Effective Products Categorization with Importance Scores and Morphological Analysis of the Titles

Leonidas Akritidis, Athanasios Fevgas, Panayiotis Bozanis

Department of Electrical and Computer Engineering

Data Structuring and Engineering Lab

University of Thessaly

The 30th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2018)

November 5-7, 2018, Volos, Greece

E-commerce

- Large growth rate:
  - E-commerce share of retail sales worldwide (2015): 7.4%.
  - E-commerce share of retail sales worldwide (2018): 11.9%.
  - Predicted E-commerce share of retail sales worldwide (2021): 17.5%[1]
- The related research problems have been rendered increasingly important.
- Effective and efficient management, processing, and mining of products data are examples of such problems.

[1]https://www.statista.com/statistics/534123/e-commerce-share-of-retailsales-worldwide/

Products Categorization

- One of the most important problems in the area.
- Given a product and a set of categories, it is required that we determine the category where the product belongs to.
- It leads to numerous novel applications:
  - Query expansion and rewriting.
  - Relevant/Similar products retrieval.
  - Personalized recommendations etc.

Attributes or not?

- Relevant work is divided into two categories:
  - those which are based on the products titles only and
  - those which take into consideration additional properties of the product (brand name, attributes, technical characteristics, etc).
- However, such metadata is not always present; even if it is present, it is frequently incomplete, ambiguous, inconsistent, or incorrect.
- The proposed method belongs to the first category and operates by accessing the titles only.

Theoretical Background

- Let Y be the set of all categories.
- The categories are usually organized into a tree structure with parent and leaf categories.
- A product can only be assigned one leaf category in the aforementioned tree.
- Each product is described by its title τ.
- The title has the words $(w_1, w_2, \ldots, w_{lx})$.

N-grams vs. words

- The proposed method performs morphological analysis of the titles and extracts n-grams of variable sizes.
- The reason of employing n-grams is that a n-gram is less ambiguous than single words.
- For instance, a brand name (e.g. *Apple*) may be correlated with multiple diverse categories (mobile phones, tablets, computers, etc).
- However, the bigram *Apple iPhone* is much more specific.

Tokens and Ambiguity

- We collectively refer to all n-grams and words as *tokens*.
- Each token has its own level of *ambiguity*.
  – Ambiguous tokens are not tightly correlated with a single category; they can be connected with multiple categories.
- Or, each token is of different importance.
- According to the previous example, longer tokens are less ambiguous, that is, more important.

Importance Scores

- Furthermore, a token is more important if it has been correlated with only a few categories.
- Based on these notifications, we introduce the following importance score for a token t.

$$I_t = l_t \log \frac{|Y|}{f_t}$$

  - $|Y|$: the total number of categories,
  - f_t: the frequency of t, i.e. the number of the categories which have been correlated with t, and
  - l_t: the length (in words) of t.

Categorization Model: Training Phase

- The training phase builds a lexicon L for the tokens. Each entry in the lexicon stores:
- The token t,
- Its frequency f_t,
- Its importance score I_t,
- A relevance description vector (RDV) which for each token-category relationship, includes a pair in the form $(y, f_{t,y})$, where y is a category and $f_{t,y}$ is another frequency value that reflects how many times t has been correlated with the category y.

Model Training Algorithm

Algorithm 1: Model training algorithm

```
initialize the lexicon L;
for each product x in the training set do
    retrieve the category y;
    extract the title τ;
    perform linguistic processing of τ;
    for each n ∈ [1, N] do
        compute all tokens T_n of length n;
        for each token t ∈ T_n do
            if L.search(t) == false then
                L.insert(t);
                set f_t ← 1;
                L.insertRDV(t,y);
                set f_{t,y} ← 1;
            else
                if L.searchRDV(t,y) == false then
                    set f_t ← f_t + 1;
                    L.insertRDV(t,y);
                    set f_{t,y} ← 1;
                else
                    set f_{t,y} ← f_{t,y} + 1;
                end
            end
        end
    end
end
for each token t ∈ L do
    set I_t ← l_t log(|Y|/f_t);
    sort RDV of t in decreasing f_{t,y} order;
end
```

Categorization Model: Testing Phase

- The testing phase is based on the lexicon L of the previous phase.
- Initially, an empty candidates list Y' is created.
- In the sequel, for each token t of each product p, we perform a search in L.
- In case it is successful, we retrieve the RDV, f_t and I_t of t. Then, we traverse the RDV and for each entry y we update the candidates list Y'.

Candidates Scoring

- The score $S_{t,y}$ of each candidate category is expressed as linear combination of the importance score of the token and a quantity $Q_{t,y}$:

$$S_{t,y} = k_1 I_t + k_2 Q_{t,y}$$

$$Q_{t,y} = \log f_{t,y} \log \frac{|Y|}{f_t}$$

- Finally, the candidates are sorted in decreasing $S_{t,y}$ order and the top candidate is selected as a category for the product.

Categorization Algorithm

Algorithm 2: Categorization algorithm

for *each product* x *in the test set* **do**
    initialize candidates list Y';
    extract the title τ;
    perform linguistic processing of τ;
    for *each* $n \in [1, N]$ **do**
        compute all tokens T_n of length n;
        for *each token* $t \in T_n$ **do**
            if *L.search(t) == true* **then**
                for *each pair* $(y, f_{t,y})$ *in the RDV of* t **do**
                    if Y'*.search(y) == false* **then**
                        Y'.insert(y);
                        set $S_y \leftarrow f(I_{t,y}, Q_{t,y})$
                    else
                        set $S_y \leftarrow S_y + I_{t,y}$
                    end
                end
            end
        end
    end
    sort Y' in decreasing S_y order;
    set $y \leftarrow Y'[0]$;
end

Model Pruning

- We may decrease the size of the lexicon L by preserving only the tokens whose importance score exceeds a threshold C:

$$C = T \max_{t,y} I_{t,y}$$

- We will demonstrate experimentally that this choice combines a significant reduction of the size of L, with infinitesimal losses in the accuracy of the algorithm.

Experimental Setup

- Dataset: 313,706 products & 230 (191 leaf) categories from shopmania.com.
- Training/Test Set sizes: 60%/40%.
- Four scenarios for the extracted n-grams:
  – N=1: Extract unigrams (single words) only.
  – N=2: Extract unigrams & bigrams.
  – N=3: Extract unigrams, bigrams, & trigrams.
  – N=4: Extract 1-, 2-, 3-, and 4-grams.
- $0.0 < T < 1.0$ with steps of 0.1.

Examined Methods

- SPC – Supervised Products Classifier – the proposed algorithm.
- LogReg – Logistic Regression.
- RanFor – Random Forests.

Accuracy Evaluation

Highest Accuracy: 95.1% for N=3, T=0.

Interesting Measurement: Accuracy 93% for N=3, T=0.7.

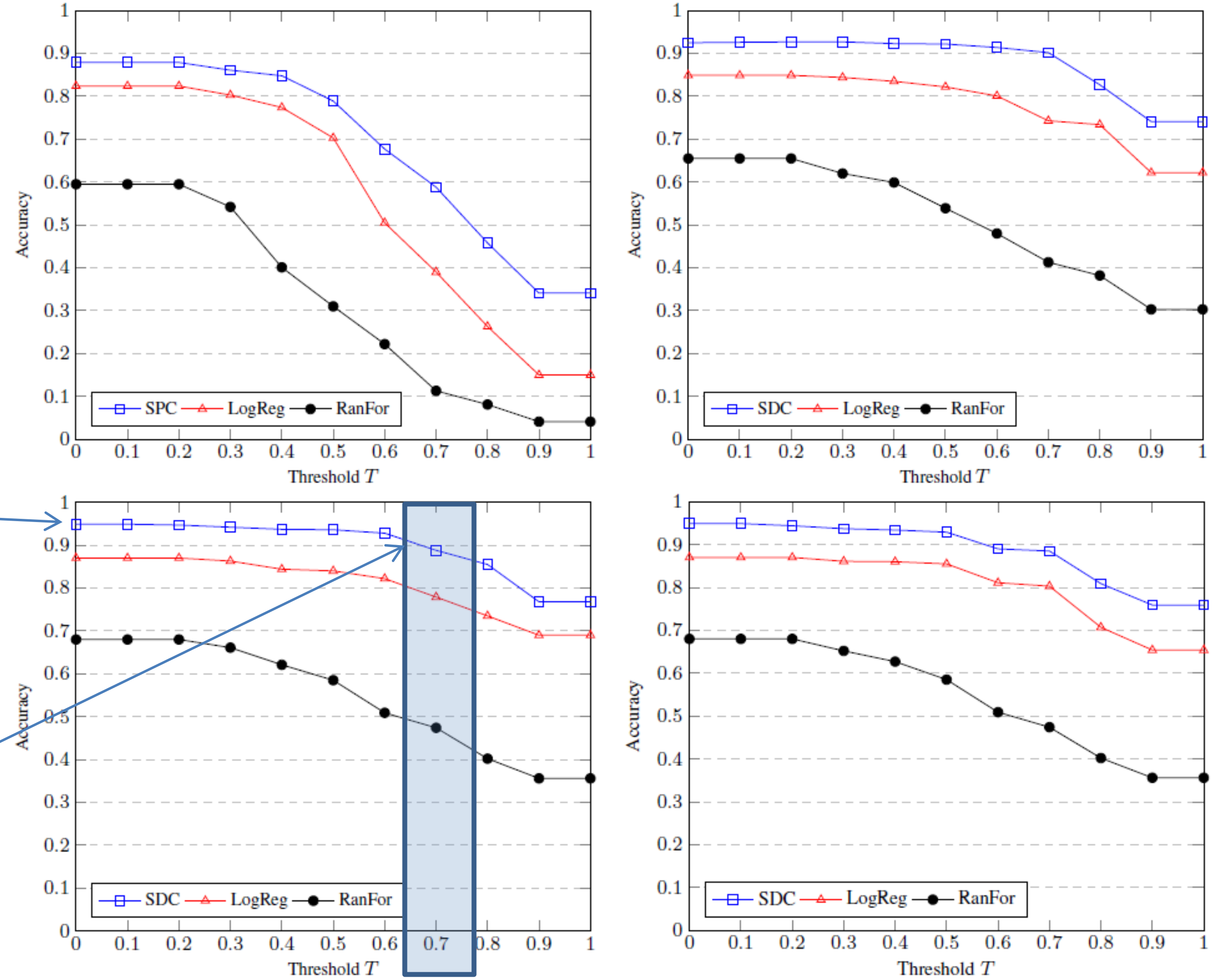

Fig. 1. Comparison of the classification performance of SPC against Logistic Regression (LogReg) and Random Forests (RanFor). Accuracy fluctuation with respect to the value of the cut-off threshold T for: i) $N = 1$ (top left), ii) $N = 2$ (top right), iii) $N = 3$ (bottom left), and iv) $N = 4$ (bottom right).

Pruned Model Evaluation

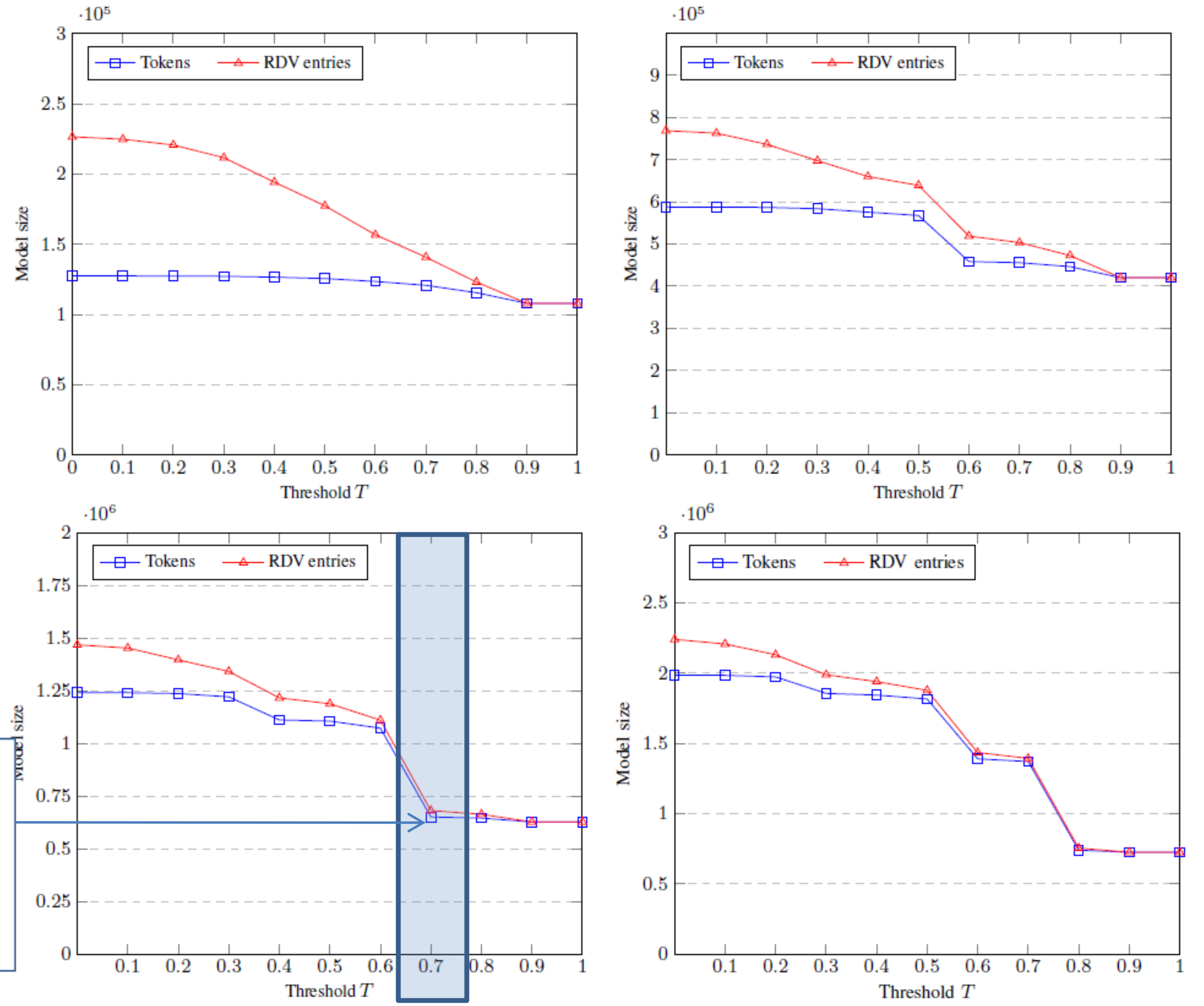

Fig. 2. Model size (overall number of tokens and RDV entries) reduction with respect to the value of the cut-off threshold T for: i) $N = 1$ (top left), ii) $N = 2$ (top right), iii) $N = 3$ (bottom left), and iv) $N = 4$ (bottom right).

Recall: For N=3, T=0.7, accuracy 93% with 47% fewer tokens and 54% fewer RDV entries

Conclusions

- We presented a supervised learning algorithm for products categorization.
- It trains a classification model based on:
  - The morphological analysis of the titles,
  - The extraction of n-grams of variable sizes,
  - The assignment of importance scores to each token.
- The method achieves ~95% classification accuracy.
- It also embodies a self-pruning strategy.
- The experiments have demonstrated that this strategy leads to a reduction of about 50% in the size of the model combined with small losses in the classifier performance.

Thank You

Any Questions?