UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Large-scale product classification for efficient matching in procurement systems

Author:
Ihor HRYSHA

Supervisor:
Samuel GRONDAHL

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Ihor HRYSHA, declare that this thesis titled, "Large-scale product classification for efficient matching in procurement systems" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"There are three constants in life... change, choice and principles."

Stephen Covey

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Large-scale product classification for efficient matching in procurement systems

by Ihor HRYSHA

Abstract

We consider the problem of recommending relevant suppliers given detailed request context in a procurement setting. The fundamental recommendation in procurement systems is that a single query has potentially hundreds of relevant suppliers associated. A complicating factor is that, for most suppliers, we do not have a complete listing of product and service offerings, in contrast with most literature in the space of product search. An additional difficulty is introduced by the fact that queries are generated by users operating within large procurement organizations, each building queries in idiosyncratic but internally consistent ways, and each organizing activities according to a unique internal product taxonomy. The central research question that we aim to address is: can we utilize this vast but inconsistently structured set of product data that allows us to derive semantic meaning across users and contexts? We propose several fully and semi-supervised approaches and benchmark them using a proprietary dataset that includes large-scale procurement data as well as supplier-provided catalogs. Finally, and uniquely, we experimentally validate the performance of our preferred model in a live production setting.

Acknowledgements

First of all, I would like to thank the entire Fairmarkit team for smooth onboarding to the enterprise-grade procurement, and especially Viktor Kushch for the opportunity to work on significant problems, the solution of which makes our world a little better.

Special thanks to my supervisor Sam Grondahl for all the ideas, conversations, and freedom of choice he has given me. Thanks to the entire Data Science department for meaningful conversations that helped bring us closer to the truth. Noah Yusen, thank you for helping to deal with such diverse data sources.

Last but not least, it is essential to mention the whole Ukrainian Catholic University community. You create a beautiful space where everyone can grow and become a better version of themselves. I am personally grateful to Oleksii Molchanovskyi for his enthusiasm and empathy. The support you have provided was a crucial part of my success in finishing the program.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
UNSPSC	**U**nited **N**ations **S**tandard **P**roducts and **S**ervices Code
SKU	**S**tock **K**eeping **U**nit
MRO	**M**aintenance **R**epair and **O**perations

Dedicated to my loving wife and children. Thank you for the support and patience.

Chapter 1

Introduction

1.1 Rise of procurement centered solutions

Information technology is increasingly penetrating our daily and business lives. In recent years, user expectations of information systems have dramatically changed. The information system was considered the only common source of truth, where one could either enter data or view reports. Nowadays, users expect the system to be an intelligent partner who can help make decisions and minimize time spent on routine tasks. User-faced information systems are increasingly integrated with cloud solutions, allowing them to collect the necessary information and learn to understand user intent better.

We can observe the rapid growth in procurement automation services over the last decade. A prominent example of public procurement is the reformation of government tenders and the creation of the ProZorro[1] information system in Ukraine. Medium and large-sized businesses try to automate the processes of their procurement departments through the adoption of systems that help keep track of all events, helping find the best suppliers. However, in the case of commercial organizations, such tenders are not always publicly available or may be difficult for supplier representatives to access. Bensch, 2012; Zhang and Wang, 2005. Therefore, procurement departments have to do a lot of manual operations to communicate, summarize and process the information received from suppliers to make a final decision and award the winner.

A well-conducted procurement request should attract as many suppliers as possible, stimulate competition between them, and, as a result, the buyer should receive the best market prices for products and services that they plan to purchase García Rodríguez et al., 2020. However, how to bring information about the intention of buying some product to the market and help the buyer choose the most relevant suppliers.

Many US-based startups recognized and understood how to exploit such a gap and started to develop solutions. Procurement startups have been snowballing over the last few years, narrowing their focus to nonstrategic procurement and tail spent optimization. A bright representative of such startups is Fairmarkit [2] — which mainly focused on tail spent optimization.

1.2 Economic motivation

There are some apparent differences between the startups and the Ukrainian ProZorro. Nevertheless, they still share the same goal, which is quite simple: to save

[1] https://prozorro.gov.ua/
[2] https://www.fairmarkit.com/

buyer organizations money by stimulating competition between suppliers on the market.

We analyzed open public procurement in Ukraine and found that about 39% had just one tender participant, the tender winner. The reasons for single-bidder tenders may vary. They can be a natural monopoly on resources by the supplier or an early indicator of unfair tender condition, ineffective information about the tender, or low market interest Fazekas, 2019. Such similarities can be found in the public procurement process of some European countries. According to various data sources, the level of single bid tenders ranges between 30-40% and depends on the country and category of supplies.

Another research conducted in Spain by García Rodríguez et al., 2020 shows the relationship between the number of participants and the relative cost savings that the procuring entity can obtain (Figure A.1). The level of such economy on the tender price reaches a plateau of 35% Fazekas, 2019. It shows the real market price for the products and services. We have done similar research on tenders conducted by governmental organizations, and Figure A.2 demonstrates an almost identical trend. Assuming that an ideal condition is a sufficient number of bidders — more than 4 per tender, we lose over UAH 137 billion (approx. USD 4.8 billion) savings due to the lack of a sufficient number of participants. The collection of similar statistics for Ukrainian commercial companies is challenging, as this information is not publicly available, and they use RFQ documents that do not have an initial price. However, empirical 3+ bids per RFQ is the expected threshold of quality of a deal. In addition to the obvious financial benefits, commercial organizations want to increase independence from suppliers through diversification. Besides, many companies are trying to position themselves as socially responsible businesses. Keeping in mind this goal, procurement departments try to attract suppliers not only for financial motivation but also to consider other factors: sustainability, support for local businesses, support of minority-owned businesses, etc.

Thus, despite the differences between governmental and commercial organizations, the fact remains that as many relevant suppliers as possible are one of the crucial factors of a successful procurement.

1.3 Thesis goals and structure

In Chapter 1, we make a brief overview of the procurement solutions and answer the question of why it is essential to recommend an optimal set of suppliers from an economic standpoint.

There is much research in a community dedicated to the recommendations in eCommerce. We start Chapter 2 by emphasizing the difference and complexity of recommendations in procurement. We review different approaches and discuss their applicability to our specific domain. In this section, we formalize the primary goal of our research - creating the classifier to use as one of the filters in the search engine. The rest of the section overviews different text classification algorithms.

We claim that for our specific task, no appropriate dataset exists. We start Chapter 3 with the requirements for the dataset we have and compare existing datasets. We continue with a detailed explanation of the data collection and data transformation pipeline. We describe our practical steps to shape our data to the format to be used in our experiments. In this section, we measure different dataset version quality.

The best version of the dataset is used in Chapter 4 to conduct experiments. In our research, we iterate each dataset sequentially, trying to improve the classifier performance. Most of our experiments aim at revealing the main properties of the data we use.

In the final Chapter 5, we summarize the results of the research and share our thoughts and suggestions about its future directions.

Chapter 2

Related Work

2.1 Automated procurement process

There exist considerable differences in the procurement process in the commercial and government organization. However, the general flow of a procurement process remains the same and is schematically shown in Figure 2.1.

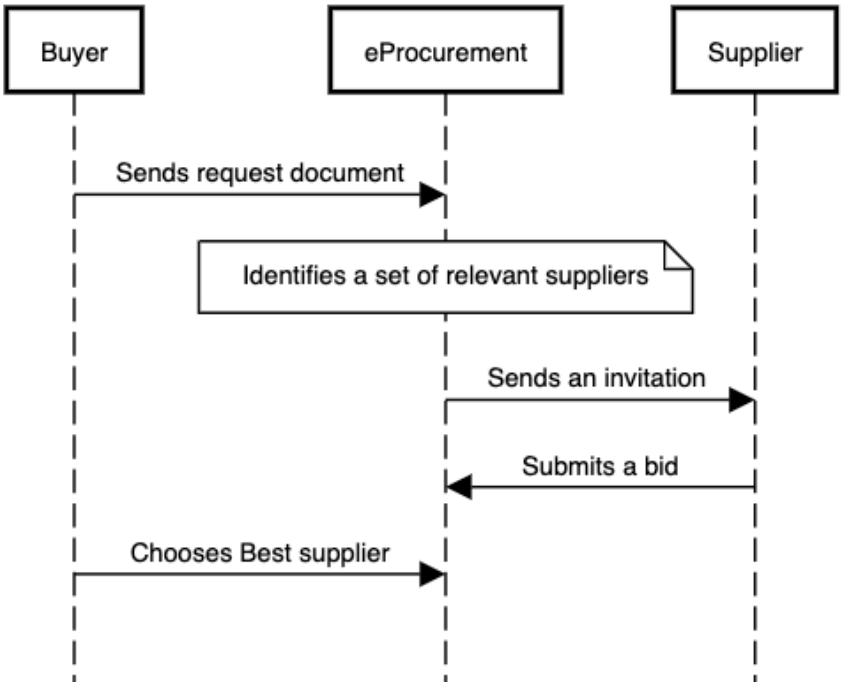

FIGURE 2.1: Automated procurement process.

The process starts with the initial step of the creation of a request document (request for quotation/proposal, tender notice). The specification of products/services organizations wish to purchase is attached in the free form to the requested document. Request document is the basis for all future transactions. In the automated setting, given the request context, the intermediate system tries to identify the best matching set of suppliers. Notified suppliers study the request and specification documents and submit a bid based on internal catalogs (listing of products/services they can offer). If the purchase is made in an auction mode, the suppliers are motivated by the fact that they can directly affect the result by lowering the price. The result in auction mode is usually determined automatically. If the purchase does not involve an auction, the buyer chooses the best (according to various criteria) bidder compared to others.

The supplier and the buyer of a particular company have their internal product catalogs. For example Commerce XML [1] and Catalog Interchange Format[2] standards were designed to expose supplier catalogs to the buyer. Nevertheless, a level of adoption of such standards among suppliers is low. As a rule, in the procurement process, the subject of procurement is described as an unstructured text. Reducing the space of unstructured product descriptions to a more compact representation, as a feature vector (embedding vector) or mapping, such unstructured data to some generally accepted taxonomy would simplify the further development of supplier recommendation systems and allow the adoption of a wide range of algorithms developed for eCommerce domain.

2.2 Problem Setting

2.2.1 General-purpose recommender algorithms

In academic literature, the "recommender system" was mentioned in Koren, Bell, and Volinsky, 2009 by providing an example of the Netflix Prize competition. It is defined as the large sparse matrix that stores relations between “product” columns and “customer” rows, which can be used to retrieve hidden connections using matrix factorization techniques. The main focus of such recommendations is to find similarities between customer preferences and recommendations based on transactions of similar customers (collaborative filtering) Aleksandrova, 2017. Another commonly used approach in practice is the analysis of products and the recommendation of similar products (content-based filtering). The implementation of such an approach requires a well-prepared catalog of products and services with descriptive features, which this type of recommendation can be based on Sarwar et al., 2001. In addition to collaborative and item-based filtering, other approaches either use a combination of the above two or use additional sources of information about the relationships between entities (social graph filtering, expert knowledge filtering).

Unfortunately, the procurement recommender system has two significant limitations that do not allow one to fully leverage the above approaches Zhang and Wang, 2005:

1. Another entity is introduced — the supplier, which is the subject of the recommendation. Even though the customer is interested in purchasing certain products, the system should recommend a list of the most relevant suppliers.

2. There is no general catalog of products and services. In contrast to the classic recommendation systems, where the concept of "product" corresponds to some record in the lookup list, procurement documentation of the client usually describes in free form text the subject of the procurement or can attach a specification document that contains information about goods in unstructured form.

Solving the problem of introducing an additional entity, the supplier converts the “client-product” matrix to the third-order tensor, which stores “buyer-product-supplier” relations Zhang and Wang, 2005. Factorization of tensors Karatzoglou et al., 2010 is a less-studied problem than factorization of matrices, but many works in the field exist that help to solve this problem. However, to prepare such a tensor, we need to have a complete list of products (or classes), not unstructured descriptions.

[1] http://cxml.org/
[2] SAP documentation

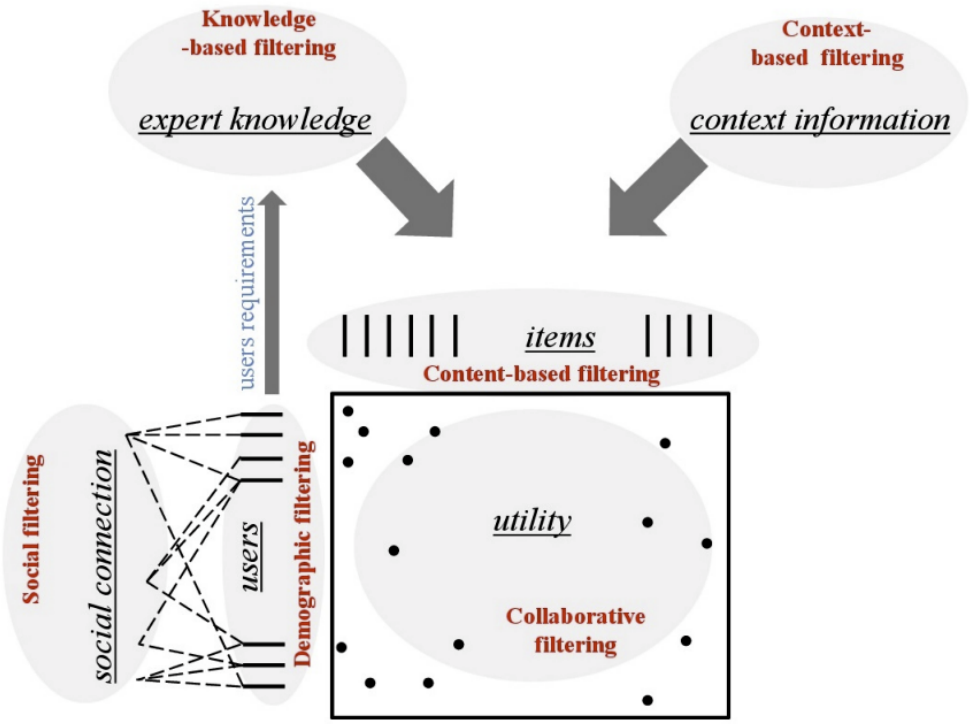

FIGURE 2.2: Filtering Techniques of Recommender Systems Aleksandrova, 2017

In our opinion, this problem is the least researched in the academic community and requires the most attention.

2.2.2 Supplier recommendation in procurement

According to Wikipedia, information retrieval is the process of obtaining information system resources that are relevant to an information need from a collection of those resources. The main applications of information retrieval techniques are mentioned above (recommender systems, search engines). Many authors do not separate them and consider recommendation systems to be zero-query search Belkin and Croft, 1992. We consider it appropriate to separate these concepts to make them less ambiguous. Although the term recommender system in procurement has historically been used to define such systems, we will understand it as a search engine since the requester aims to find the most relevant suppliers according to a previously specified query (description of products and services, previous transaction history, geolocation, etc.). In the following section, we will try to separate the concepts of recommender system and search engine and give an overview of the latter, as well as identify places where the task of product classification is appropriate.

2.2.3 Product classification in supplier search

Modern search engines are usually complex composite systems, which consist of two fundamental stages:

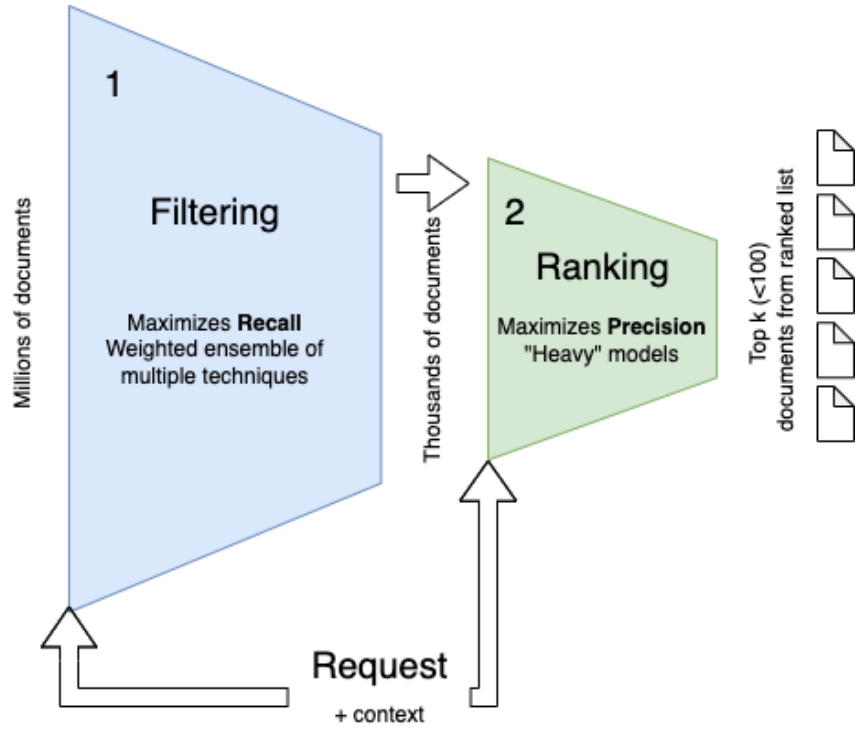

FIGURE 2.3: Modern search engine architecture.

1. **Filtering**. In this step, the system's task is to reject negative instances as quickly as possible. The result of this step should contain as many true positive instances as possible, but they can occur in the wrong order.

2. **Ranking**. The filtered data obtained in step 1 is sorted for a specific user request context. As a result, we get a sorted list with more relevant documents at the top. This step is computationally more complex.

On the *filtering* stage, a weighted ensemble of techniques is used to find the most relevant documents. The search query is compared to all documents in the search collection. Basic search techniques can be divided into the following types:

- *Calculation of token frequency*. Classic methods that work on the inverse index by counting the number of tokens in each document (term frequency) and weighing them to the frequency of occurrence in the whole dataset (inverse document frequency) Robertson, Zaragoza, et al., 2009; Chris Manning, Pandu Nayak, 2018.

- *Search in a certain taxonomy*. All documents are classified into one or more classes beforehand. The query is classified at runtime, and the result is matched within the prepared classes in the database Ziegler, Lausen, and Schmidt-Thieme, 2004; Isinkaye, Folajimi, and Ojokoh, 2015.

- *Dense vector representations search*. At the stage of preparation, embedding is stored in some database. At runtime, the vector representation of the request is compared with all embedding in the dataset Mu, Yang, and Yan, 2019; Li et al., 2019.

- *Tree-based search*. This method does not require prior preparation of the dataset. The model contains vector representations in the form of a binary tree, in which the leaf nodes contain links to the specific documents Chang et al., 2021.

The above approaches are well-studied in eCommerce domains where the search subject is products and services. In our research, we suggest applying this knowledge to the procurement domain, where the subject of the search is a list of optimal suppliers, and information about the product is only a part of the search query.

2.3 Approaches to product classification

2.3.1 Fully supervised approach

The first attempts to build a general-purpose classifier based on United Nations Standard Products and Services Codes (UNSPSC) were made in the early 2000s. For example, the authors of GoldenBullet Ding et al., 2002 tried to create a tool that would help content managers do less routine work; the authors of another work developed the whole approach around the data of a particular procurement service Abbott and Watson, 2011. The project's goal was to accelerate the transition from custom taxonomy to UNSPSC. The authors of both papers note that one of the biggest problems was the lack and imbalance of data. Therefore, in addition, they are widely trying on Information retrieval technologies (Apache Lucene) and other more specific tools (jColibry) Abbott and Watson, 2011.

2.3.2 Self- and unsupervised approaches

Several approaches try to take into account the position of the word in the sentence and the "influence" of its direct neighbors. For example, word2vec can be trained as a shallow neural network that tries to guess direct word neighbors given a particular word in context (or vice versa predict the word in the context given its neighbors) Mikolov et al., 2013. The trained model can be used to map words to some space of predefined dimensionality. One of the downsides of using word2vec model is that it works strictly with the dictionary of the previously known words. So for some rare or derived words, there could be undefined embedding. The author of fastText, Bojanowski et al., 2017, tried to solve the problem of unknown words in dictionaries and dictionary length by using the letter n-grams, splitting, and vectorizing unknown words, or storing infrequent words in the dictionary as a set of n-grams. For some cases, Simple Word-Embedding-Based Models show close to state of the art (SOTA) results Shen et al., 2018. Alternative approaches such as Doc2Vec Le and Mikolov, 2014 suggest training in addition to encoding the sentence sequence number in the dataset by introducing the concept of document context into the input matrix.

2.3.3 General and domain specific SOTA

For most NLP tasks, de facto, standard models are built on the encoder-decoder architecture using multi-head attention. One of the most prominent techniques is BERT Devlin et al., 2018. For example, in the transformers library Wolf et al., 2020, the models are usually pre-trained on large datasets, and the end-user needs to fine-tune the model on their specific dataset. The trained model can be utilized as an end-to-end tool or an embedding model. The apparent advantages of these transformers are the following a) they capture more accurately the semantics of words through the mechanism of "attention" Vaswani et al., 2017 b) they can be used as a universal tool through the support of many languages.

RoBERTa by Liu et al., 2019 was pretrained with the Masked Language Modeling (MLM) objective. Taking a sentence, the model randomly masks 15% of the words on the input, then runs the entire masked sentence through the model and has to predict the masked words. Such procedure is different from traditional Recurrent Neural Networks (RNNs) that usually see the words sequentially or from autoregressive models like GPT, which internally mask the future tokens. It allows the model to learn a bidirectional representation of the sentence.

DistilBERT by Sanh et al., 2019 is one of the lightest transformers models, smaller and faster than BERT, which was pretrained on the same corpus in a self-supervised way, using the BERT base model as a teacher. It means it was pretrained as an automatic process to generate inputs and labels from those texts using the BERT base model. More precisely, it was pretrained with three objectives: MLM and Distillation loss/Cosine embedding loss. The latter two were used to align with the teacher model and generate hidden states and probabilities as close as possible to the BERT base model.

Besides the original transformers, several attempts were made by teams of large marketplaces to approach the extreme classification of products and services. For example, the Amazon Chang et al., 2021 team suggested considering the problem of searching for products given a request as a multilabel classification problem where each label corresponds to a particular product. They use a tree-based algorithm for semantic matching using XR-Linear (PECOS). Inference takes sublinear (log) time. A beam search algorithm is used to get b most relevant clusters (leaf nodes with products). The same team tried to use BERT Chang et al., 2019 for a similar problem and made an attempt to adopt transformer-based architectures for the XMC task. They were inspired by the information retrieval approach (index→match→rerank), where the BERT model improves only the matching step.

Researchers from Walmart Labs handle the product classification as a mapping task of new products coming to their platform with a particular node in taxonomy Sun et al., 2014. They named their solution Chimera, which combines the output of the models' prediction and human-crafted business rules for classification. In cases where the machine can not reliably determine the category, Chimera sends the task to crowdsourcing. If the latter is unable to make an unambiguous decision, system forwards the task to an in-house expert, who creates new categories or sets of new rules.

The task of classifying all goods can be considered an open-world task, which tries to include the definition of an unknown category in the model. In the recent paper Xu et al., 2019, researchers use a two-step approach also motivated by information retrieval techniques. The first step, which they call ranker, finds a list of top-k nearest examples for each seen class and forms clusters. The second step, meta-classifier, produces the probability that x belongs to the seen class c based on top-k examples from this class.

As we can see, there is no single right way to solve the product classification problem with an extreme number of classes, especially due to the fact that many researchers consider it in different ways Tsagkias et al., 2021. Most papers describe a stack of several models and business applications that work well in certain conditions and with specific business data. So we are inspired by the diverse ideas of the industry leaders and understand that we can get most answers from the deep understanding of data we possess.

2.3.4 Different approaches to hierarchical classification

Many ways to exploiting the hierarchical structure of the target categories during classification procedures have been presented by Silla and Freitas, 2011, and Stein, Jaques, and Valiati, 2019 grouped them into three primary groupings:

1. *flat*: during the training and testing phases, ignores the hierarchy by flattening it to the level of leaf nodes - standard multi-class classification;
2. *global*: a single classifier while taking the hierarchy into account and may use a top-down strategy at the testing phase;
3. *local approaches*: uses the hierarchy structure to build classifiers using local information, i.e., only the data that belongs to a particular node is considered to learn one or many classification models per each node.
   (a) *local classifier per node* (LCN) a binary classifier for each node;
   (b) *local classifier per parent node* (LCPN) a multi-class classifier for each parent node plus root classifies;
   (c) *local classifier per level* (LCL) a multi-class classifier for the entire hierarchy level.

All systems constructed utilizing this local categorization technique employed a top-down strategy during the testing phase. They predict a class at the top level and then use that knowledge to forecast deeper beneath the candidates' nodes based on the previous step only in a recursive fashion until they reach a leaf node or meet the blocking conditions.

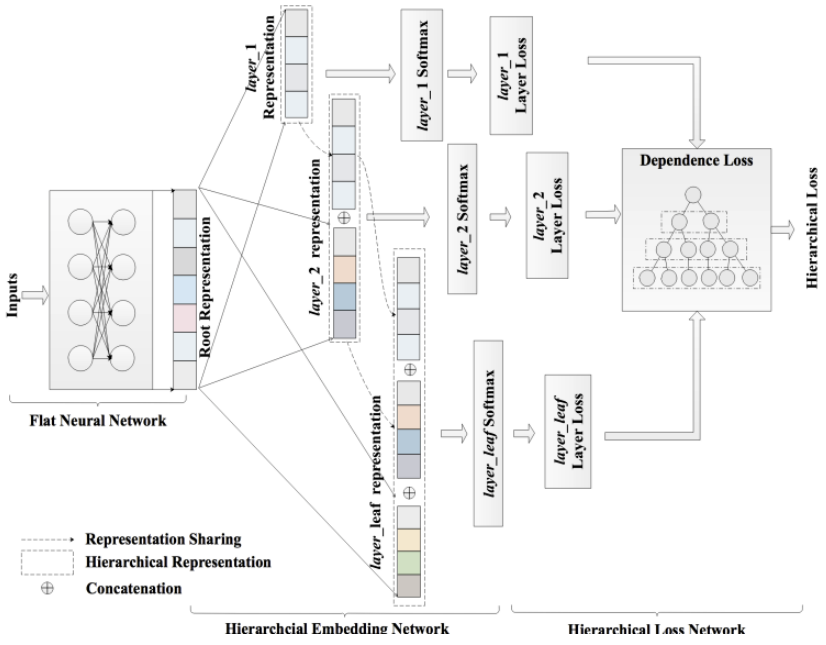

FIGURE 2.4: Deep Hierarchical Classification architecture by Gao et al., 2020

One of the earliest end-to-end hierarchical deep learning approaches was introduced by Gao et al., 2020. The authors proposed a neural network that is composed

of three steps. The first one, the Flat Neural network, giver root representation produces level representations in Hierarchical embedding network. Level representations are the concatenations of current level representation with the parent representations. Representations, constructed in such a way, transformed by the softmax function are used to calculate the hierarchical loss.

Rhinobird team(Yang et al., 2020), the winning team of the SWC2020MWPD challenge, used the Transformers Bert model embedding step and applied Dynamic Masked Softmax to deal with hierarchy.

The contest organizers built upon this idea and proposed an RNN head that sequentially predicts a node for each level in the product hierarchy, achieving slightly better results with simpler model architecture.

They used encoded by transformer's [CLS] token embedding. Inside the classification head, they concatenated it with a hidden state. Based on the concatenated matrix, a linear layer predicts the first level in the product hierarchy. A second linear layer updates the hidden state. The updated hidden state is fed back into the RNN to predict the next level in the product hierarchy. This procedure is repeated in a loop until it reaches the bottom-most level. Generally speaking, the use of *global* approach to building a hierarchical classifier with the intention to teach the model to distinguish relations given the hidden state of the previous level.

Chapter 3

Dataset

3.1 Overview of existing datasets

The problem of text classification is one of the main ones in NLP. Many datasets are de facto standards for building models and comparing quality. However, the specific nature of our work imposes certain limitations:

1. *Syntactic differences of product titles*. Most textual information consists of product titles, which are nouns and adjectives. Thus, full sentences are not represented in most datasets.

2. *Asymmetry*. Suppliers describe goods in more detail, and buyers are limited to the abbreviated name.

3. *Many data providers*. One dataset combines products from different data providers.

4. *Labeling with open standard*. The taxonomy of product classes used for labeling the dataset should be open and widely spread.

Over the past couple of years, we can see the rise of product-based datasets Tsagkias et al., 2021. For example, the WDC Product Data Corpus by Primpeli, Peeters, and Bizer, 2019 is compiled based on open data from the Common Crawl [1] project. It contains information from many providers, but there is no single taxonomy that can be used as a label, so it cannot be utilized for our classification task. The same research group Zhang et al., 2020 prepared a handcrafted dataset for the MWPD2020 challenge labeled with GS1[2] taxonomy, which contains good properties, but the coverage of classes is insufficient for our task. Icecat Dataset [3] has a professionally curated catalog of products labeled with UNSPSC taxonomy, whereas it primarily focuses on a narrow domain of consumer electronics and IT hardware products. Lin, Das, and Datta, 2018 overview another dataset submitted by the Rakuten data challenge 2018, which was collected based on data only from one provider, and was labeled with their proprietary taxonomy.

Thus, to the best of our knowledge, there is no such dataset that could be used to address our particular requirements. For this purpose, we decided to create our own dataset, which is a combination of product descriptions from different suppliers and purchase order history data from different buyers. In this section, we will describe the process of collecting and preparing data that would satisfy the limitations mentioned above. Various taxonomies, their advantages and disadvantages, and the motivation for choosing the UNSPSC taxonomy are described in detail.

[1] https://commoncrawl.org/

[2] https://www.gs1.org/standards/gpc

[3] https://icecat.biz/en/menu/channelpartners/index.html

3.2 Data collection

At the initial stage of our research, the only available data was historical transaction information about spending uploaded by buyers to the Fairmarkit[4] platform. Such information was unified to one structure but remained heterogeneous by its nature, as different companies build procurement processes differently. For example, all buyers categorize their spending using different internal taxonomies, and its depth varies from the 1st to 4th level of them.

Historical procurement data of a buyer for a particular supplier contains only partial information about the products and services that were purchased in the past. Unfortunately, such historical data does not contain full information about all products that a supplier can offer. Not all suppliers are ready to prepare product catalogs with a unified structure for machine processing.

We decided to take a proactive approach and began collecting full catalog information from the supplier's website. As a proof of concept, we decided to analyze most popular suppliers' that were mentioned in buyer's spend data. Most of the selected suppliers are global companies with the main focus on North America and European Union. The products offered by selected suppliers fall into so-called Maintenance, Repair, And Operations(MRO) domain and are purchased by any type of buyers. Examples of such suppliers are given in dataset example table B.1.

While this collection of information should increase the visibility of the supplier for the buyers, it can also overload their servers with inefficient traffic. We decided to perform such data extraction in the least active business hours and make no more than one request per second. The broad scraping task is simplified due to the wide adoption of the Semantic Web approach Berners-Lee, Hendler, and Lassila, 2001; Guha, Brickley, and Macbeth, 2016. We followed best practices of extracting data from web resources and build our data ingestion pipeline as two steps flow:

1. **Crawler**. As an input, the crawler takes a list of domains, explores them, and collects a list of pages from the sitemaps. Then it tries to "understand" if the given domain is an eCommerce resource, and if so, it transfers all the pages to the *Scraper*.

2. **Scraper**. A scraper downloads each page from the given list and tries to get different types of meta-information on pages (RFDa, JSON-LD, etc.) to obtain high-quality structured product information.

Data Source	Amount	Labeled with UNSPSC taxonomy	Labeled with specific taxonomies
Supplier's catalogs	7 537 458	30.1%	94.9%
Buyer's historical data	13 970 232	21.0%	36.1%
All sources	21 507 690	24.4%	56.7%

TABLE 3.1: Data sources and the amount of labeled data.

As we can see from Table 3.1, only a small subset of data is labeled with UNSPSC codes. This subset could be used for training. However, most of the data is either unlabeled(75.6%) or labeled with provider-specific taxonomy.

[4]Tail spend optimization platform https://www.fairmarkit.com/

3.3 Diversity of product information

The main two parties in the procurement process, a supplier and a buyer, provide product data with different levels of detalization. This data serves different purposes for both parties. The main goal of the buyer's procurement department is to express their needs, given the data they possess. As we can see from table 3.2 in some cases, a buyer may not know the specific name of the product or all its characteristics. For example, *1 GALLON OF OIL PAINT* is pretty generic, but it clearly indicates two main properties that are important for the buyer - volume and type of paint. In addition to unstructured information about the product title, in rare cases, buyers may have information about the serial number, SKU, model, or specific manufacturer. Statistics on the occupancy of such fields in our dataset are presented in table 3.2. As a rule, the need is not limited to one product, and several products may be listed in a document request (RFQ).

Field	# records populated,% median length	Examples
Title	**13 970 149** 100% 37	Apple USB-C to Lightning Cable KALE LEAVES FROZEN 1 GALLON OF OIL PAINT.
Category	**5 041 067** 36.1% 23	extr/mix/pump/pell - extruder, mixers, melt pumps and pelletizers recruit:preemplscrn : recruitment hvac and refrigeration : gen ind
Brand	**1 446 946** 10.3% 8	ULINE N/A MRC
Identifier	**421 935** 3.1% 9	S-1259BL UNKNOWN 2474-XXXX

TABLE 3.2: Buyer product data

A supplier possesses more comprehensive information about the catalog of the products sold and a full list of its characteristics. Product title is used for marketing purposes, e.i. a supplier tries to provide maximum information while being limited by title length constraints. In our dataset, the average length of the title field is 43 characters which is aligned with reported statistics in similar datasets Zhang et al., 2020. The table 3.3 represents the main product fields statistics collected from suppliers' websites. As an example, we provide supplier product data that might be relevant for buyers *1 GALLON OF OIL PAINT* request.

As we can see from the example, the title is a composite of leaf-level supplier category, brand, and key properties. The main difference in suppliers' product data is additional fields - properties, photos, and full descriptions. *Properties* field contains semistructured key-value map. Full description is a full text that describes the main product properties mixed with autogenerated marketing text. The full description along with properties fields are the least represented ones in our dataset, so we decided not to use them in our final dataset.

We also decided to exclude fields that contain unique identifiers (such as serial numbers, models, and product codes) from our final dataset. In most cases, such information is presented in a product title and generates tokens with a low frequency.

Field	# records populated, %	Example
Title	**7 537 456** 100%	BEHR 1 gal. White Oil-Base Semi-Gloss Enamel Paint
Category	**7 149 848** 94.9%	Paint > Paint Colors
Brand	**4 007 524** 53.2%	BEHR
Identifier	**6 676 880** 88.6%	380001
Properties	**2 505 923** 33.2%	Approximate Coverage (sq. ft.) : 400 Base Material : Oil Based Color Family : White Hexadecimal Value : F8F9F5 Number of coats recommended : 2 Sheen : Semi-Gloss Surface Material Use : Brick,Drywall,Metal,Plaster,Stucco,Wood Transparency: Solid
Full description	**3552842** 47.1%	BEHR Oil-Base Semi-Gloss Enamel is formulated for easy application with a roller, sprayer or brush. It is ideal for application on siding and trim. This mildew-resistant formula will help protect both interior and exterior surfaces from scuffs, rust and household chemicals. Ideal for metal and wood doors, trim and cabinetry Excellent flow and leveling Durable hard finish ...
Image	**6 368 495** 84.5%	-

TABLE 3.3: Supplier product data

Wirojwatanakul and Wangperawong, 2019 during the modeling phase have successfully used product image embedding and applied different fusion mechanisms to combine data of different modalities. We decided not to use images for training as, in most cases, we do not have images during the evaluation phase.

The only required field used in the model training process is the Product Name. A significant part of the dataset is unlabeled, and we try to use this data during an unsupervised step of our pipeline in later experiments in section 4.7.

3.4 Data preprocessing

The resulting dataset has been collected from over 36 data providers. We manually reviewed the primary data providers, and for specific cases, we applied provider-specific transformations. Nevertheless, in most cases, the transformation pipeline looked the same:

1. Filter out serial numbers and codes from titles with regular expressions
2. Cleaning Personally Identifiable Information.
3. Clear short titles(less than 2 token) and small tokens(less than 3 characters)
4. Filter out non English titles

Field	Buyer Data	Seller Data	Data type	# records populated, %
Title	+	+	string(250)	**3872954** 100%
Category	+	+	string/categorical	**3606278** 93.3%
Full description	+	-	string	**895426** 23.1%
Code	+	+	string(8)	**3872954** 100%

TABLE 3.4: Resulting labeled dataset structure

We applied deduplication to the subset of Title and Code fields. Such steps have significantly improved data quality in our base. Table 3.4 shows a labeled subset of the data after transformation and deduplication. Data contained in Title and Code fields is the main input to the model. In most cases, the full path to the root node is given, but sometimes only a leaf class is provided. Effect of Category and Full description fields on model performance is described in Section 4.5.

3.5 Taxonomy depth selection

The choice of taxonomy motivated by the amount of labeled data in our dataset. However, in addition to choosing the proper taxonomy, it is also essential to choose the appropriate depth level to which the labels will be aligned. Samples labeled with high levels will be too generic. Labeling samples with lower taxonomy levels might be expensive in usage and understanding. Users will spend a lot of time distinguishing between differences in similar categories.

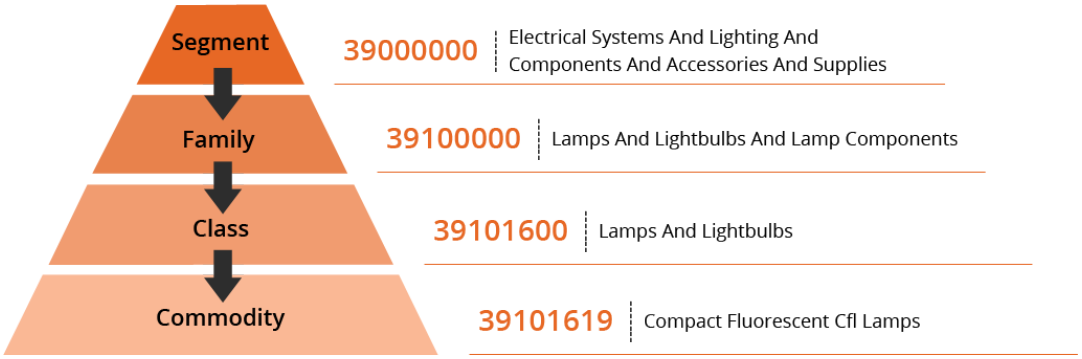

FIGURE 3.1: UNSPSC taxonomy levels example. SunTec, 2019

Analyzing the distribution of available data in Table 3.5, we can see that most of the data is labeled using the *Commodity* level. No data is lost when using the **Segment** level, but the model trained on such dataset will be too generic. We have empirically chosen the third taxonomy level as an optimal balance between data loss and class coverage. Our assumption is also confirmed by Payne and Dorn, 2011, who claim that the three-level taxonomy is the optimal choice for procurement tasks.

So at this stage of our dataset creation pipeline, we filtered out all records labeled explicitly with *Segment* or *Family* levels, amounting almost to 141K samples. We also translated all records of *Commodity* level to *Class* level. On the one hand, we have lost 5524 classes, but on the other, transition to parent level gives us a median of 96

Level	Samples		Classes		Samples in class				
	#	loss	#	loss	min	Q1	med	Q3	max
1. Segment	3872954	0	**57**	7074	143	2214	10192	56925	1011274
2. Family	3846989	25965	**378**	6753	1	43	414	2987	585555
3. Class	3732000	140954	**1607**	5524	1	10	96	637	420905
4. Commodity	3258773	614181	**7131**	0	1	4	24	140	276038

TABLE 3.5: Statistics of labeled sample in our dataset by level in UNSPSC taxonomy

samples per class compared to 24 while using the 4th level. This step allowed us to obtain a unified dataset labeled with the third level of UNSPSC taxonomy, which consists of 3,732K samples labeled with 1607 classes.

3.6 Data processing pipeline

Schematically, all the steps for creating a dataset described in this section, which will be used for further experiments, are shown in Figure 3.2.

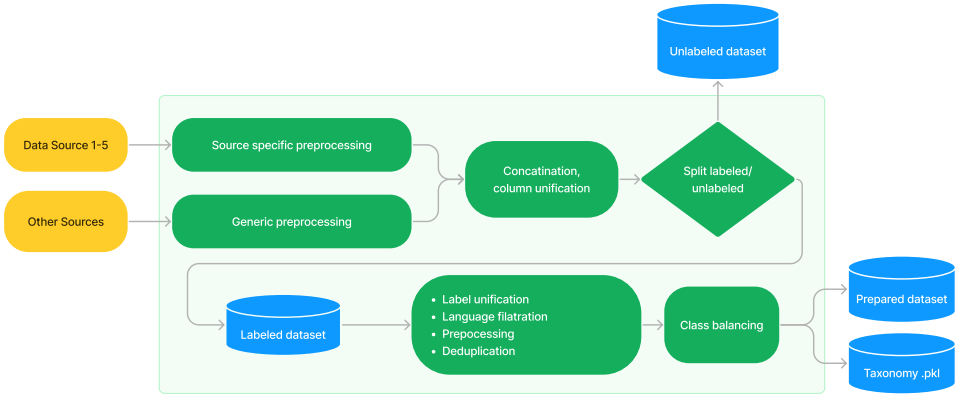

FIGURE 3.2: Dataset creation pipeline.

It is worth noting that, by design, our pipeline provides the functionality of switching between labelling taxonomies. Our dataset consists of two files: the flat prepared dataset file labelled with the leaf level nodes of taxonomy, and, additionally, we store a file that contains a graph-based structure(tree) that represents the relation between parent nodes in taxonomy. Taxonomy file is a serialized Python wrapper class over the DiGraph class of NetworkX [5] library Hagberg, Swart, and S Chult, 2008. The taxonomy class in our implementation serves two main purposes. It helps to calculate hierarchical metrics, and it is also used to train hierarchical versions of classifiers that are presented in Section 4.6.

[5] https://networkx.org/

3.7 Final dataset

After applying balancing as described in section 4.2, we partially lost data (33%) and could cover fewer classes (713), but on the other hand, we achieved better quality.

Field	Buyer Data	Seller Data	Data type	# records populated, %
Title	+	+	string(250)	**701850** 100%
Category	+	+	string/categorical	**701850** 100%
Full description	+	-	string	**321046** 45.7%
Code	+	+	string(8)	**701850** 100%

TABLE 3.6: Final dataset after deduplication and balancing

Statistics on the final dataset are shown in Table 3.6.

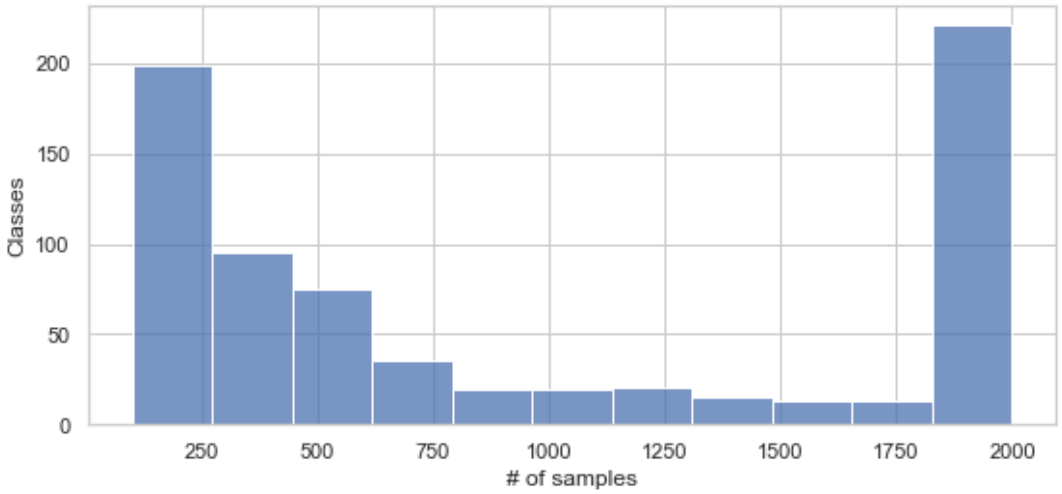

FIGURE 3.3: The distribution of samples per class in the final dataset.

The distribution of samples per class in the final dataset is presented in Figure 3.3. We can see that the long tail of 205 overrepresented classes is limited to 2000 samples. Sample of final dataset is given in B.1.

Chapter 4

Experiments

In the beginnig of this chapter, we describe the metrics we collect to compare model's quality in all of our experiments. We use various approaches trying to establish a good baseline.

In the last three sections, we try to consider the various properties of our data and take them into account when modeling. The following properties of our dataset did affect the choice of modeling techniques:

1. Multiple fields with product information in a different format (title, description, category).
2. Hierarchical structure of labels (product categories).
3. The presence of a significant part of unlabeled data.

In the following section, we will describe all the experiments aimed at incorporating those properties into our models.

4.1 Metrics

For all our experiments we report three metrics. The main metric, we use to compare model quality is macro $F_1(mF_1)$. In this metric, each class equally contributes to the final score. To understand whether the improvements is in influenced by better performance in dominant classes we report weighted $F_1(wF_1)$. Another additional metric we use is hierarchical $F_1(hF_1)$. This metric captures predictions on all levels of taxonomy and could explain the best path in the graph with the classifier.

The most widely spread metrics for classification are:

Accuracy	A	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	P	$\frac{TP}{TP+FP}$
Recall	R	$\frac{TP}{TP+FN}$
F_1	F_1	$\frac{2 \cdot P \cdot R}{P+R}$

For our imbalanced datasets, reporting of F_1, which is the harmonic mean of precision and recall, makes sense. In our multi-class imbalanced setting, the *macro-averaging* is used to understand how well the classifier can distinguish between different classes. Marco-averaging compares each class separately with the other classes, treating them as one.

$$mP = \frac{\sum_i^{|C|} P_i}{|C|}; \qquad mR = \frac{\sum_i^{|C|} R_i}{|C|}; \qquad mF1 = \frac{2 \cdot mP \cdot mR}{mP + mR}; \tag{4.1}$$

where,

C - set of classes.

Additionally, we will report *weighted* average F1:

$$wP = \frac{\sum_i^{|C|} P_i \cdot n_i}{\sum_i n_i}; \qquad wP = \frac{\sum_i^{|C|} R_i \cdot n_i}{\sum_i n_i}; \qquad wF1 = \frac{2 \cdot wP \cdot wR}{wP + wR}; \tag{4.2}$$

where,

C - set of classes, n_i - count of samples in class.

The above metrics do not consider the node level to measure the misclassification impact, Kiritchenko, Matwin, Famili, et al., 2005 propose an approach that extends the traditional precision and recall. Instead of taking account of only the real and predicted leaf nodes, their measures augment the objects considering that each tuple belongs to all ancestors of the class it has been assigned to, except for the root node. The authors call these measures hierarchical precision (hP) and hierarchical recall (hR), which are suitable to calculate a hierarchical F_1 measure (hF_1) as defined in equations 4.3. Stein, Jaques, and Valiati, 2019 compare hierarchical measure based on the sets of nodes in the path with the one that uses the least common ancestor proposed by Kosmopoulos et al., 2015. Nevertheless, for our case, with a single label on leaf level and tree-based category, both *hierarchical* measures can be simplified to:

$$hP = \frac{\sum_i |A_i \cap \hat{A}_i|}{\sum_i |\hat{A}_i|}; \qquad hR = \frac{\sum_i |A_i \cap \hat{A}_i|}{\sum_i |A_i|}; \qquad hF1 = \frac{2 \cdot hP \cdot hR}{hP + hR}; \tag{4.3}$$

where,

$\hat{A}_i$ - the set of all ancestors nodes, including label node, predicted,

A_i - the set of all ancestors nodes, including label node, real.

Table 4.1 is the result of the prediction of a hierarchical model on some test data. As in UNSPSC, code hierarchy is encoded considering code as a set of nodes ('12183600' is a set of '12', '1218', '121836'). For simplicity, we provide only code in the table. The root node of the category tree is not used in hierarchical metrics.

y_i	$\hat{y}_i$	A_i	$\hat{A}_i$	$A_i \cap \hat{A}_i$
12234500	12234500	12, 1223, 122345	12, 1223, 122345	12, 1223, 122345
16310800	16120800	16, 1631, 163108	16, 1612, 161208	16
48011400	31010000	48, 4801, 480114	31, 3101	$\emptyset$
$\lvert\sum\rvert$		9	8	4

TABLE 4.1: Calculation of hierarchical F_1 score.

Thus, we received the following results:

$$hP = \frac{4}{8}; \qquad hR = \frac{4}{9}; \qquad hF1 = \frac{8}{17};$$

Our experiments will operate both plain and hierarchical versions of classifier models, so both types of metrics will be reported.

4.2 Class imbalance

At first glance, our dataset has significant coverage of 1607 distinct classes. However, not all of this information can be used for model training. The long tail distribution of elements with a significant number of classes with less than ten samples makes it almost impossible to use it in training. In fact, we observed the Pareto rule in action,

where 20% of a class covers 80% of the data. We think that the main prerequisites for such imbalance are:

1. *Not all types of products are widely sold on the Internet*. The top of the most frequent categories in our dataset 43211500 *Computers* are widely spread in eShops.

2. *Categories that have a limited set of products*. For example, the number of unique products in the 11101700 *Base Metals* category will not frequently change over time. In our dataset, there are 85 samples, and most of them are approximate duplicates.

3. *Mistakes*. Mistakes made at the data input stage lead to the emergence of categories with a small number of elements. While manually checking some classes with a single sample, we identified that some of them do not correspond to the chosen class.

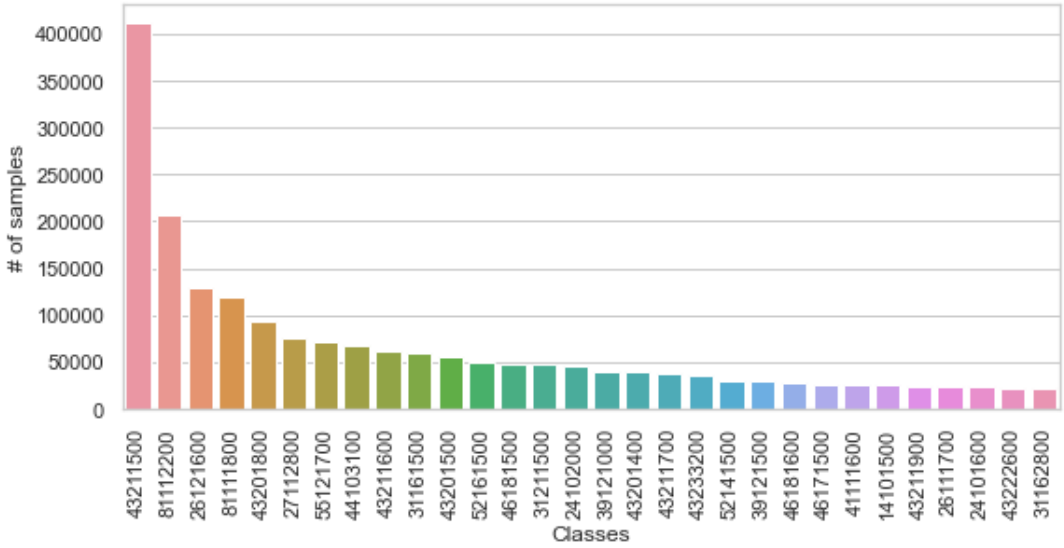

FIGURE 4.1: Distribution of top 30 classes.

For most machine learning algorithms, imbalances of dataset classes lead to worse results Batista, Prati, and Monard, 2004. A number of data and algorithmic solutions to the class-imbalance problem have already been suggested and aggregated by Kotsiantis, Kanellopoulos, Pintelas, et al., 2006. At the data level, these solutions include: random oversampling with replacement, random undersampling, directed oversampling (no new samples are created, the choice of samples to replace is informed rather than random), directed undersampling (the choice of examples to eliminate is informed), oversampling with informed generation of new samples, and combinations of the above techniques. At the algorithmic level, solutions include adjusting the various classes' costs to counter the class imbalance.

4.2.1 Data approach to class imbalance

Data augmentation for a text can be pretty tricky, and one could achieve great results dealing with text only when such information is in the form of vectors Padurariu and

Breaban, 2019. Therefore, we leave the oversampling experiments for further research. As for our dataset, we focused on the undersampling techniques. For convenience. we have used imblearn Python package [1] presented by Lemaître, Nogueira, and Aridas, 2017. The Random Undersampler strategy was chosen for our pipeline.

Name	Number of		Samples in class					Metrics	
	Samples	Classes	min	Q1	med	Q3	max	mF_1	wF_1
Initial dataset	3814K	1607	1	10	96	637	420K	-	-
No upper bound *L 100;*	2584K	713	101	251	295	2802	282K	73.5	89.3
Fully balanced *L 2000; U 2000;*	412K	206	2000	2000	2000	2000	2000	88.3	88.3
Soft Capping *L 100; q0.95 + 10%;*	1832K	713	101	251	595	2802	41K	79.7	88.4
Hard Capping *L 100; U 2000;*	682K	**713**	101	251	595	2000	2000	**80.8**	83.9

TABLE 4.2: Comparation of different undersampling techniques applied to our dataset

We compared the results of different oversampling techniques from Table 3.5. We trained FastText model with default hyperparameters for 50 epochs for each derived dataset. At this particular stage, we mostly were focused on the combination of good class coverage and high *macro* F_1. Our initial dataset contained 15 classes with only one sample. So we empirically defined a lower boundary with at least 100 samples per class and *No upper bound*. We've attempted to create *Fully balanced* dataset, and despite the highest mF_1 score(88.3), we were not satisfied by class coverage of 206 - most of the data was lost.

For *Soft capping* we apply dynamic upper boundary as a 95th percentile on sample counts; additionally, for each class above, we add a 10% sample. On the contrary, for *Hard Capping* dataset, we set a constant upper bound and downsample all classes that were above it. Both methods produce datasets with 713 classes and last showed better by 1.1% result in mF_1. We will consider this dataset as the main one for our further experiments.

4.2.2 Algorithmic approaches to class imbalance

After balancing on data level in our dataset, a difference between a dominant class and the smallest one can reach 20x at the worst. So it also makes sense to tolerate data imbalance on the model level. The loss function is a hyperparameter for Facebook fastText model implementation. Thus, detailed experiments on hyperparameter typing are described in section 4.3. For most deep learning models from the transformer library (Wolf et al., 2020), the default loss function for the sequence classification head is a cross-entropy criterion. The most straightforward way to deal with class imbalances is by incorporating the inverse-frequency class weights w_c to penalize errors of dominant classes:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c} \tag{4.4}$$

[1] https://imbalanced-learn.org/

where x is the input, y is the target, w_c is the inverse of class frequency, C is the number of classes, and N spans the minibatch dimension

Another improvement to the weighted loss function was proposed by Cui et al., 2019. Authors introduce the concept of *Effective number of samples*; the main idea is that as the number of samples increases, the additional benefit of a newly added data point diminishes:

$$w_c = \frac{1}{E_{n_c}} \qquad E_{n_c} = \frac{1-\beta^{b_c}}{1-\beta} \qquad l_n = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c} \tag{4.5}$$

E_{n_c} is effective number of classes and w_c its inverse and β is hyperparameter (0.999 used in our case)

As we can see from Table 4.3 we have gained additional improvement where we used class weights. The inverse-frequency of class weighting provides better results with +1.8% improvement on $mF1$.

Model	mF_1	wF_1
RoBERTa + CE loss	79.7	83.7
RoBERTa + Inverse Frequency Weighted + CE loss	80.8	84.8
RoBERTa + Inverse Effective Number of Samples + CE loss	80.5	84.9

TABLE 4.3: Improvements on model performance, while using weighted loss function

After conducting experiments with class balancing, we are guided by the following rules to build the final version dataset:

1. Exclude all classes that contain less than 100 samples
2. Use all samples without changes from the classes with the amount ranging between 100 and 2000 samples per class.
3. For all classes with more than 2000 samples per class, apply hard capping to 2000 samples.
4. Consider algorithmic penalization of loss function for model architectures where it is applicable during further experiments.

4.3 Baseline classifiers

As shown in earlier work on the classification of products and services using UNSPSC Abbott and Watson, 2011; Ding et al., 2002, the basic model for that task was Naive Bayes probabilistic model. We decided to take this approach as the starting point for our research. In the same paper, the authors also used the third level of UNSPSC (class level) as a label. The method showed promising results on the initial dataset.

As an improvement, we suggested using word vector representation models: word2vec and fastText. This models generate embeddings on word level, so we join non-zero word vector by averaging as proposed in Joulin et al., 2016a for word2vec

generated embeddings. For fastText embeddings, motivated by the original implementation [2] we additionally apply normalization of word embeddings before averaging. On contrast doc2vec, which was designed to catch relations between words in the context of a sentence, producing *Paragraph vector* and makes results comparable with the above two methods.

We used the Gensim package Řehůřek and Sojka, 2010 for described above training of unsupervised models and wrapping them with a sklearn Buitinck et al., 2013 TransformationMixin. This approach helped us to simplify usage in the sklearn pipeline as a transformation step. Embeddings generated by the transformation were used to train simple classification models. We also used Logistic Regression and Random Forest classifiers. Additionally, we assumed that each embedding could capture different important title properties, and, for this reason, we decided to combine different embeddings pairwise.

Embedding Model / Classifier	mF_1	wF_1	hF_1
Naive Bayes Classifier	0.601	0.753	0.800
word2vec / Logistic Regression	0.593	0.710	0.780
doc2vec / Logistic Regression	0.350	0.441	0.467
fastText / Logistic Regression	**0.633**	0.752	0.789
word2vec + doc2vec / Logistic Regression	0.594	0.710	0.780
doc2vec + fastText / Logistic Regression	0.621	0.750	0.768
word2vec + fastText / Logistic Regression	0.631	0.768	0.796

TABLE 4.4: Experiments on the baseline classifiers.

Both classification of prepared word2vec and fastText embeddings showed results close to Naive Bayes. We can observe a significant difference between macro and weighted F_1 scores, up to 10 points, which means that model works better in predicting dominant classes. Any of the embedding combinations does not introduce significant improvements, and the classifier shows the result close to the best embedding of the combination Table 4.4.

Despite the fact that fastText showed the best results, there exists a main drawback of the presented approach. Embedding and classification are two separate models, and loss function errors from the classifier are not propagated to the embedding model. The original implementation tries of fastText to overcome this issue in *train supervised* mode. This mode introduces an additional fully-connected layer on top of the averaged word-level embeddings.

Hyperparameters	mF_1	wF_1	hF_1
hierarchical softmax	0.703	0.751	0.791
negative sampling	0.732	0.796	0.833
negative sampling/dim 200	0.743	0.803	0.839
softmax	0.805	0.837	0.864
softmax/dim 200	0.812	0.842	0.869
softmax/dim 100/ngrams 2	**0.834**	**0.868**	**0.891**

TABLE 4.5: Experiments on fastText supervised model.

We tried different settings of fastText supervised model, and in all cases, it performs better than the separate embedding transformation and classification model on top. The results obtained in this experiment are presented in Table 4.5. We have

[2] https://github.com/facebookresearch/fastText/blob/master/src/fasttext.cc

tested different loss functions, and softmax showed the best results in all cases. However, the training of the models using softmax was relatively more time-consuming, about 69 minutes, compared to the 3-4 minutes for hierarchical softmax approximation. Additionally, we grid searched over the dimensionality of latent space. By default model uses 100 dimensionalities. There is a slight improvement while changing to 200 and no change to model performance. On the other hand number of word n-gramms increases model performance by almost 3 points. We will refer to the best model from this experiment as the *baseline* for the following experiments.

4.4 Transformer-based classifiers

In the previous experiments, we tried to reproduce the historical development of different approaches used to text classification, especially product titles. At the current moment, the best result in most NPL tasks is achieved by the transformer-based model.

The transformer is an umbrella term that refers to the family of encoder-decoder models with multihead attention layers. In general case each model from transformer library consists of two parts: "body" - the language model which was pretrained on big dataset and top layer "head" that are designed to solve specific task (classification, sentiment analysis, named-entity recognition). We used the classification head with the default settings and changed different bodies to understand the best balance between train time and model performance.

Hyperparameters	F_1	wF_1	hF_1
Baseline	0.834	0.868	0.891
XLNet	0.645	0.747	0.799
DistilBERT	0.651	0.751	0.803
RoBERTa	0.804	0.847	0.876

TABLE 4.6: Experiments on the transformer-based model. A default classification head is used with a different body model.

All the models converged fast and were early stopped with patience in five sequential steps. We started with default hyperparameters. Due to hardware limitations, the batch size was set to 32 and changed to 16 for XLnet. We have tuned the learning rate. Table 4.6 shows the best results for each of the model underlying model bodies. As we can see, RoBERTa showed the best performance while remaining relatively fast to train. It took 4h 30m (2.5 epochs) on Tesla V100 graphical card with 16GB of RAM to train the best model.

4.5 Effect of using additional descriptive fields

There are multiple strategies for incorporating data of different modalities in a classifier. For example, fusion model architecture is proposed, where different layers encode each field, and the returned vectors are concatenated before passing to the final fully-connected layer Wirojwatanakul and Wangperawong, 2019. Nevertheless, we will take the most straightforward way for our experiments by concatenating different fields and their combinations with the title. While using RoBERTa model, input was trimmed to 500 tokens due to the input sequence limitation.

As we initially expected, the category field, which is populated for most samples, significantly boosted both models' performance. Unfortunately, product description

Model	Fields	mF_1	wF_1	hF_1
fastText	Title	0.834	0.868	0.891
fastText	Title+Cat	0.907	0.915	0.934
fastText	Title+Desc	0.829	0.859	0.893
fastText	Title+Cat+Desc	0.900	0.911	0.928
RoBERTa	Title	0.804	0.847	0.876
RoBERTa	Title+Cat	0.896	0.912	0.930
RoBERTa	Title+Desc	0.814	0.853	0.887
RoBERTa	Title+Cat+Desc	0.887	0.903	0.928

TABLE 4.7: Experiments on combination of different descriptive fields for both model types

in combination both with title and with title + category field has shown either minor degradation or no improvements. Therefore, we decided to use categories in addition to title fields for our further experiments.

4.6 Hierarchical label structure

Since our labels are organized into UNSPSC taxonomy, we can leverage such hierarchy during modeling and/or evaluation. In previous experiments we were using flat approach. In the current section, we will try to apply LCPN with fastText submodels and a global approach with Transformer models. Additionally, we will propose custom improvement over LCPN.

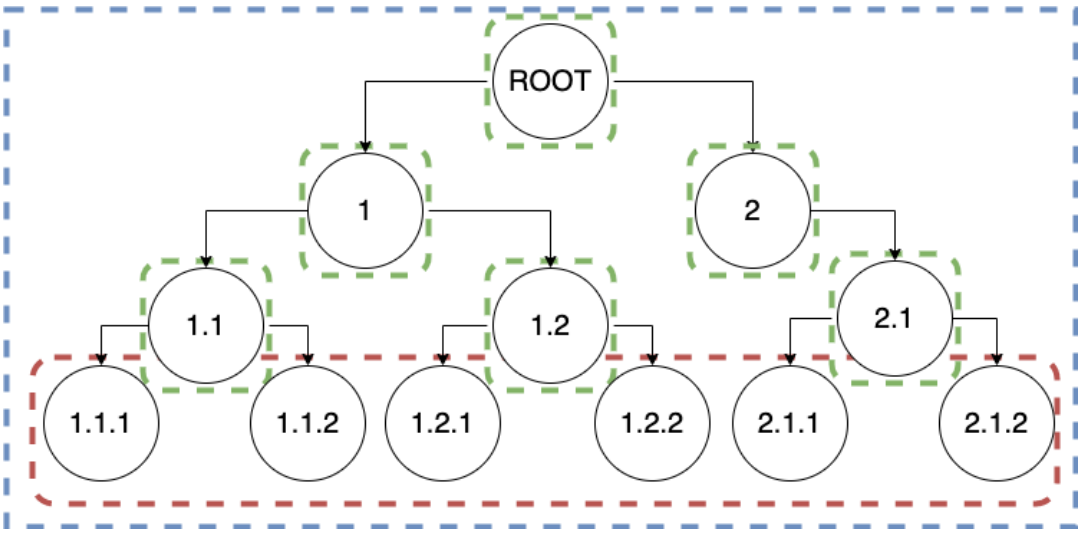

FIGURE 4.2: Approaches used for our experiments. Flat - one multi-class classifier - red line. Blue line denotes global approach which we use with Transformer models. And LCPN, green boxes, which are used with fastText models.

4.6.1 Improvement of local classifier per parent node

The quality of the classifier predictions is dependent on the number of classes the model should distinguish. Theoretically, UNSPSC could have 99 class per level at most, but there is only 22 at most in our dataset. For example, if we train the classifier

only on the first level of the label hierarchy, the accuracy could reach *0.93*. The most natural way to start is a local classifier per parent node. We need *K* classifiers in the vanilla setting, where K is the amount of parent nodes plus root. Despite the simplicity, we found two main problems with the application of LCPN to our dataset:

1. Multiplication of error. The deeper the taxonomy is, the more erroneous our classifier becomes, as the probability of unsuccessful prediction affects next-level prediction. For example, with 3 levels of depth with an accuracy of 0.93 at each level, we will get $(0.93)^3 \approx 0.804$ of the final prediction. So we should consider the tradeoff between the number of classes for a single classifier and the depth of prediction tree.

2. Hardware limitation. Modern models consume RAM at evaluation time, and, in the case of the LCPN approach, the result composite classifier will be linearly scaled by a factor of *K*.

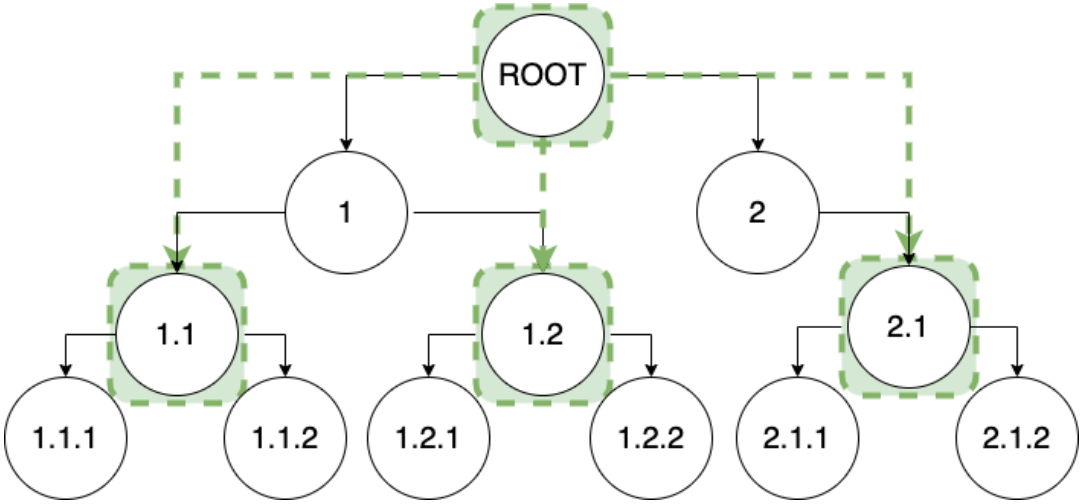

FIGURE 4.3: *Jumping local classifiers per parent node* are green boxes. Green arrows show the flow of prediction, omitting the second level of taxonomy.

To address the above two problems, we propose a new method, to which we will refer as **Jumping local classifier per parent node** (JLCPN). The main idea is to omit the middle levels of taxonomy and create a classifier for root and leaf parents level (second level in our case). As we can see from the figure 4.3 classifier's tree simplifies the depth of UNSPSC taxonomy. For our particular case, choosing the described above level is optimal. However, the proposed approach could also be generalized to dynamic search over the space of depth and class quantity per subclassifier, exposing them as hyperparameters of the hierarchical model.

JLCPN allowed us to reduce the composite model by 220 second-level subclassifiers. Nevertheless, even when applying this improvement, each subclassifier with optimal hyperparameters that were found in section 4.5 makes a 900 Mb memory footprint. It leads to a ≈50Gb footprint when the full composite model is fully loaded to the memory. It is a hardware constraint for us at the current moment, so we applied the *quantization* technique.

Joulin et al., 2016b adapted product quantization method to store word embeddings. They managed to circumvent quantization artifacts without a significant loss

in accuracy. They claim that their improvement decreases a memory footprint by two orders of magnitude than fastText while having the optimal balance between memory usage and accuracy.

Model/Hyperparameters	mF_1	wF_1	hF_1
fastText Flat *Cat	softmax/dim 100/ngrams 2*	0.907	0.915	0.934
fastText Flat *Cat	softmax/dim 100/ngrams 2	quantized*	0.897	0.901	0.924
fastText LCPN *Cat	softmax/dim 100/ngrams 2	quantized*	0.889	0.899	0.913
fastText JLCPN *Cat	softmax/dim 100/ngrams 2	quantized*	**0.917**	**0.926**	**0.940**

TABLE 4.8: Hierarchical LCPN models and quantization experiments

As we can see from Table 4.8 we started from the best result we achieved in the previous experiment, which we refer to as Flat. The quantized version of the Flat model had minor degradation in performance. We consider this as a reasonable cost for removing hardware constraints. The quantized model consumes 9x less RAM compared to the full version. Quantization allowed us to build a vanilla hierarchical LCPN version of the model. LCPN also showed performance degradation, and we can explain this by the Multiplication of Error mentioned above. Therefore, the best results were achieved by our JLCPN with more than 1% in mF_1 and 0.6% in hF_1. It is also worth mention that in out production setting, user can provide top level category. This prior knowledge allows us to omit root-level classifier and go straight to the corresponding subclassifier. The effect is hard to measure but we expect it to improve the overall quality of the prediction.

4.6.2 Hierarchical classification with transformer-based models

Theoretically, we could apply LCPN(or leaner JLCPN version) approach to any classifier model. Nevertheless, memory constraint is hard to deal with for transformer-based models. As a quick fix, we tried to use a single pretrained body of transformer that will be used as embedding transformation of inputs. Moreover, multiple classification heads will form the JLCPN hierarchical classifies. Such an approach has not shown any improvement and actually produced worse results than the flat version. It could be explained by the fact that the only trainable parameters were the head's fully-connected layer weights. Furthermore, the body remained in frozen mode. So, we decided to test RoBERTa with RNN head for our dataset.

Hyperparameters	mF_1	wF_1	hF_1
RoBERTa Flat	0.896	0.912	0.930
RoBERTa RNN head	0.910	0.922	**0.953**

TABLE 4.9: Experiments on RNN head of transformer-based model

A model with an RNN head allowed us to improve performance by 1.4 points in mF_1 over the flat version. Even though the current version model is slightly worse in mF_1(-0.7% compared to JLCPN), it showed the best hierarchical F_1 0.953.

4.7 Effect of Using unlabeled data

Only a part of the documents was labeled in our initial dataset, and an unlabeled subset did not participate in the training. One possible approach to using such an inhomogeneous dataset for our task is to divide it into two steps. At first, using all the available data in our dataset, we train an unsupervised (also called self-supervised) model that encodes words (make embeddings) in the most optimal way. In the second step, the classifier is trained on the embeddings for the data obtained for the labeled part.

In the next improvement step, we decided to incorporate knowledge of unlabeled product data in our pipeline. Each model or its implementation has its own way of the combination of supervised and unsupervised approaches.

Authors of fastText, Grave et al., 2018, created pretrained sets of vectors for 157 common languages, including English[3]. Their models were trained on publicly available Common Crawl and Wikipedia datasets. For our experiments, we have taken all product title information that we had in our dataset and trained unsurprisingly set of vectors with same dimensionality (300) and other setting as Mikolov et al., 2018 did.

Then we used both *our* and *open* sets of vectors as preloading a model before supervised-mode training. It is worth mentioning that we have not seen any examples of work that supports doing so. Many practitioners claim that any remaining influence of the original word-vectors may have diluted to nothing, as they were optimized for other tasks.

Furthermore, the goals of word-vector training are different in unsupervised mode (predicting neighbors) and supervised mode (predicting labels). However, a similar type of transfer learning is common for transformer-based models, so we decided to empirically try this approach.

Liu et al., 2019 claim that transformer-based models were originally trained on big datasets, including the ones mentioned above. The library itself provides a convenient way to train Language Models from scratch [4], but our dataset is relatively small, and the cost of training a full model is high (Sharir, Peleg, and Shoham, 2020). Domain-specific Language Modelling is used to overcome this issue. The pretrained model body is fitted to the domain using the same objective function as it was originally trained on. In the case of RoBERTa, MLM objective is a cross-entropy loss in predicting the masked tokens.

Hyperparameters	mF_1	wF_1	hF_1
RoBERTa Flat	0.896	0.912	0.930
RoBERTa MLM Flat	**0.903**	0.918	0.935
fastText Cat | dim 300	0.838	0.870	0.897
fastText Open | Cat | dim 300	0.839	0.870	0.898
fastText Our | Cat | dim 300	0.842	0.875	0.901

TABLE 4.10: Experiments on the impact of unlabeled data

All the models shown in Table 4.10 are trained in a flat (non-hierarchical manner); title and category fields have been used. For fastText models, dimensionality

[3] https://fasttext.cc/docs/en/crawl-vectors.html
[4] https://huggingface.co/blog/how-to-train

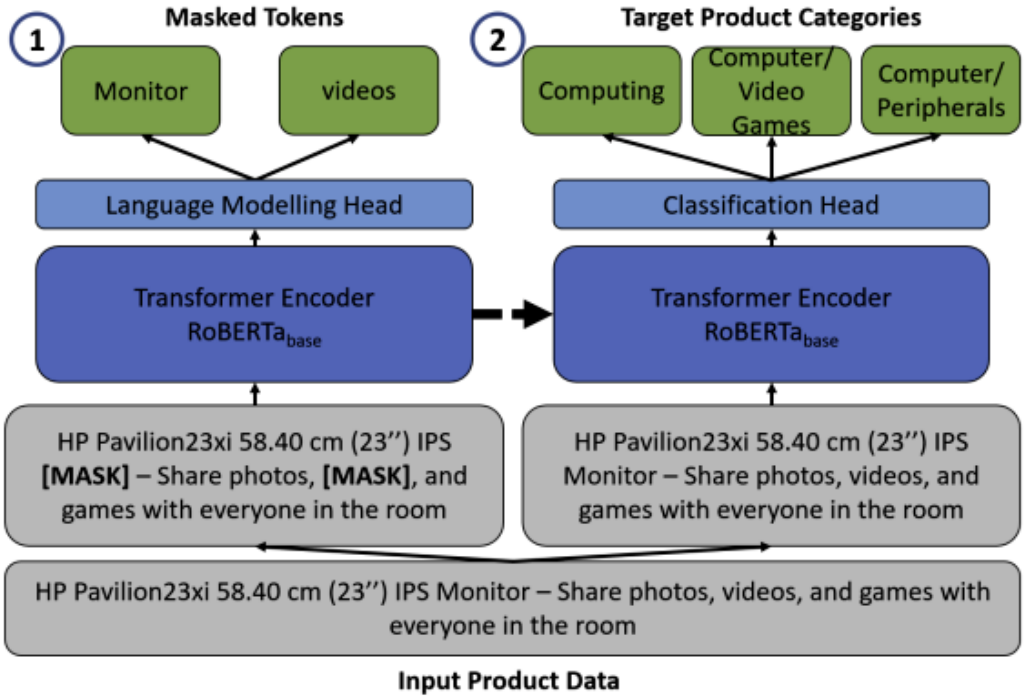

FIGURE 4.4: Overview of the (1) pretraining and (2) fine-tuning procedure, which combines a transformer model with a language modelling head first and afterwards with a task-specific classification head. Brinkmann and Bizer, 2021

300 was applied, and all other hyperparameters were set to default. Any of the pretrained vector sets used in the experiment has not changed metrics significantly. We can conclude that the effect of initial vectors is decaying during supervised training.

Similarly to Brinkmann and Bizer, 2021 we obtained small improvement to the baseline (only 0.7 point in mF_1) on domain-specific trained model body. Nevertheless, we believe that this approach could be improved by combining the hierarchical approach described above.

Chapter 5

Results

5.1 Future work

In our research, we tried to solve the applied problem of improving suppliers' recommendations (search) in terms of incomplete information about the catalog of products from suppliers. We have suggested that if there is no direct match of the title of the product in our historical records, then the best approximation may be suppliers that can offer products that belong to the same category as the products in the request. To achieve this, we decided to build the product classifier model and used the generally accepted taxonomy as labels.

Our research efforts were equally spread between dataset preparation and model selection steps. Our research was built sequentially: the next experiment was dependent on the results of the previous one. The main conclusions and suggestions for future work are as follows:

1. Information was collected from different data sources, and we have managed to unify it into a single format by identifying similar fields. However, some data source information remains too diverse. So we plan to train at least two classifiers separately for suppliers and buyers.

2. We applied a couple of undersampling strategies to overcome major disproportion in data classes. Nevertheless, we used rigid boundaries. Making such boundary search a part of the pipeline will also make sense. For some classes, it is also applicable to use oversampling approach.

3. In order to overcome the remaining but still significant imbalance after random undersampling, we have tested two weighted techniques. Both of them showed better results than the unweighted classification head. There are more robust algorithmic approaches that can be applied to our dataset.

4. Our pipeline handles the transition between different taxonomies used as labels by design. However, we have only used UNSPSC. As a future step, we propose using domain-specific taxonomy and building transition riles.

5. We have shown that fastText, being not a novice but an industry-proven solution, shows quite similar and sometimes better to SOTA model results. Models that require fewer resources and time to train are much faster to iterate on. We are planning to iterate more on transformer-based models in the future.

6. More data is not always the best choice. We concatenated the title field with category and description. Moreover, while usage of category data showed improvements, the usage of the description field showed degradation in performance. We suggest trying different separate embedding strategies for different fields and using a resulting vector in the classifier as a future improvement.

7. Among the different approaches to hierarchical classification, we have chosen LCPN. We proposed an improvement to the LCPN, omitting middle-level nodes, making the tree of the label taxonomy shallower. To the best of our knowledge, we are the first who apply such an approach. As a possible direction for future work, we offer to generalize this approach to any tree structure and maximum nodes per classifier as a hyperparameter of such a model.

8. Due to existing hardware limitations, the LCPN approach is not applicable to transformer-based models. We applied the RNN head to iterate over taxonomy layers. This method has not shown any significant improvements, so experiments with other architectures(GRU, LSTM) are a good choice for future work.

9. Last but not least, we have tried to use an unlabeled part of our data. Experiments showed minor improvements over the baseline. Nevertheless, we have not tested this model with other techniques described above due to the time limitations, and we leave it for future work.

5.2 Application of the classifier in supplier search

We've backtested the classifier on the main task - finding the best suppliers. We've tried two approaches how to combine search engine results and categories predicted by the classifier:

1. *Filtering*. In the first stage, all documents(product information attached to a particular supplier) in the database are filtered using the categories predicted by the classifier. This reduced subset is used by a search engine. Unfortunately, this approach showed worse results compared to a search on a full database. Considering specific cases, we concluded that the main reason is insufficient coverage, as the classifier did not cover most of the categories encountered in incorrect results.

2. *Boosting*. With this approach, the first stage calculates the search engine score for all documents. For those documents where there is a coincidence of categories within the query, the score is multiplied by a factor. Additionally, we cut off all predictions with a probability below 0.7. This approach helped to improve search results (we were mainly focused on recall@10).

For example, for the search phrase, WHITE BOARD CLEANER previous version of the system return a list with top whiteboard sellers. The classifier correctly classifies such request as cleaning and janitorial supplies, which lead to a more relevant list of suppliers.

5.3 Discussions

We set ourselves the ambitious goal of learning to classify all possible products and services. However, in reality, such a task turned out to be:

1. Impossible. Most of our research was spent collecting and improving the dataset. However, a significant number of strict assumptions were made and the resulting dataset covered only a small part of the underlying taxonomy.

2. Unpractical. We analyzed the use of the model and possible future usage scenarios, which significantly narrowed the model's scope.

Therefore, we decided to **create our own taxonomy**, which will connect suppliers with buyers and will supplement the information about the product they exchange. We plan to replace the UNSPSC taxonomy with our taxonomy.

Another drawback of our dataset was the large percentage of unlabeled data. We only used buyer and supplier categories representation as additional tokens concatenated with a title for our classifier. However, most samples contain a semistructured representation that can be transformed into a tree-based taxonomy. Our initial task of classifying goods turns into a taxonomy mapping, a much narrower task in our case. For example, for one source with 1.2m samples, we only need to map 400 supplier categories. With a little effort, most of the dataset can be labelled.

At the moment, the users do not influence the choice of categories. We plan to use humans in the loop and influence users' choice of categories as well as the evaluation of classifier performance. We believe that a crowdsourcing approach will help us get a sustainable flow of high-quality labeled data. We assume that we will be able to link most categories of buyer/supplier in the early stages of receiving such information in the data processing. Furthermore, such information could be just used as-is in the recommendation stage.

Appendix A

Relation between the received offers of bidders and savings

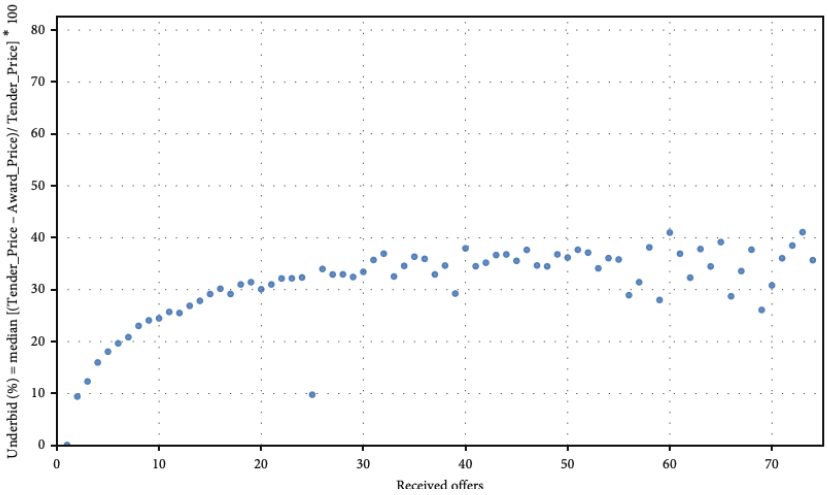

FIGURE A.1: Relation between the received offers of bidders and savings in Spain by García Rodríguez et al., 2020

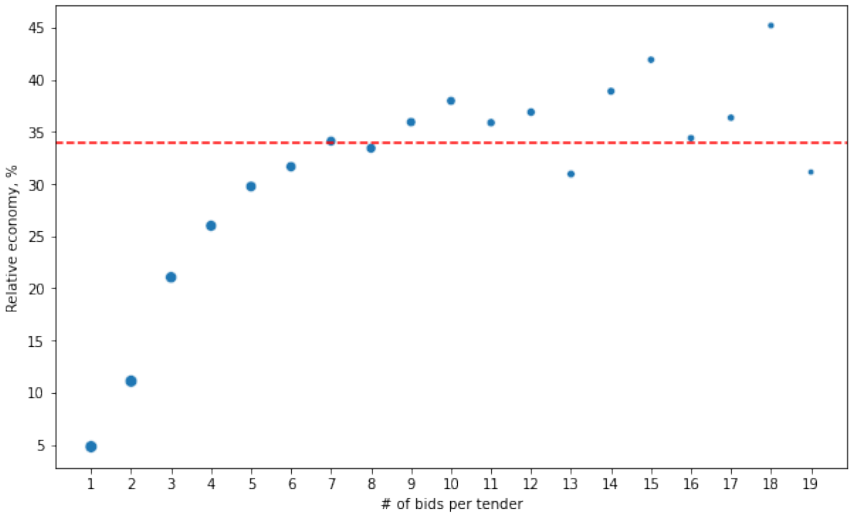

FIGURE A.2: Relation between the received offers of bidders and savings in Ukraine

Appendix B

Example of prepared dataset

Data Provider	Tite/Category	UNSPSC
Suppler's Catalogs		
beaed.com	Tag Coverall 10.5x6 Blk/Fl Grn Relief/Control Valve	55121503
globalindustrial.com	Aquaverve Touchless Cold Water Cooler, Silver W/ Black Trim Category: Water Coolers	48101710
grainger.com	8 in Work Boot, 10, EE, Men's, Black, Composite Toe Type, 1 PR Category: Safety-Toe Work Boots and Shoes	46181605
grainger.com	Transom Kit, Material: Steel, Overall Height: 10 ft Category: Wire Partition Components	30152001
graybar.com	Pendant Stations, Pushbutton, Yellow Category: Pendant Stations	39121561
graybar.com	C-Rail, 72 in. Long Category: DIN Rails	39121410
hdsupplysolutions.com	Sylvania® LED Bulb 10W A19, 2,700K, Package Of 6 Category: LED	39101628
hdsupplysolutions.com	Kwikset® Delta Satin Nickel Dummy Lever Category: Commercial Exit/Entry	46171503
lawsonproducts.com	Hex Cap Screw Grade 8 Category: Hex Cap Screws and Hex Bolts	31161501
shi.com	CTO5 400PD G6 MINI I3 8GB 500GB W10P 64 Category: Desktops and Workstations	43211507
shi.com	CTO MEDIATEK MT7921 WLS 6 +BT5.2 WLAN Category: Network Interface Cards	43201409
Buyer's historical data		
buyer 1	ULTRASONIC RAIL INSPECTION SERVICES ALL REVENUE TRACK, 2 TIMES A YEAR FOR THEE ± 145 MILES PER EACH TEST, PRICE PER MILE (FY24)	25000000
buyer 1	MLT, 24V	25000000
buyer 2	Techlog Product Owner 2021- Q3-Q4- Techlog Product Owner 01Jul2021 - 31Dec2021	81111504
buyer 2	Air King 30" 3 speed industrial oscillating fan #9175	27110000
buyer 3	Professional Services - Stack power cable	43221700
buyer 3	ASCO Rebuild Kit	27110000

TABLE B.1: Example of prepared dataset

Bibliography

Abbott, Alastair A and Ian Watson (2011). "Ontology-aided product classification: a nearest neighbour approach". In: *International Conference on Case-Based Reasoning*. Springer, pp. 348–362.

Aleksandrova, Marharyta (2017). "Matrix factorization and contrast analysis techniques for recommendation". PhD thesis. Université de Lorraine.

Batista, Gustavo EAPA, Ronaldo C Prati, and Maria Carolina Monard (2004). "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD explorations newsletter* 6.1, pp. 20–29.

Belkin, Nicholas J and W Bruce Croft (1992). "Information filtering and information retrieval: Two sides of the same coin?" In: *Communications of the ACM* 35.12, pp. 29–38.

Bensch, Stefan (2012). "Recommender systems for strategic procurement in value networks". In.

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001). "The semantic web". In: *Scientific american* 284.5, pp. 34–43.

Bojanowski, Piotr et al. (2017). "Enriching word vectors with subword information". In: *Transactions of the association for computational linguistics* 5, pp. 135–146.

Brinkmann, Alexander and Christian Bizer (2021). "Improving hierarchical product classification using domain-specific language modelling". In: *Proceedings of Workshop on Knowledge Management in e-Commerce*.

Buitinck, Lars et al. (2013). "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

Chang, Wei-Cheng et al. (2019). "X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers". In: *arXiv preprint arXiv:1905.02331*.

Chang, Wei-Cheng et al. (2021). "Extreme multi-label learning for semantic matching in product search". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2643–2651.

Chris Manning, Pandu Nayak (2018). *Introduction to Information Retrieval*. https://web.stanford.edu/class/cs276/handouts/lecture12-bm25etc.pdf. Accessed: 2022-06-17.

Cui, Yin et al. (2019). "Class-balanced loss based on effective number of samples". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Ding, Ying et al. (2002). "Goldenbullet: Automated classification of product data in e-commerce". In: *Proceedings of the 5th international conference on business information systems*. Citeseer.

Fazekas, Mihály (2019). "Single bidding and non-competitive tendering procedures in EU co-funded projects". In: *Brussels: European Commission, Directorate-General*

for Regional Policy. https://ec. europa. eu/regional_policy/en/information/publications/reports/2019/singlebidding and-non-competitive-tendering.

Gao, Dehong et al. (2020). "Deep hierarchical classification for category prediction in e-commerce system". In: *arXiv preprint arXiv:2005.06692*.

García Rodríguez, Manuel J et al. (2020). "Bidders recommender for public procurement auctions using machine learning: data analysis, algorithm, and case study with tenders from Spain". In: *Complexity* 2020.

Grave, Edouard et al. (2018). "Learning Word Vectors for 157 Languages". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Guha, Ramanathan V, Dan Brickley, and Steve Macbeth (2016). "Schema. org: evolution of structured data on the web". In: *Communications of the ACM* 59.2, pp. 44–51.

Hagberg, Aric, Pieter Swart, and Daniel S Chult (2008). *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Isinkaye, Folasade Olubusola, Yetunde O Folajimi, and Bolande Adefowoke Ojokoh (2015). "Recommendation systems: Principles, methods and evaluation". In: *Egyptian informatics journal* 16.3, pp. 261–273.

Joulin, Armand et al. (2016a). "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759*.

Joulin, Armand et al. (2016b). "FastText.zip: Compressing text classification models". In: *arXiv preprint arXiv:1612.03651*.

Karatzoglou, Alexandros et al. (2010). "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering". In: *Proceedings of the fourth ACM conference on Recommender systems*, pp. 79–86.

Kiritchenko, Svetlana, Stan Matwin, A Fazel Famili, et al. (2005). "Functional annotation of genes using hierarchical text categorization". In: *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*.

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix factorization techniques for recommender systems". In: *Computer* 42.8, pp. 30–37.

Kosmopoulos, Aris et al. (2015). "Evaluation measures for hierarchical classification: a unified view and novel approaches". In: *Data Mining and Knowledge Discovery* 29.3, pp. 820–865.

Kotsiantis, Sotiris, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. (2006). "Handling imbalanced datasets: A review". In: *GESTS international transactions on computer science and engineering* 30.1, pp. 25–36.

Le, Quoc and Tomas Mikolov (2014). "Distributed representations of sentences and documents". In: *International conference on machine learning*. PMLR, pp. 1188–1196.

Lemaître, Guillaume, Fernando Nogueira, and Christos K Aridas (2017). "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning". In: *The Journal of Machine Learning Research* 18.1, pp. 559–563.

Li, Jie et al. (2019). *The Design and Implementation of a Real Time Visual Search System on JD E-commerce Platform*. DOI: 10.48550/ARXIV.1908.07389. URL: https://arxiv.org/abs/1908.07389.

Lin, Yiu-Chang, Pradipto Das, and Ankur Datta (2018). "Overview of the sigir 2018 ecom rakuten data challenge". In: *eCOM@ SIGIR*.

Liu, Yinhan et al. (2019). "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692*.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26.

Mikolov, Tomas et al. (2018). "Advances in Pre-Training Distributed Word Representations". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mu, Cun, Binwei Yang, and Zheng Yan (2019). "An empirical comparison of FAISS and FENSHSES for nearest neighbor search in hamming space". In: *arXiv preprint arXiv:1906.10095*.

Padurariu, Cristian and Mihaela Elena Breaban (2019). "Dealing with data imbalance in text classification". In: *Procedia Computer Science* 159, pp. 736–745.

Payne, Joe and William R Dorn (2011). *Managing Indirect Spend: Enhancing Profitability Through Strategic Sourcing*. Vol. 557. John Wiley & Sons.

Primpeli, Anna, Ralph Peeters, and Christian Bizer (2019). "The WDC training dataset and gold standard for large-scale product matching". In: *Companion Proceedings of The 2019 World Wide Web Conference*, pp. 381–386.

Řehůřek, Radim and Petr Sojka (May 2010). "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, pp. 45–50.

Robertson, Stephen, Hugo Zaragoza, et al. (2009). "The probabilistic relevance framework: BM25 and beyond". In: *Foundations and Trends® in Information Retrieval* 3.4, pp. 333–389.

Sanh, Victor et al. (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108*.

Sarwar, Badrul et al. (2001). "Item-based collaborative filtering recommendation algorithms". In: *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295.

Sharir, Or, Barak Peleg, and Yoav Shoham (2020). "The cost of training nlp models: A concise overview". In: *arXiv preprint arXiv:2004.08900*.

Shen, Dinghan et al. (2018). "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms". In: *arXiv preprint arXiv:1805.09843*.

Silla, Carlos N and Alex A Freitas (2011). "A survey of hierarchical classification across different application domains". In: *Data Mining and Knowledge Discovery* 22.1, pp. 31–72.

Stein, Roger Alan, Patricia A Jaques, and Joao Francisco Valiati (2019). "An analysis of hierarchical text classification using word embeddings". In: *Information Sciences* 471, pp. 216–232.

Sun, Chong et al. (2014). "Chimera: Large-scale classification using machine learning, rules, and crowdsourcing". In: *Proceedings of the VLDB Endowment* 7.13, pp. 1529–1540.

SunTec (2019). *UNSPSC Data Classification Services*. https://www.suntecindia.com/unspsc-data-classification-services.html. Accessed: 2022-05-25.

Tsagkias, Manos et al. (2021). "Challenges and research opportunities in ecommerce search and recommendations". In: *ACM SIGIR Forum*. Vol. 54. ACM New York, NY, USA, pp. 1–23.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Wirojwatanakul, Pasawee and Artit Wangperawong (2019). "Multi-label product categorization using multi-modal fusion models". In: *arXiv preprint arXiv:1907.00420*.

Wolf, Thomas et al. (2020). "Transformers: State-of-the-art natural language processing". In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45.

Xu, Hu et al. (2019). "Open-world learning and application to product classification". In: *The World Wide Web Conference*, pp. 3413–3419.

Yang, Li et al. (2020). "Bert with Dynamic Masked Softmax and Pseudo Labeling for Hierarchical Product Classification." In: *MWPD@ ISWC*.

Zhang, Xuirui and Hengshan Wang (2005). "Study on recommender systems for business-to-business electronic commerce". In: *Communications of the IIMA* 5.4, p. 8.

Zhang, Ziqi et al. (2020). "MWPD2020: Semantic Web Challenge on Mining the Web of HTML-embedded Product Data." In: *MWPD@ ISWC*.

Ziegler, Cai-Nicolas, Georg Lausen, and Lars Schmidt-Thieme (2004). "Taxonomy-driven computation of product recommendations". In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pp. 406–415.