Label Frequency Transformation for Multi-Label Multi-Class Text Classification

Raghavan A K[†], Venkatesh Umaashankar[‡], Gautham Krishna Gudur[†]
[†]Global AI Accelerator (GAIA), Ericsson; [‡]Ericsson Research

Introduction

Goal: To classify short texts describing German books into one or multiple classes, 8 generic categories for Subtask (a) and 343 specific categories for Subtask (b). Our system (***Team Raghavan***) comprises of three stages.

- Transform multi-label multi-class problem into single-label multi-class problem – Build a **Category Model**.
- Build a **Class Count Model** – to predict the number of classes a given input can belong to.
- Transform single-label problem into multi-label problem back again by **selecting top-k predictions** from the category model (with optimal k value predicted from the class count model).

The code for our solution: `https://github.com/oneraghavan/germeval-2019`.

Dataset

GermEval 2019 Task 1 dataset consists of German books crawled from `randomhouse.de`, with the following major attributes extracted:

• Title • Description • Author Name • ISBN • Book Release Date

Total **343 categories** across three levels of hierarchy with 8, 93 and 242 categories respectively. The label distribution is very imbalanced. The top-level distribution is shown in Figure 1.

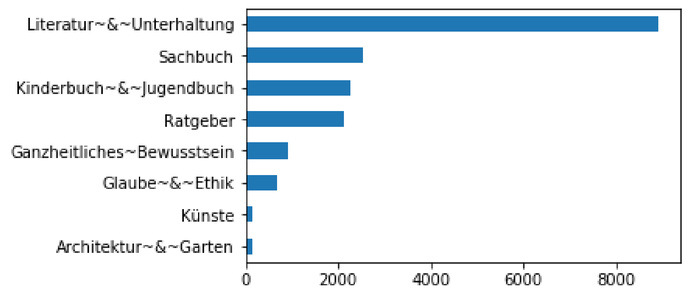


Figure 1: Top-level Label Distribution

Conclusion

In this work, we have successfully demonstrated that traditional approaches like ***Linear Support Vector Machines***, with a class count predictor model can effectively model Multi-label Multi-class Hierarchical Text Classification of German blurbs – *GermEval Task 1*. The authors would like to emphasize that conventional machine learning solutions would help in better interpretability, and when pipelined/fused with the right set of techniques, can effectively save a lot computational resources and time.

References

[1] Benjamin Heinzerling and Michael Strube. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[2] Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242, pages 133–142, 2003.

[3] Y. Tang, Y. Zhang, N. V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39:281–288, 2009.

Pre-Processing

The XML tags for each feature were parsed using the Python library – ***BeautifulSoup***.

Byte-Pair Encoding (BPE):

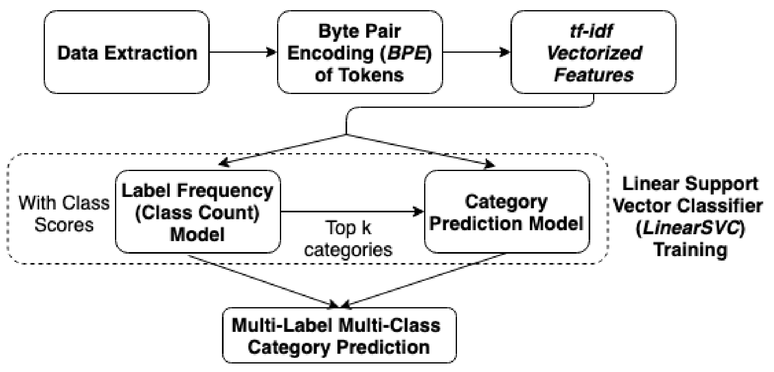
- ***BPEs*** [1] create tokens from the titles and descriptions (using *BPEmb* package).
- Identifies most common consecutive bytes of German words and replaces with a byte that does not occur within the data.
- *'Ein Blick hinter die'* might be converted into *'__ein __blick __hinter __die'* after BPE with a vocabulary size of 25,000.

Term Frequency-Inverse Document Frequency (tf-idf):

- *tf-idf* [2] vectors are from the BPE tokens. Shows significance of a word in a document in whole corpus.
- n-grams utilized – uni-grams, ***bi-grams*** and tri-grams. Bi-grams yielded better efficiencies, compared to rest.

Authors' data treated as a binomial attribute (returns 1 if book was authored by that author, else 0). ***Year*** from publication date, and ***Group ID***, ***Publisher ID*** from the 13-digit ISBN are represented as categorical features.

Overall Block Diagram



Model Description

The corpus has heavy class imbalance across all levels, we choose a ***Support Vector Machine (SVM)*** model [3] which utilizes a *one vs rest scheme*, returns the best-fit hyperplane that categorizes data in n-dimensional space.

Squared-hinge loss is used for maximum-margin classification that penalizes the violated margins more strongly (quadratically).

$$l(y) = max(0, 1 - t \cdot y)^2$$

where y is the classifier score; $y = w.x + b$, w, b are the parameters of the hyperplane, x is the input variable(s) and the intended output t=±1.

Two *LinearSVC* models – ***(a)*** one for predicting count of classes a book could belong to (***Class Count Model***); ***(b)*** other is the ***Category score predictor Model***. *LinearSVC* was chosen as it uses the ***liblinear*** framework, and scales well with increase in number of features.

The two models are pipelined for effective classification of categories, input features for both models remain the same. Two sets of target variables for the respective models are created – ***categories*** for class score predictor model and ***count of categories*** for class count predictor model. *Motivation behind model:* High correlation between the number of categories a book belongs to, and its corresponding class with highest score.

Experiment

For building an end-to-end classifier system, we build a Classifier Class extending the ***scikit-learn*** Base estimator API, with the respective fit and predict functions, and two *LinearSVC* models are created.

In the fit (training) phase, both models are trained with their respective target variables. We utilize a k-fold cross-validation strategy ($k = 4$) and obtain the corresponding class scores. The class count model is trained with the obtained class scores for the whole training data again. The class (category) predictor model is then finally trained with top k category predictions from the class count model. Retraining the entire set of models takes just under 2 minutes.

The experimental setup utilized for our solution is as follows: **(1)** Intel® Xeon® Processor E5-2650 v4 30M Cache, 2.20 GHz, 12 Cores, 24 Threads **(2)** 250 GB RAM **(3)** CentOS 7.

Model Parameters

Table 1: Optimal Parameters for *LinearSVC*

Parameter	Optimal Value
C	1.0
Tolerance for stopping	0.0001
Loss	Sqrd. Hinge Loss
Penalty	L2
Optimization algorithm	Dual
Max Iterations	3000

Table 2: Class Weights for Top-level Categories to handle Class Imbalance in *LinearSVC*

Category	Class Weight
Kinderbuch & Jugendbuch	1.8
Ratgeber	3
Sachbuch	2
Glaube & Ethik	2
Künste	6
Architektur & Garten	6
Literatur & Unterhaltung	1
Ganzheitliches Bewusstsein	1

Results

Table 3: k-fold Cross-Validation (k=4) – f1-micro scores on *Subtask (a)* and *Subtask (b)*

CV Fold	*Subtask (a)*	*Subtask (b)*
Fold 1	0.833	0.384
Fold 2	0.943	0.471
Fold 3	0.950	0.484
Fold 4	0.900	0.397

Team Raghavan achieves ***Rank 4*** in *Subtask (a)* during the test phase (f1-score of ∼0.86). However, in the post-evaluation phase, we achieve an f1-score of 0.878 (***first position***) in *Subtask (a)*. The additional 0.02 gain in f1-score was achieved by adding ISBN based features – *Group ID* and *Publisher ID*.

Table 4: Evaluation Metrics (f1-micro scores) for Test Data on *Subtask (a)* and *Subtask (b)*

Phase	*Subtask (a)*	*Subtask (b)*
Validation	0.851	0.4098
Test	0.857	-
Post-Eval.	0.878	0.3947