Towards Pipeline Construction for Product Matching Task

Olga Cherednichenko, Oksana Ivashchenko & Maryna Vovk

National Technical University »Kharkiv Polytecnic University« , Ukraine.
E-mail: olga.cherednichenko@khpi.edu.ua , oksana.ivashchenko@khpi.edu.ua, maryna.vovk@khpi.edu.ua

Abstract Online shopping services are commonplace for people around the world for acquisitions. The product offers on websites are mostly given with some text descriptions and are often accompanied by specification tables. The same product can be found in many different e-shops, but information about the offers of this product can vary greatly on different electronic platforms. The task of product description categorization is not new, but the solution heavily depends on the data set of product descriptions. The pipeline is the scheme of step-by-step data processing. The aim of the paper is to construct the pipeline, for product description categorization, which should be flexible. The pipeline should allow getting access at any step during data processing and modify it by adding new functions, changing activity order, and implementing particular methods. We propose an approach to construct a flexible pipeline for product matching. The case study is given and experimental results are discussed.

Keywords: product matching, pipeline, text mining, item categorization, software.

University of Maribor Press

DOI https://doi.org/10.18690/um.fov.3.2022.12
ISBN 978-961-286-583-2

1 Introduction

Today, online shopping services are commonplace for people around the world. The rapid development of e-commerce caused an increasing number of products sold online. Product on websites is given along with some text descriptions. Specification tables, i.e., HTML tables that contain such specifications as price, manufacturer, country of origin, etc. are often presented (Ristoski, 2018). The broad concept of product matching is related to the process of finding and buying goods. The same product can be found in many different e-shops, but information about the item can vary greatly. In addition, there are no global identifiers for such products, and product offers are often unrelated. Therefore, there is no easy way for consumers to find all the necessary information and the best prices for the goods they are looking for. To improve user interaction, there are many product aggregators that try to integrate and classify products from many e-shops and many different vendors (Ristoski, 2018).

2 Related works

The issue of item matching has been already studied by the researchers (Hoffmann, 2015; Kannan, 2011; Köpcke, 2009). The similarity function is proposed to compare items according to their attributes. This method presumes attribute categorization using a specific similarity function that processes this attribute (Hoffmann, 2015). The regular expressions are used for classification. The unstructured product offers could be matched using semantic processing of item descriptions (Kannan, 2011). Product matching sort out product offer deriving from great variety of e-commerce websites that concern to the same real-world product. The product matching is solved as a binary classification problem. Pair of product offers is compared if they describe the same/similar product or not. The available frameworks for item matching are analyzed (Köpcke, 2009) and the conclusion claims that all matching frameworks work offline, have drawbacks and don't give the general scheme to solve the studied task. Product offers are usually given on trading platforms as a textual description and are specification table (https://ir-ischool-uos.github.io), for example HTML tables provides the information about the price or the country of origin.

In order to solve this task, the set of semantic technologies is required. The technology should be adjusted to the product domain, product offer matching, and product taxonomy matching (https://ir-ischool-uos.github.io). Existing solutions use text manipulation and transformation, clustering and classification algorithms, but they lack specific structure definition for their data processing. A lot of researches focus on implementing a pipeline, which is customizable, flexible and robust at the same time as a solution for this issue.

There are developed pipelines, which could be used. SpaCy allows creating own pipelines consisting of reusable components (https://spacy.io). This includes spaCy's default tagger, parser and entity recognizer, but also your own custom processing functions. Scikit-learn pipeline (https://scikit-learn.org) can be used to chain multiple estimators into one. This is useful as there is often a fixed sequence of steps in processing the data, for example feature selection, normalization and classification. The use of mentioned above pipelines presumes that person can make some changes in code and has developer skills. But real managers who work in the sphere of e-commerce mostly are not specialized in programming. They need a tool for non-professional developer, but for average user, who can use existing blocks and libraries. At the same time proposed pipeline should be tailored to different languages, diverse product peculiarities. The developed pipeline should allow getting access at any step during data processing and modify it by adding new functions, changing activity order, and implementing particular methods.

Therefore, the aim of the given research is to improve the use of text analysis pipelines for ad-hoc product description mining for the product matching task.

3 Methods and Materials

Item matching deals with identifying product offers deriving from different websites or from different sellers that refer to the same real-world product. To study the problem of product matching it is necessary to work with datasets of product descriptions. Usually, this data is presented as text descriptions and images of goods.

The Web Data Commons project (http://webdatacommons.org) has released the publicly available product data corpus originating from e-shops on the Web. In our research, we use Shoes as a product category from the data corpus. The product data corpus is offered in JSON format. The following attributes are used for describing the product offers in the corpus: title, description, brand, price, the content of the specification tables, and category to which is the offer was assigned. The initial dataset is huge and can be considered as a special kind of text.

Text mining deals with issues of the inference of structured information from collections of unstructured input texts. Approaches that are applied in text mining requires task-specific text analysis processes that may consist of several interdependent steps. Usually, these processes are realized with text analysis pipelines (Wachsmuth, 2015). One of the main problems is text analysis pipelines are mostly constructed manually because their design requires expert knowledge. To work with the problem of product matching, methods for processing proposal descriptions are used can be divided into the following stages: preliminary data processing; breakdown and work with individual tokens; implementation of special stages of processing such as clustering, classification, etc. A certain structure is used and the data must be adjusted, for text data, it is the use of alphabetic characters, lowercase text, and so on. The pipeline is implemented of such processing steps.

To implement a pipeline, we should understand what it is, what it consists of, and how to implement it for use. In computing systems, pipeline means a logical queue of actions to which instructions for sequential data processing are transmitted. It is an organized process of storing, placing, and transmitting data in turn. The main feature of the pipeline is that the output of processing the previous function is the input to the next function.

In the given work we introduce the paradigm of the pipeline construction and execution for product matching task. The product matching task is worded following. Given a collection of product text descriptions A, process A in order to infer all information as a structured set of information types B. Depending on the task, the types in B may represent different semantic concepts, linguistic annotations, sets of tags, and similar.

The decomposition of a text analysis process into single steps is a prerequisite for identifying relevant information types. Usually, this decomposition and pipeline construction are made manually, which prevents the use of pipelines for special text such as product descriptions. Our research focuses on implementing a pipeline, which is customizable, flexible and robust at the same time as a solution for this issue.

The main idea is the improvement of product description processing. We are implementing a pipeline concept for data processing which includes: data preprocessing; filtering; block of main data processing functions (clustering; token evaluation; core of tags creation). The pipeline contains a data clustering step. We consider K-means and SVM methods, because of their common usage in such classes of tasks.

Due to NLP and machine learning algorithms working with numeric data we need vectorization of the textual data. Vector is a numeric representation of a text value that encodes its meaning. The most common vectorization approaches are Bag-of-words; TF-IDF; Word Embeddings (Jacob, 2018; Joulin, 2017; Mikolov, 2013).

We have chosen the Word Embeddings approach with the usage of the Word2Vec model because of its high-level accuracy compared to others. Figure 1 displays the processing steps (developed by authors) that we proposed in the pipeline, which can be configured.

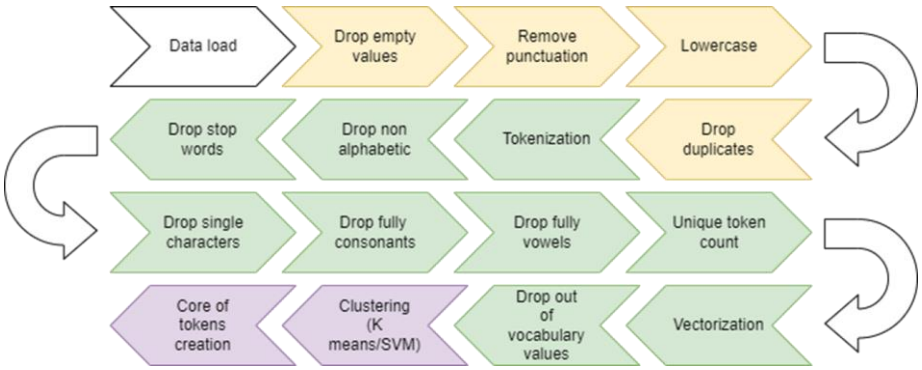

Figure 1: Proposed pipeline steps

Thus, we are going to perform experiments and analyze results. Different methods for clustering text data are used, generating a core of tokens according to clusters is performed.

4 Experiments

An experiment on the operation of the pipeline is conducted. WDC Product Data Corpus (http://webdatacommons.org) is considered. For the offer category in the research we have chosen shoes.

For implementation we use publicly available python libraries with tokenization/clustering algorithms:

- spaCy - NLP library for tokenization;
- gensim - ML tools;
- scikit-learn - ML tools library with pipeline concept.

The aim of the experiment is to test the work of the pipeline, using various methods for clustering text data, creating a core of tokens based on clusters and comparing their relevance indicators on trading platforms. The first step in the processing chain is to pre-process the dataset, which includes standardizing text data and filtering empty values. The results of the preliminary cleaning of the dataset is given in the table 1.

Table 1: The results of the preliminary cleaning of the dataset

Step name	Dataset size	Size difference
Initial dataset	664176	0
Empty values drop	551239	112937
Punctuation removal	551239	0
Lowercasing	551239	0
Duplicate values drop	392528	158711

The next step after cleaning and standardizing text values is to vectorize the data and split the data into atomic tokens. Functions for tokenization and vectorization of words using Tok2Vec and Word2Vec models, functions for token filtering and validation are implemented. The results of these tokens processing functions are given in the table 2.

After vectorization and filtering, the data should be clustered. Displayed (figures 2-4) pie charts show K means clustering for vectors from 3 vectorization models: spaCy Tok2Vec model with static vectors; Google trained Word2Vec model (GoogleNews-vectors-negative300); Custom trained Word2Vec model (WDC corpus). Each slice represents number of tokens in each cluster.

Table 2: Results of tokenization and filtering

Step name	Dataset size	Size difference
Initial dataset	3702438	0
Non alphabetic values drop	3490404	212034
Stop words drop	3257529	232875
Single characters drop	3055223	202306
Fully consonants values drop	2539365	515858
Fully vowels values drop	2525143	14222
Unique values count	100725	2424418
Vectorization	100725	0
OOV (out of vocabulary) values drop	44621	56104

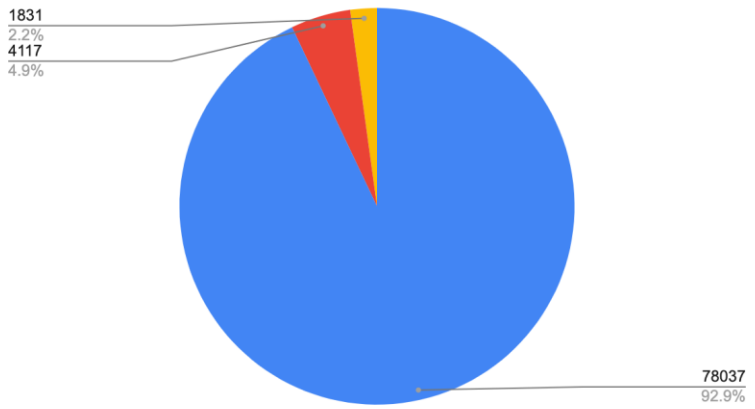

Figure 2: Custom trained Word2Vec model

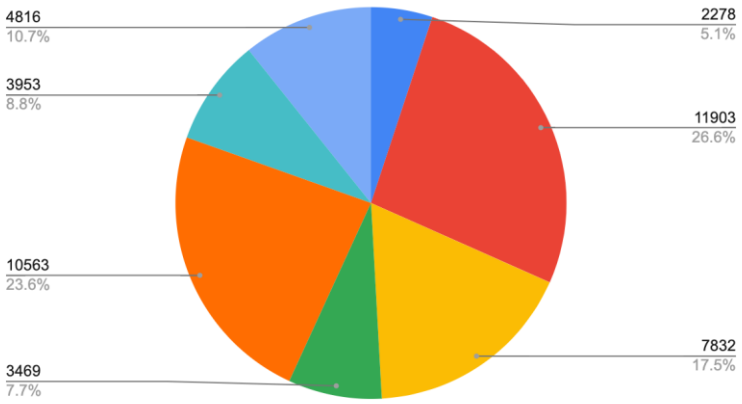

Figure 3: Google trained Word2Vec model

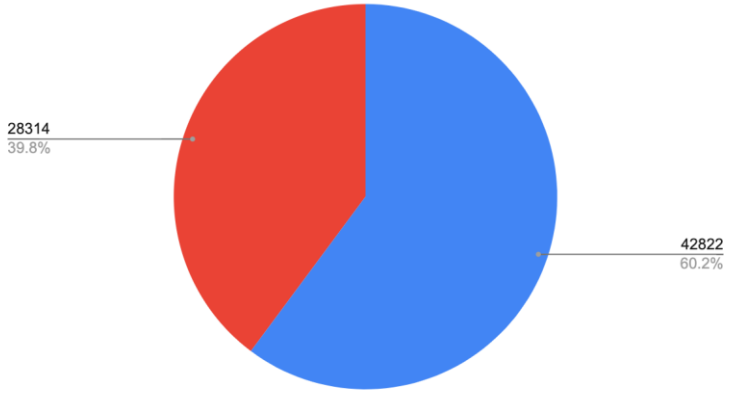

Figure 4: spaCy Tok2Vec model with static vectors

The results show that Pipline was able to cluster the results of the Tov2Vec vectorization model into only two clusters, so these results can not be used to train the SVM model. Instead, we use the results of clustering the Word2Vec model from Google, which split the data into 7 clusters, and our trained Word2Vec model, which split the data into 3 clusters. The results of SVM clustering show that the use of clusters from our trained model Word2Vec is not effective, because 1-2 clusters came out, and the use of Word2Vec trained Google gave results with 4-5 clusters.

The next task was to assemble token cores for each of the clusters. We will use only those results where more than 2 clusters were obtained - the results of clustering by the K method of average Word2Vec models from Google and our trained model, the results of SVM clustering based on clusters of the Word2Vec model from Google. Using different combinations of vectorization and clustering, we created cores of tags for each cluster. Tokens were compared with each other using semantic similarity within their cluster, tokens with highest results were assigned to the core.

In addition to the shoe category selected for the experiment, the developed pipeline can be used to process a variety of text data by adjusting the order of processing steps.

The experimental results show that the choice of particular algorithms for each step of the text processing pipeline is quite ad-hoc and requires the tool to create the adjustable and flexible pipeline.

5 Results

The main goal is to make the pipeline as flexible as possible and allow users to change it as they wish: add their own functions, set their own order of action, use other methods for data processing. The main action and goal of users or external systems that can use and integrate the implemented Pipeline module is to download data for processing and create a configuration file to perform Pipeline operations.

First of all, to process data, you need to be able to load it into memory and determine which block of data to work with. The data is defined in a table format and uploaded to the object for management. Next, you need to allow users to use a series of data processing features that will be used in the pipeline. More specific functional requirements for the implementation of the software component are given in table 3.

Table 3: Functional requirements for pipeline component implementation

Name	Description
REQ-1	Able to load a dataset into memory for work
REQ-2	Function for deleting special and punctuation characters that cannot be processed
REQ-3	Data standardization function - lowercase translation
REQ-4	"Blank" filtering function
REQ-5	Duplicate removal function
REQ-6	Data tokenization method
REQ-7	Function to remove "empty" tokens
REQ-8	Vectorization of tokens, deletion of invalid data
REQ-9	Token filtering function by length, must be full words, some characters must be deleted
REQ-10	The function of removing tokens that consist entirely of vowels or consonants
REQ-11	Unique token counting function
REQ-12	Function of clustering vectors into groups
REQ-13	Function of estimating semantic similarity of vectors
REQ-14	The function of creating a token kernel beyond the threshold of semantic similarity

The software implementation has the structure of a python library and modules that can be integrated into other applications. The implemented solution can be used as an internal or external component of such systems, the main objects of which are the implementation of a pipeline controller interacting with pre-processing processors, a token processor interacting with the vector model, and a main processor for data clustering. For the data transmission of these components, the internal Systems Network Architecture (SNA) is used - a proprietary network architecture with a full stack of protocols, designed to connect computers and their resources (https://www.ibm.com). Communication with the pipeline component does not take place via Internet protocols. Processing modules should be separated in the developed component: data pre-processing module; tokenization module; token processing module; main module of controllers; modules of additional functions, etc.

Pipeline implementation modules are written in Python, using clustering and vectorization libraries, as well as structuring and data transformation libraries: gensim, spaCy, sklearn, numpy, pandas. Both code libraries and individual script files containing a set of functions can be considered as a module. Modules allow you to organize related functions, classes, or any block of code in a single file. Thus, it is best practice to split large blocks of code into smaller modules, especially for large amounts of code for production-level projects in Data Science.

Pipeline implementation is strictly divided into the following modules and directories: 1) src - main folder with source code and functions; 2) base.py - the main file with the implementation of the pipeline class; 3) module of data processing functions - data_processing: a) pre-treatment module; b) token module; c) module of basic functions-handlers; 4) the config module, which is responsible for the validation of the config file; 5) auxiliary modules scripts, and utils.py; 6) directories for data storage - data, dumps.

The main components of the modules are the implemented classes. Classes provide a means of combining data and functions together. Creating a new class creates a new object type that allows you to create new instances of this type. Each instance of a class can have attributes added to it to maintain its state.

Pre-processing is the processing of "raw" data - dataset processing. In our case, it contains:

- loading the dataset into memory for work;
- cleaning of special symbols, numbers, punctuation marks;
- lower case transfer for further processing;
- delete empty values, null values;
- remove duplicates.

This pipeline step is the first, and its results are important for further processing. As mentioned earlier, a configuration file is used to use and change the order of processing functions. It has a mandatory preprocessing parameter, which specifies the procedure for processing text descriptions from a dataset.

Vectorization of tokens is one of the key aspects of pipeline development, because it is with the form of vectors that further actions are performed. For vectorization, several methods were considered, the modules of which are implemented in existing libraries: Doc2Vec from gensim; Tok2Vec from spaCy.

The following functionality should also be included in the pipeline: functions of counting the number of unique tokens; clustering by groups; comparison of tokens by semantic features; creating a token core.

The implemented pipeline is adaptive and by changing processing steps you can achieve the needed results to process the text data. The implemented pipeline is modular, it has library architecture. As a result, we got a pipeline that can be used in a variety of text processing evaluations. Customizable configuration allows building different adaptive scenarios. By adding necessary processing pipes we can improve our solution without changing the whole structure.

7 Discussion and Conclusion

Matching product offers is a complicated issue demanding refined and adapted entity resolution approaches. We proposed to build a flexible pipeline for product matching, as the other researches did (Nguyen, 2011). The valuable difference of our approach is the pipeline allow getting access at any step during data processing and modify it by adding new functions, changing activity order, and implementing particular methods. The results of the experiments showed that it is quite difficult to build and use generally applicable pipeline with preliminary defined clustering models. The trained model provides acceptable results only on similar datasets. As a result, the analysis and elaboration of the needs for the development of a software component are started, which allows processing the text data of product proposals for their classification. The main task of the proposed solution is to consistently perform data processing functions from users or third-party systems, thus validating and transforming the data into the necessary structure for further analysis and processing. The concept of a pipeline for processing and clustering text descriptions of product offers was developed. The implemented solution has a structure of modules and libraries written using the Python 3.9 programming language, which makes it easy to integrate into other systems. In future work, it is supposed to develop the automated tool to pipeline construction for the product matching task.

References

Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, (2017). "Bag of Tricks for Efficient Text Classification", Conference: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2

Devlin Jacob, Chang Ming-Wei, Lee Kenton, Toutanova Kristina, (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

Doc2Vec https://www.tutorialspoint.com/gensim/gensim_doc2vec_model.htm

Henning Wachsmuth (2015) Text Analysis Pipelines. Towards Ad-hoc Large-Scale Text Mining

Hoffmann, U., Silva, A., Carvalho, M. (2015) Finding Similar Products in E-commerce Sites Based on Attributes. Published in AMW

Kannan, A. (2011) Matching unstructured product offers to structured product specifications.KDD '11: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining pp. 404–412

Köpcke, H., Rahm, E. (2009) Frameworks for entity matching: A comparison, Data Knowl. Eng.

Nguyen, H. Fuxman, A. Paparizos, S. Freire, J. and Agrawal R. (2011) Synthesizing products for online catalogs. Proceedings of the VLDB Endowment, 4(7):409–418.

Ristoski P., Petrovski P., Mika P., Paulheim H. (2018) A Machine Learning Approach for Product Matching and Categorization. In: Yahoo Labs, London, UK

Robert C. Martin, (2018) Clean Code: A Handbook of Agile Software Craftsmanship 1st Edition

Scikit-learn Machine Learning in Python https://scikit-learn.org

Semantic Web Challenge ISWC2020 - GitHub Pages https://ir-ischool uos.github.io/mwpd/

SpaCy Industrial-Strength Natural Language Processing https://spacy.io/usage/processing-pipelines

Systems Network Architecture https://www.ibm.com/docs/en/zos-basic skills?topic=implementation-what-is-systems-network-architecture-sna

Tok2Vec https://spacy.io/api/tok2vec

Tomas Mikolov, et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781

WDC Product Data Corpus http://webdatacommons.org