NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Classification problem in real estate corpora

Furniture detection in real estate listings

Alexandra Ordina

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

FURNITURE DETECTION IN REAL ESTATE LISTINGS

by

Alexandra Ordina

Internship report presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

Co Advisor: Nuno Miguel da Conceiçao Antonio

Co Advisor*:* Ricardo Costa Dias Rei

September 2021

ABSTRACT

The real estate market has been a conservative industry, but it has recently seen an increased number of data-driven solutions and software products aimed at real estate professionals. Casafari is at the forefront of this development and is rapidly building a portfolio of data services. All of these services rely on clean data extracted from multiple websites. The current project contributed to several of the company's products by developing a classifier that detects if a mention of furniture is present in the online property listings. As a result, a new filter was introduced on the company's metasearch page, and a new furniture attribute was made available for an improved valuation of the property. In addition to the improvements in the website navigation and property valuation accuracy, the current project contributed to the improvement of theory. Several state-of-the-art NLP models were tested and evaluated against the developed classifier, with some showing very competitive results without any training and reliance on pre-labeled data. However, for these models to be considered for production, there needs to be a significant improvement in speed and GPU requirements.

KEYWORDS

NLP; Text Classification; Zero-shot learning; Keywords; Supervised Learning; Real Estate

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

SaaS	Software-as-a-Service
NLP	Natural Language Processing
B2B	Business-to-Business
ML	Machine Learning
PropTech	Property Technology (Real Estate Start-ups)
CRISP-DM	Cross-Industry Standard Process for Data Mining. Process model with six phases that naturally describes the lifecycle of a data science project.
BOW	Bag-of-words. Representation of text that describes the occurrence of words within a document.
SVM	Support Vector Machines. Set of supervised learning methods used for classification, regression, and outliers' detection.
HMM	Hidden Markov Models. Statistical models used in language modeling, specifically for part-of-speech tagging.
GloVe	Global Vectors. Model for distributed word representations
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer
RoBERTa	Robustly Optimized BERT Pretraining Approach
ELMo	Embeddings from Language Models. Deep contextualized word representation technique.

1. INTRODUCTION

1.1. COMPANY PRESENTATION

Casafari is a SaaS company providing property sourcing for real estate agents and brokers, banks, private equity funds, developers, consultancies, and direct buyers. Its business model is to build data services around property sourcing for the B2B market in Europe. It is currently servicing several European markets with the most substantial presence in the Iberian Peninsula: the first market to open was Portugal, followed by Spain, Italy, and France.

The company's main product is a meta-search engine that crawls daily publicly available real estate listings from agencies and private sellers, providing "the cleanest and most complete real estate database"[1], eliminating duplicates for properties listed in multiple sources. The auxiliary products are based on data analysis of this aggregated historical and close-to-real-time data. Casafari offers property tracking with notifications to clients whenever a change in price, status, and other relevant information for selected properties occurs. It also offers market analysis, property valuations, property portfolio analysis, competitive analysis, benchmark reports, data support for press reports, and other services.

For Casafari, each real estate listing is a data point. By utilizing Natural Language Processing (NLP) and Machine Learning (ML), the company can identify each property's location name, type, and relevant features such as the type of floor, orientation, view, available parking places, storage area, swimming pool, or garden.

1.2. PROJECT OBJECTIVES AND REQUIREMENTS

1.2.1. Business requirements

The internship was within Casafari's data science team and focused on the project dealing with text classification. The company wanted to develop a model detecting whether a listed property is *furnished* or *unfurnished*. The model was required to support multiple languages: English, Spanish, Portuguese, French and Italian, and be able to incorporate additional language requirements in the future, when the company starts operating in other European markets.

The raw data parsed from real estate websites is not uniform: websites have different structures, filters, level of details, and the listings are published in multiple languages. Sometimes the source listings do not have an explicit filter for furniture, and only the free-text description has an indication

[1] from https://www.casafari.com/

of whether the property is furnished or unfurnished. Aggregating data from multiple sources in one search platform requires building a uniform structure, and automatic assignment of labels.

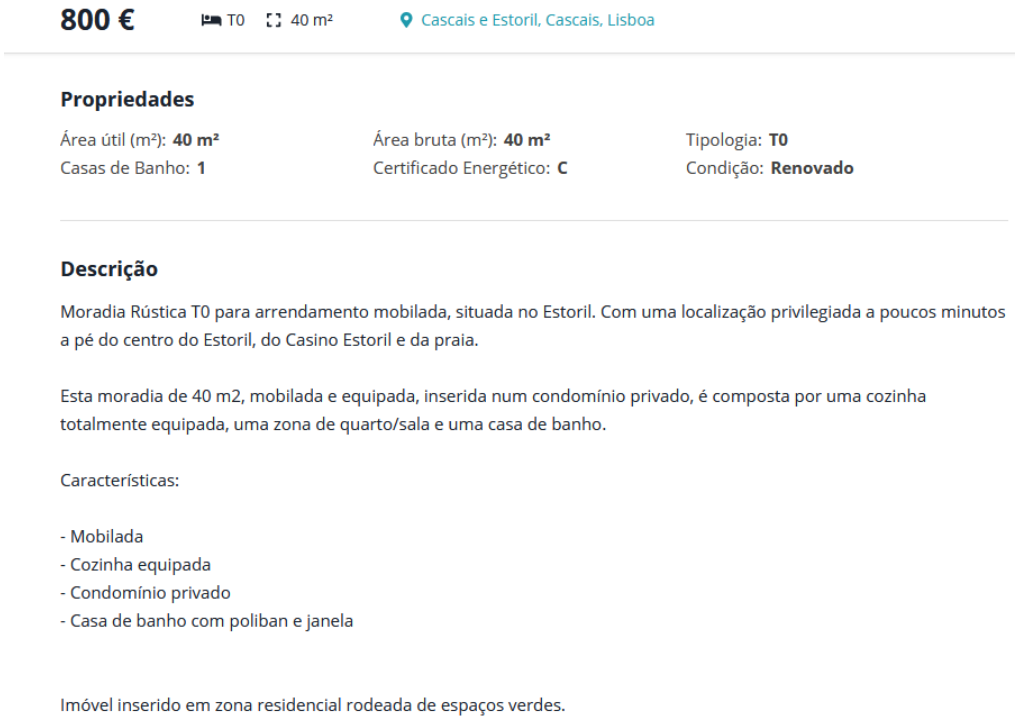

Figure 1. Example of title, features and description of a listing. Source: imovirtual.com

Aggregating the data from multiple realtors and classifieds, the company needs an algorithm that automatically considers all available information and detects the feature.

The research approach was to align the model development with the existing development for other features (elevator, garage, floor, orientation, etc.). In addition, it was also decided to research alternative models using more recent findings in data science and specifically state-of-the-art models in the NLP field and explore their application in production. This research went beyond the scope of the course materials and covered deep learning natural language models for text classification working with small-sized training data.

Upon discussion with data and business teams, the following business requirements were identified for the project:

- Interpretability of result. The filter for furniture needs to be deployed in production and to be visible to clients on the front-end application. Inside the application, the clients can flag the errors in listing's labels, and the customer success team makes sure the errors are corrected. For the business process to remain unchanged, the way the label is assigned should be interpretable to non-data experts.

- Consistency of result interpretation.
- Ease of maintenance. Correcting mislabeled data points is an integral part of the current business process, so the developed model should allow quick and easy fixes. In addition to that, modularity of the code is essential. Adding more labels (for example, the air conditioner is planned for future releases) should not require retraining the existing model for current labels. Also, adding another language should not require extensive retraining for current labels as well.
- Fast execution time. The model should be as fast as possible and aim at the shortest prediction time per listing as classification is performed close to real-time. The listings are added to the database hourly.
- Ability to scale. Ideally, the model should achieve almost infinite scalability, as the company aims to have exponential growth in the number of listings and websites parsed over time.
- Avoidance of false positives. Attributing features to property that does not have them should be reduced to zero if possible or kept at a minimum. The focus should be on the correct detection of furniture when the property is furnished. False positives will have a negative impact on the business and credibility of the model by clients. Realtors use the Casafari database to select properties for prospective buyers. The cost of false positives would mean showing false information and offering a property that does not meet the buyer's requirements, which should be avoided. Also, usually, the rental price increases if property is furnished, and a more conservative approach was considered suitable for the model. Failing to detect furniture for some properties would mean that these properties would not be displayed to the broker if the furniture filter is on. However, a more general search with other valuable characteristics such as the number of bedrooms and bathrooms, size of the property, or location can be used to display those records. For the same reason, cases where furniture is optional or partial furnishings are provided needed to produce an "unfurnished" label.

The project's final deliverable was a classifier model that considers the business objectives and requirements plus the summary of research on the state-of-the-art models and their suitability for the production deployment.

This project was built upon the following resources:

- Labeling resources from customer success team;
- Access to Casafari database with 37.000.000 listings from over 11.000 websites;

- Access to previous research and repository with the current classifier trained for different labels (elevator, garage)

One of the project constraints was the low availability of labeled data and the need to request manual labeling from the customer success team to start the project. As manual labeling has a business cost associated with it, it was agreed to request a maximum of 200 - 250 samples per each of the five languages. Another constraint was the hardware limitation, as there was no large GPU for running the model in the production environment.

The target duration of the project was set to five months and one month for research. The first month was allocated to data exploration and business understanding, with regular tasks of data collection, data processing, and data quality analysis: setting up JSON configurations for the crawler to parse accurate data from the websites, developing python scripts for quality checks and evaluation the quality health of specific locations (for example, tracking possible errors in identified listing location), data sources and targeted data fields (for example, property types and property features). Several meetings with business process owners were scheduled to review the requirements for the project and walk through the data lifecycle to fit the model in the current data flow. Stakeholders' actions upon completion of the project were also agreed. The Quality Assurance team needed to include the new automated field in the quality checks. The Customer support and Sales teams needed to update the documentation and communicate to the customers the release of the new filter. The Product team needed to include the new feature in the scheduled release notes.

The current project objective is similar to the objective of a previously developed classifier (already deployed in production) to predict whether the property has a garage and elevator. Due to this similarity, research on the other model was done to inherit the good practices, filter out the models which were already tested and found unsuitable, and incorporate the lessons learned from the previous implementation.

The research showed that previous development focused on implementing a Neural Network to distinguish between negative and positive labels for elevator and garage. Bad performance with a small training set and lack of labeled data on a sufficient scale made it unsuitable for deploying in production. A more straightforward method was adopted based on a rule-based algorithm. A neural network was deemed unsuitable from a business perspective as the company preferred not to use the model working as a "black-box," which could not provide the required transparency to the users.

The following model constraint was identified upon research into previous implementations and initial data analysis. The model will have to deal consistently with the ambiguity of the information in the

published listing, sometimes open to interpretation even for human annotators. The text fields can be conflicting between themselves, or text fields can be conveying information opposite to what is conveyed in the listing photos. Below two examples illustrate this issue. Description indicates that the property is fully furnished. At the same time, the features mention the property is rented empty in the listing in Figure 2, and in Figure 3, the description indicates the house is rented without furniture. However, there is furniture on the listing photos.

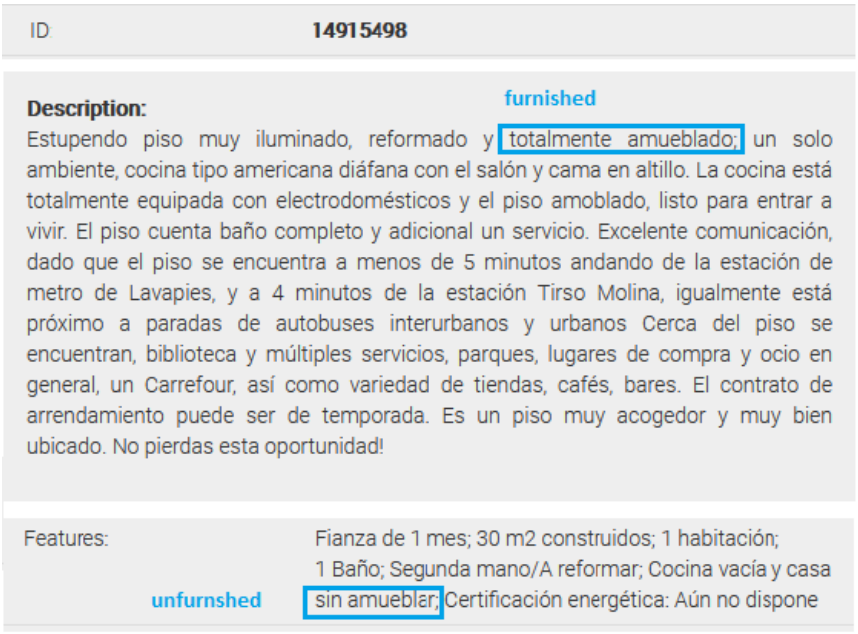

Figure 2. Example of ambiguity in a listing. Source: Casafari app

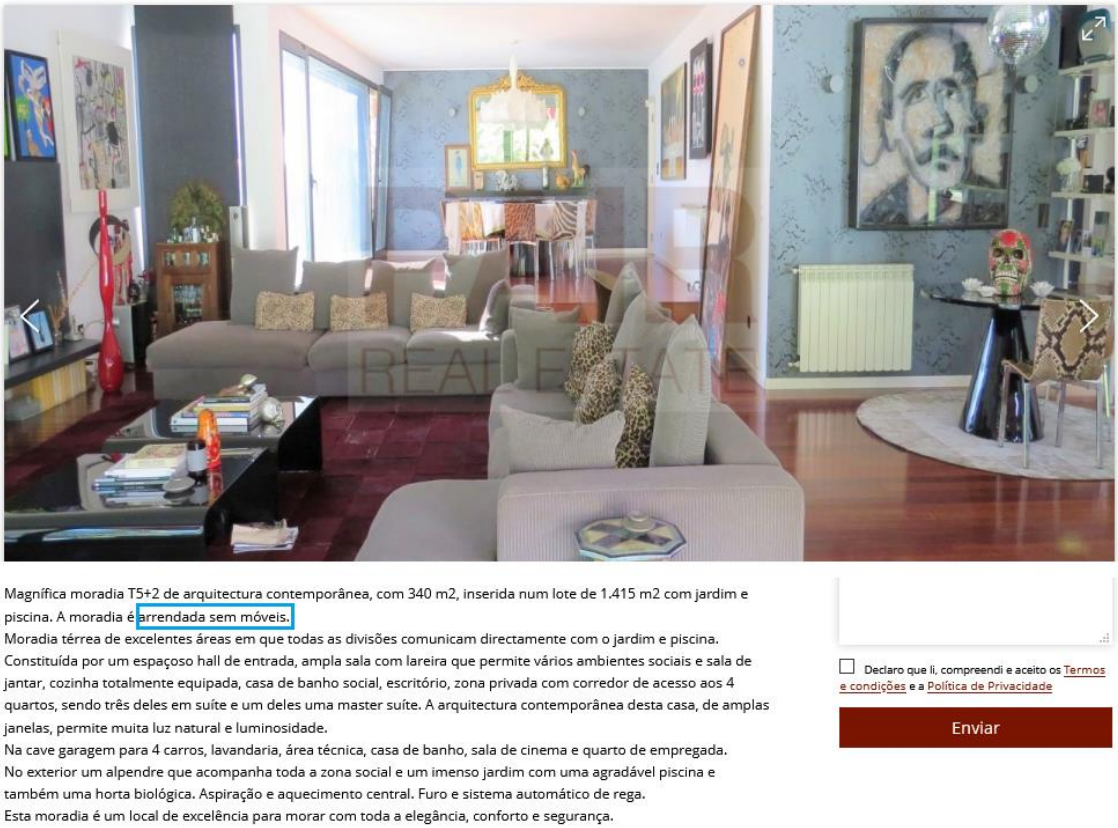

Figure 3. Example of ambiguity in listing. Source: imovirtual.pt

Upon consulting with the project stakeholders on the business side, it was decided to select textual information as the primary source of truth should the ambiguity occur and ignore the visual content. As for the ambiguity between textual fields, the agreement was always to output negative labels if there is conflicting information. However, it was agreed that these decisions could be reviewed after customer feedback, and the model script should be easy to adjust without investing many resources and time.

Another constraint is the model output depends on the quality of the parsed text fields in the database. It can perform worse for the listings with low-quality text fields, such as incomplete features or missing descriptions.

To measure the project's business success, the expectation was set to predict if the property is furnished or unfurnished with at least 80% accuracy, desired to go as high as 85-90%, with consistent predictions for listings where the presence of furniture is arbitrary. The prediction speed needed to be reasonably fast and not exceed by much the current processing speed for the deployed classifier for the elevator and garage. A slight increase in processing time is acceptable and would be reviewed against the performance of current classifiers by a senior data scientist.

As for the data mining goals and criteria, the accuracy score for the selected listings published in the year 2021 and labeled by the customer success team will be used as the main measure and the average speed for predicting one listing, however other performance metrics detailed in section 4 will also be used to control the model's response to ambiguity and make sure the model is fit to be used with new unseen bookings.

2. LITERATURE REVIEW

2.1. DATA PRODUCTS IN THE REAL ESTATE MARKET

According to the Real Estate Innovations Overview report published by KPMG in 2020 (KPMG Real Estate Advisory, 2020) 2600 mln USD of investment capital was poured into PropTech companies in 2019, which shows there is a huge market in data in real estate and it continues to grow. According to this annual publication, the investment in digital, IT, and PropTech collaboration is driven by a need for: Improved efficiencies, Cost-reduction, and Enhanced decision making.

One of the key aspects of data-driven decision-making is extracting knowledge from enormous data from different sources. Data is often unstructured, and there is a lot of duplication when the same property is published for sale or rent on different websites. Providing a "real estate experience free of clutter and redundancy" (Dima Williams, 2020) is one of the things that drives real estate market players to success and lets them lead the change in the market.

Real estate has always been a conservative industry. Still, more and more companies develop products powered by data mining, machine learning, natural language processing in addition to their more traditional selection. Companies created as online marketplaces, such as Idealista from Spain (https://www.idealista.pt/en/), Rightmove from the UK (https://www.rightmove.co.uk), and Zillow from the USA (https://www.zillow.com/), are developing auxiliary services in market analytics, property valuations, surveying to name a few.

More than 595 PropTech companies whose business and products are innovating the real estate industry were cited in the KPMG 2020 report, and their number is growing. Among companies who are fulfilling the need of real estate professionals for data insights and data-driven tools, an American company, Co-Star, has been named by Real Estate Technology Trends paper (KPMG, 2019) as the "most widely used across all real estate organizations" for market research and valuation tools.

Companies such as Co-Star develop models that consider multiple variables to estimate the value of the property. One of the fastest-growing trends in the housing industry is the high demand for rental units, knows as "rent generation," and the shift in preference from owning to renting real estate(Deloitte, 2020). This shift means that there is a lot of demand for data-driven "buy-to-rent" investments and estimating the rental yield. Models providing such an estimation use many variables: type of the property, its size, location, and also the provided furnishings.

"Renters can increase the price more significantly for short-term rentals because the demand for a furnished unit is greater than for renters looking for a long-term lease. Even for longer-term rentals,

the price increase averages at least 15% to 20%".
(https://www.zumper.com/manage/resources/furnished-vs-unfurnished-rental/).

Every algorithm requires clean data to perform well. Still, when using information automatically extracted from different sources, this cannot be assured, so there is the need to create algorithms that ensure that this data has the correct values.

The current project addressed this need by developing a model that detects the property's furnishings based on the textual information about the property published online.

2.2. TEXT MINING

2.2.1. Text mining applications

According to different estimates, more than 80 percent of data is typically composed of unstructured or semi-structured data (Hotho et al., 2005; Talib et al., 2016). Textual data is everywhere in our personal and professional lives: our e-mails hold our personal and professional interactions, we read the digital press and other online publications, post online reviews for products we bought or movies we have watched, we permit to record customer service calls which are transcribed, we share content and connect with others via social media posts, we create client records in CRM tools, make posts on forums and so on.

These enormous amounts of textual data can be a source of important insights and help with business decisions. Manual processing and analyzing text data would be a slow, expensive and inefficient process. Text mining makes this process faster and more efficient. According to a widespread definition, text mining refers to the process of extracting interesting and non-trivial information and patterns from unstructured or semi-structured textual data (V. Gupta & Lehal, 2009; Hotho et al., 2005; Talib et al., 2016). Text mining "uses techniques from information retrieval, information extraction as well as natural language processing (NLP) and connects them with the algorithms and methods of KDD (Knowledge Discovery in Databases), data mining, machine learning and statistics" (Hotho et al., 2005)

Many industries have recognized the value of text mining and started to use it to achieve various business objectives. Text mining is used to process customer complaints, call center transcripts, and service tickets, contributing to a better customer experience, customer relationship management, and customer service quality. Insurance companies are combining text and data mining in fraud-prevention models. News agencies use text mining to automatically tag news reports with categories, names of important persons, and places. Marketing professionals use text mining tools in social media analysis. Text mining is at the core of machine translations and search engines. Text mining is used in sentiment

analysis, which is a technique that allows identifying if the sentiment of a client towards the product or service is positive, negative, or neutral by processing customer reviews, social media posts, and comments.

The most popular text mining techniques can be divided into the following categories: information extraction, information retrieval, categorization, clustering, and summarization. Some experts distinguish text visualization as a separate category.

Information extraction "identifies key phrases and relationships within text"(V. Gupta & Lehal, 2009) Text mining can help extract entities, attributes, and relationships, and typically the relevancy is checked using precision and recall metrics.

Information Retrieval (IR) refers to "a process of extracting relevant and associated patterns according to a given set of words or phrases"(Talib et al., 2016). An information Retrieval system is typically based on a set of algorithms which goal is to help find the most relevant documents based on user requirements. Most common representation of information retrieval is a search engine like Google or Yahoo. Still, it is also used in product search in e-shop or online library catalogs, and in general, wherever there is a search bar, an IR system is involved. The IR models typically rely on two concepts - bag of words (BOW) representation of the text document and TF-IDF method to measure word and text document relevance. TF-IDF is a combination of Term Frequency (TF) which finds out the occurrence of the query words in documents, and Inverse Document Frequency (IDF) estimates query words relevance.

Categorization or classification refers to "identifying the main themes of a document by placing the document into a pre-defined set of topics"(V. Gupta & Lehal, 2009). It is used in spam filtering or personalized commercials. Text classification can be binary, multi-class, or structured depending on the number and nature of categories.

The most basic classification method is a manual rule designed to reflect the domain knowledge. For example, the rule can be "if text contains word "x" THEN it belongs to category "Y". The significant advantages of the manual rule-based approach are its interpretability and the fact that it works well when the category is well-defined and can be easily distinguished based on surface features like special vocabulary. However, it does not handle uncertainty well, and the rules are dependent on expert knowledge, can be inconsistent, and need manual input and maintenance.

Automatic classification can be done by many models like Naive Bayes, Logistic Regression, Neural Networks, KNN, and SVMs. Automatic classification relies on the availability of the training data for the model to learn from, and the most popular models can be divided into generative probabilistic models and discriminative approaches.

Clustering is a popular unsupervised technique in both data and text mining that groups similar objects by discovering natural structures in the data. Similar objects can be documents, terms, phrases, sentences, or web pages on text mining. The key problem here is how to define and measure the "similarity" of text documents. "A basic clustering algorithm creates a vector of topics for each document and measures the weights of how well the document fits into each cluster" (V. Gupta & Lehal, 2009).

Summarization refers to the process of automatically generating a compressed version of a specific text that holds valuable information for the end-user. Text summarisation integrates and combines various methods for text categorization like decision trees, neural networks, regression models, and swarm intelligence.

2.3. TEXT CLASSIFICATION

2.3.1. Rule-based text classification

One of the simplest and earliest methods in text classification is rule-based, meaning there is a rule or set of rules designed by a human with domain knowledge to assign the text into a specific category (*How Does Text Classification Work? | Unite.AI*, n.d.)

For example, an algorithm that uses a "if...then" logic to assign text to a topic, news, politics, or finance would be a rule-based model. Compared to more advanced classification methods rule-based approach provides interpretable results, and this is its main advantage. It does not require large amounts of training data as it primarily relies on the knowledge of an expert who designs the rules. However, the approach also has many limitations: it requires domain knowledge and resources to design and maintain the rules. It is impossible to define an all-encompassing set of rules.

2.3.2. Dictionary-based text classification

A dictionary-based approach relies on a prepared list of words and classifies text by matching text samples with the dictionary. This dictionary can be either prepared manually by a domain expert or produced using automatic algorithms to extract keywords from domain-specific corpora, such as TF-IDF, mentioned earlier.

This approach also produces very interpretable results and is simple to maintain: the dictionary can be improved by adding more keywords. Keywords matching is not a suitable method for highly complex texts with rich vocabulary, as it would require considerable resources to build and maintain the dictionary. However, it can give good results for domain-specific texts where it is common to use standard terminology, including but not limited to real estate.

This approach does not handle the ambiguity of human language well as it does not capture the context and semantics of words.

2.3.3. Deep learning methods in text classification

While rule-based and dictionary-based classification relies on domain knowledge, machine learning models are able to classify text based on text data observation. This ability typically means that a large training corpus is required for the model to learn from and then be able to perform well with unseen text.

Popular choices of classification algorithms include Naïve Bayes, support vector machines (SVM), hidden Markov model (HMM), gradient boosting trees, and random forests (Minaee, Kalchbrenner, et al., 2020). These models are typically trained on very high dimensional and sparse features, and feature engineering step is crucial to the classifier's success. In comparison with the classical machine learning methods, neural networks use dense representations of words in a low-dimensional vector space called word embeddings. Word embeddings are superior to keywords because they can capture semantic relationships of words in the text and recognize the context. The most popular word embeddings are word2vec, introduced by Mikolov in his paper in 2013 (Mikolov et al., 2013) and GloVe, developed by Pennington in 2014 (Pennington et al., 2014).

In recent decades there was a lot of research and development using deep learning techniques in text classification. Deep learning techniques were inspired by the way the human brain works. Deep learning has the potential to reach high accuracy levels with minimal engineered features (*A Comprehensive Guide To Learning Text Classification*, n.d.).

Recently, larger embedding models have been developed such as BERT (Devlin et al., 2018), trained on 3.3 billion words and GPT, developed by OpenAI (Radford & Narasimhan, 2018) which are based on a novel architecture called Transformer Neural Network (Vaswani et al., 2017). This architecture was introduced by Google researchers in 2017 and described in the paper “Attention is All You Need” (Vaswani et al., 2017). A transformer is an architecture for transforming one sequence into another one with the help of two parts (Encoder and Decoder). Transformers apply the so-called self-attention mechanism to compute in parallel for every word in a sentence or document an “attention score” to

model each word's influence on another. This process can be compared to how human interpreters would work to translate a sequence of sentences from one language to another: by taking note of “key terms” that are important to the translation and give it context (*What Is a Transformer?. An Introduction to Transformers And… | by Maxime | Inside Machine Learning | Medium*, n.d.).

BERT (Bidirectional Encoder Representations from Transformers) is one of the most widely known and popular examples of Transformers architecture. One of its followers is RoBERTa, developed by Facebook AI team (Liu et al., 2019). RoBERTa is a retraining of BERT on a much larger training set of 160 GB with an improved training methodology.

Since 2018 there has been developed several large-scale Transformer-based pre-trained language models. Transformer-based pre-trained language models use much deeper network architectures. They are pre-trained on much larger amounts of text corpora to learn contextual text representations by predicting words conditioned on their context (Minaee, Kalchbrenner, et al., 2020). Such models have been made possible by the significant computational power and access to massive text corpora available to tech giants such as Google and Facebook. Using pre-trained models allows to avoid hours of training and massive amounts of memory for the training data and unlock the state-of-the-art NLP capabilities.

These pre-trained language models show outstanding results and became state-of-the-art for many NLP tasks, including text classification. However, their main problem is they need a lot of GPU, making it very difficult and costly to implement in production. According to authors of DistilBert, “the growing computational and memory requirements of these models may hamper wide adoption”(Sanh et al., 2019). There have been done some developments in reducing the size of these gigantic models. The most common tools include distillation (training a smaller “student” model based on the larger “teacher” model), quantization (approximating the weights of a network with a smaller precision), and weights pruning (removing some connections in the network).

Knowledge distillation idea has been introduced by Rich Caruana from Cornell University (Bucila et al., 2006) and developed further by Hinton and co-authors from Google team in 2015 (Hinton et al., 2015), and it means that the knowledge acquired by a large model, or an ensemble of models can be transferred to a single small model without significant drop in prediction results. There are multiple methods of knowledge distillation, which differ in the main three components: knowledge types, distillation strategies and the teacher-student architectures, which have been summarized in a recent survey paper published in 2021 in International Journal of Computer Vision by a team of authors from UBTECH Sydney AI Centre, School of Computer Science and Birkbeck College (Gou et al., 2021).

Model quantization "approximates floating-point numbers with lower bit width numbers, dramatically reducing memory footprint and accelerating performance" (*Faster and Smaller Quantized NLP with Hugging Face and ONNX Runtime | by Yufeng Li | Microsoft Azure | Medium*, n.d.). As an example, reducing the precision format from the usual float32 (32-bit stored double values) to a lower form like float16, int8 and so on.

Finally, pruning is one of the oldest model compression methods. Typically, pruning can be divided in structured and unstructured pruning. When it comes to unstructured pruning, individual weight connections are removed from the network by setting them to zero. Alternatively, in structured pruning there is a pruning strategy, for example removing weights with low magnitude (M. Gupta & Agrawal, 2020). Several implementations also differ in the time at which the pruning is done: after the large network is trained, or before the training. Survey paper, published this year by M. Gupta and P. Agrawal from Microsoft (M. Gupta & Agrawal, 2020), outlines the main pruning techniques and describes their differences.

2.3.4. Text Classification Pipeline

A text categorization model can be divided into six stages (Mirończuk & Protasiewicz, 2018): data acquisition, data analysis and labeling, feature construction and weighting, feature selection and projection, training of classification model, and solution evaluation.

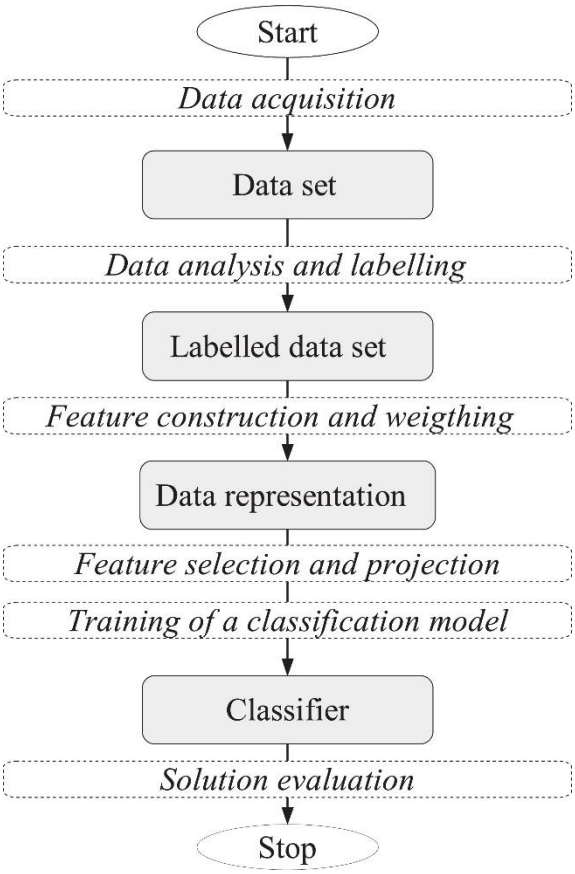

Figure 4. Typical text classification steps. Source:(Mirończuk & Protasiewicz, 2018)

3. METHODOLOGY

This chapter describes the methodology followed for building the model. We will briefly explain the project framework that was adhered to for this work, present the software used, and discuss the overall project pipeline.

3.1. PROJECT FRAMEWORK

3.1.1. CRISP-DM

The project follows Cross Industry Standard Process for Data Mining (CRISP-DM) methodology, and the project goal was aligned with business requirements. The typical cycle of a data science project consists of 6 phases, as in the below diagram (Figure 5):

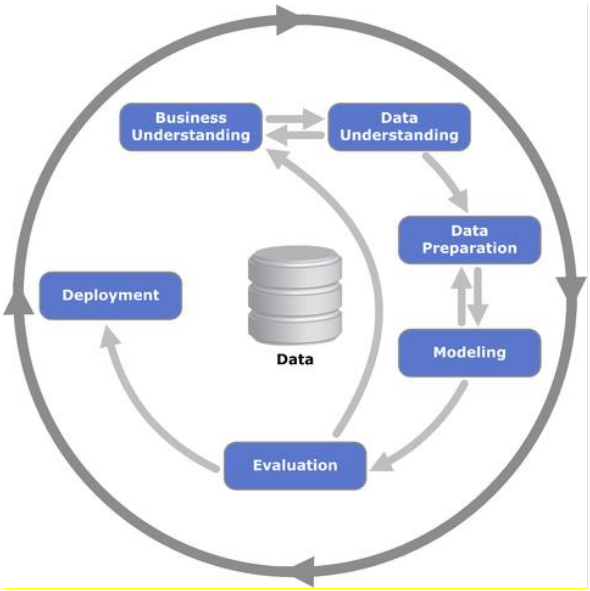

Figure 5. The life cycle of a data mining project. Source: CRISP-DM guide.

CRISP-DM approach can be adapted to any industry. It brings a universal structure to data science projects, making it easier to follow and build upon for data professionals. By following this methodology, we ensured that the project is properly documented for future use and its adoption in the data team. Another advantage of using CRISP-DM is aligning the technical aspects of the data science project with the business goals and considerations.

3.1.2. Agile

This project also followed the Agile project methodology mindset, which is a company standard. Agile is an iterative approach to project management and software development (*What Is Agile? | Atlassian*, n.d.) as outlined in Figure 6 below. Casafari adopted Jira software to manage its data science, business analytics, and software development projects. By documenting the project in Jira, we ensured that the stakeholders have access to project documentation and can track the progress of the tasks as they are being completed. It also helped to engage a wider audience and spread knowledge about the project across non-technical teams. After the model was accepted for deployment, Jira was also used to coordinate with the other developers, run the model for all the listings in the database retrospectively, and assign the front-end team's tasks, so the new functionality is included in the next software release.

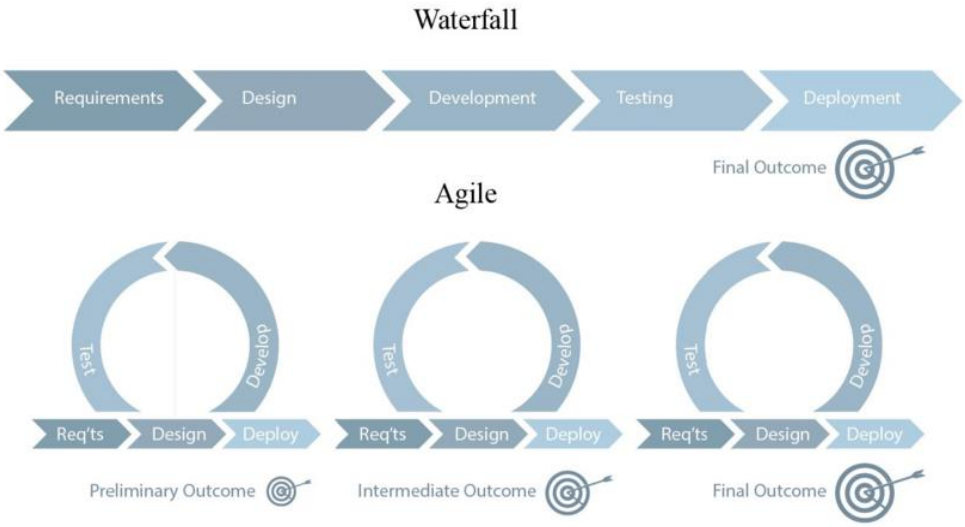

Figure 6. Agile process

3.2. PIPELINE

We can distinguish between two separate pipelines: one to train the model and the second to produce real-life predictions for the web application. Training pipeline (Figure 8) will be used to do initial training of the classifier, and any subsequent retraining, such as when a new language needs to be added or to debug the model during the quality control and based on feedback from customer-facing teams. Once the model is deployed in production, it will be integrated into the production pipeline as described in Figure 9. Each time a new listing is parsed and added to the database, it will go through the production pipeline. It will ultimately appear on the front-end application, together with the furniture label produced by the classifier.

3.2.1. Training pipeline

Preparing the datasets is a critical step in the pipeline as it helps achieve that the developed model will be able to respond to the multiple business requirements: 1) dealing well with a large variety of textual information from very different sources; 2) managing the ambiguity of published data in an explainable way, close to the way a client would handle this ambiguity. Due to its essential role in the result, the datasets were generated with the above goals as described next.

The training dataset was generated from an initial 2500 listings for rental properties. The dataset was fed into the model with the simple set of "seed" features (short dictionary of initial keywords, standard negations, and standard rule, which determines a positive or negative label based on the combined sequence of features, title, and description). From these listings, 100-150 were selected based on the market coverage of the agent and to balance positive and negative labels. Then selected samples were sent to human annotators from the customer service team, who were asked to validate the sample and add a flag for cases of ambiguity. Ambiguity was defined as the inconsistency between the title, description, and features attributes.

Additionally, the annotators were requested to highlight the samples with a conflict between textual information and photos. The current project was not using photo attributes. However, it was agreed that this additional information would aid in determining possible model improvements and the potential need to add photo attributes to the model.

The test dataset was selected at random to make sure the model would predict well the unseen instances.

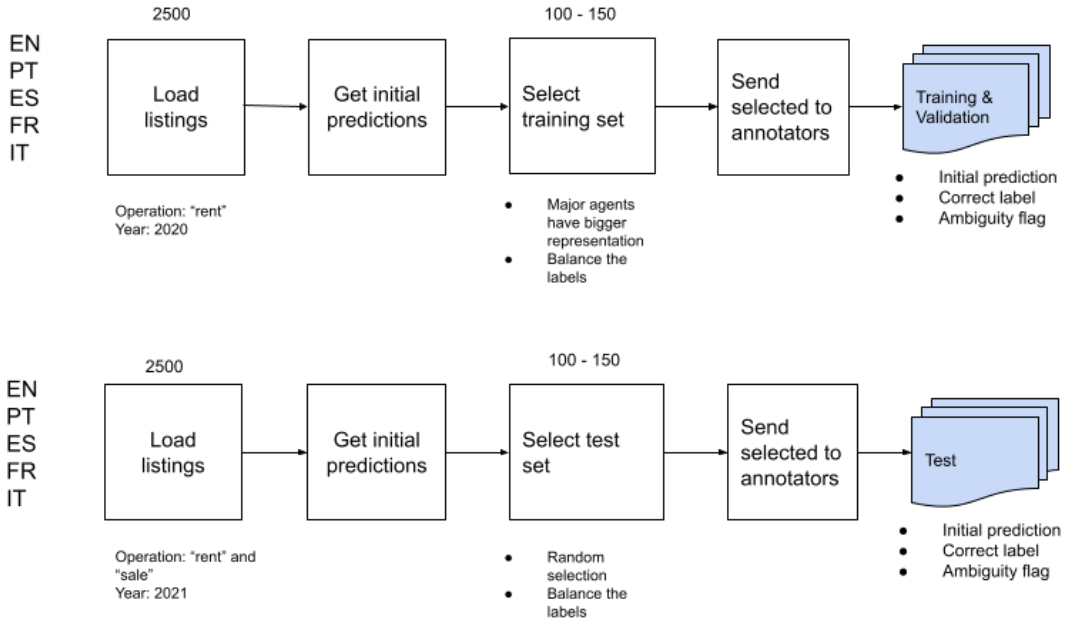

Figure 7. Process adopted for building the datasets

The "seed" features were generated manually as a simple dictionary in JSON format. However, more complex approaches were also considered while training the model and building keywords dictionaries and negations, which will be described in later chapters.

Once the datasets were labeled by annotators, training the model step consisted of expanding the dictionary and adjusting the classification rule until the model correctly assigned the "furnished" or "unfurnished" label to the samples that were assigned incorrect labels in the first run. All the mislabeled samples were analyzed.

The next step was to run the model against the test samples previously unseen to the classifier and evaluate the results. Errors were analyzed, and some corrections were made in the keywords and negations. Finally, the speed was tested and evaluated against the baseline from the production model.

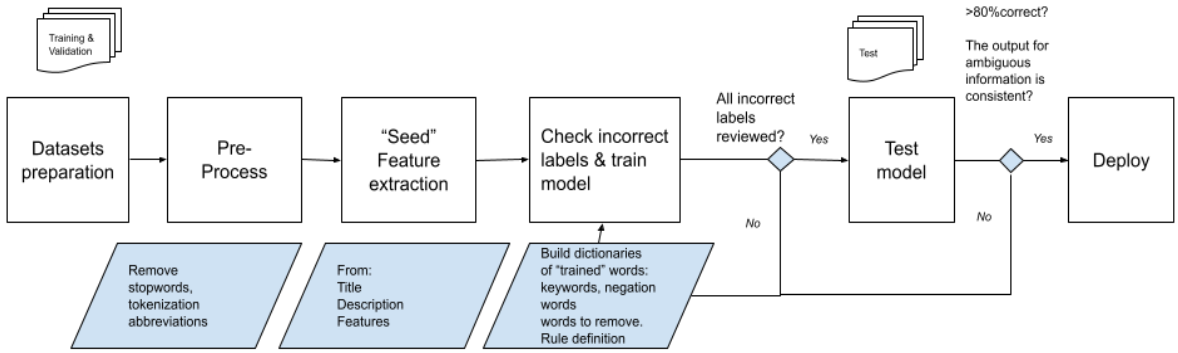

Figure 8. Training pipeline

Generally, the text samples belong to a particular domain. Therefore, there is no large diversity in style, as one would find in fiction literature, or complexity of terms as in scientific and academic literature. This lack of diversity makes the dictionary-based approach selected for modeling suitable for production as it is possible to achieve good results with relatively simple "seed" features.

3.2.2. Production Pipeline

Once the model results on the test set were found appropriate, the model was ready to be deployed in production. The script was incorporated into the production cycle. Its outputs were fed into the front-end application, where the clients could see and select the filter "furniture". This selection could be made when browsing for properties on the main metasearch screen or checking the market analytics metrics for a specific area of interest (for example, checking the price distribution for houses and apartments in the Lisbon area). The diagram of the production pipeline can be consulted in Figure 9 below. This pipeline was followed when new listings were added to the database. For successful deployment, it was essential to have the model code reviewed following the company policy and coordinate with the other teams for the next steps, as described in Chapter 1.2.1 *Business requirements*. The task was created for the front-end team to add a filter on the website that would take the model's output and visualize it. Similar actions were taken to communicate the new filter to custom customer-facing and quality assurance teams. In addition, without having a label assigned to every listing in the database, it was impossible to finish the deployment. In addition to the above steps, the model was run once for all the listings in the database to produce and store the label.

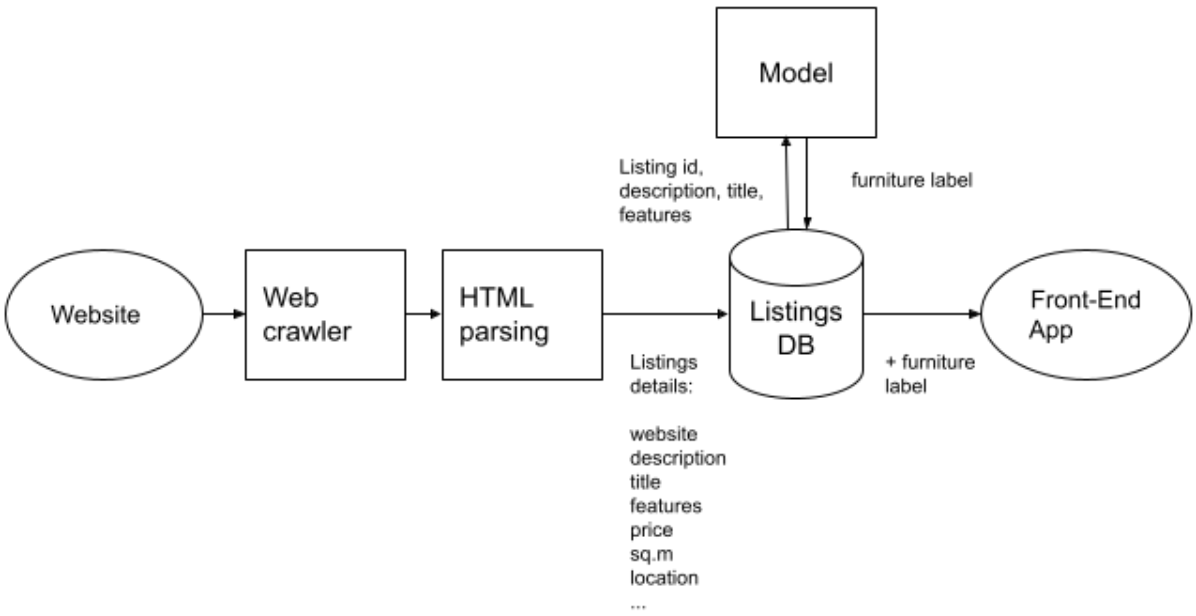

Figure 9. Production pipeline

3.3. SOFTWARE USED

In addition to Jira, which was used to calibrate the project, define the steps, set expected deadlines and track progress, collaborate with other teams, and provide project reporting, the following software was used to develop the code for the model:

- Python (including but not limited to the following libraries: dataclasses-json, fasttext, gensim, nltk, numpy, pandas,pymysql, python-dotenv, scikit-learn, unidecode, xlrd) (tensorflow, transformers)
- Jenkins as an automation, CI & CD solution
- Gemfury as a Python packages private hosting is used to upload models as packages.
- MySQL for querying the database
- Github as version control and collaboration tool

3.4. DATASET DESCRIPTION AND DATA PRE-PROCESSING

The data used to develop this algorithm is obtained from crawling different real estate websites for property listings.

The raw data we have access to in Casafari's database is by and large unlabeled as there is no clear-cut information on most of the features we will be looking for. The source table for this project is the table with parsed real estate listings. Each listing in the table has 82 attributes, which represent what typically can be found in a published property listing for sale or rent: listing unique id, location, website address, property reference, title, photos, property type and condition, property features, number of rooms, bedrooms, bathrooms, size of the property, property's description, price for sale or rent and so on. In addition, some attributes might not be directly visible in the original publication but are important for the correct management of the parsed data. Examples of this importance are the listing's creation date, the date it was delisted, listing's status that shows if the property is active or has been reserved or sold. Finally, remaining attributes in the database are generated by the data services that are running regularly and contain the output of the current models, such as structured feature labels, property's geo-position, and some others. Once this project is finished and the developed model is deployed in production, the "furnished"/"unfurnished" label will be added to these generated fields. It is important to note that the language in which the listing is published is not present among database fields. As a rule, the publications are in the local country's language, so listings for properties in Portugal are mainly in Portuguese. However, this is not always the case. Some international brokers publish in English or provide English translation after the original text. Many websites have the option to switch languages, and the data collection team policy is to parse the original language. However, due to human error, some websites are parsed in English instead of the local language.

The source of information for the model would be the *title, description, and feature list* of the published listing, so we will only be looking at those attributes in the SQL table to build the training and test datasets.

Initially, a sample of 2500 rental listings published in the year 2020 was generated for each market. Title, description, and features were combined into one text string so that each sample had a unique property id and the text string. The listings were pre-processed to prepare and clean up the datasets for predictions with the main steps as follows:

- The language was detected using a pre-trained model from open-source library *fasttext* developed by Facebook. This pre-processing was done first in the data cleaning pipeline, as some of the following steps require a language-specific treatment;

- Using regular expressions, separated the words that were merged during parsing. For example, this allows to eliminate parsing anomalies such as *furnitureFirst* where whitespace between two words was deleted;

- Lower-cased the text;
- Removed unnecessary non-words characters. Unicode library was used to clean the Unicode characters which create noise in the text;
- Tokenized the text: converted text string into a list of words (tokens) to allow better modeling;
- Replacement of common abbreviations that are semantically important with their full-text equivalent. Examples include *w/,* or *s/* which stand for *with* and *sem (without), or NC,* which stands for *Non-communiqué (not disclosed)*.

After pre-processing was done, the "seed" features were prepared: dictionary of keywords, dictionary of negation words, and the base rule definition adopted from a similar classifier deployed in production. The model with these baseline settings was run for all the listings for each of the five languages. Once each sample had been assigned the detected language and the initial prediction, we selected 150 listings for each language - English, Portuguese, Spanish, Italian and French - to send them for labeling to annotators. The selection was made to ensure the best representation of the market and the ability of the model to label the listings coming from different sources. The listings were grouped by agent, and agents were sorted by their size, defined by the number of unique properties they published in 2020. More samples were taken from bigger market players while smaller agencies were also represented. Each agent has a specific format of how the listings are published: description differs in size and its contents, some websites have features list, and some do not, the writing style of the authors differs also. This approach ensures first that the most frequent representations of the furniture feature are tested and validated. At the same time, grouping by agent allows capturing the diversity in style and format of various sources.

For testing, we used a dataset composed of listings extracted randomly from the period following the training dataset.

As the developed classifier is dictionary-based, there was no need to let the model learn from the training, validate if the training has been sufficient on the validation set, and evaluate based on the test set, as it happens typically with the machine learning pipeline. However, the selected approach of having two separate sets of data with different selection criteria allowed to optimize the model and make sure the final dictionaries and the classification rule will be suitable for most of the listings.

3.5. DATA PREPARATION AND FEATURE EXTRACTION

As the main purpose of the first run of the model was to prepare language-specific "golden standard" balanced datasets to serve as a good representation of the listings in the database, the approach was

to prepare the initial settings for the model with minimal time and resource and focus on building these features at subsequent runs of the model in its training phase.

The “seed” keywords were built based on synonyms of *furnished* and *furniture* in all five languages and contained around 4-5 elements. For example, for English language dictionary of keywords contained “furniture”, “furnished”, “furbished”, and dictionary of negations contained “zero”, “no”, “not”, “without”, “possibility”, “possible”, “optional”. After analysing prediction errors, several other words were added: for example, “decorated” and “decoration” to achieve a complete set of *trained words.*

During the previous implementation, word embeddings were tested as a method to generate the keywords automatically. However, this step did not significantly reduce the preparation time, and manual resource was still required to analyse prediction errors. Word embeddings were also generated to strengthen the documentation and filter out this method for future implementations. Word embeddings were generated from a dataset of around 1,000,000 listings descriptions using *gensim* library. Twenty words were added to the keywords dictionary from the top most similar vectors with the word “furnished”. It was observed that using word embedding allows catching uncommon spellings and misspellings (i.e., *furnished* and *furnihed*). It also allows catching words similar in meaning. However, it does not improve the model results significantly, and a simpler approach was adopted by taking the synonyms and manually building upon them by sampling model errors.

Also, it was observed that sometimes the keywords used in the wrong context could be misleading, so another step was added to the data preparation to remove the unwanted context. For example, in Spanish, “amueblado” is used commonly both for equipped kitchen (“cocina amueblada y equipada”) and furnished property (“casa amueblada”), and the model was making a prediction error for situations when the property was rented without furniture and with equipped kitchen. A third dictionary was created to fix these errors with keyword word combinations that are irrelevant and misleading, such as “cocina amueblada" (equipped kitchen). These combinations were removed from the corpus.

The three dictionaries (keywords, negations, words to remove) were completed after analyzing the prediction errors.

3.6. MODEL

The feature extraction model attempts to extract information on the characteristics each property listing has based on the available textual information. It does this by verifying which words in the corpus of the published listing - composed of the title, description, and features list - matches a set of *trained words* built from “seed” keywords by analyzing model errors.

For example, the model looks for the word “furnished” inside the listing textual fields. If it is found, then it identifies a positive match. The second level of this model tries to identify false positives by looking at the neighbors of the matched word, in our example “furnished”, and verifying if there is any word there that could indicate a false positive. An example of these words is “optional”, “no”, etc.

It is implemented by collecting a list of n-grams from each sentence in a text corpus, where n-gram is a word or sequence of words that would be matched with the list of keywords to find if there is an intersection. If a keywords dictionary contains word phrases in addition to single words, the maximum length of n-gram would be corresponding to the number of words in the word phrase to be able to match them correctly. If the English language keyword dictionary contains unigrams and bigrams (single words and two-word phrases) to be able to match all of them correctly, all sentences in the listing would be split into the list of all possible uni- and bi-grams, like in the following example for the sentence “it can be rented unfurnished” both uni- and bi-grams would be generated: ['it', 'can', 'be', 'rented', 'unfurnished', 'it can', 'can be', 'be rented', 'rented unfurnished', 'it can be', 'can be rented', 'be rented unfurnished'].

In addition to building a good set of keywords and negations, several parameters of the classifier rule can be fine-tuned to get better results. Several windows sizes were tested, and the optimal window to look for negation around keywords was found and set to 2 for window following the keyword and 3 for window preceding the keyword.

In case of uncertainty, the model favors negative labels, to avoid false positives in line with business requirements. For this reason, if a property is partially furnished, the model will identify it as unfurnished, and if the furniture is optional, the model will also identify it as unfurnished. While the “seed” classifier rule was looking at all textual information coming from title, description, and features, during training, it was modified to deal with ambiguity consistently. The model assesses what is written in the listing's title, description, and features. In case the information in these fields conflicts, it will output a negative label (for example, if the description contains the words “this furnished house” and features have a phrase “rented without furniture”, the output for the listing will be “unfurnished”).

The rule was developed to detect a conflict between explicit positive and negative labels and ignore the absence of a match. For example, if the model has identified no keyword matches in title and features and found a match with keywords in the description, the final prediction would be “furnished”.

3.7. EVALUATING DEEP LEARNING MODELS

3.7.1. Research scope

Deep learning models which require a large training corpus were filtered out at the initial stage of the research as one of the main constraints of deploying the model in production is scarcity and the high cost of labeled data. Due to this constraint and the fact that the model was required to scale well with new languages, we focused on the pre-trained multi-lingual models, as they respond well to both criteria and unlock the potential of state-of-the-art NLP without extensive training.

We will also be looking at zero-shot learning models, as the advantages of the zero-shot model is the fact that it requires no training, no prior custom pre-processing, no human resources to label the datasets, no data science resources to build keywords dictionaries, and has the potential to work across all target languages. For the purpose of this project, we understand zero-shot learning (ZSL) from a recent NLP perspective, where the model is able to do something without not being trained to do it (*Zero-Shot Learning in Modern NLP | Joe Davison Blog*, n.d.). For example, the models trained to recognize Natural Language Inference (NLI) are successfully used to classify text.

The research part of this project was built based on the resources provided at the community page of Transformers library: https://huggingface.co/models. The approach was to test the pre-trained models for the task of furniture extraction and compare the results with the developed dictionary-based model. We were only working with the out-of-the-box implementations since developing our own version would require computing resources both for development and production which are not available.

As raw pre-trained models are mostly intended to be fine-tuned for a specific task, only models fine-tuned for text classification were taken from the models' library for the goal of this project. In addition, due to the unavailability of training data and high cost of labeling, only the models tagged with both "text classification" and "zero-shot-learning" (zero shot classification tag) were explored and evaluated against the test dataset with real estate listings to see if they can be put into production: 9 models in total (see Appendix I for the full list of models and links to the code).

3.7.2. Models

Initially, we tested *xlm-roberta-large-xnli* model to have a baseline comparison to the previously developed dictionary-based model. It was tested on the same datasets in five available languages: English, French, Spanish, Portuguese, and Italian. The model is fine-tuned on XNLI, The Cross-Lingual Natural Language Inference Corpus, which includes 15 languages: Arabic, Bulgarian, Chinese, English,

French, German, Greek, Hindi, Russian, Spanish, Swahili, Thai, Turkish, Urdu, and Vietnamese. The base model is trained on 85 more languages so that the model will work to some degree for any of those in the XLM RoBERTa training corpus. A full list of languages can be found in Appendix A of the paper “Unsupervised Cross-lingual Representation Learning at Scale”(Conneau et al., 2019). Therefore, a Roberta-trained classifier can be run without additional training or language detection required for our baseline dictionary-based model.

The underlying model is trained on the task of NLI, which takes in two sequences and determines whether they contradict each other (contradiction), imply each other (entailment), or neither (neutral). This train can be adapted to the task of zero-shot classification by treating the sentence we want to classify as one NLI sequence (called the premise) and turning a candidate label into the other (the hypothesis). If the model predicts that the constructed premise entails the hypothesis, we can assume that the label applies to the text.

The model was tested for each scenario, identified previously for the dictionary-based model: the samples where the presence of furniture is clear-cut and expressly mentioned, the samples that clearly indicate the property is unfurnished, the samples with no indication of absence or presence of furniture, and finally the samples with information conflict in different sections of the listing. The model results were compared to results of the dictionary-based model for each language separately.

When dealing with the first two scenarios, the model was able to perform well without any adjustments, with the simple candidate labels "furnished" and "unfurnished". However, for samples with no indicators for furniture or absence, the model gives a very similar score to both labels, with a slight preference to "furnished". It was possible to achieve the desired label “unfurnished” and correct false positives in these cases by transforming the negative label from “unfurnished” to "unfurnished or unlikely furnished". Similar behavior was observed for all languages. Adding the third label, “unknown” was also tested, and a few variants of the new label such as "unfurnished or partly furnished", however "unfurnished or unlikely furnished" performed best. Replacing the default hypothesis template with a more suitable “The property is rented {}” has also slightly improved the overall scores.

Initially, the labels and hypothesis template were fed into the model in English, while the sequences to classify were in the original language. This approach was tested first as the goal was to see the performance without a prior step of language recognition, necessary in a dictionary-based model. The results were satisfactory, each local language showing an accuracy score of a little under 80%, which was our target from a business perspective. However, for further improvement and language-specific

comparison, the model was also tested with the translation of both labels and hypothesis templates to target languages.

It was observed that the multilingual pre-trained models perform differently depending on the language, while prediction speed does not differ across languages. Since the model achieved the best scores for English language samples, it was decided to explore further the options that would improve the speed results based on the English test set. To improve the speed, we considered the distillation of *xlm-roberta-large-xnli.* However, this was rejected as the development would require much more extensive resources than what is available to this project and outside of this project's scope. Instead, available off-the-shelf distilled models were tested and benchmarked against the large *xlm-roberta-large-xnli: distilbart-mnli*, *distilbert,* and *distilroberta*. In addition, several other large models were also tested. Full results of the tests can be found in the following chapter in section 4.2, and further descriptions of the models with links to the code are in Appendix I.

4. RESULTS AND DISCUSSION

4.1. METRICS

To evaluate the performance of our dictionary-based model and compare it with the alternative implementations, we used one of the most popular metrics for evaluating text classification: accuracy, precision, recall, and F1 score. These metrics operate with concepts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) (Minaee, Cambria, et al., 2020).

True positives and true negatives are the instances where the model correctly predicted positive and negative labels. In contrast, false positives are the number of negative instances predicted as positive, and false negatives are positive instances incorrectly identified by the model as negatives.

Accuracy measures how many correct predictions the classifier made out of the total number of samples in the dataset. Precision measures the proportion of all positive predictions correctly and actually matches the real positive samples. Recall measures how many of the positive cases were correctly predicted as positive. This definition means that the recall of “furnished” label shows how many listings for furnished properties are correctly classified divided by how many furnished listings are in the dataset. The F1 Score is used when we look to achieve a balance between precision and recall.

In addition to the above metrics, we measured the speed of the models as this is an important factor in model deployment in production. The speed was measured based on the average time the model takes to classify one sample in 5 iterations on the full test set.

4.2. EVALUATION

4.2.1. Evaluation of dictionary-based classifier

After training our model, building dictionaries, and fine-tuning the classification rule, the goal of the model is to predict new occurrences. The following result was achieved for each of the five target languages (Table 1):

Table 1. Scores for dictionary-based furniture detection model (test set)

	accuracy	F1	precision	recall
EN	0.976744	0.98	1.00	0.97
ES	0.982759	0.98	0.99	0.98
FR	0.991453	0.99	1.00	0.99
IT	0.967213	0.97	0.98	0.97
PT	0.969298	0.97	0.98	0.97

All languages are well above the target 80% accuracy score and are acceptable for deployment in production. The cases of ambiguity produce a consistent result, and the average speed of prediction is 0.038 seconds.

Despite its excellent overall performance, the dictionary-based classifier could not correctly predict cases with double negatives. For example, for a text sample “can be rented without furniture”, the model would predict the property is unfurnished as there is a negation word in the surroundings of the keyword. It is worth mentioning that the percentage of these cases is minimal.

The dictionary-based classifier cannot handle the nuances of the language well. Examples such as “arrendado com mobília da sala” (rented with living room furniture) would output a positive label. A human annotator would understand that partial furnishings are implied. Incorrect predictions could be avoided in these cases by placing the phrase in the dictionary of words to remove. However, this would not be a robust fix as there are too many variations of the phrase, potentially leading to false negatives.

Finally, the dictionary-based model does not recognize context and is not able to recognize negations relating to furniture from negations relating to another word in the sentence. For example, for text sample “El piso se entrega con muebles y electrodomesticos, no le falta ni un detalle” (the property is rented with furniture and appliances), the model outputs “unfurnished” as the classifier finds negation ”no” in the vicinity of the keyword “muebles” (furniture). These cases are extremely rare.

4.2.2. Evaluation of zero-shot models

In comparison, *the xlm-roberta-large-xnli* model showed outstanding English language results while other languages achieved lower overall scores. Table 2 shows accuracy, precision, recall, and F1 scores for the model that was set up with a hypothesis and candidate labels in the English language, thus not requiring the step of language recognition. It achieves a 91.8% accuracy score for English samples, and

it is only 5% lower than the dictionary-based model, which will be put in production. However, the model underperforms for other languages, especially for Portuguese, for which the accuracy drops by 20%.

Table 2. Scores for xlm-roberta-large-xnli furniture detection model. Hypothesis template and candidate labels in English (test set)

	accuracy	F1	precision	recall
EN	0.9180	0.9180	0.9180	0.9180
ES	0.8276	0.8276	0.8276	0.8276
FR	0.7863	0.7863	0.7863	0.7863
IT	0.7992	0.7992	0.7992	0.7992
PT	0.7544	0.7544	0.7544	0.7544

With hypothesis template and candidate labels in target languages, the model improved the scores by 3-5% for Spanish, Italian and Portuguese, while achieving 91% for French, which was an improvement of almost 13% (Table 3). Thus, a possible deployment of this zero-shot model could be with a preliminary language detection step, which would determine which language-specific parameters to use for the classifier, as done in dictionary-based model development.

Table 3. Scores for xlm-roberta-large-xnli furniture detection model. Hypothesis template and candidate labels in target languages (test set)

	accuracy	F1	precision	recall
EN	0.9180	0.9180	0.9180	0.9180
ES	0.8700	0.8700	0.8700	0.8700
FR	0.9100	0.9100	0.9100	0.9100
IT	0.8278	0.8278	0.8278	0.8278
PT	0.7192	0.7192	0.7192	0.7192

Only Portuguese language scores were below the target score for production, while other languages, even inferior to the dictionary-based model, are competitive and can be considered for production. The Spanish, Italian, and Portuguese scores could also be further improved by exploring more language-specific pre-trained models like Portuguese-Bert.

Double negatives are overall better handled by a zero-shot model, compared to dictionary-based model in cases which are a simple double negation like “is not rented without furniture”. However, in

cases with more subtle negations both models fail. In examples such as “can be rented without furniture”, “there is a possibility to rent the house without furniture” neither of the models predicts “furnished” label. However, zero-shot model has better chances of capturing the correct meaning if the context allows it.

When it comes to handling the nuances of the language, zero-shot model and dictionary-based models have both incorrect predictions for the given test data. In earlier mentioned example “arrendado com mobília da sala” (rented with living room furniture) zero-shot model also outputs incorrect “furnished” prediction, as it is not able to attribute the living room furniture to partial furnishings, similar to dictionary-based model. In samples containing a description of different pieces of furniture available in the property, like “bed” or ”table”, instead of keyword “furniture” (zero-shot model generally produces correct “furnished” label, however a few “unfurnished” predictions are also present. In those cases, it was possible to make the model produce positive prediction by altering the context, however this behaviour of the model showed the lack of consistency. Altering the context words cannot be a part of production cycle, and it was done purely to research these inconsistent predictions.

In addition, dictionary-based model produces consistent negative predictions for samples with the conflict of information between title, description, and features text fields, while zero-shot model results do not have a pattern that is as easy to interpret due to the mix of positive and negative labels.

Zero-shot model is more versatile compared to dictionary-based model and is able to recognize the context and which part of the sentence the negations refer to. In the example ‘‘El piso se entrega con muebles y electrodomesticos, no le falta ni un detalle” (the property is rented with furniture and appliances) zero-shot model outputs the correct “furnished” label.

In terms of speed performance, the tests showed that the dictionary-based model is far superior to *xlm-roberta-large-xnli* model. Zero-shot pre-trained model is simply too big and slow to be deployed as is. Distilled models like *distilroberta* and *distilbert* come closer to dictionary-based models, even though the difference is still large. Still, unfortunately, they lose accuracy to a greater extent than it is acceptable for production. The full results can be consulted in Table 4 below.

Table 4. Pre-trained NLP models results tested for furniture detection (test set)

model	accuracy	precision	recall	model speed
valhalla/distilbart-mnli-12-1	0.930233	0.930233	0.930233	2.26759
joeddav/xlm-roberta-large-xnli	0.918605	0.918605	0.918605	3.94398
valhalla/distilbart-mnli-12-6	0.918605	0.918605	0.918605	3.14771
valhalla/distilbart-mnli-12-3	0.918605	0.918605	0.918605	2.54813
facebook/bart-large-mnli	0.906977	0.906977	0.906977	4.5232
valhalla/distilbart-mnli-12-9	0.895349	0.895349	0.895349	4.04507
joeddav/bart-large-mnli-yahoo-answers	0.790698	0.790698	0.790698	4.80242
distilroberta-base	0.77907	0.77907	0.77907	0.668273
Distilbert-base-uncased	0.22093	0.22093	0.22093	0.523209

Without a significant speed improvement, none of the tested pre-trained models is a valid competitor to the dictionary-based model. They use far too many computing resources to efficiently run in the production environment for each new listing in the database.

4.2.3. Maintenance

To monitor the performance of the dictionary-based model, some samples must be randomly selected and sent for human evaluation. If the classifier consistently outputs an incorrect label, the model needs to be retrained by either expanding the dictionaries or changing the rule itself. In addition to this, the clients and customer service team can flag the errors directly in the website application. The data science team can use this data to do quality control of the deployed model.

When the company enters a new market, the classifier needs to be trained for the new language.

5. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORKS

The project produced a successful classifier which was then deployed in production. There is no negative feedback from the clients or customer-facing teams to the new filter implemented in the application, and no error flagged until now. The classifier's output can be further integrated into the company's data services, specifically its valuation model and search for comparable properties and advanced market analytics.

The dictionary-based classifier was an excellent choice for the production as it produced consistently good scores with minimal speed. Its results can be explained to non-technical business experts and clients. Its maintenance and quality control can be done as part of the established process in the company without changes and do not require additional training or resources.

The dictionary-based model can be adapted fast should a change in business requirements occur. Suppose there is a need in the future to add the “unknown” label to existing “furnished”/“unfurnished” labels or introduce a new “partially furnished” label. In that case, the development can be done with minimal time and resources.

Based on additional research in deep learning NLP models, it was observed that some models produce comparable and acceptable results without any need for human labeling or model training. However, no pre-trained model was deemed appropriate for the production environment due to its size, GPU requirements, and slow prediction speed. Exploring the options, we identified that several techniques like distilling could be used to improve the prediction speed. The main challenge will be to find a trade-off between a decrease in speed and a decrease in scores, which was not achieved in pre-trained distilled models but can be explored in further development of our own distilled version finetuned on real estate listings data. This development would require large GPU and resources that were not available for the current project, and it was left outside of the project scope.

An additional consideration is the interpretability of the deep learning NLP models, as they remain a “black-box” implementation, which is unacceptable from a business perspective. For the moment, the advantages of the zero-shot model, such as no resources for labeling and training and the potential of working across multiple languages, do not outweigh the reluctance of business stakeholders for a “black-box” implementation.

In conclusion, a dictionary-based approach is easy and fast to implement and was a good fit for the business goal.

Further development can be done by introducing image recognition to detect furniture in the listings and allocating the resources to distil a company version of the NLP model. The latter would only make sense if the strategy shifted towards introducing more powerful models, which are harder to interpret.

6. BIBLIOGRAPHY

A Comprehensive Guide To Learning Text Classification. (n.d.). Retrieved May 10, 2021, from https://www.digitalvidya.com/blog/text-classification/

Bucila, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model Compression Cristian. *Kdd*, *54*(1), 1–9. http://users.iems.northwestern.edu/~nocedal/PDFfiles/dss.pdf%0Ahttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.7952&rep=rep1&type=pdf%255Cnhttp://books.nips.cc/papers/files/nips15/AA03.pdf%250Ahttps://ieeexplore.ieee.org/document/85783

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2019). *Unsupervised Cross-lingual Representation Learning at Scale*. http://arxiv.org/abs/1911.02116

Deloitte. (2020). *Real Estate Predictions 2020 | Article 5 Proptech on the move*. https://www2.deloitte.com/content/dam/Deloitte/pt/Documents/RealEstate/repredictions2020/deloitte-re-predictions-2020-5.pdf

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). {BERT:} Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, *abs/1810.0*. http://arxiv.org/abs/1810.04805

Dima Williams. (2020, January 5). *How Big Data Will Impact Real Estate Buying, Selling and Developing - Mansion Global*. https://www.mansionglobal.com/articles/how-big-data-will-impact-real-estate-buying-selling-and-developing-210771

Faster and smaller quantized NLP with Hugging Face and ONNX Runtime | by Yufeng Li | Microsoft Azure | Medium. (n.d.). Retrieved July 26, 2021, from https://medium.com/microsoftazure/faster-and-smaller-quantized-nlp-with-hugging-face-and-onnx-runtime-ec5525473bb7

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*, *129*(6), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z

Gupta, M., & Agrawal, P. (2020). *Compression of Deep Learning Models for Text: A Survey*. 1–53. http://arxiv.org/abs/2008.05221

Gupta, V., & Lehal, G. (2009). A Survey of Text Mining Techniques and Applications. *Journal of Emerging Technologies in Web Intelligence*, *1*. https://doi.org/10.4304/jetwi.1.1.60-76

Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network*. 1–9. http://arxiv.org/abs/1503.02531

Hotho, A., Nürnberger, A., & Paass, G. (2005). A Brief Survey of Text Mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, *20*, 19–62.

How Does Text Classification Work? | Unite.AI. (n.d.). Retrieved May 9, 2021, from https://www.unite.ai/how-does-text-classification-work/

KPMG. (2019). *2019 Real Estate Technology Trends survey results*. https://assets.kpmg/content/dam/kpmg/us/pdf/2019/09/real-estate-tech-trends-print.pdf

KPMG Real Estate Advisory. (2020). *Real Estate Innovations Overview, 5th Annual Edition: Vol. 5th Annual* (Issue July). https://assets.kpmg/content/dam/kpmg/nl/pdf/2020/sectoren/real-estate-innovations-overview-2020.pdf

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. In *arXiv*. arXiv.

Mikolov, T., Chen, K., Corrado, G. s, & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR*, *2013*.

Minaee, S., Cambria, E., & Gao, J. (2020). *Deep Learning Based Text Classification: A Comprehensive Review*. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2020). Deep Learning Based Text Classification: A Comprehensive Review. *ArXiv*, *1*(1), 1–43.

Mirończuk, M. M., & Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. In *Expert Systems with Applications* (Vol. 106, pp. 36–54). Elsevier Ltd. https://doi.org/10.1016/j.eswa.2018.03.058

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1532–1543. https://doi.org/10.3115/v1/d14-1162

Radford, A., & Narasimhan, K. (2018). *Improving Language Understanding by Generative Pre-Training*.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2–6. http://arxiv.org/abs/1910.01108

Talib, R., Kashif, M., Ayesha, S., & Fatima, F. (2016). Text Mining: Techniques, Applications and Issues. *International Journal of Advanced Computer Science and Applications*, *7*. https://doi.org/10.14569/IJACSA.2016.071153

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-December*, 5999–6009.

What is a Transformer?. An Introduction to Transformers and… | by Maxime | Inside Machine learning | Medium. (n.d.). Retrieved May 9, 2021, from https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

What is Agile? | Atlassian. (n.d.). Retrieved June 7, 2021, from https://www.atlassian.com/agile

Zero-Shot Learning in Modern NLP | Joe Davison Blog. (n.d.). Retrieved June 4, 2021, from https://joeddav.github.io/blog/2020/05/29/ZSL.html

APPENDIX I. FULL LIST OF MODELS

Pre-trained language models suitable for text classification with little or zero training data and available for target languages (from HuggingFace library https://huggingface.co/models):

No	Model	Description	Scores
1.	facebook/bart-large-mnli	This is the checkpoint for bart-large after being trained on the MultiNLI (MNLI) dataset.	Accuracy: 0.906977 Precision: 0.906977 Recall: 0.906977
2.	valhalla/distilbart-mnli-12-1	Distilled version of bart-large-mnli, using the No Teacher Distillation technique proposed for BART summarisation by Huggingface, here.	Accuracy: 0.930233 Precision: 0.930233 Recall: 0.930233
3.	joeddav/xlm-roberta-large-xnli	This model takes xlm-roberta-large and fine-tunes it on a combination of NLI data in 15 languages.	Accuracy: 0.918605 Precision: 0.918605 Recall: 0.918605
4.	joeddav/bart-large-mnli-yahoo-answers	This model takes facebook/bart-large-mnli and fine-tunes it on Yahoo Answers topic classification. It can be used to predict whether a topic label can be assigned to a given sequence, whether or not the label has been seen before.	Accuracy: 0.790698 Precision: 0.790698 Recall: 0.790698
5.	valhalla/distilbart-mnli-12-3	Distilled version of bart-large-mnli created using the No Teacher Distillation technique proposed for BART summarisation by Huggingface, here.	Accuracy: 0.918605 Precision: 0.918605 Recall: 0.918605
6.	distilroberta-base	This model is a distilled version of the RoBERTa-base model. It follows the same training procedure as DistilBERT.	Accuracy: 0.77907 Precision: 0.77907 Recall: 0.77907
7.	valhalla/distilbart-mnli-12-9	Distilled version of bart-large-mnli created using the No Teacher Distillation technique proposed	Accuracy: 0.895349 Precision: 0.895349 Recall: 0.895349

		for BART summarisation by Huggingface, here.	
8.	valhalla/distilbart-mnli-12-6	Distilled version of bart-large-mnli, using the No Teacher Distillation technique proposed for BART summarisation by Huggingface, here.	Accuracy: 0.918605 Precision: 0.918605 Recall: 0.918605
9.	distilbert-base-uncased	This model is a distilled version of the BERT base model. DistilBERT is a transformers model, smaller and faster than BERT, pre-trained on the same corpus in a self-supervised fashion, using the BERT base model as a teacher.	Accuracy: 0.22093 Precision: 0.22093 Recall: 0.22093