TepesAI

Smart corruption detector

1. Overview

TepesAI is a complex multi-component approach to detecting and combating public figures corruption. The system leverages the fact that all people working in public institutions in Romania have to disclose their revenue and assets every year. Using self made automatic tools, we crawl the statements from public databases, we perform text extraction on them, we process the text to fit our models, then we feed the data to statistical & neural network based anomaly detectors. With this approach, we can provide an objective, unbiased corruption detection based on publicly available data.

2. State of the art

TepesAI - State of the art

3. Our solution

Our application consists of the following components:

1. Crawler

For data gathering, we were presented with two options:

- Manually search & download the statements
- Setup an automatic tool which can download tens of statements at a time

We focused on the second approach, since for the anomaly detector to be accurate, we need to train it using an ample amount of data. Since there aren't any public databases which expose API endpoints for us to make requests to, we had to develop a more sophisticated crawler using the automation tool Selenium which is able to mimic human behavior and thus avoids rate limitations. The developed crawler can be run from the command line and is highly customizable, accepting parameters for each query field from the websites that expose the asset statements. The crawler is capable of downloading multiple pages of statements and can extract both .pdf files and .csv tables containing data about public figures.

2. Normalizer

With the public statements gathered, we now need to extract useful information from them. That's where the normalizer comes in, its job being extracting data from the pdfs and processing it. The challenge in our case with extracting the data from the pdfs is that we have multiple types of content inside of them:

- Fully digital pdfs
- Digital pdfs which were later scanned as images then converted to pdfs
- Handwritten statements which were later scanned and converted to pdfs

For the purpose of this tool, we'll limit our scope to the first two types of pdfs since the later one needs a complex approach for handwritten text extraction in romanian language.

The fully digital pdfs can be parsed with common pdf libraries and simply have their content extracted through their metadata. The challenge comes from the second type, where we have to perform ocr (optical character recognition) on the pdf, and then extract the text as we did in the first type approach. The issue with this type is that the current ocr technology while being quite developed is nowhere near the needed level, especially in the context of the romanian language. As a consequence, we receive artifacts in the ocr'd text, more exactly incomplete words which can't be used in our models. To overcome this problem, we implemented a spellchecker which corrects the incorrect words using Levenshtein distance and maps them to the nearest correct word using a dictionary of romanian words for reference.

3. Aggregator

With the data extracted, we now need to process it further and convert it into a useful model for training and testing our neural network based anomaly detector. The aggregator has to be able to extract data from multiple statements formats and convert them in the same standardized model.

4. Anomaly Detector

The final challenge is to actually use the generated data and based on it decide if a person is corrupted or not. While the task seems pretty simple at first, a simple boolean response of true / false if the person is corrupted or not, the actual process of getting to that result is fairly complicated. We used a multi step approach. At first, we use the generated data as time series and put it through the pre-trained value predictor. The neural network uses LSTM layers and predicts how a future politician statement should look. This approach is similar to the ones used in crypto / stock price prediction. With the generated values, we then compare them to the actual values from that year's statement.

Another approach would be to assign each person a corruption score, and run a deep neural network which trains a model for regression. Such, we would need a set of training data with a corruption score assigned to each person - for example, we could hand-pick training data as known corrupted individuals and assign them a corruption score.

Based on this data, we could then assign each person we want to evaluate such a number. The factors we could take into consideration would be salary, debts and investments.

4. Challenges and results

Workflow:

The crawler will search and download the declarations from specific popular figures from Romania.

Rezultatele căutării: 120 declarații

Nume Prenume	Institutie	Functie	Localitate	Judet	Data completare declaratie	Tip declaratie	Vezi declaratie	Distribuie
CHIRICA M MIHAI	Comuna Berezeni	Consilier local	Berezeni	Vaslui	18.05.2022	Declaratie de avere	Vezi document	
CHIRICA M MIHAI	Comuna Berezeni	Consilier local	Berezeni	Vaslui	19.05.2022	Declaratie de interese	Vezi document	
Chirica I Mihai	Municipiul Iasi	Primar	Iasi	Iasi	14.06.2022	Declaratie de avere	Vezi document	
Chirica I Mihai	Municipiul Iasi	Primar	Iasi	Iasi	14.06.2022	Declaratie de interese	Vezi document	
CHIRICA MIHAI	Consiliul Local	PRIMAR	Iasi	Iasi	07.06.2021	Declaratie de avere	Vezi document	
CHIRICA MIHAI	Consiliul Local	PRIMAR	Iasi	Iasi	07.06.2021	Declaratie de interese	Vezi document	
CHIRICA MIHAI	Consiliul Local	PRIMAR	Iasi	Iasi	01.01.1900	Declaratie de interese	Vezi document	
CHIRICA MIHAI	Consiliul Local	PRIMAR	Iasi	Iasi	01.01.1900	Declaratie de avere	Vezi document	

After that, based on an approach we decided to implement, the normalizer would transform and fit each politician's declaration into a model politician with the following attributes:

```
class PoliticianModel:
    def __init__(self, first_name, last_name, functions, declaration_type, investments, assets, vehicles, debts, salary):
        self.first_name = first_name
        self.last_name = last_name
        self.functions = functions
        self.declaration_type = declaration_type
        self.assets = assets
        self.vehicles = vehicles
        self.investments = investments
        self.debts = debts
        self.salary = salary
        self.isCorrupt = 2  # 0 - not corrupt, 1 - corrupt, 2 - unclassified
```

Having the model built, the aggregator is merging all of them in a csv file, which will be used as train/test data for the anomaly detector.

Ciolos	Dacian	europarlamentar la Parlamentul	DECLARATIE DE AVERE	390107 EUR	No Assets	Nissan Qashqai 2008	0	83860 EUR
Ciolos	Dacian	europarlamentar la Parlamentul	DECLARATIE DE AVERE	257866 EUR	No Assets	Nissan Qashqai 2008	0	84940 EUR
Dragnea	Liviu	Deputat la Parlamentul Romani	DECLARATIE DE AVERE	592120 LEI	No Assets	[SKODA SUPERB 2006 BMW X5 2009	0	76838 EUR
Dragnea	Liviu	Deputat la Parlamentul Romani	DECLARATIE DE AVERE	653206 LEI	No Assets	[SKODA SUPERB 2006 BMW X5 2009	0	161912 LEI
Dragnea	Liviu	Deputat la Parlamentul Romani	DECLARATIE DE AVERE	725879 LEI	No Assets	[SKODA SUPERB 2006 BMW X5 2009	0	153384 LEI
Draghici	Mircea Gheorghe	Deputat la CAMERA DEPUTATIL	DECLARATIE DE AVERE	127318 LEI	65000 EUR	[MERCEDES 2016 MERCEDES 2016]	570000 LE	145024 LEI
Draghici	Mircea Gheorghe	Deputat la CAMERA DEPUTATIL	DECLARATIE DE AVERE	1076544 LEI	426432 LEI	[MERCEDES 2016]	1080200 L	140040 LEI

With the dataset built, we can now use a series of statements and predict the values for a future statement. With the predicted values we then compare it to the values from actual future statements.

```
model = keras.Sequential()

model.add(Bidirectional(
  CuDNNLSTM(WINDOW_SIZE, return_sequences=True),
  input_shape=(WINDOW_SIZE, train_dataset.shape[-1])
))
model.add(Dropout(rate=DROPOUT))

model.add(Bidirectional(
  CuDNNLSTM((WINDOW_SIZE * 2), return_sequences=True)
))
model.add(Dropout(rate=DROPOUT))

model.add(Bidirectional(
  CuDNNLSTM(WINDOW_SIZE, return_sequences=False)
))

model.add(Dense(units=1))

model.add(Activation('linear'))
```

While the components on their own don't seem that intimidating, merging them together and creating a complete, end to end system which is able to achieve the end goal of detecting anomalies in personal assets statements and highlighting corruption where it's suspected.

Each component presents its difficulties and since the approach is built upon a cascading system, each failure in the component chain can disrupt the whole flow.

For example, the Crawler can fall into edge cases, for example providing no found statements based on some query parameters or finding too many statements and being unable to display them because of internal website policies.

The next difficulty comes from the Normalizer component, which has the hard job of extracting and processing data from non-standardized pdfs. Along with the problem mentioned in a previous section, related to the artifacts generated by the ocr, we also need to find an automatic pdf type detection approach. For this, we employ a chain resolver technique where we first assume that the pdf is fully digital and we try to validate the extracted data. If the validation fails, we perform ocr on the pdf then we extract & process the data and we perform once again a validation. If the validation fails again, it means it's an unsupported pdf type and we dismiss it. The difficulty of this approach is in finding a constant value to validate against since the pdfs have different content.

- We need to have time series data so that we can detect corruption (detect huge spikes in total assets value from one year to another) and most of the corrupted politicians aren't elected for more than 4 years (insufficient data)
- Most politicians are careful with their corruption, they tend to hide their assets through family members which adds a layer of complexity (very hard to track)
- Assets in the statements are declared in multiple currencies (eur, usd, ron) and while it's easy to convert to one single universal currency, we need to take into account the conversion rate from that period.
- Another problem with the assets statements is that we have items such as cars which don't have an associated value (We'd need to implement a crawler which searches for the median price of the vehicle on multiple websites)
- Another issue would be that the aggregator has to solve is finding a model which actually provides useful information for training the neural network and testing its behavior.

5. Future work

While the current state of the project is nowhere near production phase, the approach looks promising regarding the near-future development of the used technologies. The main areas of improvement would be:

- to develop and extend the functionality of the normalizer. That would require further improvement in the ocr approach for the pdf files, as well as developing a more complex algorithm to generify the distinct formats of the declarations
- to further investigate the anomaly detector approach on a larger dataset of people. While the concept of corruption is still ambiguous, the deep learning algorithms could provide a way to quantify the level of the corruption of a politician based of his declaration

6. Conclusion

TepesAI is a complex multi-component approach to detecting and combating public figures corruption. The system leverages the fact that all people working in public institutions in Romania have to disclose their revenue and assets every year. Using self made automatic tools, we crawl the statements from public databases, we perform text extraction on them, we process the text to fit our models, then we feed the data to statistical & neural network based anomaly detectors. With this approach, we can provide an objective, unbiased corruption detection based on publicly available data.