LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Data Quality Metrics Project

Status: 06.04.2022

Initial tasks:

- ☐data set analysis
- ☐literature research
- ☐reasonable metrics extraction
- ☐data preprocessing
- ☐information extraction
- ☐information classification

Data access & security

- ☒data search & preprocessing
- ☒literature research
- ☒metrics extraction
- ☒information processing
- ☒tasks implementation
- ☐performance analysis

Background & Prerequisites

Source code:

Reference & link to GIT: https://github.com/eshmaneva/DS-Practical.git

I. 1. Data search

Due to complications with the data access and security, opensource data sets were used to test existing classification and summarization methods

Data set (summary generation): Amazon Product Data (Mobile Electronics Reviews)
Source: https://jmcauley.ucsd.edu/data/amazon/

Data set (classification tests): Amazon Fina Food Reviews
Source: https://www.kaggle.com/snap/amazon-fine-food-reviews

I. 2. Data preprocessing

Used data is processed directly via code:

- for summary generation task: due to the absence of the train_data part of the data set, data is being split, shuffled and a part of it pretrained
- for classification task: (tbd due to ROBERTA being in implementation phase)

II. Literature research

For the current project following findings and resources proven to be useful:

- 1st try to find a best suitable library: https://www.upgrad.com/blog/python-nlp-libraries-and-applications/
- Understanding text preprocessing & its implementation: https://t-redactyl.io/blog/2017/06/text-cleaning-in-multiple-languages.html
- Understanding text summarization and its implementation: https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70
- Understanding and using (Distil)BERT & Tensorflow: https://medium.com/geekculture/hugging-face-distilbert-tensorflow-for-custom-text-classification-1ad4a49e26a7
  https://huggingface.co/docs/transformers/v4.17.0/en/model_doc/bert#transformers.BertTokenizerFast
  https://towardsdatascience.com/hugging-face-transformers-fine-tuning-distilbert-for-binary-classification-tasks-490f1d192379
- Implementing BERT for all kinds of summary creation tasks: https://towardsdatascience.com/summarization-has-gotten-commoditized-thanks-to-bert-9bb73f2d6922
  https://pypi.org/project/bert-extractive-summarizer/
- BERT finetuning for all kinds of summary creation tasks: https://arxiv.org/pdf/1908.08345.pdf
- RoBERTa and abstractive summary: https://anubhav20057.medium.com/step-by-step-guide-abstractive-text-summarization-using-roberta-e93978234a90
- Understanding performance metrics in NLP: https://towardsdatascience.com/the-ultimate-performance-metric-in-nlp-111df6c64460
- Implementing performance metrics in NLP: https://pypi.org/project/rouge-score/
  https://arxiv.org/pdf/1908.08345.pdf

III. Metrics extraction

Based on the structure of the open-source dataset, stated in (I.1.), following metrics were chosen for further analysis and processing:

Numerical data: Helpfulness (rule-based numerical analysis → review helpful? Y/N)
Text data: ReviewText, Summary

IV. Information processing

Information processing was done accordingly within the code

Further information tbd (code in review & bug controlling)

V. Main tasks implementation (summary creation & classification)

- Within the solution of the classification task following libraries and classes were implemented and used

Used libraries: NumPy (ver. 1.19.5), TensorFlow (ver. 2.7.0), transformers (ver. 4.7.0), sacremoses (ver. 0.0.45)

Implemented classes:

DistilBertTokenizerFast
https://huggingface.co/docs/transformers/model_doc/distilbert#transformers.DistilBertTokenizerFast
TFDistilBertForSequenceClassification
https://huggingface.co/docs/transformers/model_doc/distilbert#transformers.DistilBertForSequenceClassification
NLTK
https://www.nltk.org/install.html
re (regular expressions operations)
https://docs.python.org/3/library/re.html
tensorflow datasets
https://blog.tensorflow.org/2019/02/introducing-tensorflow-datasets.html

Further classes tbd (code in review & bug controlling)

- Within the solution of the summarization task following libraries and classes were implemented and used

Used libraries: Datasets (ver. 1.0.2), transformers, rouge score

Implemented classes:

tbd (code in review & bug controlling)

Project motivation and description

Motivation:

Before starting a vehicle model series production, the production process of it is tested within vehicle concept and prototype engineering. Data is collected, specifically, to track occurring quality defects needing the rework. For ease of presentation, this data will be referred to as “Prototype”.

When a vehicle model goes into series production in a plant, during production, again, very similar data is collected to record each quality defect that again need to go into the rework. This data will be referred to as "Production". For defects occurring during series production respective tickets are raised. All these tickets should be resolved by the end of vehicle production.

Quality defect recordings in either of the mentioned data source exhibit a human, free text description. To better maintain these defects a more manageable, superordinate data source – referred to as "Knowledge-Base" – is built with the purpose to summarize similar quality defects in "Production" and "Prototype". Specifically, each recording in these data should describe a prebuilt, known defect cluster. Besides similar defects, this data source should summarize similar steps conducted to fix those defects.

The goal of the project is to derive reasonable metrics for text data in the data sources, analyze the free text description data and create a summary of it.

Due to data security issues, the research for the most similar structured data was conducted, resulting into using the Amazon Product Reviews data (narrowed to utilizing of the Quality Food and Music Instruments reviews). Based on the new data, summarization of the ReviewText was set as the main task of the project.

Theoretical aspects:

Summarization aims to condense some text data into a shorter version while preserving most of its meaning. Generally, machine summarization is split into two types: extractive (important sentences are extracted as they appear in the original document) and abstractive summarization (summarize important ideas or facts contained in the document without repeating them verbatim). The first one can be compared to highlighting the most important parts of the text with the marker while the latter one is supposed to be comparable to a person-written summary.

To be able to test both summarization methods, the standard BERT library (Bidirectional Encoder Representations from Transformers) was chosen. Although, for easier and faster computation the DistilBERT (distilled version of BERT[1]) is being used. Additionally, Robustly optimized BERT approach[2] (RoBERTa) was applied and being tested for creating both abstractive and extractive summaries.

[1] DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter: https://arxiv.org/abs/1910.01108

[2] Abstractive Text Summarization Using RoBERTa: https://anubhav20057.medium.com/step-by-step-guide-abstractive-text-summarization-using-roberta-e93978234a90

After creating the summary, it should be classified. Another model was created for that part, using BERT and NLTK libraries which is currently working with over 90% accuracy.

VI. Performance analysis

tbd (code not yet in implementation phase)