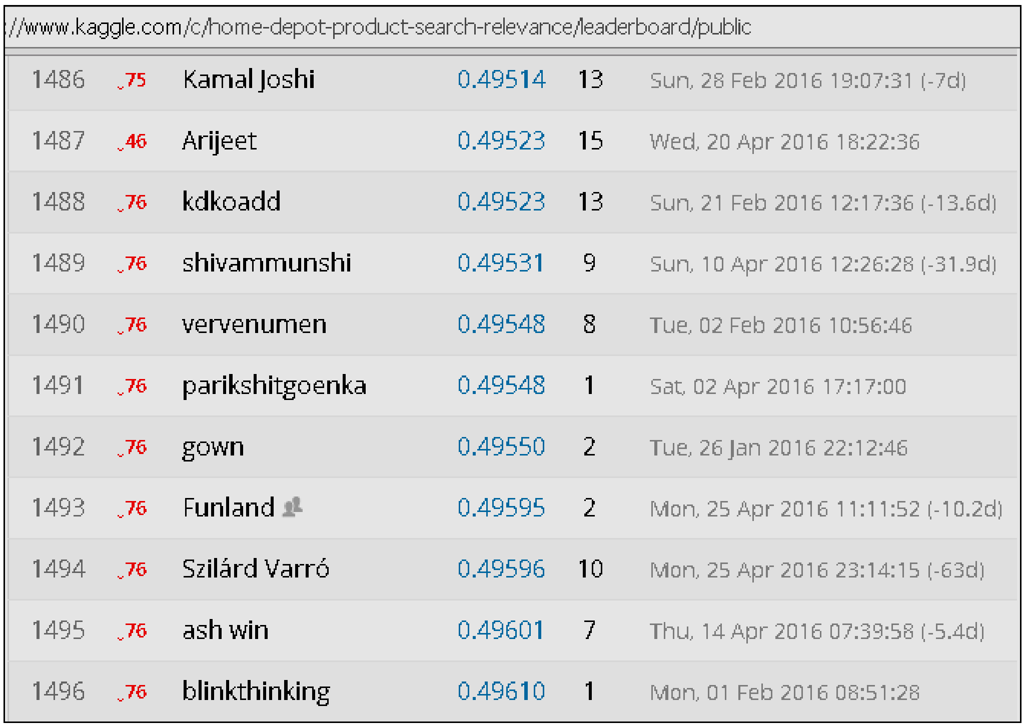
Table 12. Optimal hyper-parameters of the predictive model.

After finding the optimal hyper-parameters, the final step is to use the predictive model with the **test** dataset. The table 13 shows the results of the predictive model with **test** dataset.

Predictive Features:	F1 – F15
Number of decision trees:	900
Number of variables:	12
RMSE:	0.495369701916983

Table 13. Results of the predictive model with the **test** dataset.

According to the leader board of the Kaggle's competition the winner of the competition got a score of 0.43294; our predictive model with a score of 0.495369701916983 would have got the 1490th place out of 2125 teams. The figure 45 shows part of the leader board of the Home Depot Search Relevance Competition at the Kaggle web site.



://www.kaggle.com/c/home-depot-product-search-relevance/leaderboard/public

1486	↓75	Kamal Joshi	0.49514	13	Sun, 28 Feb 2016 19:07:31 (-7d)
1487	↓46	Arijeet	0.49523	15	Wed, 20 Apr 2016 18:22:36
1488	↓76	kdkoadd	0.49523	13	Sun, 21 Feb 2016 12:17:36 (-13.6d)
1489	↓76	shivammunshi	0.49531	9	Sun, 10 Apr 2016 12:26:28 (-31.9d)
1490	↓76	vervenumen	0.49548	8	Tue, 02 Feb 2016 10:56:46
1491	↓76	parikshitgoenka	0.49548	1	Sat, 02 Apr 2016 17:17:00
1492	↓76	gown	0.49550	2	Tue, 26 Jan 2016 22:12:46
1493	↓76	Funland	0.49595	2	Mon, 25 Apr 2016 11:11:52 (-10.2d)
1494	↓76	Szilárd Varró	0.49596	10	Mon, 25 Apr 2016 23:14:15 (-63d)
1495	↓76	ash win	0.49601	7	Thu, 14 Apr 2016 07:39:58 (-5.4d)
1496	↓76	blinkthinking	0.49610	1	Mon, 01 Feb 2016 08:51:28

Figure 45. Part of the leader board of the Home Depot Search Relevance Competition at the Kaggle web site.

The figure 46 shows the importance of the predictive features of the model applied to the **test** dataset.

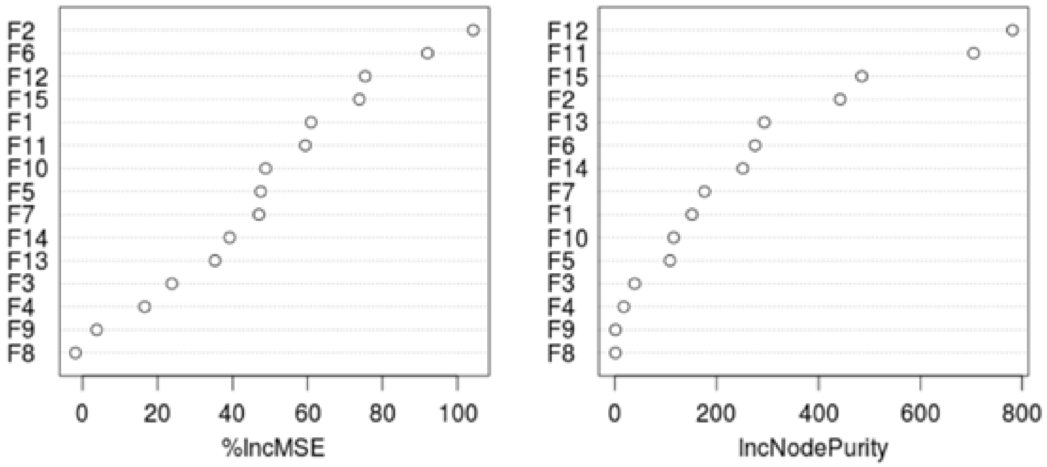


Figure 46. Comparison of the importance of the predictive features of the model applied to the **test** dataset.

Figure 47 shows the dendogram of the result of the Random Forest algorithm applied to the **test** dataset.

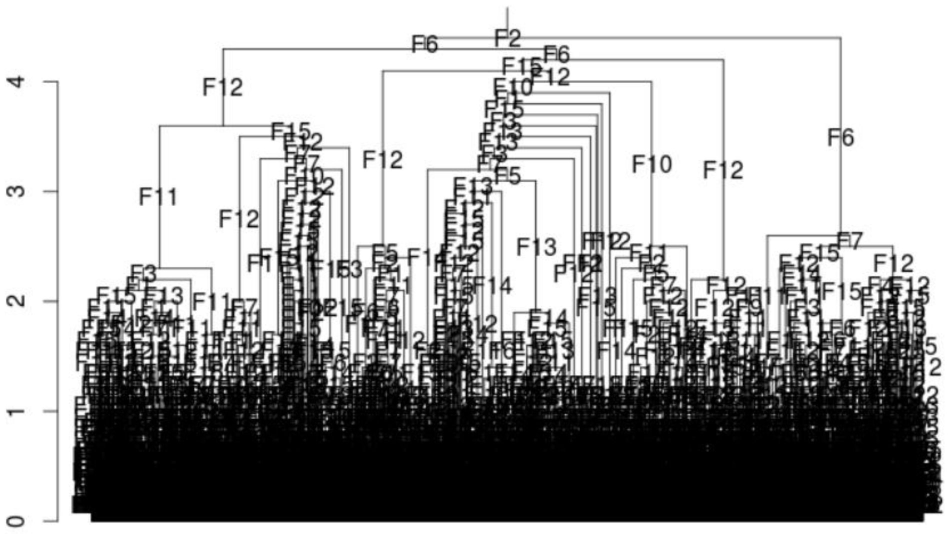


Figure 47. Dendogram of the Random Forest algorithm with 900 decision trees and 12 variables applied to the **test** dataset.

7. Conclusions

- The combination of NLP techniques like word lemmas and the distributed word embeddings contributed to improve the accuracy of the predictive model for the present project.

- Word2vec is a promising technology that can improve Natural Language applications like sentiment analysis, word prediction, translation, etc. It helps to describe syntactic and semantic relationships between words and phrases in a simple way, yet a lot of computing power is needed.

- The success of a project depends in the quality of data. Data pre-processing is a key part of building a predictive model, despite it takes effort and it is time consuming, it is a mandatory step. There are many actions that can be taken in this stage like case conversion, tokenization, lemmatization, selection of variables, word spell correction.

- Data Science is a trendy topic nowadays. IT professionals can benefit from data analysis, because of the vast amounts of data that is available in the Internet, the increase of computing power and data storage. Data Science also gives contributions to companies that solve their problems and take advantage of useful knowledge.

- Regression analysis is a powerful statistical tool for prediction; it can be used in many fields having accurate results.

- Natural Language Processing is an important field of Artificial Intelligence which allows humans and computers to interact in easier ways, its development will bring outstanding applications in the next years.

- Augmented Intelligence is a new term which says that machines will give humans more intelligence and will empower them to evolve and improve their quality of life. The original term Artificial Intelligence was thought to give machines the ability to simulate human intelligence, but actually the Artificial Intelligence of machines has surpassed human intelligence in many fields.

- Among other applications, R language is a useful tool for Data Science, packages like "tm" and "NLP" allow us to do complex tasks like tokenizing, removing stop words and punctuation, stemming, etc. Likewise R's Machine Learning packages help us building a predictive model.

- Kaggle.com is an outstanding platform for Data Science practitioners; its competitions promote the use of Data Science, solving real-world problems. It's also a good place for learning Data Science, networking and promoting Data Science careers.

8. References

[1] Kaggle. Your Home for Data Science.
www.kaggle.com

[2] Home Depot. Products and Services
www.homedepot.com

[3] Prithvi Raj, Vineesha, Varun, Swaroop: Data Mining Process
http://www.slideshare.net/prithviness/data-mining-process

[4] BioMed SpaceStat. 2014
https://www.biomedware.com/files/documentation/spacestat/Statistics/Multivariate_Modeling/Regression/About_Aspatial_Linear_Regression.htm

[5] Google Books Ngram Viewer
https://books.google.com/ngrams

[6] Adrian Colyer: The amazing power of word vectors, the morning paper blog, April 2016.
https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

[7] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: Efficient Estimation of Word Representations in Vector Space, arXiv:1301.3781v3, 2013

[8] Christian S. Perone: Terra Incognita blog.
http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

[9] Machine Learning Mastery Blog
http://machinelearningmastery.com/start-here/

[10] Will Y. Zou, Richard Socher, Daniel Cer, Christopher D. Manning: Bilingual Word Embeddings for Phrase-Based Machine Translation, Department of Electrical Engineering and Computer Science Department Stanford University, Stanford, CA 94305, USA

[11] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, Ting Liu: Learning Semantic Hierarchies via Word Embeddings, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1199–1209, Baltimore, Maryland, USA, June 23-25 2014.

[12] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou', Steven Skiena: The Expressive Power of Word Embeddings, arxiv.org/abs/1301.3226v4, 2013.

[13] Joseph Turian, Lev Ratinov, Yoshua Bengio: Word representations: A simple and general method for semi-supervised learning, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 384–394, Uppsala, Sweden, 11-16 July 2010.

[14] Rami Al-Rfou, Bryan Perozzi, Steven Skiena: Polyglot: Distributed Word Representations for Multilingual NLP, arxiv.org/abs/1307.1662v2, 2014

[15] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, Tie-Yan Liu: A Probabilistic Model for Learning Multi-Prototype Word Embeddings, Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 151–160, Dublin, Ireland, August 23-29 2014.

[16] Samy Bengio, Georg Heigold: Word Embeddings for Speech Recognition, Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech (2014)

[17] Angeliki Lazaridou, Nghia The Pham, Marco Baroni: Combining Language and Vision with a Multimodal Skip-gram Model, arXiv:1501.02598v3 [cs.CL] 12 Mar 2015.

[18] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, Yorick Wilks: A Closer Look at Skip-gram Modelling, NLP Research Group, Department of Computer Science, University of Sheffield

[19] KDNuggets. Data Mining, Analytics, Big Data and Data Science
www.kdnuggets.com

[20] Snowball Stop words
http://snowball.tartarus.org/algorithms/english/stop.txt

[21] Ixa-pipe tools for tokenizing
https://github.com/ixa-ehu/ixa-pipe-tok

[22] Mikolov word embeddings
https://docs.google.com/open?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM

[23] Metric used for evaluating the predictive model. Root Mean Squared Error.
https://www.kaggle.com/wiki/RootMeanSquaredError