Claims Knowledge Graph dataset Classification

Technical Project

Ensiyeh Raoufi February 20th, 2022

I. Definition

Project Overview

Fact-checking is a process that seeks to verify sometimes factual information, to promote the veracity and correctness of reporting. Studies of post hoc (after the text is published) fact-checking have made clear that such efforts often result in changes in the behavior, in general, of both the speaker (making them more careful in their pronouncements) and of the listener or reader (making them more discerning about the factual accuracy of content).[1]. The dataset is used for this project is ClaimsKG, a dataset that is collected from fact-checking sites (such as Politifact or Snopes) by LIRMM in collaboration with several European research teams. The basis of the dataset is a knowledge graph that provides data about claims and their metadata. "A knowledge graph, also known as a semantic network, represents a network of real-world entities—i.e. objects, events, situations, or concepts—and illustrates the relationship between them. Knowledge graphs are typically made up of datasets from various sources, which frequently differ in structure. Schemas, identities, and context work together to provide structure to diverse data."[2]

In this project, I have done feature engineering on the ClaimsKG dataset and have trained several models on the feature-extracted data that classify the claims in True, False, and Mixture classes. Taking a good result from this model, one can predict if a certain claim is True, False, or a Mixture of them. The claim has some metadata such as a specific author, headlines, keywords, source of review, and some more features. The result is helping automatically fact-checking of news, because we will investigate that having certain information about a claim, with how much precision and recall we can classify it. So it is worth solving this problem to check the veracity of any represented claims.

Problem Statement

I'm solving two problems on claims classification of the ClaimsKG dataset.

1. I'm facing a classification problem to classify each claim to be False, Mixture, or True, and, because I have the label of each claim, I should do a type of supervised learning. A good prediction will help us to precisely estimate whether a claim with specific metadata would be reviewed by a special reviewer as true, false, or a mixture of true and false (in mixture case some parts of the claim are true and the other parts would be false). It has benefits spreading the inference among the society and avoiding politicians and other public speakers to say wrong information in their talks.
2. I should classify claims to be in two classes: {TRUE or FALSE} vs. {MIXTURE}. In this case, by getting a high score prediction from our models, we would know how much our estimator can declare a boundary between definite true/false claims and mixture claims reviewed by resources.

In both of the above problems, I'm using machine learning models to train classification models that their prediction indicates:

Based on metadata and the text of a claim, which of the classes the claim belongs to.

For solving this problem I first created a comprehensive table from all other 3 data tables of False/Mixture/True that I was extracted from ClaimsKG's web interface Claim Explorer. Then I have replaced the NaN cells with selections from unigrams or bigrams, Lemmatized and stemmed the words in the whole table, and removed the stop words using nltk library. At the last preprocessing step, I vectorized the whole table based on each class set of unigram and bigram choices, using some NLP metrics, and then used it as my data source. The source table contains all the claims important features in float amounts. Each float amounts indicate the feature's TF-IDF (Term Frequency and Document Frequency) based on the claim label. For example, I did a TF-IDF vectorization on the whole selections of the single words and unigrams that are existed in the True labeled claims and I did this job on the two other claim classes. Finally, I've merged all the three class vectorized table and have gotten the source table for training a classification model on it. A big part of this project consists of:

- Cleansing data and extracting information
- Filling the NaN cells of the table of the whole claims with appropriate n-grams
- Stemming, lemmatizing, removing the stop words **Note: For filling the NaN cells of the named_entities_article column, I've Used unigram and bigrams from the headline and named_entities_claim to recognize the top 2 article topics. Because the number of NaNs in this column is more than 9000 and the number of words, especially in the named_entities_claim column is too much and, because we are searching for both single words and 2-grams, it's the bottleneck of my program to fill the NaNs of this column. Not using the Colab GPU, it takes about 40 minutes for running.

Offer: We can use only single words for article named-entity recognition.** * Vectorizing the table and getting appropriate numerical feature columns * Concatenating columns and creating neat tables * Merging all the tables to one comprehensive table that we can use as a source for our model training

When my source table is ready then we should solve the main problem. We should do the following jobs to get the results:

- Training a Scikit_Learn Random Forest model based on the given hyperparameters (number of estimators and leaves)
- Fitting a Multinomial Naive Bayes Classifier
- Using Support Vector Machine Classifier (SVC) model with various kernels like Linear, Polynomial, and, Gaussian
- Fitting K-Nearest Neighbors Classifier on the source data and making predictions

- Evaluating all models on some suitable classification metrics like f1-score, recall, and precision
- Training the dataset with the auto-ML library Autogluon Tabular predictor to find a benchmark for our model
- Evaluating the model scores of this auto-ML predictor and comparing to our results
- Finding the best model for predicting our data

Metrics

I evaluated my models using metrics like f1-score, recall, precision, and balanced accuracy which are suitable metrics for my unbalanced classification problem. My data is unbalanced because the number of True claims is about 20% of the whole dataset and, each of False and Mixture claims forms about 40% of the dataset. We use balanced accuracy in binary and multiclass classification problems to deal with imbalanced datasets. It is defined as the average of recall obtained in each class. Also as we know precision is intuitively the ability of the classifier not to label as positive a sample that is negative and, recall is intuitively the ability of the classifier to find all the positive samples. Also, f1-score defines as below:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

f1-score is a good metric that can be interpreted as a harmonic mean of precision and recall. In the multi-class and multi-label case, the average of the F1 score, also known as balanced F-score or F-measure is the average of the F1 score of each class with weighting depending on the average parameter. If the weighted average is set to this metric, it calculates metrics for each label, and finds their average weighted by support (the number of true instances for each label). So these are proper metrics for my unbalanced dataset.

II. Analysis

Data Exploration

I extracted the true, false, and mixture claims using ClaimsKG's web interface Claim Explorer. There are 27588 goal claims (4404 True, 12350 False, 10834 Mixture).

Because this explorer export the results to a CSV form just for a maximum of 10000 rows, I exported each of the true, false, and mixture records in separate CSV files. Because the distribution of all types of claims is not uniform, and both of the false and mixture claims have more than 10000 records, we lost 3184 claims to extract. But in comparison with the data set size and also, regarding a low difference between the number of total false and mixture claims, I guess it's not going to make a mistake. The data is extracted from Claim Explorer are contained in three files:

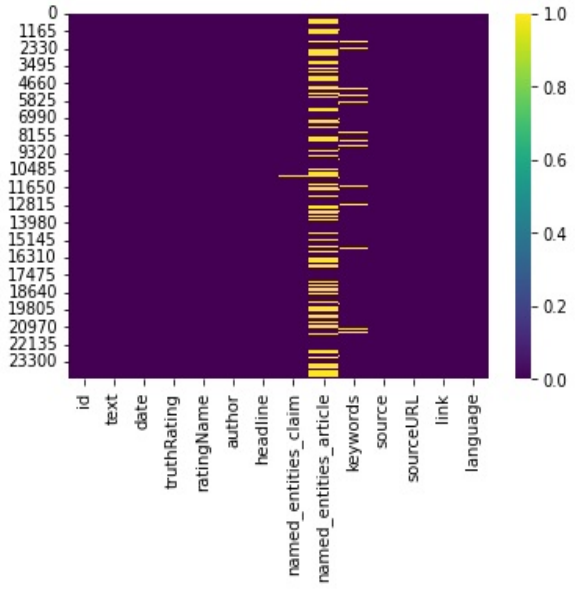
- claimskg_false.csv - contains 10000 False claims and metadata about each claim (text, date, author, keywords, source, etc.)
- claimskg_mixture.csv - contains 10000 Mixture claims and metadata about each claim (text, date, author, keywords, source, etc.)
- claimskg_true.csv - contains 4404 True claims and metadata about each claim (text, date, author, keywords, source, etc.)

Each of these 3 csv files contains below features:

- id (object) - Claim link
- text (object) - Claim text
- date (object) - Claim published date
- truthRating (int64) - 1:False, 2: Mixture, 3: True - Label of classified claim
- ratingName (bool) - False, Mixture and True claim rating names
- author (object) - Name of the author
- headline (object) - Headlines of the claim
- named_entities_claim (object) - every cell contains an instance of a claim
- named_entities_article (object) - every cell contains an instance of an article
- keywords (object)- Keywords used in the claim
- source (object) - Claim reviewer source name
- sourceURL (object) - Claim reviewer source link
- link (object)- Link to the claim review
- language (object) - Language of the claim

Note: You may need to know this dataset is extracted from a knowledge graph containing all the relations between claims and their features. Some of the relations are one-to-many and some other are many-to-many. For example, the relation between "claim proposition" and its "representation" is one-to-many."A claim proposition refects the meaning of one or more semantically equivalent claim utterances expressed in different linguistic forms or contexts."[3] This means that a claim proposition can have several representations (in text format, dictionary format, etc.), but a presentation is representing just one specified claim proposition. There are some other many-to-many relations in this knowledge graph. For example, the relation between the "claim utterance" and its "linguistic representation" is many-to-many. "A claim utterance is the act of expressing a claim proposition in a specific natural language and form (like text or speech)."[3] It reflects that a claim utterance may have several linguistic representations (for example several different texts and speeches pointing to the same claim) and, a linguistic representation may be the representation of several different claims utterances.

First, we merged all 3 CSV files into a table. By watching the raw merged table, we can see that the "language" consists of 'English' for all raws. So we should drop it because it has no impact on predicting the labels. Also, we can see that each of the rating names is equal to a numerical rating label in the 'truthRating' column. By drawing a heatmap of the NaN cells, We consider a lot of NaNs, especially in the 'named_entities_article' column, that should be filled using other features. You can watch this heatmap here:



Because the type of the article has a strong relationship with its headline and the main body concepts of the claim text, I chose to fill the NaNs in this column by extracting the unigram and bigram keywords of both 'headline' and 'named_entities_claim' columns. I made the selection of bigrams to also keep the order of the words in the text. I didn't use 3-grams or higher n-grams because of their computational costs. I used a KeyBERT model for keyword extraction.

After filling the NaNs of all columns using the above strategy (by getting keywords from suitable columns), it's time to make the whole table easier to understand by any model. So I lowercased all the textual data, removed punctuations, extra spaces, and English stop words (because all the claims were presented in English), lemmatized all the necessary columns, and stemmed all the words. The head of the table looks like below now:

	id	text	date	truthRating	ratingName	author	headline	named_entities_claim	named_entities_article	keywords	source	sourceURL
0	http://data.gesis.org/claimskg/claim_review/36...	'There will be no public funding for abortion ...	2010-03-21	3	True	Bart Stupak	Stupak revises abortion stance on health care ...	Abortion rights,Barack Obama,Bart Stupak,Ben N...	abortion	Abortion,Health Care	politifact	http://www.politifact.com
1	http://data.gesis.org/claimskg/claim_review/e6...	Central Health 'is the only hospital district ...	2011-03-15	3	True	Wayne Christian	State Rep. Wayne Christian says Central Health...	Austin American-Statesman,Harris County Hospit...	NaN	Abortion	politifact	http://www.politifact.com

If we take a look at the distribution of the sources of claims of this edited table, we can observe that Politifact at the first and Snopes at the second, are the most contributing sources for truth rating the claims. Also, by drawing the histogram of the rating name of the claims we can obviously see the unbalanced classes of the data (False, Mixture, and True), through the whole dataset

I changed every text column to a numerical form. All machine learning models use mathematics and numbers to compute data. Since the input here is textual, we will use the TF-IDF (Term Frequency and Document Frequency) scheme and, because our classes are unbalanced and have different counts of words, we don't use BoW (Bag of Word) and just care about the importance ratio of the words **in each class**, and at last, you can see the edited version of my table in the following:

```
   abc abc news abort accid  ... york time youtub target_labels target_names
0  0.0      0.0   0.0   0.0  ...    0.0   0.0                1        False
1  0.0      0.0   0.0   0.0  ...    0.0   0.0                1        False
2  0.0      0.0   0.0   0.0  ...    0.0   0.0                1        False
3  0.0      0.0   0.0   0.0  ...    0.0   0.0                3         True
4  0.0      0.0   0.0   0.0  ...    0.0   0.0                3         True

[5 rows x 544 columns]
Shape of the data:  (24416, 544)
```

Algorithms and Techniques

After editing my table and before vectorization, I got a fast classification prediction based on all textual features of claims using Transformer Hub. TensorFlow Hub is a repository of trained machine learning models ready for fine-tuning and deployable anywhere. Building Sequential model with 3 layers, it gives the f1-score of [0.58, 0.36, 0.26] for [1, 2, 3] classes using input hyperparameter of {batch_size:16, epochs:50} which is quite low (1:False, 2: Mixture, 3: True). Once a Sequential model has been built, every layer has an input and output attribute and these attributes can be used to do neat things, like quickly creating a model that extracts the outputs of all intermediate layers in a Sequential model. Although we can use features extracted with the Sequential model, I was more interested in feature engineering on my own. So I used the TF-IDF (Term Frequency and Document Frequency) scheme, because our classes are unbalanced and have different counts of words. I've done vectorization calculating TF-IDF on every 7 textual columns (except for the columns that contain links) of each of our 3 classes. I set a limitation of a maximum

of 150 features and some other restrictions for my Tfidf Vectorizer to embed the data simultaneously, and the result was amazing. We got the total features of [683, 728, 680] for the [1, 2, 3] classes. After merging all the classes on all the common and distinct features, we obtain a table having just 550 features for all the classes! after removing duplicate values columns we have a table of the shape (24416, 544), that contains all the important numerical features for applying any classification model. I've fitted several models (Random Forest, SVM Classifier (SVC), KNN, etc.) for both multi-class and binary class classification and, evaluated the models on my metrics. Finally, I've trained an AutoGluon Tabular predictor for benchmarking my results from the other models.

Benchmark

If my problem was a Kaggle challenge, I could submit my results and give back a score to benchmark my trained models. Also, there are some online platforms like "Google's Vertex AI" where one can upload their CSV files and give back the prediction metrics. Unfortunately, because these kinds of sites incur costs, I can't use them. So I tried a different Auto-ML training like Autogluon Tabular predictor to compare the results with the previously trained models on my dataset. We know that this predictor gives us the best model from its various models and, at the end, we can compare the metric scores of both training jobs and see how our models are performing well and are convenient.

III. Implementation

First, we should install the required packages. I ran my whole project on my Google Colab and used its GPU especially for preprocessing the data. In this project I've upgraded the NumPy package, I've installed AutoGluon that can use its tabular predictor at the end of the program for benchmarking. I've installed keybert to use it for keyword extraction and filling the NANs and, installed seaborn to use it for drawing plots and some other packages that you can see in the very first cell of my program.

The main parts of the implementation can be split into 3 steps:

1. Cleansing data, modifying the tables and merging them to create a neat textual table
2. Vectorizing the data to create a table of numerical features
3. Train a random forest and several other models (KNN, SVC, etc.) and evaluate the models on the test data using our metric scores
4. Using an AutoGluon Tabular predictor to compare our model metric scores with

It is worth mentioning that preparing the data to become a neat, usable data frame is the toughest and the most time-consuming job. As a Forbes report indicates: "Data scientists spend 60% of their time on cleaning and organizing data." [4] and I experienced this on this project.

IV. Results of the Model Evaluation

After vectorizing the dataset we are ready to train models on our feature extracted data. First, we can take a look at the features having the most TF-IDF average in every class of labeled claims. We can observe that 'Snopes' and 'Politifact' together with 'Trump' and 'Obama' have the top 4 average of TF-IDF of the words in False labeled claims. 'tax', 'Snopes', 'say' and 'state' have the top 4 average of TF-IDF of the words in Mixture labeled claims. 'Politifact' and 'Snopes' together with 'state' and 'say' have the top 4 average of TF-IDF of the words in True labeled claims. From these results, we could investigate that Snopes is tending more than Politifact on the claims classification to be in Mixture category (because the name of the Politifact is not even in the top 10 high TF-IDF average features of the Mixture class), and Politifact -i.e. the most contributing source of truth checking for this dataset- is focusing on the classifying the claims to be True or False. So Politifact is going further to detailed partitioning of the claims.

Multi-class classification

SKlearn Random Forest Classifier is the first model that we use to classify our data. "Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time."[5] Fitting this classifier for 300 estimators and a minimum sample leaves of 3, we got a balanced accuracy of 0.86 and a weighted average F1 score of 0.9 on the test data, which is quite fine. Here is the screenshot of the results:

	precision	recall	f1-score	support
1	0.88	0.96	0.92	3067
2	0.92	0.98	0.95	3012
3	0.99	0.65	0.79	1337
accuracy			0.91	7416
macro avg	0.93	0.86	0.88	7416
weighted avg	0.92	0.91	0.91	7416

By looking at the most important features of this classifier, we can observe that 'politifact', 'asp', 'snopes' and 'articl' are the most important features to affect the result of the predictions. So we would be suspicious about the tension of the sources to classify a certain type of claims (such as True or False). So I ran this model another time on the dataset with removed source label features and we can see an increase of 9% in the weighted average, as you can see below:

	precision	recall	f1-score	support
1	0.88	0.87	0.87	3064
2	0.76	0.95	0.85	3017
3	0.99	0.47	0.64	1335
accuracy			0.83	7416
macro avg	0.88	0.76	0.79	7416
weighted avg	0.85	0.83	0.82	7416

So we can think about the biased classification of the sources as a conjecture, i.e. the willingness of a fact-checking source to classify and publish more False (or more True) facts.

Fitting a Multinomial Naive Bayes Classifier, we got a balanced accuracy of 0.78 and a weighted average F1 score of 0.81 on the test data. This algorithm guesses the label of a claim, such as True, False, or Mixture, using the Bayes theorem. It calculates each label's likelihood for a given sample and outputs the label with the greatest chance.

Using a Support Vector Machine Classifier with a Linear kernel, with a decision function shape of one-vs-one (one class should be distinct from every other class, that is exactly our problem), we got a well-balanced accuracy of 0.86 and a weighted average F1 score of 0.90 on the test data. However using the Polynomial kernel can't give a high score prediction on the test, which is normal because it's not suitable enough for multi-class classification. Applying a Gaussian kernel to our SVC, we got metric scores of above 80%, but it's still not good enough as a Linear OvO kernel. Fitting a K-Nearest Neighbors Classifier would not give us a metric score higher than 75%.

Training an Autogluon predictor for 10 minutes, we can observe that a LightGBM_BAG_L2 model classifies the unseen data at the best with a balanced accuracy of 91%.

Binary class classification

Fitting a Random Forest Binary Classifier on the dataset with joined True and False classes, is giving us a high balanced accuracy and f1-score of 95 percent, the scores are just like using a Binary Linear SVC. However, using a Multinomial Naive Bayes Binary Classifier our balanced accuracy decreases to 86%. a KNN Binary classifier is not working so well just like the multi-class classification.

V. Conclusion

In both multi-class and binary classification, we can observe that the LightGBM_BAG_L2 model (that implements Ensemble Selection) is trained in stack level 2 can predict our data at the best. A model with "_L2" suffix is a not base model, meaning it depends on other models. This best model is a Stacker Ensemble Model using 10 models (view details in the predictor leadership) to distributed gradient boosting and classify the data based on decision tree algorithms.

In multi-class classification, the best model's balanced accuracy is 91%, however, we could predict on the test data using a single Linear SVC with a decision function shape of one-vs-one with a balanced accuracy score of 90%.

In binary classification, the best model's balanced accuracy and f1-score is 96%, however, we could predict on the test data using a single Linear SVC with 95% metric scores.

We conclude that a Support Vector Machine Classifier is the best classifier having a minimum computation costs in both multi-class and binary-class classification.

You can see the results of all classifiers and predictors and also, several various plots and tables in the .ipynb file.

Thanks for your attention.

References

[1] https://en.wikipedia.org/wiki/Fact-checking#:~:text=Research%20suggests%20that%20fact%2Dchecking,spreading%20false%20or%20misleading%20claims.

[2] https://www.ibm.com/cloud/learn/knowledge-graph#:~:text=A%20knowledge%20graph%2C%20also%20known,the%20term%20knowledge%20%E2%80%9Cgraph.%E2%80%9D

[3] http://users.ics.forth.gr/~fafalios/files/pubs/CKG2019.pdf

[4] https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=14537dd96f63

[5] https://en.wikipedia.org/wiki/Random_forest