Report: Machine Learning challenge

Feature engineering

We computed basic characteristics of text features - **title**, **abstract** and **topics (keywords)**- those included: number of characters, number of words, number of stopwords, readability scores, number of capitals, etc. Some of these variables were also log-transformed if they have a higher correlation with a target in this form. We dropped calculated text features that correlated with target < 0.05.

We tried using word embeddings on **abstracts** (in Orange software) with pretrained GloVe embeddings for English but the results were less promising in terms of predictive power compared to the topic modelling using Latent Dirichlet Allocation (LDA). Therefore, for **abstracts,** we applied LDA. As for **titles** and **topics (keywords)** variables, we employed a special topic modelling technique called "Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model for short text clustering" (GSDMM) which is more suitable for short texts than LDA (Yin & Wang, 2014; Pelgrim, 2021). We didn't calculate the most probable topic for each document rather we calculated probabilities of belonging to each topic. Those papers that have missing values for **abstracts** or **topics (keywords)** were assigned 0 probability for each topic. Thus, for each document (paper) we had k variables based on a specific topic model for each text variable. We experimented with several topics for each variable (title, abstract, topics) testing its predictive power using cross-validation and holding other parameters and variables constant. We decided to stop on 9 topics for abstract, 20 topics for titles and 20 topics for topics (keywords). We also calculated tf-idf for top-50 most frequent words and bigrams in **titles** and **abstracts.** We decided to have the top-50 most frequent tokens to have a balance between too many features which could lead to a curse of dimensionality and too few features which could be not so informative.

The basic format of **doi** is '10.prefix/suffix'. We extracted prefixes from doi and based on it created dummy variables for top-3 most frequent prefixes. We counted the number of **authors** per article. We additionally tried using the author names to calculate the average number of citations per author for our train data, but we were confronted with the problem of information leakage as we discovered that the intersection of authors in the training data and the test data is low which led to overfitting as testing on hold-out validation data revealed. Therefore we decided to collect additional author metrics from Google Scholar using web scraping. We ended up with the following metrics: h-index, i-index, total citations for only one of the available authors on Google Scholar and approximately 40% of all publication's authors on combined training and test data.

We created a set of dummy variables based on cleaned and standardised **venues'** names for the top-7 most frequent **venues** in combined train and test data. We also used external data about conference ranking (A+, A, B, C) and merged it with our data based on **venue** variables.

The variable **Year** has several missing values which were imputed with the median. **References** were log-transformed since EDA showed that log-transformed references have a higher correlation with log-transformed citations. **Information about open access** was used as a dummy variable. In all analyses, we used log-transformed versions of our **target citations** since evaluation is based on R^2 of log-transformed citations.

Machine Learning Algorithms and Tuning

We used the Lazy Predict library (Auto ML package) to test which type of ML models work better for our task. Based on these results we choose the top-3 best performing models for further investigation: LightGBM, Histogram-based Gradient Boosting and Random Forest. We used permutation-based feature importance to investigate what features are most predictive of citations. We revealed that the most important features are: number of topics (keywords), log-transformed number of references, year, log-transformed number of authors, LDA topics revealed in abstracts. For both Random Forest and Histogram-based Gradient Boosting we conducted a random grid search based on 5-fold-cross (and 10-fold) cross-validation to find the best hyperparameters. As both models

underperformed compared to LGBM (best R^2 of 0,45 for Random Forest and 0,47 for Histogram-based Gradient Boosting) we forgo in-depth with fine-tuning the LightGBM. The choice of LightGBM was also partly dictated by the fact that it is much faster to train. For LightGBM we tuned: number of trees, number of leaves, min number of observations in child nodes, learning rate, L1 and L2 regularization, % of columns sampled.

Discussion of the performance of our solution

The first submitted model reached an R^2 of 0.4979 on the test data. After this, we added additional information about the authors as well as played with feature engineering (number of topics for topic modelling, standardization, adjusting of the number of categories for doi and venues, adding tf-idf for words and bigrams) and fine-tuned LGBM which led to an increase in R^2 reaching 0.5370 on the test data for the final submission.

Name of the account under which we made our submission: MikhailB

Detailed specification of the work done by group members:

Milan Fedorik: Having no skills in machine learning and text data, I contributed with EDA in MS Excel helping us to identify key relations, data cleaning and processing of the featured venue, small research on possible packages to be used and rankings of conferences (venue), investigation of year variable (possibility to use dummy variable instead of a year).

Balázs Gönczy: I contributed with the following: Web scraping and external data integration from Google Spreadsheets, LDA topic modelling on abstracts, TFIDF vectorization, Using autoML package to find best performing type of models, Code refactoring for submission.

Mikhail Bogdanov: EDA in Orange, prototyping first models in Orange, trying word embeddings and different topic modelling algorithms in Orange, implementation of GSDMM topic modelling for titles and topics (keywords) in Python, extracting features from text variables, fine-tuning LightGBM, getting predictions for the test data in the required format, creating and finalising the submission.

Katharina Pritzl: Investigation of raw data in Python (mainly visualizations of correlations and detecting of missing values), data cleaning and preprocessing of the feature author and doi, as well as the processing of year, references and is_open_acess as described above. Testing and Implementing of the additional data regarding the authors which were collected by other team members. Implementation of Random Forest and Histogram-based Gradient Boosting and hyperparameter tuning for these two models.

Literature

1. Cic.tju.edu.cn (2021). CORE Computer Science Conference Rankings. Link: http://cic.tju.edu.cn/faculty/zhileiliu/doc/COREComputerScienceConferenceRankings.html
2. Matsui, T., Kanamori, K. & Ohwada, H. (2014). Predicting Future Citation Count Using Bibliographic and Author Information of Articles. *International Journal of Machine Learning and Computing*, *4*(2), 139–141. https://doi.org/10.7763/ijmlc.2014.v4.401
3. Pelgrim, R. (2021, June 21). *Short-Text Topic Modelling: LDA vs GSDMM - Towards Data Science*. Medium. https://towardsdatascience.com/short-text-topic-modelling-lda-vs-gsdmm-20f1db742e14
4. Team, S. (2021, December 1). ScraperAPI - The Proxy API For Web Scraping. ScraperAPI. https://www.scraperapi.com
5. Yin, J., & Wang, J. (2014, August). A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 233-242).