מכון טכנולוגי חולון
Holon Institute of Technology

פרוייקט מעשי לתואר B.Sc. במדעים

ניתוח פסקי דין בעברית

Hebrew Court Verdicts Analyzer

מגישים: איתי אופיר - 204318034, עמנואל בן חפר - 309991644

בהנחיית: ד"ר יונתן שלר **חתימה:** <חתימות המנחים>

תאריך הגשה: <תאריך הגשת הספר לחתימת המנחה>

מכון טכנולוגי חולון
Holon Institute of Technology

הצע ת פרויקט שנתי (דו- סמסטריאלי) בפקולטה למדעים
מכון טכנולוגי חולון- H.I.T.

שימו לב! יש למלא את הטופס באופן דיגיטלי. אין לשנות מבנה הטופס .

טופס סרוק או עם פרטים חסרים לא יתקבל .

סמן ב **X** את מספר הקורס :	**X**	מדעי המחשב	1	0	0	6	6
		מתמטיקה שימושית	6	0	2	1	2

שנה עברית			
ת	ש	פ	א

סמסטר תחילה:
א

תאריך הגש ת ההצעה (סטודנט) : 20.11.2 **סוג פרויקט (תעשייתי\מחקרי:)** תעשייתי

א. פרטי הסטודנט/ים:

שימו לב! ניתן לקחת פרויקט שנתי רק פעם אחת במהלך כל תואר.

שם מלא	איתי אופיר	**שם מלא**	עמנואל בן חפר
מחלקה	מדעי המחשב	**מחלקה**	מדעי המחשב
תעודת זהות	**204318034**	**תעודת זהות**	**309991644**
טלפון	**050-2056611**	**טלפון**	**050-7281548**
דואר אלקטרוני	io.zr174@gmail.com	**דואר אלקטרוני**	**immibh@gmail.com**

חתימת הסטודנט **חתימת הסטודנט**

שימו לב! ניתן לקחת פרויקט שנתי רק פעם אחת במהלך כל תואר.

שם מלא		**שם מלא**	
מחלקה		**מחלקה**	
תעודת זהות		**תעודת זהות**	
טלפון		**טלפון**	
דואר אלקטרוני		**דואר אלקטרוני**	

חתימת הסטודנט **חתימת הסטודנט**

ב. שם הפרויקט

שם הפרויקט בעברית:

ניתוח פסקי הדין

Figure 1. Signed Project Proposal

Table of Contents

Summary

The Supreme Court's website offers people access to its vast database of verdicts. To search and sift through the hundreds of thousands of cases, one could use the built-in search engine available by the site. While the simple searching options are quite capable, the advanced search options are lacking. Specifically the ability to run morphological search queries, which currently, is not supported.
This project focuses on adding morphological search capabilities as well as classifying each case. The project is based on a similar project from previous years. This iteration of the project offers a great improvement to the previous one. The system scrapes all cases available by the Supreme Court up to 1997. Using multithreaded scraping functions. The information gathered will be parsed and normalized. The normalization process consists of deterministic pre-processing steps. Those steps handle the representation variance of the parsed legal personal and non legal personal names.
The cases will be classified using machine learning methodologies. The verdict's summary is extracted from the JSON formatted verdict. Then, it is tokenized by a TF-IDF vectorizer which sets them to a model comprehended format. The vectorizer checks word frequencies, eliminates words that repeat constantly and high rank words that unify the current verdict summary from all the rest. Those tokenized verdicts are passed to the model as training input. The model is tested and evaluated by predicting pre-splitted test data. The model ultimately will predict the most likely category. All valid cases will be uploaded to the latest version of Elasticsearch, to do so we've updated the deprecated elasticsearch-hebrew-analyzer plugin that was used in the previous project version. This plugin allows us to perform morphological searches in Hebrew in Elasticsearch. In addition, overall performance, project structure and architecture was improved to allow the system to be accessible for future development.
Over 85% of cases were validated, parsed and uploaded successfully to Elasticsearch. An improvement from the previous project version. The Elasticsearch version was successfully updated from the deprecated v5.5.3 previously used to the latest v7.10.2 version.

Introduction

Searching for information has always been a difficult task. In the past, if you wanted to look up information on a certain topic, you would need to go to a library or consult with professionals or academics. Technology, and the internet in particular, has improved our ability to search information. Even on the most esoteric topics. The massive shift of digitizing information in the past decades has made a significant impact on many industries. This transformation in the way we view information has made all information increasingly accessible for the public.

In Israel, many government institutes are still having issues modernizing their systems. This slowness to progress is the root of many Israelis' frustrations with the government's institutional bodies. Recently, the Israeli government has declared it is planning to improve all government digital infrastructures (Conseil de L'Europe, 2020, 104). This is an important step in the right direction.

One sector that will benefit from this investment is the legal industry. As many cases are decided using previous precedences, the ability to look up historical data on verdicts and procedures is crucial. In the past, the only way you could access information on old cases was to physically go to the courts' library and search for the right case, as well as trying to find similar or other relevant information. This is still true today for very old cases that are not yet uploaded to the court's database. The recent advancements in the court's infrastructure has made this information accessible for all. This includes creating dedicated websites with the ability to search the many files and memos produced and written by the court systems.

Israel is a unitary state with a single system of general law courts. The Basic Law: The Judiciary, establishes three levels of courts: the Supreme Court, district courts and magistrates' courts (Israel Ministry of Foreign Affairs).

Currently, only the Supreme Court enjoys a dedicated website with its own search engine, while all other courts use a single website with limited capabilities. Searching for specific keywords, for example, is not supported by the official search engine website.

As the Supreme Court holds a higher value than other courts, setting a precedence in the Supreme Court directly affects lower level courts, the ability to allow access to such information is crucial.

There are currently over 200,000 cases in the Supreme Court's search engine dating back as far as 1997. The current search engine used in the website is quite capable. Allowing for searches to include specific keyword matching, procedure number, date ranges, document details, presiding judge and a few more attributes. Adding an improved morphological search will directly affect all users of the system, from the general population to law academics and legislators to big data researchers performing complicated queries.

While the existing search engine is quite capable for basic searches, the search engine lacks advanced searching options. While researching for this project, we've noticed that the search engine includes an "extended morphological" keyword search option but testing the feature proved that it does not work.

As of now, there is no direct way of accessing the Supreme Court's database and searching information using an API. In addition, there is no advance data processing on the existing information, for example category classification.
Offering advanced features to the existing system is not a straightforward effort. We have no ability to directly improve or work on the Supreme Court's database. This means our solution must use the existing infrastructure and be built around it.
There are a few websites providing searching capabilities for various courts. Some also include the Supreme Court and morphological search:
Psakdin - The search function is limited to paying users or registered lawyers, making it difficult to assess the quality of the search engine, but it does seem to have morphological capabilities.
Nevo Publishing Ltd - The official publisher of the Supreme Court's . Provides the most extensive search engine we could find. The service is only available for paying customers.
Dinim Veod - Provides morphological searching capabilities but the accuracy is somewhat lacking.
Our goals are to expand and improve on the existing project that was started two years ago - AccessibleCourtData (Revah & Shalumov, Github). We will update to the latest Elasticsearch version by updating the deprecated elasticsearch-hebrew-analyzer plugin.
We'll improve the current parsing capabilities and add a category for each case using machine learning to analyze the case details. Finally, to ensure these improvements are added in the best possible way, we'll improve the overall performance and project architecture and structure to ensure the project is development friendly and accessible for future iterations.

Design and Implementation

High Level Design

We've chosen to stay with the current implementation of the main.py file. We will reference all main files of each component in the central main.py file. The alternative would be to implement each component as a microservice and add a communication protocol between the services. We've decided not to go with this option in order to save time and effort as we had a lot to implement without it.
As all components are independent from each other, we'll run all components in parallel. We will use the existing method used -
$ nohup python3 ./main.py <component number> ... &
nohup (IBM) stands for "no hang up". It is used to execute commands that ignore the hangup (HUP) signal. To ensure the command continues to run even after the user logs out, we will add the '&' operator as stated in the documentation. The output will be routed to each appropriate file (for example, 'scraper.out' for the scraper's output).
We do not have access to the Supreme Court's database. The only way we can collect cases is by using the existing search engine in the Supreme Court's website and scrape all the cases we can find. The Scraper's main purpose will be to iterate over the Supreme Court's database of cases using their website's UI. All scraped information will be saved to the filesystem in JSON format.
The project is hosted on one of HIT's dedicated servers which run on the Ubuntu OS. For this reason we've chosen to use Firefox as the main browser of choice. Firefox is the default browser for Ubuntu and comes pre-installed. This saves us from installing any additional dependencies such as Chrome or Edge. We've faced several hard resets on the server during development, this decision quickly proved to be the right one. The current chosen Firefox webdriver was matched to the server's Firefox version. If the Firefox version will be updated in the future, the webdriver version will need to be updated as well. To assist in the development process we will add support for scraping on Chrome as well. In addition, we will add support for MacOS and Windows OS.
In order to prevent re-scraping existing cases on each activation, we'll use the existing MongoDB service. We'll have a dedicated collection (table), containing all relevant dates. Each date will have a 'status' attribute. The possible states are: 'available', 'error', 'done'. Initializing the database will add the dates with the 'available' status. If the Scraper fails to scrape a certain date, the status will be updated to 'error'. Otherwise, dates that all cases have been scraped successfully will be updated to 'done'.
As each date contains many cases, we've implemented a unique name for each case. The case will be saved in the following format:
<date>__<case number in page>.json
For example, '01-01-1997__4.json', indicating the 4th case on the first of January, 1997.

This will allow us to easily identify the case by simply reading its unique name.
Testing the existing Scraper showed promise but was relatively slow. We'll implement a multi-threaded scraping function to allow us to substantially increase the scraping speed. To implement the multi-threaded function, we will use the 'ThreadPoolExecutor' function of Python's 'concurrent.futures' library. Each date will use one thread, this will allow us to control the amount of concurrent workers (threads) and subsequently, the amount of dates been scraped at the same time.
To preserve computing resources, we'll run the Firefox browser using the headless mode. This means the browser will be running in the background, without displaying any windows. We've noticed a drop in performance when using the scraper without the headless mode. The degradation was amplified when we ran multiple threads.
To scrape and crawl the pages we'll use the existing Selenium infrastructure, this will allow us to save a considerable amount of development time. The existing Scraper proved to be reliable as a standalone service. We will refactor the existing code to create a unified look and feel, as well as add our proposed changes and features.
The Parser will start the processing pipeline. The cases will be collected from the Scraped cases directory. Badly scraped cases will be filtered out. This will include files with no texts or missing key attributes. The main part of the parser will be to process the case details into its appropriate attributes - in a key-value manner. As most of the information scraped in each case is stored in a single bulk text. The parser will iterate over the text and break it down to each attribute, in accordance with the Elasticsearch schema we are using. The Elasticsearch schema used can be found in appendix A. The Enricher will need additional attributes that are not part of the original parsing. We will set up the required attributes at this stage. This means the output of the Parser should always be a valid and Elasticsearch-compliant case. The saved cases will be stored in directories describing its status: parsed successfully - 'success', failed validation - 'failed_validation', failed parsing - 'failed_parse'.
We have designed and implemented a specific object that handles the data enrichment process. The enricher is a base class for all further enrichers. This will let us use polymorphism to call the enricher without worrying about which derived type it is. We have implemented two enrichers for now: Normalizer, for names, and a Classifier, for the verdicts. The Normalizer is in charge of normalizing the parsed names in order to achieve a unified form for different inputs which are actually the same term. This will produce a more accurate picture of the results. The verdicts Classifier is in charge of extracting the verdict summary and using machine learning to decide its category. This is done by tokenizing the extracted verdicts and passing them as training input to a MultinomialNaiveBased model. The model performance is being evaluated by passing pre splitted test data and predicting the category.
The Elastic module will collect all cases that have successfully passed all stages: Parser, Enricher - normalization, Enricher - classification.
One issue we've faced with uploading cases to Elasticsearch is the fact the ID used by the Supreme Court is the same ID for all following cases discussing the same case. If we only rely

on the given ID as an ID for Elasticsearch we will overwrite many cases. To prevent overwriting we'll create a unique Elasticsearch ID. Our suggested naming convention is to concatenate the case's date to the existing ID. This will allow us to quickly understand the case in question as well as identify the exact preceding of the questioned case:
<case id>-<case date>
For example, '6380-97-01-01-1998', will represent the case number 6380/97 from 01/01/1998.
The last checkpoint would be to compare the final file planned for upload versus the schema design. If the validation passes, the file will be uploaded to Elasticsearch.
Another key update will be to upgrade the Elasticsearch version from v5.5.3 to v7.10.2 (the latest version as of writing the project). v5.3.3 is a deprecated version which is no longer supported (Elastic). This version limit was set by the 'elasticsearch-analysis-hebrew' plugin (synhershko, Github) that was used. 'elasticsearch-analysis-hebrew' is an Open Source project that has been out of production for quite some time (the plugin is maintained in the paid version). The current Hebmorph plugin is supported only up to v5.5.3. We will need to update the plugin in order for it to be usable in later versions of Elasticsearch. This will allow us to ensure continuous development in the future. An important note, each upgrade of Elasticsearch will require upgrading the plugin version - even if no actual changes were made to the code itself.
Our results will be displayed using a custom dashboard created in Kibana.

Components Integration Design

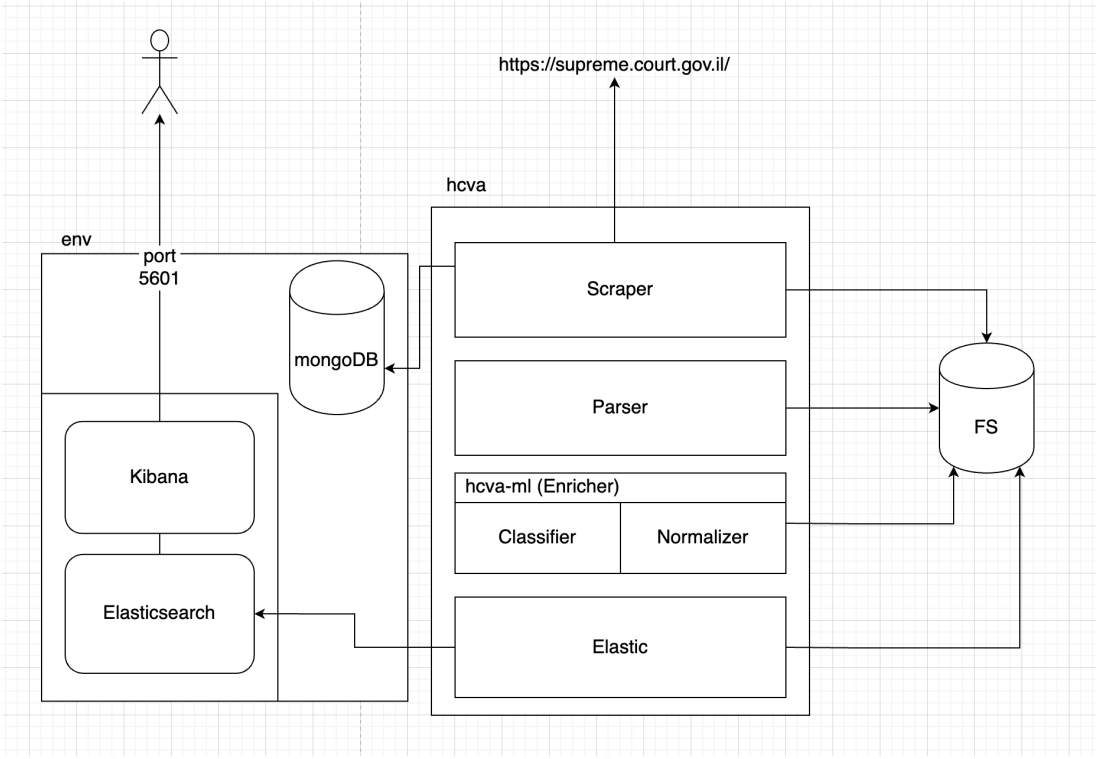


Figure 2. Components Integration Design

Environment and Repository Design

Work environment

Our project ran on a remote linux machine, so some initial set up was in order before we could start. Prerequisites included installing:

- Git
- Docker
- Python3
- Pip3

The latest installation commands can be found in appendix B.

HCVA-env-set-up Structure

As the environment set up is based on the docker-elk repository (deviantony, Github), it follows its general directions. Some additional features were added, these mainly consist of the 'elasticsearch-analysis-hebrew' plugin, MongoDB and mongo-express. To align with the format of docker-elk, our work was done on the 'release-7.x' branch and not on the main branch.

- Elasticsearch
    - Contains the Dockerfile and configuration files for Elasticsearch, the plugin installation was added to the Dockerfile
- Extensions
    - Includes the "built-in" extension provided by docker-elk.
- Kibana
    - Contains the Dockerfile and configuration files for Kibana
- Logstash
    - Contains the Dockerfile and configuration files for Logstash
- Mongo-express
    - Contains the Dockerfile file for Mongo-express, this was done to preserve the styling format of docker-elk
- Mongo
    - Contains the Dockerfile files for Mongo
- Other files:
    - The main configuration files, include the .env and docker-compose.yml which serves as the main file activating all services

HCVA Structure

- Cases
    - Contains all the cases that were scraped. The cases are stored in accordance with their current status. The top level contains the different components of the project, so: Classified, Elastic, Normalized, Parsed, Scraped. The second level of each directory contains the status within that component. For example, the Elastic directory also contains: failed upload, failed validation, successful upload.
- Env
    - Contains all code related to the environment setup.
        - The env-set-up directory is the submodule that contains the whole environment setup for the project. In addition, we have 3 ready-made scripts for ease of use.
        - Init_env.sh - activates the docker containers of the env-set-up submodule
        - Shutdown_env.sh - shutting down the containers
        - Remove_env.sh - adds the -v flag to the shutdown process, removing the volumes of the containers. This should be used only for hard resets.
- Hcva
    - The project's directory, divided into the different components:
        - Elastic
            - Divided into 3 main parts:
            - Index
                - Contains the index used to initialize the Elasticsearch engine
            - V7
                - All code relevant for the version 7.10.2 of Elasticsearch, including its validation functions
            - Legacy code
                - Is split between the index, v5 and validation directories
        - Parser
            - Contains the new schema data, alongside all the parser's code
        - Scraper
            - Contains the webdrivers and the scaper's code, divided into the crawler, scraper and the main driver which is responsible on the multi-threaded functions
        - Utils
            - All utility code, including the database, logger, date and so on
- Logs

        - Contains all logging files for each component
- Requirements
    - Contains the requirements.txt file
- Resources
    - Contains all the markdown files elaborating on certains parts of the project.
- Other files:
    - Dot files - containing configuration data
    - Project metadata files, includes: license, README, setup.py.
    - Activation files - the drivers of the project:
        - Init_app.sh - running all the project's components together
        - Main.py - the main driver of the project
        - Shutdown_app.sh - shuts down all processes

HCVA-ML Structure

- hcvaEnricher
    - Resources
        - Contains all resources (except the 'models' folder) needed for the module to operate, these include various csv, txt and json files
    - Main files:
        - The main.py file acts as the driver of the library, followed by the Enricher acting as the main class implemented by its subclasses: Normalizer and Classifier
- Images
    - Contains images for the README.md file
- Other files:
    - Set up files:
        - These files are used to package the module into a consumable library, they include: MANIFEST.in, setup.py
    - Models*
        - This folder is not included in the repository because of its size. You should download the folder from the project's Google Drive (appendix C)

Components

HCVA-env-set-up (HESU)

We've decided to continue the usage of the docker-elk repository and expand on its original structure and purpose, we forked the docker-elk repository and added our changes in a dedicated new repository (Immanuelbh, Github), to preserve the docker-elk branching structure, we've decided to work on the already created 'release-7.x' branch, this is where all the added details below are implemented. In case it is needed, the 'release-5.x' branch supports the legacy code previously written, running the branch will install Elasticsearch v5.5.3 and the matching 'elasticsearch-hebrew-analyzer' plugin.
Our environment now contains a couple of new features like MongoDB, mongo-express and also an updated version of the elasticsearch-analysis-hebrew plugin. The repository works as a standalone unit. This means it has no dependencies on the actual project and can run regardless of the HebrewCourtVerdictsAnalyzer (HCVA). For ease of use we've decided to add the HESU as a submodule in the HCVA. This allows us to integrate with the module seamlessly while still being able to keep the environment set up as a standalone repository, offering the environment for other developers who'd like to use the latest version of Elasticsearch with hebrew morphological searching capabilities.
In order to run HESU we must first ensure we have Docker and GIT installed on our machine. The full Docker and GIT installation instructions can be found in appendix B and also in the HCVA resource files. You can clone the repo as a standalone environment, but we suggest using the submodule mechanism for ease-of-use. To do so, clone the project repository using the following flag:
$ git clone --recurse-submodules
https://github.com/Immanuelbh/HebrewCourtVerdictsAnalyzer.git
Notice the --recurse-submodules flag added to the clone command. The default branch of the cloning is 'release-7.x' so there is no need for any additional configuration when cloning.
The HESU repository is essentially a collection of nicely packaged Dockerfiles that allow us to add, change or delete any service we need in that framework. For development, we recommend Docker Desktop. For the dedicated server we can only install the CLI version as Docker Desktop is not supported on Ubuntu .
An important point to note: the default JVM maximum heap size (Xmx) and initial heap size (Xms) set in the repository (under docker-compose.yaml) is intended for the dedicated server and is set to 4GB, this can be increased, but be aware that changing these values can cause the Elasticsearch to fail on start up. This might be because the values set are too high or too low. Please try to follow the logs and try to adjust accordingly.

This was felt during development, trying to run HESU on our local machines failed when using the default values, but succeeded when decreasing the values to 2GB each.
To run the environment as a standalone module, simply navigate to the HESU folder, make sure you are on the 'release-7.x' branch using the following command:
$ git status
If you are on the main branch, you can switch branches, using:
$ git checkout release-7.x
To start everything for the first time, type in (this is only needed when first initializing / changing branches / making changes):
$ docker-compose build
Once everything has finished building, run the final command:
$ docker-compose up [-d]
This will start up the container. We recommend adding the -d (detached) flag so the container can run in the background. Otherwise, the logs will be written directly to the terminal.
To shutdown the container in detached mode, you can enter:
$ docker-compose down [-v]
This will shutdown all containers. Optionally, you can add the -v (volumes) flag to remove all data saved or created by the containers. This is especially useful during development, when you need to hard reset the data.
If you are working with the environment directly from the HCVA repository (recommended). We've made the process much easier. From the root directory of HCVA, navigate to the HESU submodule:
$ cd env/
You can now use the 3 pre-made scripts to run the same commands previously mentioned:

- init_env.sh - runs the containers in detached mode
- shutdown_env.sh - stops the containers
- remove_env.sh - stops the containers and removes all volumes

MongoDB was added to HESU to keep track on the dates used for scraping. MongoDB in itself does not have a UWe interface. To help in development, we've added mongo-express, a web-based MongoDB admin interface. To connect and view the database, simply navigate to:
localhost:8081
Elasticsearch does not support Hebrew out-of-the-box (Elastic). In order to add Hebrew support, we must use an external plugin. The most-up-to-date plugin we could find was 'elasticsearch-hebrew-analyzer' made by Itamar Syn-Hershko (synhershko, Github). As the latest version supported by the plugin was for Elasticsearch v5.5.3 and this version was deprecated almost 2 years ago (Elastic, EOL), we wanted to update the plugin to support newer versions of Elasticsearch. During the development of the project, the latest version was v7.10.2. The plugin is split into 2 major components: Elasticsearch-hebrew-analyzer and Hebmorph.

Elasticsearch-hebrew-analyzer is the actual plugin for Elasticsearch. It contains the needed framework of an Elasticsearch plugin including all necessary configurations. The plugin relies on Hebmorph as the engine for Hebrew text processing.
HebMorph is an “open-source effort for making Hebrew properly searchable by various IR software libraries, while maintaining decent recall, precision and relevancy in retrievals” (synhershko, Github, 2019) meaning it is not tightly coupled to the plugin and can be used in other programs.
Updating the plugin was a challenging task as there was no clear documentation of the process. This combined with the fact we were working on unfamiliar code ment we had to experiment and find the solution on our own.
The fixes included changes from the configuration side of the plugin: adjusting versions of Elasticsearch as the plugin version is tightly coupled to the Elasticsearch version. This means using a plugin on Elasticsearch version X requires the plugin to declare its version as X as well. Other configuration updates included disabling certain parameters. These parameters arose from the maven build process and had to be disabled by the suggestion of the build process. Our theory is that these parameters are inner dependencies that in turn affect our plugin. Disabling them did not seem to have any effect on the plugin's performance in itself, so these parameters were kept off. Apart from the configuration updates, some code implementations had to be adjusted as well. The main change was removing the deprecated route registration and adding the ‘routes()’ function instead. This function returns all the routes used by the plugin.
Once all fixes were complete, we had to create new releases for each version. This allowed us to consume the new versions with a direct URL.
A new release was created for the plugin (Immanuelbh, Github). This wraps the plugin in a zip file we will need for the next step.
No changes were made to the Hebmorph engine as it did not directly affect the Elasticsearch versioning process. We only created a new release to be consumed by the plugin. The Hebmorph engine has not been updated in a long time and can be improved in the future. Further details can be found in the ‘Future research’ chapter.
Updating the Elasticsearch two major versions meant we were facing breaking changes we had to account for (Elastic, 7.10 - Setup Upgrade). Luckly, after experimenting and testing the new version, we concluded we only had to update our index - following the removal of mapping types (Elastic, 7.10 - Removal of Types), further details on this subject can be found in the Elasticsearch section.
To install the plugin in Elasticsearch, we had to add the installation command to the Elasticsearch dockerfile. We also had to add the ‘--batch’ flag to overcome the additional permissions required by the plugin. This is due to the automatic installation of the plugin, adding the flag automatically assigns ‘yes’ to the permissions request notification. Removing the flag will cause the plugin installation to fail as the response will lack the ‘yes’ confirmation.

HebrewCourtVerdictsAnalyzer

The current project was forked from the previous iteration of the project: AccessibleCourtData (liron7722, Github). A lot of effort was put into refactoring the existing code. Making it more readable and accessible for future versions, as well as fixing errors and improving performance.
To run the HCVA program, you must first ensure that HESU is running and all containers are online. To initialize HESU, navigate to the 'env' directory and run the 'init_env.sh' script. Please note that you might need to adjust the memory allocation of the JVM in order to run the environment in development versus the dedicated server. More information can be found in the HCVA-env-set-up section of this chapter.
Before running HCVA you must install the required python modules of the program. We recommend installing the dependencies into a virtual environment to prevent system dependency and for easier installation. To do so, run the following commands from the root folder:
For Windows:
$ pip install virtualenv
$ cd HebrewCourtVerdictsAnalyzer
$ virtualenv --python C:\Path\To\Python\python.exe venv
$.\venv\Scripts\activate
For MacOS:
$ python3 -m venv venv
$ source venv/bin/activate
To install the required modules, type in:
(venv) $ pip3 install -r requirements/requirements.txt
The project is also dependent on the HCVA-ML library which is installed during the requirements.txt installation. The HCVA-ML library needs an additional folder, 'models' that's not included in the installation. The 'models' folder needs to be placed in the root folder of the project. More information on the folder can be found in appendix C.
Once the 'models' folder is added, you should be ready to start the 'init_app.sh' file. Run the following command from the root folder to start the script:
$ ./init_app.sh
You can view a successful output in the appendix D. Notice no information is printed to the terminal. All logs are printed to the corresponding files, named after the component that is running. These logs can be found in the 'logs' directory.

Breakdown

For easier development, each component can run independently. You can do so by adding the component number of your choice:

$ python3 main.py <component number>
1 - Scraper
2 - Parser
3 - Enricher
4 - Elastic

Scraper

The Scraper is dependent on MongoDB for deciding which date the Scraper should scrape next. The default connection url is already set in the .env file under the 'MONGO_DB_URI' attribute, but this can be modified in the future if the username and password (for example) would be changed. For now, these values were kept unchanged. When running the component for the first time the database is initialized, we populate the database with all dates, Sunday to Friday, from 1/1/1997 to the current date. We set the status of all dates to 'available'. From this point on, all subsequent runs (including the current one) will pull all dates with 'available' and 'error' status. Next, we call the scraping function with the multi-threaded ThreadPoolExecutor class. This allows us to cut down on the scraping time dramatically. The default allocated thread count is 8 but this can be modified by updating the 'NUM_OF_CRAWLERS' value in the .env file.
A big change we've implemented from the last iteration of the project was to change the way the threads were previously allocated. The old implementation allocated 4 threads that held 4 browser windows. This means these threads were only released once they finished scraping all dates, otherwise they would remain in use. This impacted the performance of the scraping process as well as the overall usability of the dedicated server, as the server often froze due to lack of resources. To improve this behavior, we now use dynamically assigned threads. Each new date page scrape is assigned 1 thread that is released once the page scrape is complete. We can now comfortably allocate double the amount of threads we previously used, as well as continue to use the server while running the Scraper in the background.
The Scraper is split into 2 components: the Scraper and the Crawler.
Scraper - Is in charge of manipulating the browser. This includes constructing the appropriate url page, scrolling on the page, as well as 'feeding' the crawler with specific elements.
Crawler - Operates the webdriver, and the elements on the web page.
Currently we've chosen to support 2 browsers: Firefox and Chrome. Firefox was the natural selection of the dedicated server as it is pre-installed on the machine. Whereas Chrome is the most common browser on personal computers and is the natural choice for development. An important note for future development, we had to download a specific webdriver version when working on the scraper as we had inconsistent results when auto-downloading the webdriver. The plus side by doing so is that it gives us the advantage of choosing our browser before activating the Scraper. This is done in the .env file using the 'BROWSER_TYPE' attribute. The disadvantage is that we must be bound to a specific webdriver version. This does not affect the

dedicated server usage as the version will most likely stay the same. We have no control on the Firefox version installed (no admin privileges). This does mean however, that newer versions of Chrome are not supported, which does impact the development process. This can be solved by updating the webdriver when testing on Chrome (Chromium, chromedriver), or by using the same version used in the project. In the 'Further Research' chapter we discuss the option of adding an automatic webdriver download which should solve this issue.
All scraped cases will be saved to the dedicated 'scraped' folder, to be used later by the Parser. To assist in the development process we've added a syncing function. This can be used when we have reseted the Docker volumes and the MongoDB database has been deleted while still having cases already scraped. The function iterates over the declared folder and updates MongoDB with the existing scraped dates. The folder can be declared in the .env file using the 'CASE_PATH' attribute and can be run by the following command:
$ python3 main.py 5

Parser

The main objective of the Parser is to process the raw data scraped prior and to build the initial base structure of the case. The cases are stored as a JSON and must meet the final requirements of the Elasticsearch index design. In the first phase of the process we remove invalid scraped cases. These will include cases that are missing document details or are missing key attributes. The invalid cases will be stored in the 'cases/parsed/failed_validation' folder.
The valid cases are processed into the desired format of the index. The parsing function takes the raw data, which is a (very long) string line and breaks it apart into the different attributes we need. If the Parser fails to build the required format, we throw the case to the 'cases/parsed/failed_parse' folder. All cases that have passed the validation and parsing process are saved to the 'cases/parsed/success' folder.

HCVA-ML

We have two types of implemented Enrichers - a name Normalizer and a case Classifier. The Normalizer normalizes the data from the parsed folder and the classifier classifies the category of the case. We dive deeper into the HCVA-ML library in the next section.

Elasticsearch

In the previous iteration of the project the communication with Elasticsearch was done through building http requests and sending them directly to the Elasticsearch service. To improve readability and to ease future development we've updated the service to use the official 'elasticsearch' module by Elastic.
In the initial startup of the Elastic module we initialize the Elasticsearch index. The updated version of the index is stored under 'elastic/index/index_v7.json'. The main difference between the v5 and v7 is that in the v7 index mapping types are no longer supported. This means there is no 'ruling' index as before, seperating into a 'Case Details' and 'Doc Details'. The index is now referred to directly under the properties - essentially creating one big index that includes both 'Case Details' and 'Doc Details'. The differences can be viewed in appendix E.
Each case we plan on uploading must contain a unique ID. This prevents overriding existing cases and allows us to easily identify the case in question. The pattern we're using is concating the case ID, which is not unique in itself, together with the date of the case:
<case ID>-<case date>
This ensures that all court appearances of the same case ID are stored in Elasticsearch and are searchable.
The last stage before the cases are uploaded is to validate the document we want to upload. We do this to prevent the document from being rejected by Elasticsearch and by doing so preventing unnecessary load on Elasticsearch. The validation is done using the validate function from the 'jsonschema' module. If the case fails the validation, it is stored in the 'cases/elastic/failed_validation' folder.
At last, we upload the case to Elasticsearch. If the upload was successful we store the case in 'cases/elastic/success'. If the upload fails for any reason, we throw the case into the 'cases/elastic/failed_upload' folder.

Kibana

To view and search on the existing cases in Elasticsearch, we can use Kibana as it provides us with an easy to use UI to work with. To connect, navigate to:
localhost:5601
When first entering Kibana, you must configure the index pattern you wish to search on. The default Elasticsearch index of the project is 'hit_index'. This can be modified in the .env file

under the ‘ELASTIC_INDEX_NAME’ attribute. Make sure you include the index defined in HCVA in the index pattern. The index pattern supports regex, so a pattern like ‘hit_*’ is a completely valid index pattern and will include the default pattern.
After initializing the index pattern you can run queries on the indexes under that pattern. To do so, navigate to the ‘Discover’ tab. Make sure the index pattern you have created is set as the pattern you’re going to query on. Enter the query in the search box and you should see your results appearing immediately.
Besides the ‘Discover’ feature for searching the cases, we’ve added a dashboard detailing the results of the final cases. These can be viewed under the ‘Dashboard’ tab - “Enrichment Results”.

HCVA-Machine Learning

Machine Learning In General

Machine Learning is a field in applied mathematics and computer science that uses mathematics in order to train a machine to predict an outcome. There are a variety of machine learning related models for various purposes. In general we approach machine-learning related problems differently depending on the data we have. The data can be mainly halved into two major representations. The former is when the data is pre-labeled. Whereas the latter is when the data is not labeled and we have to label it before we further process our needs. The first is mainly named “Supervised Learning” and the latter is mainly named “Un-Supervised Learning”. Supervised - Learning approach uses tagged data in order to train a model. This means that we know precisely which object relates to which class before the training process. Whereas while facing Unsupervised - Learning we have no labels - we do not know which object relates to which class. To classify this kind of data we need clustering algorithms. Using these algorithms allows classification during the process. The Project Approach is to use the first approach on our classification problem. We train a model in order to further predict a verdict category. In the following sections, the classification and the normalization procedures will be elaborated fully.

Introduction

The hcva-ml module is the machine learning related part of the project. We have used machine learning methodologies in order to enrich our data. We have designed and implemented an “Enricher” base class that supports future development. Every derived enricher can be accessed and called in the same manner. We have designed and implemented two derived sub-classes of the Enricher base class - “Classifier” and “Normalizer”.

Classifier

The verdicts can be classified into multiple categories and sub-categories. We have used tagged data in order to sort the verdicts by categories. The sorted verdicts had been tokenized by a Tfidf Vectorizer and used as an input for a Multinomial Naive Bayes model. The classification process is explained in the classification section.

Normalizer

There are a variety of representations for each name in the parsed verdicts. The normalization process tries to yield an injective function for all parsed input. The normalization is achieved by applying pre-defined deterministic steps elaborated on the Normalization section below.

Enrichment Process Class Architecture

The architecture has been designed specifically in order for further enrichments to take place. This can be achieved by creating an abstract base class “Enricher”. The Enricher abstract class has the abstract method “enrich”. The “enrich” abstract method must be overridden before it is used and it throws a “not implemented error” accordingly. The responsibility of overriding the method is on the derived class. This architecture allows us, the architects of the system and the users, to use polymorphism and to achieve further functionality with relatively high abstraction - we do not need to know who is the enricher we are activating, just that he has overridden the “enrich” method.

The class diagram is as follows:

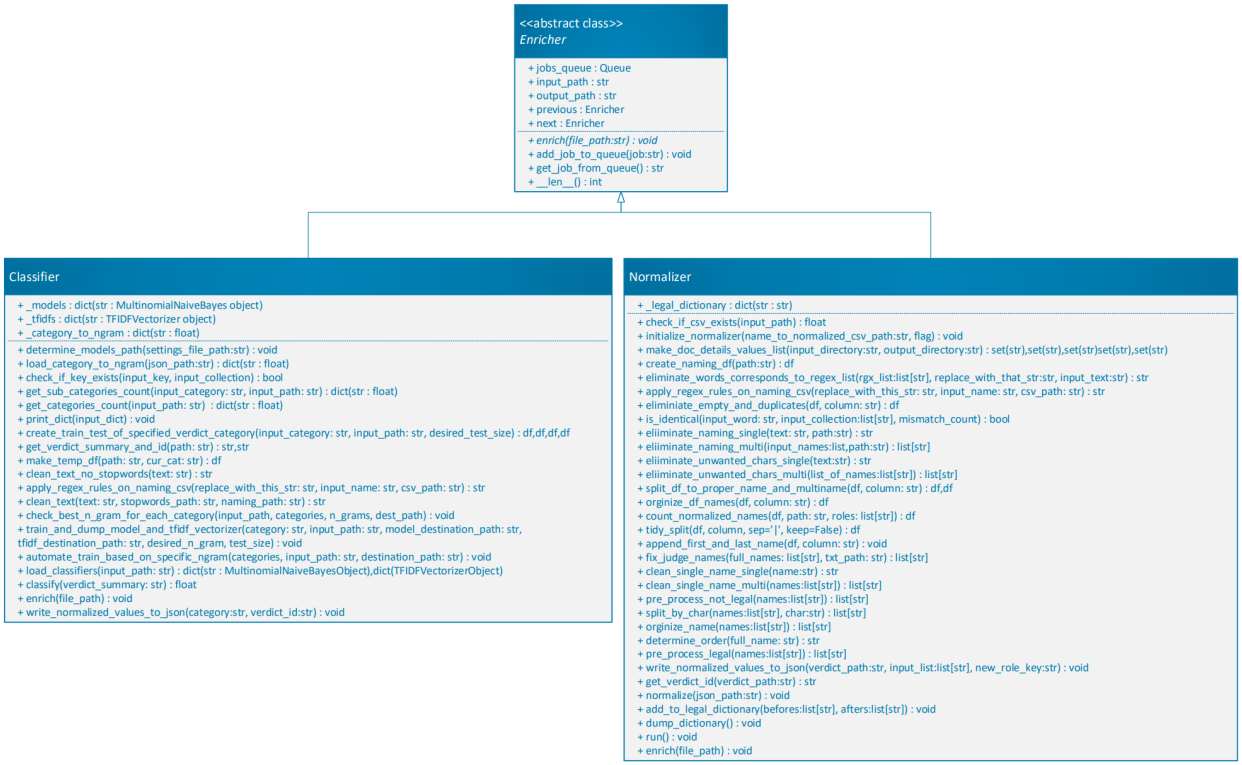


Figure 3. Enrichment Class Diagram

For convenience, enlarged pictures are presented below:

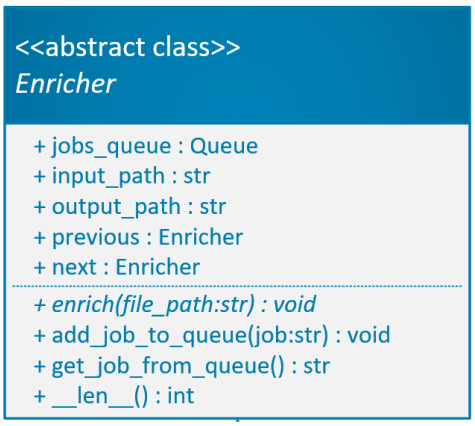


Figure 4. Enricher Class Diagram

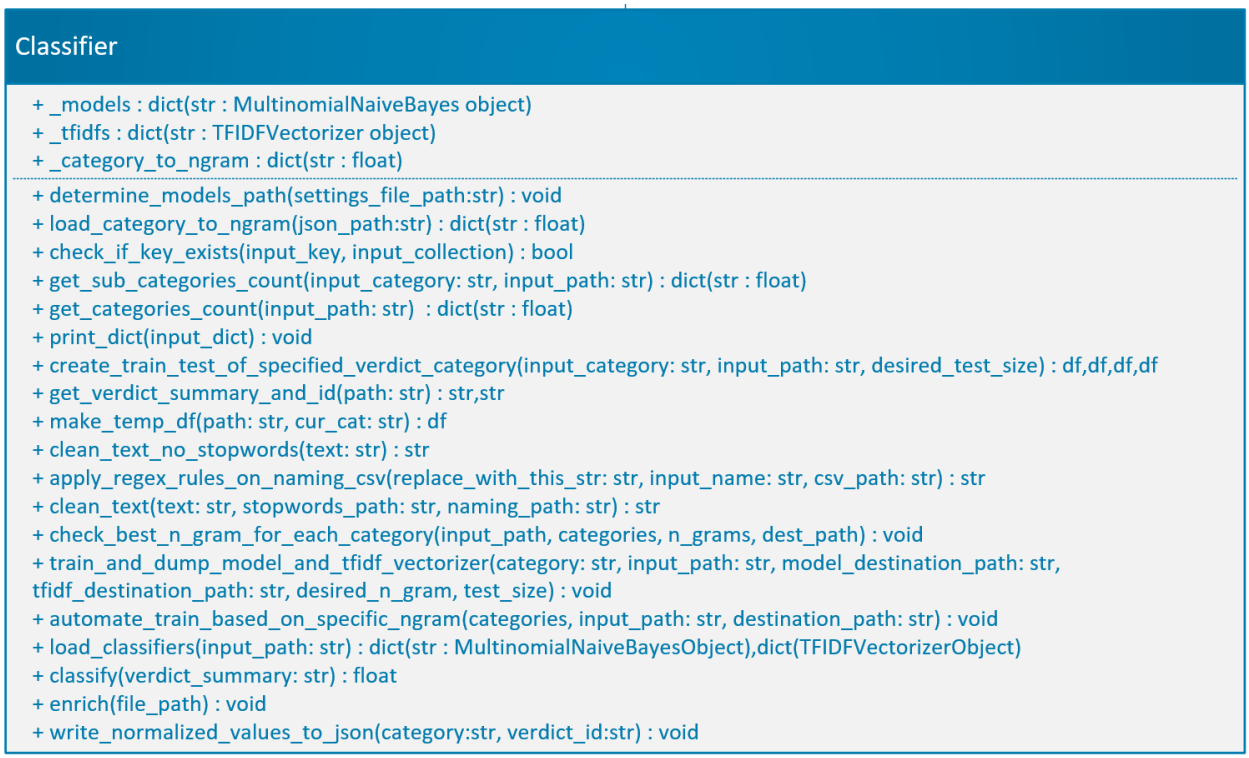


Figure 5. Classifier Class Diagram

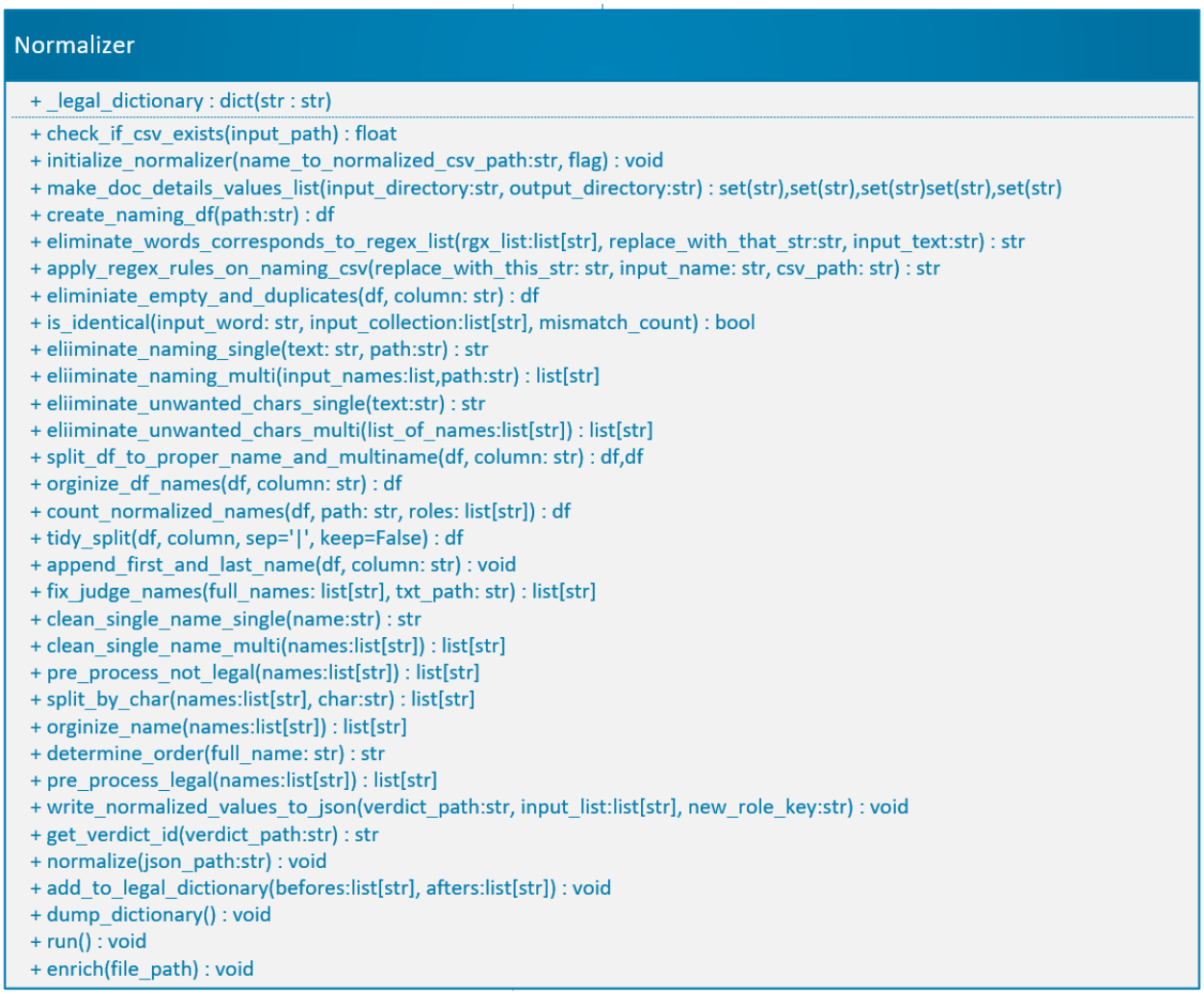


Figure 6. Normalizer Class Diagram

Enrichment Process Pipeline Diagram

Design Goal

Our design supports a multi-enriched pipeline - many enrichers can be concatenated when the first is being connected to the parser and last is connected to the upload module.

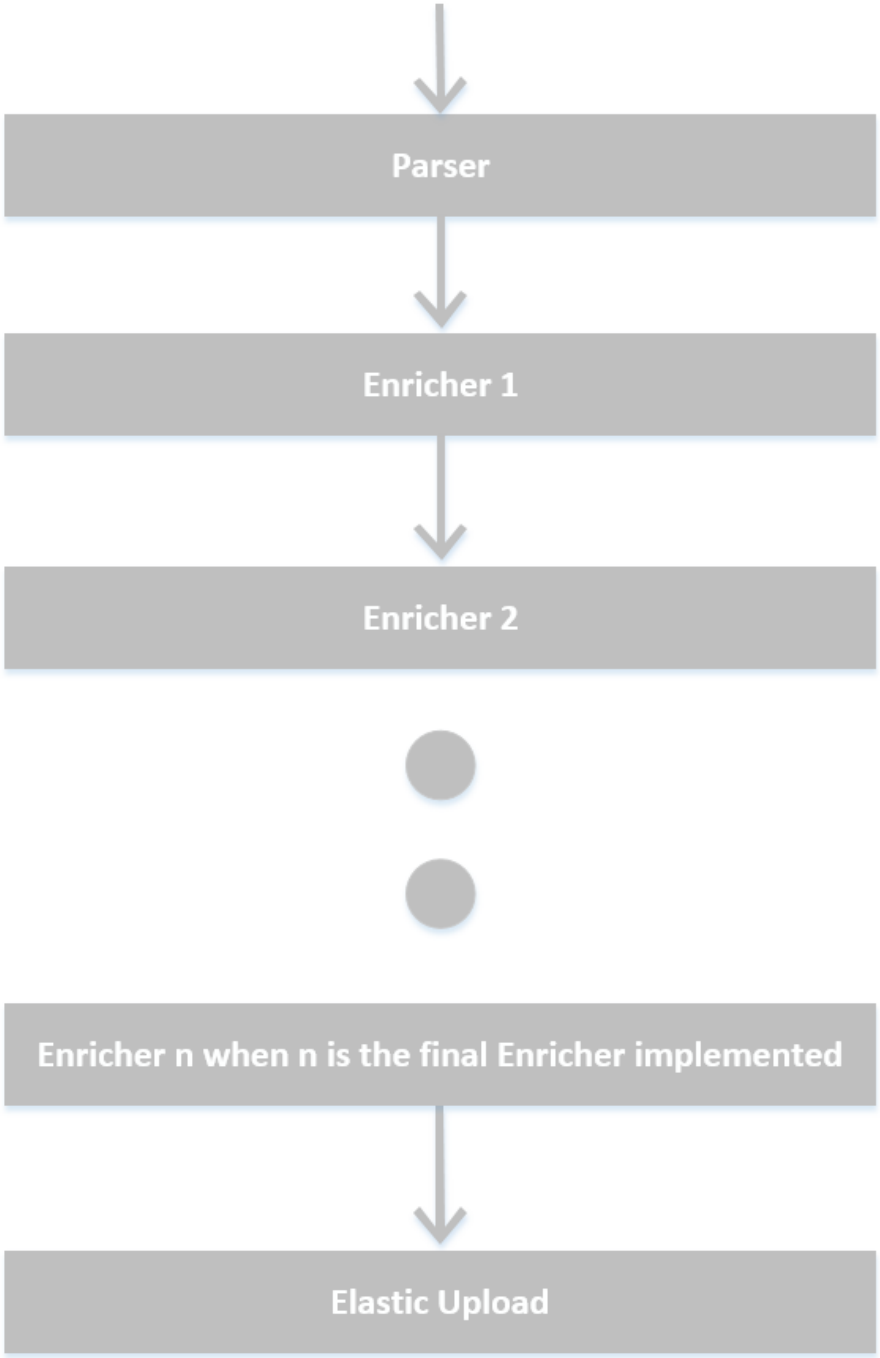


Figure 7. Enrichment Process Pipeline Design Goal

Current Project Iteration Approach

As explained, we have implemented two Enrichers - a Normalizer and a Classifier.

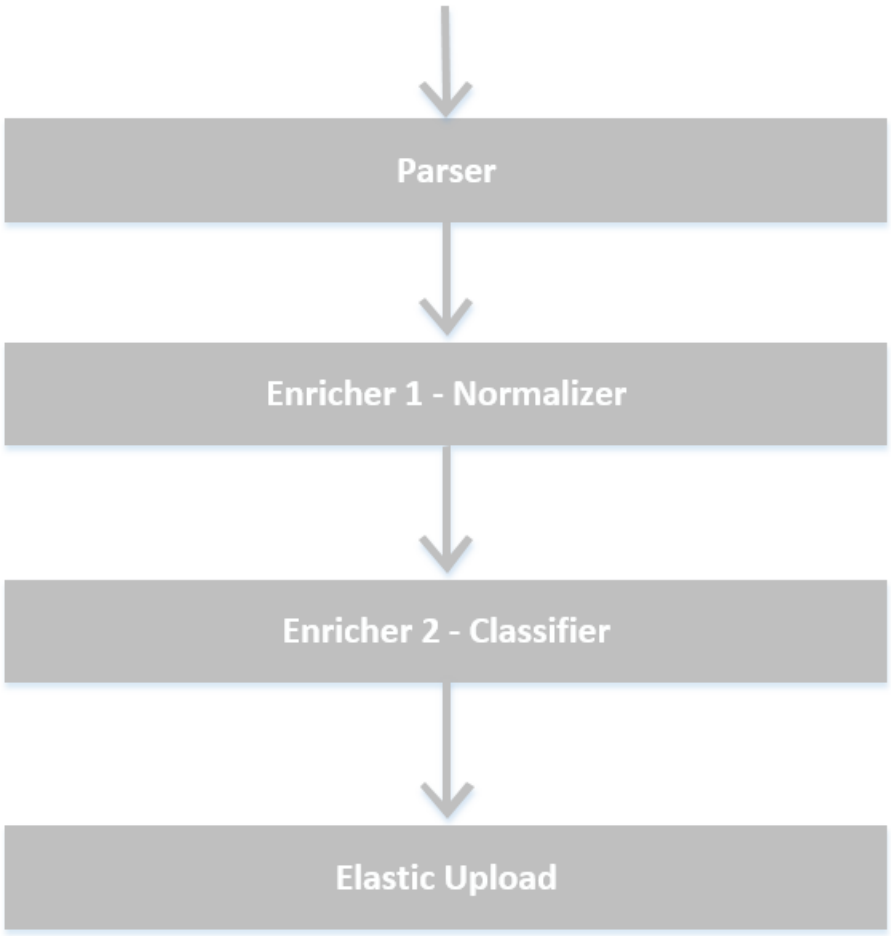


Figure 8. Enrichment Process Pipeline Project Approach

Pipeline - Normalization

In the histogram below the Normalization pipeline is shown.

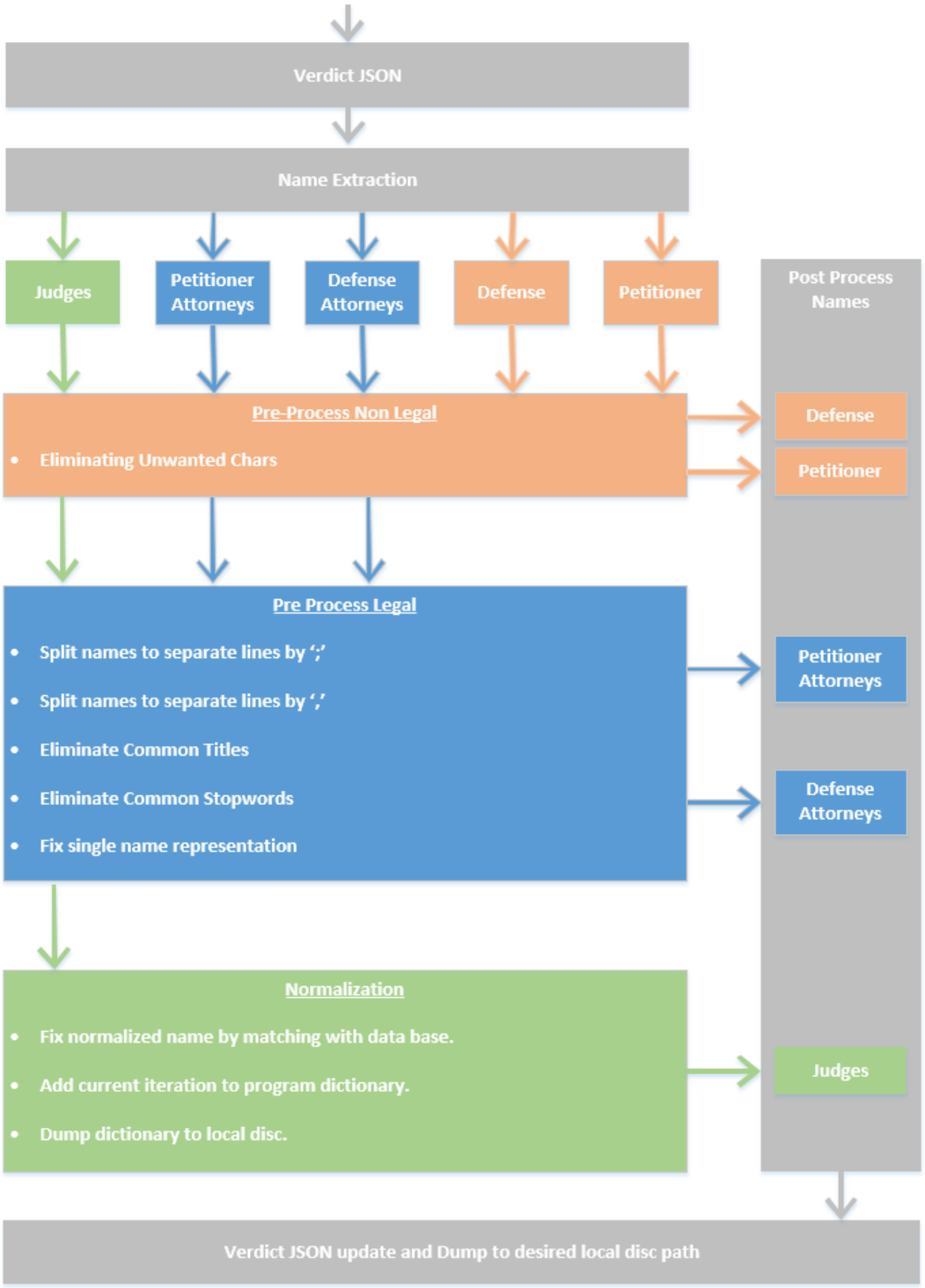


Figure 9. Enrichment Process Pipeline - Normalization

Pipeline - Classification

In the histogram below the Classification pipeline is shown.

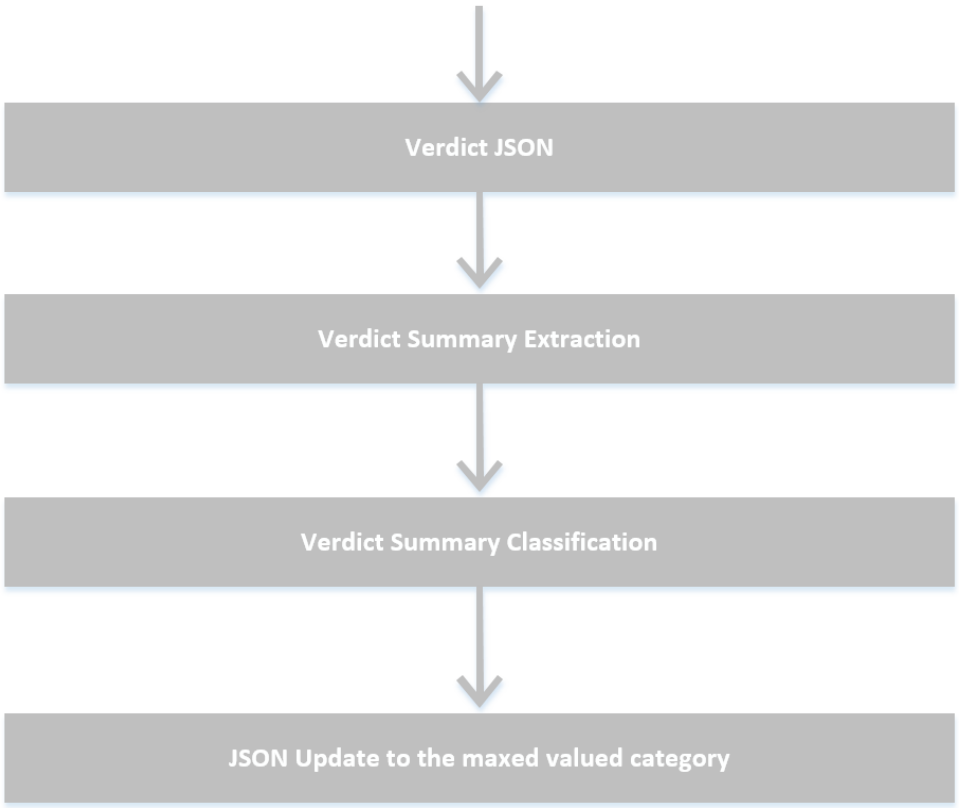


Figure 10. Enrichment Process Pipeline - Classification

Classification

Pre-Train

All the sections below elaborate on the data preparation before the training process. As already mentioned, supervised learning depends on pre-labeling of the data. In order to do so, we have built a mechanism that extracted verdicts with their categories. The verdicts appeared on formal court documents. The amount of pre-labeled data we had and the mechanism output is shown below.

1. Data labeling

We had a portion of tagged data which was sorted by categories and saved to a folder tree.

```
Administrative : 9959
Civil : 8047
Constitutional : 1932
Criminal : 4410
Family : 148
International law : 120
Labor and Employment : 532
National security, military, and the territories : 2943
Religious : 224
Social security, Health Insurance, Pension : 310
```

Figure 11. Pre-Train Data Labeling Count

Those categories and their subcategories were sorted by their names and resulted in a folder tree. The figure below is a sample of the overall resulted folder tree with all of the categories and sub-categories.

Figure 12. Pre-Train Folder Tree

That yields the following verdict distribution:

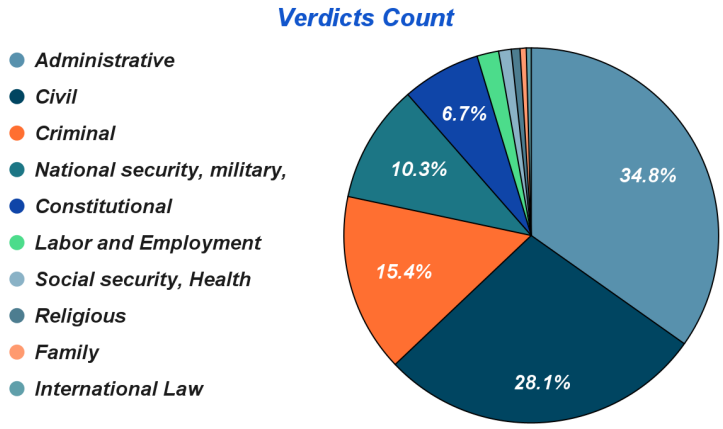


Figure 13. Pre-Train Verdict Count

2. Data Structure initialization

We have used a Data Frame ("pandas.DataFrame") in order to store the data. As you may have noticed, the verdicts count vary drastically. Therefore, in order to achieve good training results for each category a balancing mechanism was needed. The mechanism sampled the same quantity of verdicts from the chosen category and from the rest of the categories. For example, 'Civil' had 8047 verdicts - hence there will be two dataframes - one of 8047 civil verdict and the other of sampled 8047 out of 20,578 verdicts remaining.

3. Train - Test split

We have used train-test-split ("Train Test Split") in order to split the data into 70% train and 30% test. That means that we give the model 70% of the data as input which allows him to learn and tune. The rest - 30% is being used as evaluation after the training process.

4. Vectorization

We have used Tfidf Vectorizer ("Tfidf Vectorizer") in order to transform the verdict summary - a string to an array of numbers. The vectorizer checks how many times a word is present in a sentence, divided by the amount of words in the whole sentence. This value is further multiplied

by the logarithm of the total number of sentences divided by the sentences where the current word is present.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Figure 14. TF-IDF Formula

term i within document j

w = current calculation value

Train

All the sections below elaborate on the Training process. The starting point of that part is when we have our pre-labeled verdict for all of the categories. The end of the process is when we have a trained model ready for verdict category prediction.

5. Training

The transformed data from the vectorization process is the input for the model training. We use a binary classification - 1 means the verdict is from the inspected category, 0 is not of that category. We have used a Multinomial Naive Bayes Model ("Multinomial Naive Bayes") as the model for the classification mission. The model calculates the priors for both binary classes. This means, what is the likelihood of an object to be of each class.

$$\frac{|Desired\ Class\ Object\ Count|}{|All\ of\ the\ Classes\ Object\ Count|}$$

Figure 15. Object Likelihood For Specific Class

For instance, when 4 sentences are present, two from each class, then the prior of either class will be an half. Then the probability for each word is being calculated according to the following formula:

$$\frac{word\ occurences\ on\ the\ specific\ class + 1}{total\ words\ on\ that\ class + |Different\ words\ on\ the\ vocabulury|}$$

Figure 16. MNB formula

Important note: the vocabulary is of BOTH classes. Class in a more specific manner is the verdict category. The final model prediction result is the maximum between the product of all the words' probabilities times the class prior.

$$\text{classification result} =$$

$$\max\left(P_{w1} * P_{w2} * P_{w3} * P_{w4} * \ldots * P_{wn} * Class\ Prior\right)$$

Figure 17. MNB Classification Result

6. Hyper-parameterization

The amount of words to take for each sentence, in a bag of words approach such as that, is called ngram. Ngram values from (1,1) to (10,10) had been set as the tfidf vectorizer input had been tested on the mnb model. Quote from the official documentation:

ngram_range : *tuple (min_n, max_n), default=(1, 1)*

The lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min_n <= n <= max_n will be used. For example an `ngram_range` of `(1, 1)` means only unigrams, `(1, 2)` means unigrams and bigrams, and `(2, 2)` means only bigrams. Only applies if `analyzer is not callable`.

Figure 18. N_gram

7. Evaluation

The results have been evaluated with classification reports and written to txt files. The category classification evaluation results for each and every category is shown on the sections below.

7.1. Administrative

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.84	0.71	0.77	2988
1.0	0.75	0.86	0.80	2988
accuracy			0.79	5976
macro avg	0.79	0.79	0.79	5976
weighted avg	0.79	0.79	0.79	5976

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.87	0.74	0.80	2988
1.0	0.78	0.89	0.83	2988
accuracy			0.81	5976
macro avg	0.82	0.81	0.81	5976
weighted avg	0.82	0.81	0.81	5976

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.86	0.75	0.80	2988
1.0	0.78	0.88	0.83	2988
accuracy			0.81	5976
macro avg	0.82	0.81	0.81	5976
weighted avg	0.82	0.81	0.81	5976

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.88	0.76	0.82	2988
1.0	0.79	0.89	0.84	2988
accuracy			0.83	5976
macro avg	0.83	0.83	0.83	5976
weighted avg	0.83	0.83	0.83	5976

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.88	0.76	0.81	2988
1.0	0.79	0.89	0.84	2988
accuracy			0.82	5976
macro avg	0.83	0.82	0.82	5976
weighted avg	0.83	0.82	0.82	5976

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.86	0.75	0.80	2988
1.0	0.78	0.87	0.82	2988
accuracy			0.81	5976
macro avg	0.82	0.81	0.81	5976
weighted avg	0.82	0.81	0.81	5976

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.86	0.78	0.82	2988
1.0	0.80	0.87	0.84	2988
accuracy			0.83	5976
macro avg	0.83	0.83	0.83	5976
weighted avg	0.83	0.83	0.83	5976

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.86	0.79	0.82	2988
1.0	0.80	0.87	0.83	2988
accuracy			0.83	5976
macro avg	0.83	0.83	0.83	5976
weighted avg	0.83	0.83	0.83	5976

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.85	0.80	0.82	2988
1.0	0.81	0.85	0.83	2988
accuracy			0.83	5976
macro avg	0.83	0.83	0.83	5976
weighted avg	0.83	0.83	0.83	5976

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.82	0.80	0.81	2988
1.0	0.80	0.82	0.81	2988
accuracy			0.81	5976
macro avg	0.81	0.81	0.81	5976
weighted avg	0.81	0.81	0.81	5976

Figure 19. Evaluation - Administrative

7.2. Civil

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.90	0.81	0.85	3066
1.0	0.83	0.92	0.87	3066
accuracy			0.86	6132
macro avg	0.87	0.86	0.86	6132
weighted avg	0.87	0.86	0.86	6132

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.92	0.86	0.89	3066
1.0	0.87	0.93	0.89	3066
accuracy			0.89	6132
macro avg	0.89	0.89	0.89	6132
weighted avg	0.89	0.89	0.89	6132

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.94	0.84	0.89	3066
1.0	0.86	0.94	0.90	3066
accuracy			0.89	6132
macro avg	0.90	0.89	0.89	6132
weighted avg	0.90	0.89	0.89	6132

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.93	0.85	0.89	3066
1.0	0.86	0.93	0.90	3066
accuracy			0.89	6132
macro avg	0.89	0.89	0.89	6132
weighted avg	0.89	0.89	0.89	6132

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.94	0.84	0.89	3066
1.0	0.86	0.95	0.90	3066
accuracy			0.89	6132
macro avg	0.90	0.89	0.89	6132
weighted avg	0.90	0.89	0.89	6132

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.93	0.86	0.89	3066
1.0	0.87	0.94	0.90	3066
accuracy			0.90	6132
macro avg	0.90	0.90	0.90	6132
weighted avg	0.90	0.90	0.90	6132

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.91	0.86	0.89	3066
1.0	0.87	0.91	0.89	3066
accuracy			0.89	6132
macro avg	0.89	0.89	0.89	6132
weighted avg	0.89	0.89	0.89	6132

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.91	0.88	0.90	3066
1.0	0.89	0.92	0.90	3066
accuracy			0.90	6132
macro avg	0.90	0.90	0.90	6132
weighted avg	0.90	0.90	0.90	6132

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.89	0.90	0.90	3066
1.0	0.90	0.89	0.90	3066
accuracy			0.90	6132
macro avg	0.90	0.90	0.90	6132
weighted avg	0.90	0.90	0.90	6132

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.89	0.89	0.89	3066
1.0	0.89	0.89	0.89	3066
accuracy			0.89	6132
macro avg	0.89	0.89	0.89	6132
weighted avg	0.89	0.89	0.89	6132

Figure 20. Evaluation - Civil

7.3. Criminal

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.88	0.94	0.91	1323
1.0	0.93	0.87	0.90	1323
accuracy			0.90	2646
macro avg	0.90	0.90	0.90	2646
weighted avg	0.90	0.90	0.90	2646

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.92	0.93	0.93	1323
1.0	0.93	0.92	0.93	1323
accuracy			0.93	2646
macro avg	0.93	0.93	0.93	2646
weighted avg	0.93	0.93	0.93	2646

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.93	0.91	0.92	1323
1.0	0.91	0.93	0.92	1323
accuracy			0.92	2646
macro avg	0.92	0.92	0.92	2646
weighted avg	0.92	0.92	0.92	2646

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.92	0.90	0.91	1323
1.0	0.91	0.93	0.92	1323
accuracy			0.92	2646
macro avg	0.92	0.92	0.92	2646
weighted avg	0.92	0.92	0.92	2646

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.91	0.93	0.92	1323
1.0	0.93	0.91	0.92	1323
accuracy			0.92	2646
macro avg	0.92	0.92	0.92	2646
weighted avg	0.92	0.92	0.92	2646

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.93	0.93	0.93	1323
1.0	0.93	0.93	0.93	1323
accuracy			0.93	2646
macro avg	0.93	0.93	0.93	2646
weighted avg	0.93	0.93	0.93	2646

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.91	0.94	0.93	1323
1.0	0.94	0.91	0.92	1323
accuracy			0.92	2646
macro avg	0.92	0.92	0.92	2646
weighted avg	0.92	0.92	0.92	2646

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.88	0.93	0.90	1323
1.0	0.92	0.87	0.90	1323
accuracy			0.90	2646
macro avg	0.90	0.90	0.90	2646
weighted avg	0.90	0.90	0.90	2646

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.88	0.92	0.90	1323
1.0	0.92	0.88	0.90	1323
accuracy			0.90	2646
macro avg	0.90	0.90	0.90	2646
weighted avg	0.90	0.90	0.90	2646

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.87	0.96	0.91	1323
1.0	0.96	0.86	0.91	1323
accuracy			0.91	2646
macro avg	0.91	0.91	0.91	2646
weighted avg	0.91	0.91	0.91	2646

Figure 21. Evaluation - Criminal

7.4. National security, military, and the territories

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.90	0.82	0.86	883
1.0	0.83	0.91	0.87	883
accuracy			0.87	1766
macro avg	0.87	0.87	0.86	1766
weighted avg	0.87	0.87	0.86	1766

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.94	0.81	0.87	883
1.0	0.84	0.95	0.89	883
accuracy			0.88	1766
macro avg	0.89	0.88	0.88	1766
weighted avg	0.89	0.88	0.88	1766

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.94	0.78	0.85	883
1.0	0.81	0.95	0.88	883
accuracy			0.87	1766
macro avg	0.88	0.87	0.87	1766
weighted avg	0.88	0.87	0.87	1766

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.94	0.77	0.85	883
1.0	0.81	0.95	0.87	883
accuracy			0.86	1766
macro avg	0.88	0.86	0.86	1766
weighted avg	0.88	0.86	0.86	1766

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.91	0.75	0.82	883
1.0	0.78	0.93	0.85	883
accuracy			0.84	1766
macro avg	0.85	0.84	0.83	1766
weighted avg	0.85	0.84	0.83	1766

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.88	0.75	0.81	883
1.0	0.78	0.90	0.84	883
accuracy			0.83	1766
macro avg	0.83	0.83	0.82	1766
weighted avg	0.83	0.83	0.82	1766

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.84	0.82	0.83	883
1.0	0.82	0.85	0.84	883
accuracy			0.83	1766
macro avg	0.83	0.83	0.83	1766
weighted avg	0.83	0.83	0.83	1766

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.82	0.82	0.82	883
1.0	0.82	0.82	0.82	883
accuracy			0.82	1766
macro avg	0.82	0.82	0.82	1766
weighted avg	0.82	0.82	0.82	1766

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.79	0.87	0.83	883
1.0	0.85	0.78	0.81	883
accuracy			0.82	1766
macro avg	0.82	0.82	0.82	1766
weighted avg	0.82	0.82	0.82	1766

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.78	0.89	0.83	883
1.0	0.87	0.75	0.80	883
accuracy			0.82	1766
macro avg	0.82	0.82	0.82	1766
weighted avg	0.82	0.82	0.82	1766

Figure 22. Evaluation - National Security, Military and The Terorities

7.5. Constitutional

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.84	0.67	0.74	646
1.0	0.73	0.87	0.79	646
accuracy			0.77	1292
macro avg	0.78	0.77	0.77	1292
weighted avg	0.78	0.77	0.77	1292

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.86	0.74	0.80	646
1.0	0.77	0.88	0.83	646
accuracy			0.81	1292
macro avg	0.82	0.81	0.81	1292
weighted avg	0.82	0.81	0.81	1292

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.85	0.79	0.82	646
1.0	0.81	0.86	0.83	646
accuracy			0.83	1292
macro avg	0.83	0.83	0.82	1292
weighted avg	0.83	0.83	0.82	1292

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.86	0.81	0.83	646
1.0	0.82	0.87	0.84	646
accuracy			0.84	1292
macro avg	0.84	0.84	0.84	1292
weighted avg	0.84	0.84	0.84	1292

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.88	0.78	0.83	646
1.0	0.80	0.89	0.84	646
accuracy			0.84	1292
macro avg	0.84	0.84	0.83	1292
weighted avg	0.84	0.84	0.83	1292

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.86	0.78	0.82	646
1.0	0.80	0.87	0.83	646
accuracy			0.83	1292
macro avg	0.83	0.83	0.83	1292
weighted avg	0.83	0.83	0.83	1292

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.81	0.78	0.79	646
1.0	0.79	0.82	0.80	646
accuracy			0.80	1292
macro avg	0.80	0.80	0.80	1292
weighted avg	0.80	0.80	0.80	1292

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.81	0.82	0.82	646
1.0	0.82	0.81	0.82	646
accuracy			0.82	1292
macro avg	0.82	0.82	0.82	1292
weighted avg	0.82	0.82	0.82	1292

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.77	0.85	0.81	646
1.0	0.83	0.75	0.79	646
accuracy			0.80	1292
macro avg	0.80	0.80	0.80	1292
weighted avg	0.80	0.80	0.80	1292

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.81	0.85	0.83	646
1.0	0.84	0.80	0.82	646
accuracy			0.83	1292
macro avg	0.83	0.83	0.83	1292
weighted avg	0.83	0.83	0.83	1292

Figure 23. Evaluation - Constitutional

7.6. Labor and Employment

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.88	0.77	0.82	160
1.0	0.79	0.89	0.84	160
accuracy			0.83	320
macro avg	0.84	0.83	0.83	320
weighted avg	0.84	0.83	0.83	320

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.93	0.70	0.80	160
1.0	0.76	0.94	0.84	160
accuracy			0.82	320
macro avg	0.84	0.82	0.82	320
weighted avg	0.84	0.82	0.82	320

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.84	0.64	0.73	160
1.0	0.71	0.88	0.79	160
accuracy			0.76	320
macro avg	0.78	0.76	0.76	320
weighted avg	0.78	0.76	0.76	320

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.83	0.69	0.76	160
1.0	0.74	0.86	0.79	160
accuracy			0.78	320
macro avg	0.78	0.77	0.77	320
weighted avg	0.78	0.78	0.77	320

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.85	0.68	0.75	160
1.0	0.73	0.88	0.80	160
accuracy			0.78	320
macro avg	0.79	0.78	0.78	320
weighted avg	0.79	0.78	0.78	320

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.81	0.74	0.77	160
1.0	0.76	0.83	0.79	160
accuracy			0.78	320
macro avg	0.79	0.78	0.78	320
weighted avg	0.79	0.78	0.78	320

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.83	0.81	0.82	160
1.0	0.82	0.84	0.83	160
accuracy			0.82	320
macro avg	0.83	0.82	0.82	320
weighted avg	0.83	0.82	0.82	320

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.75	0.84	0.79	160
1.0	0.82	0.72	0.77	160
accuracy			0.78	320
macro avg	0.79	0.78	0.78	320
weighted avg	0.79	0.78	0.78	320

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.76	0.86	0.81	160
1.0	0.84	0.73	0.78	160
accuracy			0.80	320
macro avg	0.80	0.80	0.80	320
weighted avg	0.80	0.80	0.80	320

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.74	0.87	0.80	160
1.0	0.84	0.70	0.76	160
accuracy			0.78	320
macro avg	0.79	0.78	0.78	320
weighted avg	0.79	0.78	0.78	320

Figure 24. Evaluation - Labor and Employment

7.7. Social security, Health Insurance, Pension

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.90	0.61	0.73	93
1.0	0.71	0.94	0.81	93
accuracy			0.77	186
macro avg	0.81	0.77	0.77	186
weighted avg	0.81	0.77	0.77	186

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.86	0.68	0.76	93
1.0	0.73	0.89	0.81	93
accuracy			0.78	186
macro avg	0.80	0.78	0.78	186
weighted avg	0.80	0.78	0.78	186

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.88	0.71	0.79	93
1.0	0.76	0.90	0.82	93
accuracy			0.81	186
macro avg	0.82	0.81	0.80	186
weighted avg	0.82	0.81	0.80	186

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.85	0.71	0.77	93
1.0	0.75	0.87	0.81	93
accuracy			0.79	186
macro avg	0.80	0.79	0.79	186
weighted avg	0.80	0.79	0.79	186

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.87	0.72	0.79	93
1.0	0.76	0.89	0.82	93
accuracy			0.81	186
macro avg	0.82	0.81	0.81	186
weighted avg	0.82	0.81	0.81	186

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.78	0.78	0.78	93
1.0	0.78	0.77	0.78	93
accuracy			0.78	186
macro avg	0.78	0.78	0.78	186
weighted avg	0.78	0.78	0.78	186

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.86	0.81	0.83	93
1.0	0.82	0.87	0.84	93
accuracy			0.84	186
macro avg	0.84	0.84	0.84	186
weighted avg	0.84	0.84	0.84	186

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.77	0.82	0.79	93
1.0	0.80	0.75	0.78	93
accuracy			0.78	186
macro avg	0.79	0.78	0.78	186
weighted avg	0.79	0.78	0.78	186

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.79	0.91	0.85	93
1.0	0.90	0.76	0.83	93
accuracy			0.84	186
macro avg	0.85	0.84	0.84	186
weighted avg	0.85	0.84	0.84	186

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.73	0.91	0.81	93
1.0	0.88	0.66	0.75	93
accuracy			0.78	186
macro avg	0.81	0.78	0.78	186
weighted avg	0.81	0.78	0.78	186

Figure 25. Evaluation - Social Security, Health Insurance, Pension

7.8. Religious

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.94	0.66	0.78	68
1.0	0.74	0.96	0.83	68
accuracy			0.81	136
macro avg	0.84	0.81	0.80	136
weighted avg	0.84	0.81	0.80	136

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.88	0.82	0.85	68
1.0	0.83	0.88	0.86	68
accuracy			0.85	136
macro avg	0.85	0.85	0.85	136
weighted avg	0.85	0.85	0.85	136

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.90	0.69	0.78	68
1.0	0.75	0.93	0.83	68
accuracy			0.81	136
macro avg	0.83	0.81	0.81	136
weighted avg	0.83	0.81	0.81	136

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.94	0.74	0.83	68
1.0	0.78	0.96	0.86	68
accuracy			0.85	136
macro avg	0.86	0.85	0.84	136
weighted avg	0.86	0.85	0.84	136

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.83	0.78	0.80	68
1.0	0.79	0.84	0.81	68
accuracy			0.81	136
macro avg	0.81	0.81	0.81	136
weighted avg	0.81	0.81	0.81	136

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.86	0.87	0.86	68
1.0	0.87	0.85	0.86	68
accuracy			0.86	136
macro avg	0.86	0.86	0.86	136
weighted avg	0.86	0.86	0.86	136

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.80	0.76	0.78	68
1.0	0.77	0.81	0.79	68
accuracy			0.79	136
macro avg	0.79	0.79	0.79	136
weighted avg	0.79	0.79	0.79	136

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.89	0.94	0.91	68
1.0	0.94	0.88	0.91	68
accuracy			0.91	136
macro avg	0.91	0.91	0.91	136
weighted avg	0.91	0.91	0.91	136

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.89	0.93	0.91	68
1.0	0.92	0.88	0.90	68
accuracy			0.90	136
macro avg	0.91	0.90	0.90	136
weighted avg	0.91	0.90	0.90	136

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.84	0.99	0.91	68
1.0	0.98	0.81	0.89	68
accuracy			0.90	136
macro avg	0.91	0.90	0.90	136
weighted avg	0.91	0.90	0.90	136

Figure 26. Evaluation - Religious

7.9. Family

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.95	0.44	0.61	45
1.0	0.64	0.98	0.77	45
accuracy			0.71	90
macro avg	0.80	0.71	0.69	90
weighted avg	0.80	0.71	0.69	90

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.96	0.58	0.72	45
1.0	0.70	0.98	0.81	45
accuracy			0.78	90
macro avg	0.83	0.78	0.77	90
weighted avg	0.83	0.78	0.77	90

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.83	0.56	0.67	45
1.0	0.67	0.89	0.76	45
accuracy			0.72	90
macro avg	0.75	0.72	0.71	90
weighted avg	0.75	0.72	0.71	90

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.85	0.51	0.64	45
1.0	0.65	0.91	0.76	45
accuracy			0.71	90
macro avg	0.75	0.71	0.70	90
weighted avg	0.75	0.71	0.70	90

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.91	0.67	0.77	45
1.0	0.74	0.93	0.82	45
accuracy			0.80	90
macro avg	0.82	0.80	0.80	90
weighted avg	0.82	0.80	0.80	90

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.74	0.82	0.78	45
1.0	0.80	0.71	0.75	45
accuracy			0.77	90
macro avg	0.77	0.77	0.77	90
weighted avg	0.77	0.77	0.77	90

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.78	0.80	0.79	45
1.0	0.80	0.78	0.79	45
accuracy			0.79	90
macro avg	0.79	0.79	0.79	90
weighted avg	0.79	0.79	0.79	90

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.82	0.91	0.86	45
1.0	0.90	0.80	0.85	45
accuracy			0.86	90
macro avg	0.86	0.86	0.86	90
weighted avg	0.86	0.86	0.86	90

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.82	0.91	0.86	45
1.0	0.90	0.80	0.85	45
accuracy			0.86	90
macro avg	0.86	0.86	0.86	90
weighted avg	0.86	0.86	0.86	90

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.74	0.93	0.82	45
1.0	0.91	0.67	0.77	45
accuracy			0.80	90
macro avg	0.82	0.80	0.80	90
weighted avg	0.82	0.80	0.80	90

Figure 27. Evaluation - Family

7.10. International law

n_gram val : 1

	precision	recall	f1-score	support
0.0	0.85	0.64	0.73	36
1.0	0.71	0.89	0.79	36
accuracy			0.76	72
macro avg	0.78	0.76	0.76	72
weighted avg	0.78	0.76	0.76	72

n_gram val : 2

	precision	recall	f1-score	support
0.0	0.95	0.53	0.68	36
1.0	0.67	0.97	0.80	36
accuracy			0.75	72
macro avg	0.81	0.75	0.74	72
weighted avg	0.81	0.75	0.74	72

n_gram val : 3

	precision	recall	f1-score	support
0.0	0.80	0.56	0.66	36
1.0	0.66	0.86	0.75	36
accuracy			0.71	72
macro avg	0.73	0.71	0.70	72
weighted avg	0.73	0.71	0.70	72

n_gram val : 4

	precision	recall	f1-score	support
0.0	0.89	0.69	0.78	36
1.0	0.75	0.92	0.83	36
accuracy			0.81	72
macro avg	0.82	0.81	0.80	72
weighted avg	0.82	0.81	0.80	72

n_gram val : 5

	precision	recall	f1-score	support
0.0	0.84	0.86	0.85	36
1.0	0.86	0.83	0.85	36
accuracy			0.85	72
macro avg	0.85	0.85	0.85	72
weighted avg	0.85	0.85	0.85	72

n_gram val : 6

	precision	recall	f1-score	support
0.0	0.86	0.86	0.86	36
1.0	0.86	0.86	0.86	36
accuracy			0.86	72
macro avg	0.86	0.86	0.86	72
weighted avg	0.86	0.86	0.86	72

n_gram val : 7

	precision	recall	f1-score	support
0.0	0.97	0.86	0.91	36
1.0	0.88	0.97	0.92	36
accuracy			0.92	72
macro avg	0.92	0.92	0.92	72
weighted avg	0.92	0.92	0.92	72

n_gram val : 8

	precision	recall	f1-score	support
0.0	0.76	0.89	0.82	36
1.0	0.87	0.72	0.79	36
accuracy			0.81	72
macro avg	0.81	0.81	0.80	72
weighted avg	0.81	0.81	0.80	72

n_gram val : 9

	precision	recall	f1-score	support
0.0	0.97	0.94	0.96	36
1.0	0.95	0.97	0.96	36
accuracy			0.96	72
macro avg	0.96	0.96	0.96	72
weighted avg	0.96	0.96	0.96	72

n_gram val : 10

	precision	recall	f1-score	support
0.0	0.79	0.94	0.86	36
1.0	0.93	0.75	0.83	36
accuracy			0.85	72
macro avg	0.86	0.85	0.85	72
weighted avg	0.86	0.85	0.85	72

Figure 28. Evaluation - International Law

7.11 Evaluation results and final model training

The classification results had been examined, and a fixed value of n_gram had been picked for all of the categories. The values picked for training are:

Category	n_gram Value
Administrative	9
Civil	9
Constitutional	4
Criminal	6
Family	8
International Law	9
Labor and Employment	7
National security, military, and the territories	2
Religious	8
Social security, Health Insurance, Pension	7

Table 1. Evaluation Results and Final model Training

8. Model Dump

The best model and tfidf vectorizer dumped using pickle ("pickle"). The procedure results are two files for each category - Multinomial Naive Bayes model and TFIDF Vectorizer.

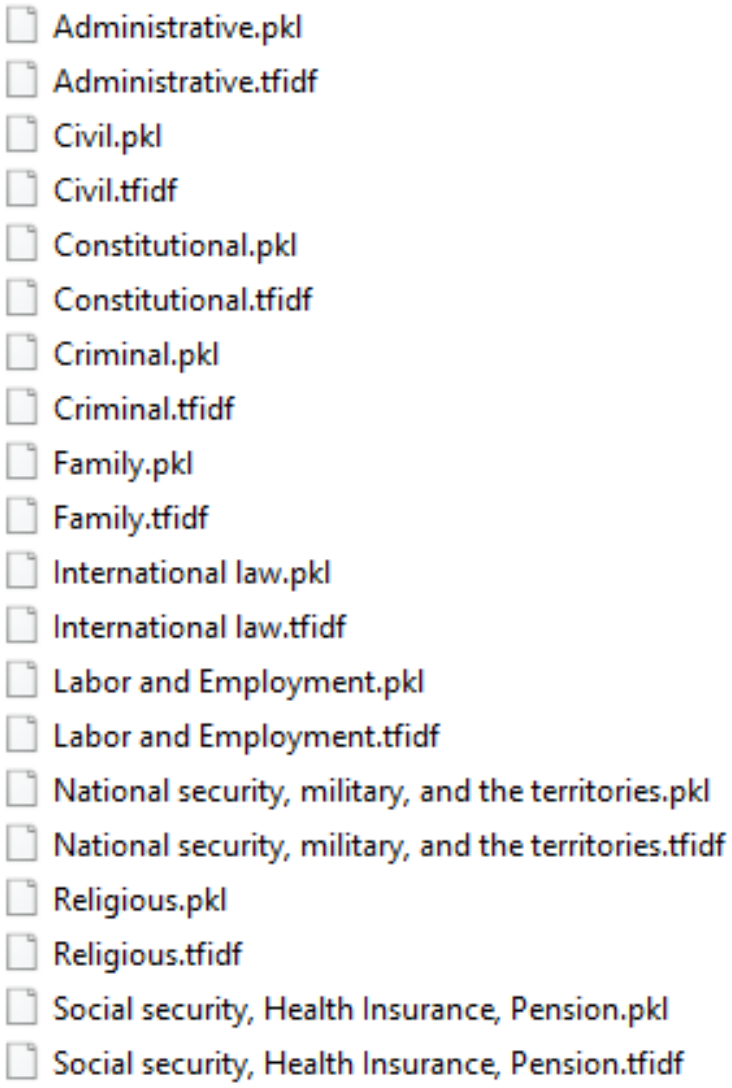


Figure 29. Model Dump

Normalization

The names we process are divided into two main categories:

1. Legal system related

2. Non-Legal system related

1. Legal system related process

In the following sections the processing steps on the legal system personal names will be shown.

1.1. Names extraction - the judges and attorney names are extracted

We access the names by calling the matching keys from the JSON dictionary and allocating them in the program's memory.

1.2. Pre-process the extracted names

The pre-processing of each and every name consists of 5 major transformations:

1.2.1. Unwanted characters elimination

- Digits - ~~54623~~
- Parenthesis content - ~~(this is removed)~~
- Leaked parsed statements - strings such as '2.' has leaked out from the parsing process - ~~2.~~
- Dash - transform : ' - ' → '-'
- Blankspaces - transform - ' this ' → 'this'

1.2.2 Common titles elimination

Israeli security titles, legal titles, emergency services, etc.

1.2.3 Common Hebrew stopwords elimination

Prepositions, pronouns, verbs, conjunctions etc.

1.2.4 Full name fix

Names with three words turn into two by applying a dash between the two last names.

1.2.5 Single name fix

Handles single multi-dashed names and fixes them.

1.3 Fixing judge names

Looking for the pre-processed name in the judges names txt file. For example, the process will turn the name: 'א חיות' into: 'אסתר חיות'.

1.4 Adding the newly examined name to the _legal_dictionary data member.

This will ensure a growing dictionary when the program runs. The dictionary has the name before normalization as a key and the normalized name as a value. The program can check if a specific input already exists and save search times in the document.

1.5 Applying normalization results to the verdict JSON.

Re-write the normalized judges names, and the pre-processed legal related personal as well as the pre-processed non-legal personal values to the program's memory stored JSON.

1.6 Dumping the new verdict JSON to the output destination.

Dump the memorized json into the location specified in the settings file.

1.7 Dumping the new dictionary to 'legal_personal.csv' destination.

The file has the name before normalization as a key and after as the value that has been collected so far by the program's memory

2. Non-legal system related process

The names are passed through the unwanted chars elimination process only.

Conclusion

Enrichment Results

We have successfully normalized and classified 212,348 verdicts. In the following sections we go in depth to explain the results, starting from the resulting JSON scheme, to the normalization and classification procedures findings.

JSON Scheme

The verdict JSON before the process:


```
{
    "Doc Details": {
        "מספר הליך": "בע\"מ 6839/00",
        "לפני": [
            "כבוד המשנה לנשיא ש' לוין",
            ""
        ],
        "העותר": [
            "זילכה אברהם",
            ""
        ],
        "מידע נוסף": [
            "בקשת רשות ערעור על פסק דינו של בית",
            "המשפט המחוזי בתל-אביב-יפו מיום",
            "9.7.00 שניתן על ידי בע\"א 4178/98",
            "סגן י' גרוס, ים",
            "א' קובו, מ' רובינשטיין"
        ],
        "המשיב": [
            "1. מ\"מ בע\"א. חניוני י.ש.",
            "2. חברת בית עובד בע\"מ",
            "3. הפניקס הישראלי",
            ""
        ],
        "בשם העותר": [
            "עו\"ד לאה (לוי) הלחמי"
        ],
        "בשם המשיב": [
            "עו\"ד ב. זיגר"
        ],
        "החלטה" :"סוג מסמך",
        "סיכום" :"1. לפני בקשת רשות לערער על פסק דינו של בית המשפט המחוזי בתל אביב-יפו לפיו הופחת הסכום שעל המשיבים לשלם למבקש מסך של 47,750 ש\"ח לסך של 430,050 ש\"ח, בתוספת הסכום שנפסק למבקש בגין כאב וסבל ובהפחתת תגמולי המוסד לביטוח לאומי ששולמו למבקש.\n2. בתשובתם הפנו המשיבים לפסק דינו של בית משפט זה בע\"א 5655 ,5671/99 (לא פורסם) שבו נקבע, תוך הפניה לע\"א 650/78 פ\"ד לח(2) 242 כי יש להפחית בתשובתו לתשובת המשיבים לא הצביע המבקש על טענה.\nאת סכום התגמולים ללא קשר לשאלה מה הסכום בו שיפה המזיק את המוסד לביטוח לאומי המצדיקה קיומו של דיון בגלגול שלישי חרף קיום ההלכה האמורה כאשר בית המשפט המחוזי היה ער לכך שחישוב הנזק נעשה על דרך של אומדן.\nהבקשה נדחית ללא צו להוצאות.\n(22.8.2001) ג' אלול תשס\"א ,ניתנה היום.\nהמשנה לנשיא",
        "02/07/2001" :"תאריך",
        "1 :"עמודים",
        "לפני מנורמל": "",
        "העותר מנורמל": "",
        "המשיב מנורמל": "",
        "בשם המשיב מנורמל": "",
        "בשם העותר מנורמל": ""
    },
    "Case Details": {...}
}
```


Figure 30. JSON Scheme Before

After:

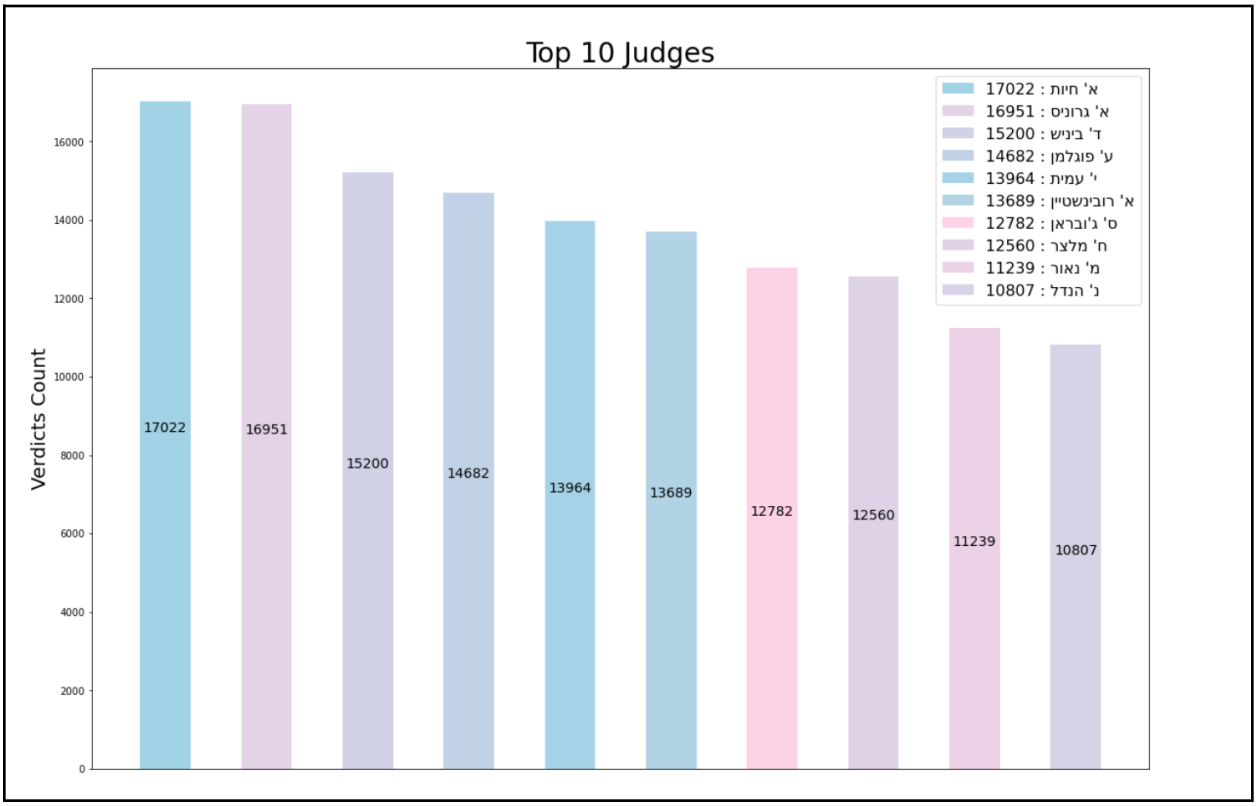

```
{
    "Doc Details": {
        "מספר הליך": "בע\"מ 6839/00",
        "לפני": [
            "כבוד המשנה לנשיא ש' לוין",
            ""
        ],
        "העותר": [
            "זילכה אברהם",
            ""
        ],
        "מידע נוסף": [
            "בקשת רשות ערעור על פסק דינו של בית",
            "המשפט המחוזי בתל-אביב-יפו מיום",
            "9.7.00 שניתן על ידי בע\"א 4178/98",
            "סגן '' גרוס, ים",
            "א' קובו, מ' רובינשטיין"
        ],
        "המשיב": [
            "1. מ\"בע .ש.א. חניוני '",
            "2. מ\"בע חברת בית עובד",
            "3. הפניקס הישראלי",
            ""
        ],
        "בשם העותר": [
            "עו\"ד לאה (לוי) הלחמי"
        ],
        "בשם המשיב": [
            "עו\"ד ב. זיגר"
        ],
        "סוג מסמך": "החלטה",
        "סיכום": "1. לפני בקשת רשות לערער על פסק דינו של בית המשפט המחוזי בתל אביב-יפו לפיו הופחת הסכום שעל המשיבים לשלם למבקש מסך של 47,750 ש\"ח לסך של 430,050 ש\"ח, בתוספת הסכום שנפסק למבקש בגין כאב וסבל ובהפחתת תגמולי המוסד לביטוח לאומי ששולמו למבקש.\n2. תשובת המשיבים נתבקשה אך לעניין הטענה כי לא היה מקום להפחית את תגמולי המוסד לביטוח לאומי מהסכום שנפסק בתשובתם הפנו המשיבים לפסק דינו של בית משפט זה בע\"א 5671/99, 5655 (לא פורסם) שבו נקבע, תוך הפניה לע\"א 650/78 פ\"ד לח(2) 242 כי יש להפחית בתשובתו לתשובת המשיבים לא הצביע המבקש על טענה.\nאת סכום התגמולים ללא קשר לשאלה מה הסכום בו שיפה המזיק את המוסד לביטוח לאומי המצדיקה קיומו של דיון בגלגול שלישי חרף קיום ההלכה האמורה כאשר בית המשפט המחוזי היה ער לכך שחישוב הנזק נעשה על דרך של אומדן.\nהמשנה לנשיא.\nניתנה היום, ג' אלול תשס\"א (22.8.2001).\nהבקשה נדחית ללא צו להוצאות.\n",
        "תאריך": "02/07/2001",
        "עמודים": "1",
        "לפני מנורמל": [
            "שלמה לוין"
        ],
        "העותר מנורמל": [
            "זילכה אברהם",
            ""
        ],
        "המשיב מנורמל": [
            "מ\"בע .ש.א. חניוני '",
            "מ\"בע חברת בית עובד",
            "הפניקס הישראלי",
            ""
        ],
        "בשם המשיב מנורמל": [
            "ב זיגר"
        ],
        "בשם העותר מנורמל": [
            "לאה הלחמי"
        ],
        "Category": "Civil"
    },
    "Case Details": {}
}
```


Figure 31. JSON Scheme After

Normalized Data

The following histograms present the top 10 most frequent names for each type.
The names are of the same groups that have been presented in the “Normalization” section.

Judges



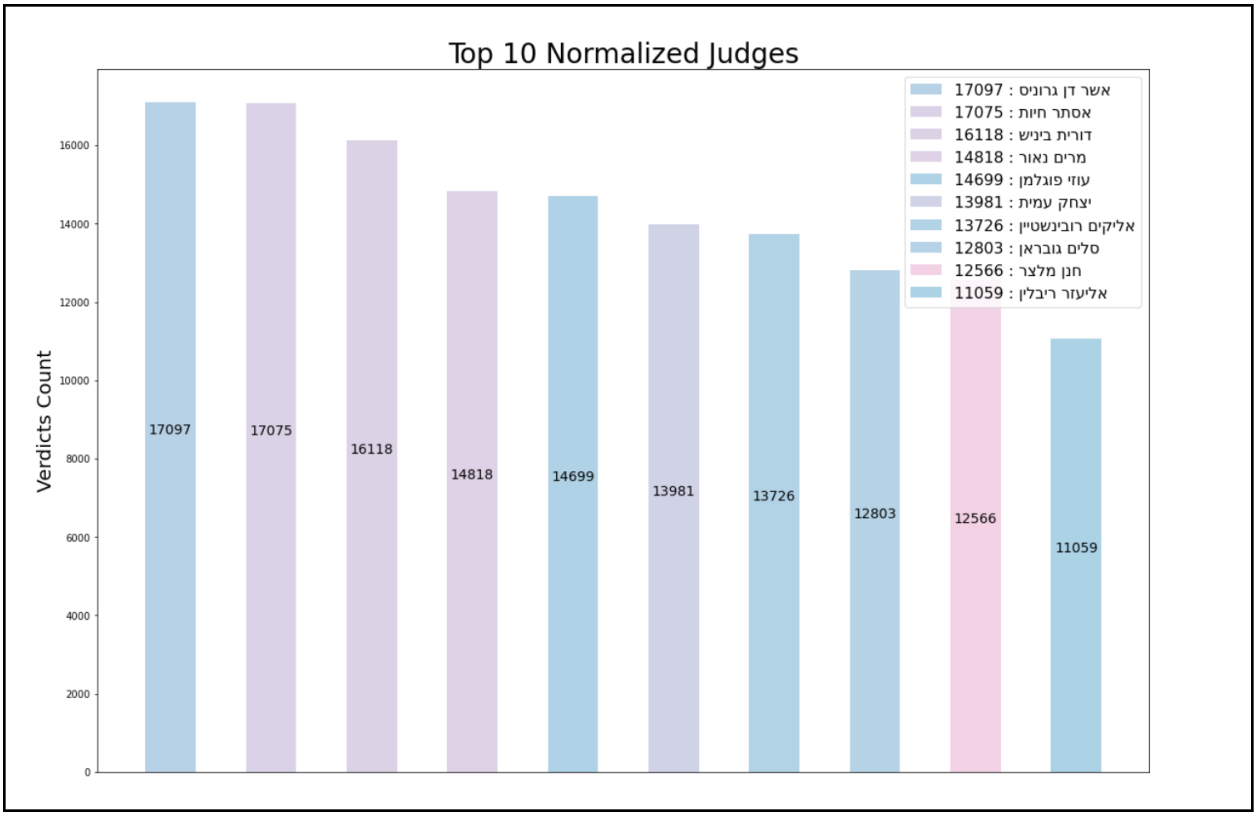


Table 2. Normalized Judges

1. 'אשר דן גרוניס' - Asher Dan Grunis - Former president of the Israeli Supreme-Court - 2012-2015.
2. 'אסתר חיות' - Eshter Hayut - Current president of the Israeli Supreme-Court - 2017-2023.
3. 'דורית ביניש' - Dorit Beinisch - Former president of the Israeli Supreme-Court - 2006-2012.
4. 'מרים נאור' - Miriam Naor - Former president of the Israeli Supreme-Court - 2015 - 2017.
5. 'עוזי פוגלמן' - Uzi Vogelman - Current Judge of the Israeli Supreme-Court - 2009 - Now. In 2023 - expected presidency successor of Ester Hayut.
6. 'יצחק עמית' - Yitzhak Amit - Current Judge on the Israeli Supreme-Court - 2009 - Now.In 2024 - expected presidency successor of Uzi Vogleman.
7. 'אליקים רובינשטיין' - Elyakim Rubinstein - Former vice president of the Israeli Supreme-Court - 2015 - 2017.
8. 'סלים גובראן' - Salim Joubran - Former Judge on the Israeli Supreme-Court 2004 - 2017.
9. 'חנן מלצר' - Hanan Melcer - Current Judge on the Israeli Supreme-Court - 2007 - Now.
10. 'אליעזר ריבלין' - Eliezer Rivilin- Former Judge on the Israeli Supreme-Court 2000 - 2012.

Petitioners

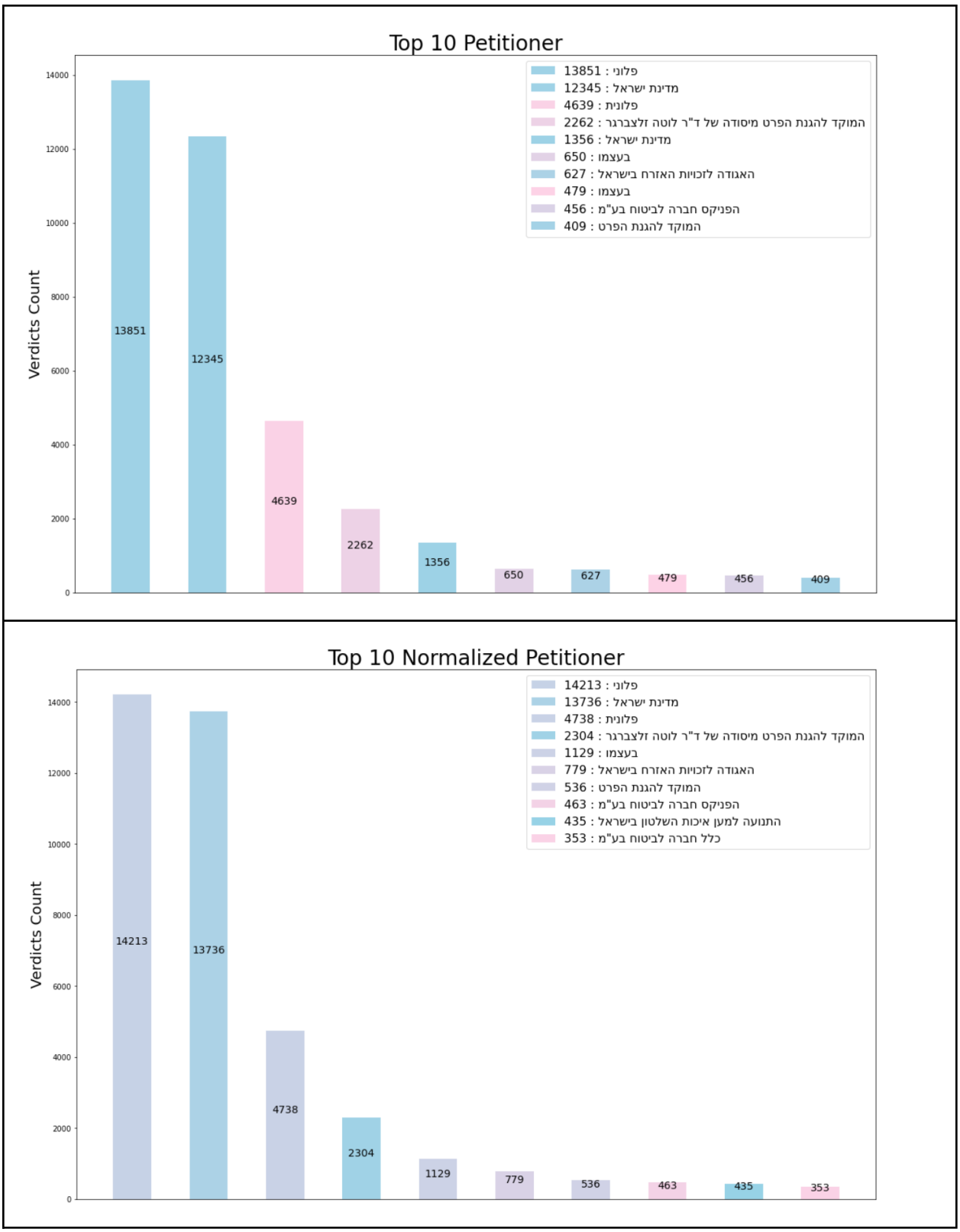
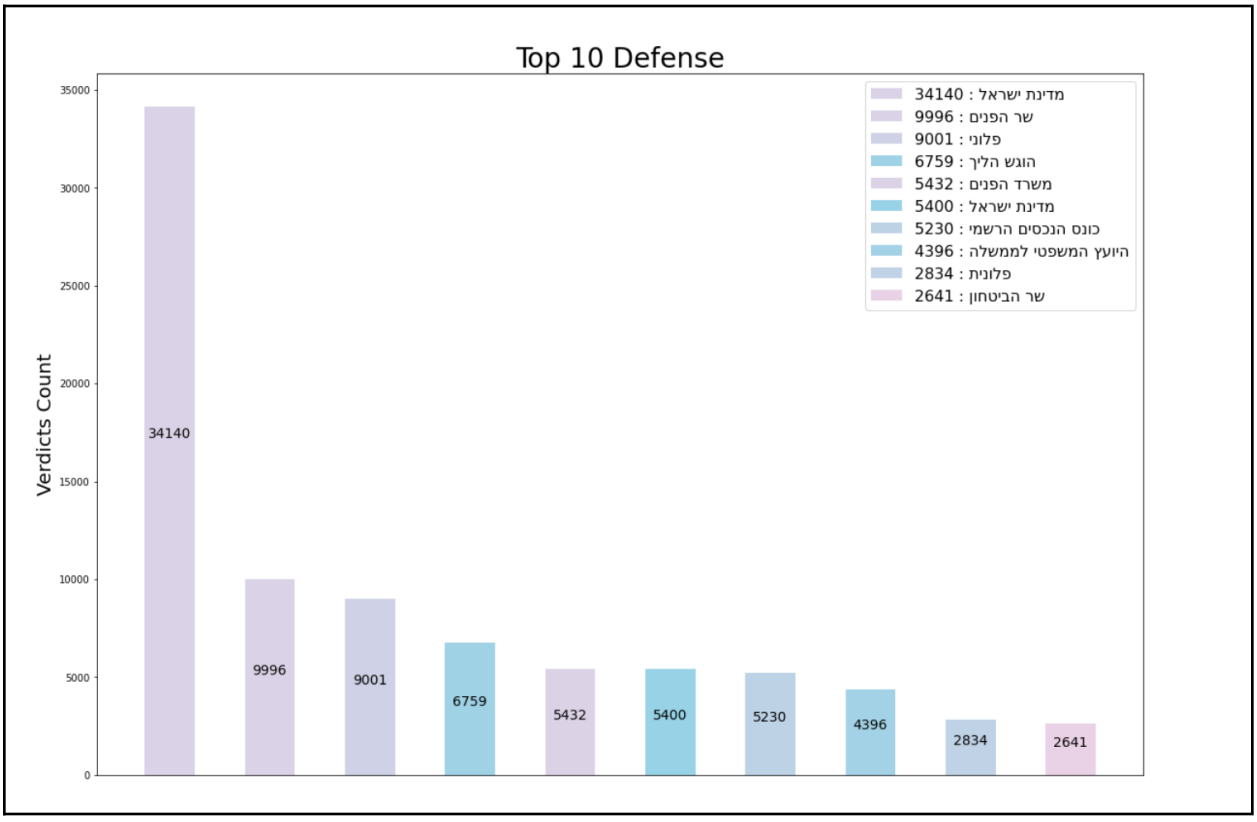


Table 3. Normalized Petitioners

1. ‘פלוני’ - Petitions that are being submitted by males anonymously
2. ‘מדינת ישראל’ - Country of israel
3. ‘פלונית’ - Petitions that are submitted anonymously and by females.
4. ‘המוקד להגנת הפרט מיסודה של ד”ר לוטה זלצברג’ - ‘HaMoked’ - Israeli based non-profit human rights organization
5. ‘בעצמו’ - The petitioner himself
6. ‘האגודה לזכויות האזרח’ - Association for civil rights in Israel - Israeli based non profit civil rights organization
7. ‘המוקד להגנת הפרט’ - Same as 3.
8. ‘הפניקס חברה לביטוח בע”מ’ - ‘The Phoenix Insurance Company Ltd.’ - An Israeli insurance company
9. ‘התנועה לאיכות השלטון’ - The movement for quality government - Israeli based non profit organization.
10. ‘כלל חברה לביטוח בע”מ’ - ‘Clal Insurance’ - An Israeli insurance company

Defense



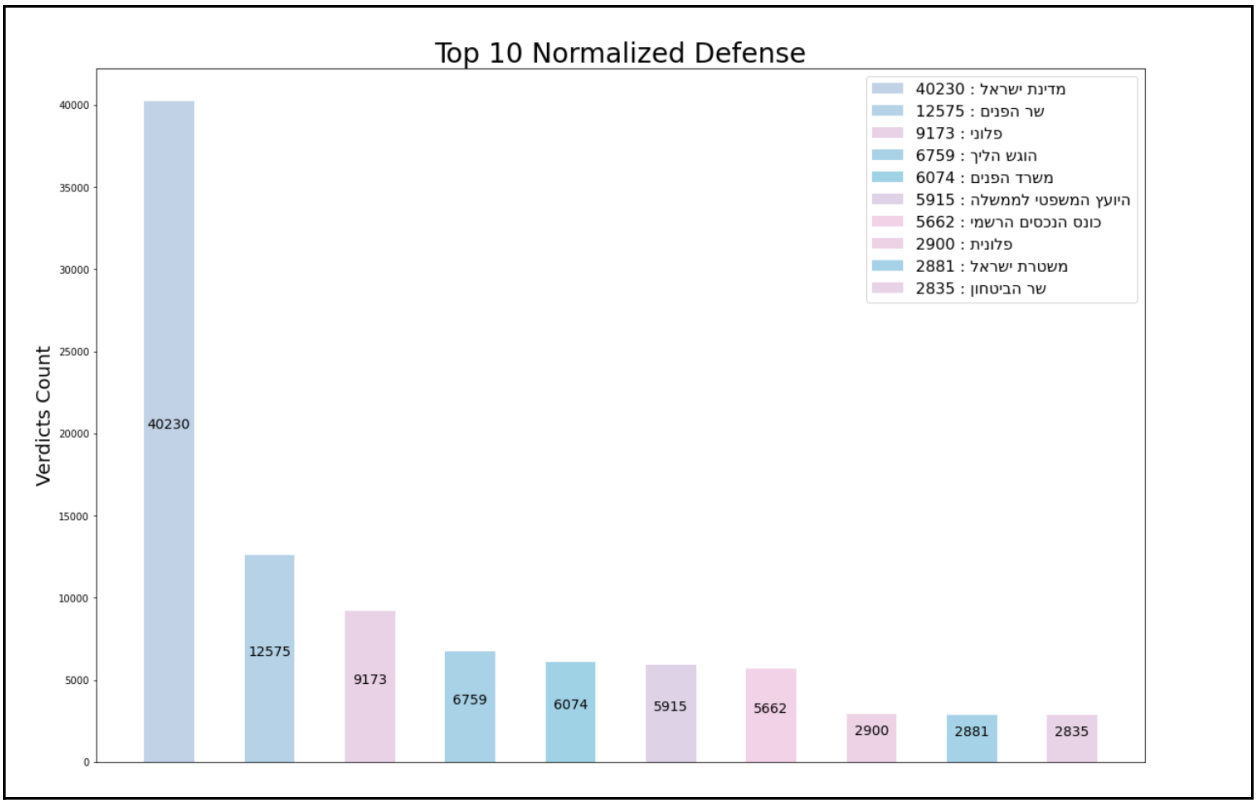


Table 4. Normalized Defense

1. ‘מדינת ישראל’ - Country of israel
2. ‘שר הפנים’- Minister of Internal Affairs
3. ‘פלוני’ - Male defense presented anonymously
4. ‘הוגש הליך’ - This result needs further examination
5. ‘משרד הפנים’ - Office of Internal Affairs
6. ‘היועץ המשפטי לממשלה’ - The Attorney General of the Israeli Government
7. ‘כונס הנכסים הרשמי’ - Formal Israeli Receivership
8. ‘פלונית’ - Female defense presented anonymously
9. ‘משטרת ישראל’ - The Israeli Police
10. ‘שר הביטחון’ - Minister of Defense

Petitioner Attorneys

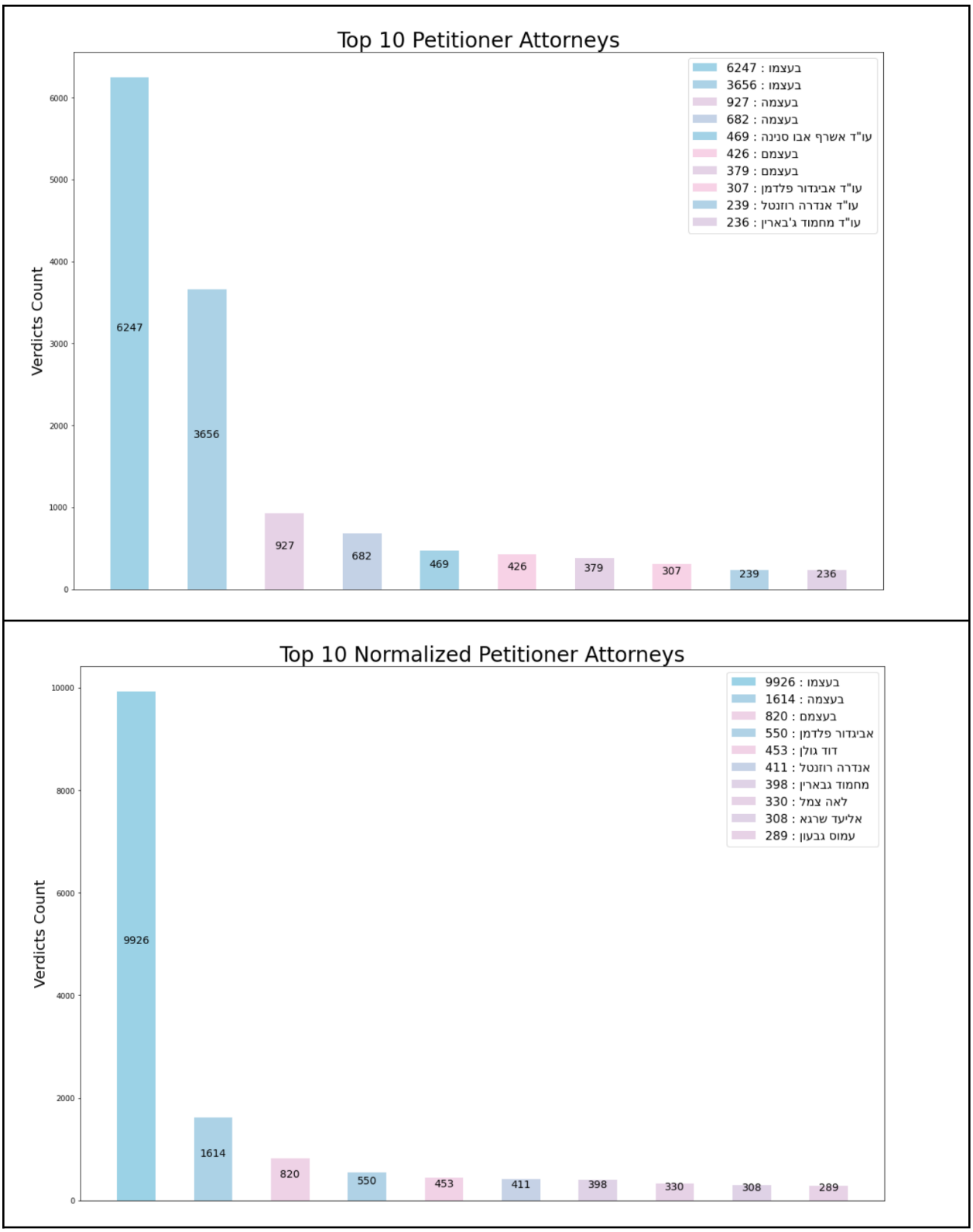
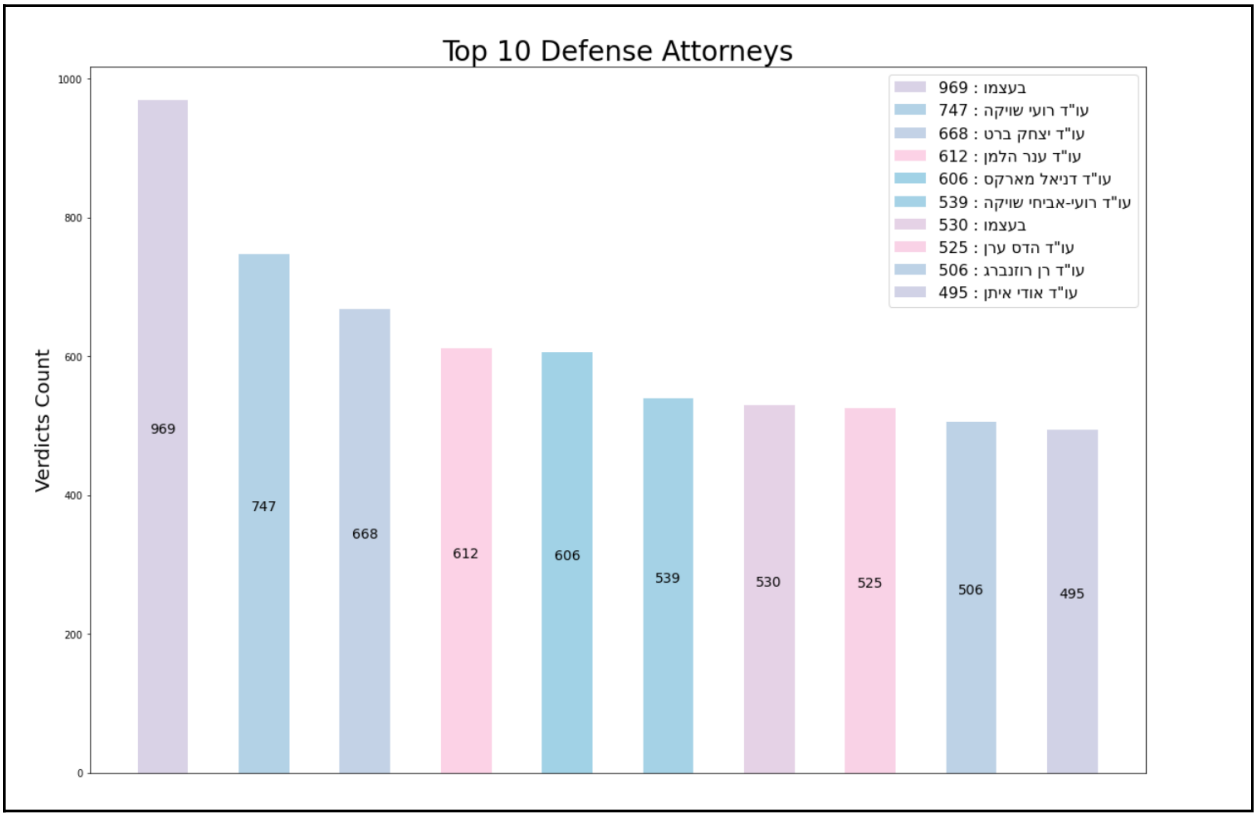


Table 5. Normalized Petitioner Attorneys

1. ‘בעצמו’ - ‘Himself’ - Male petitioned without representation and anonymously
2. ‘בעצמה’ - ‘Herself’ - Female petitioned without representation and anonymously
3. ‘בעצמם’ - ‘Themselves - Males petitioned without representation and anonymously
4. ‘אביגדור פלדמן’ - Avigdor Feldman
5. ‘דוד גולן’ - David Golan
6. ‘אנדרה רוזנטל’ - Andre Rozental
7. ‘מחמוד גבארין’ - Mahmoud Jabrin
8. ‘לאה צמל’ - Leah Zemel
9. ‘אליעד שרגא’ - Eliad Sharaggah
10. ‘עמוס גבעון’ - Amos Givon

Defense Attorneys



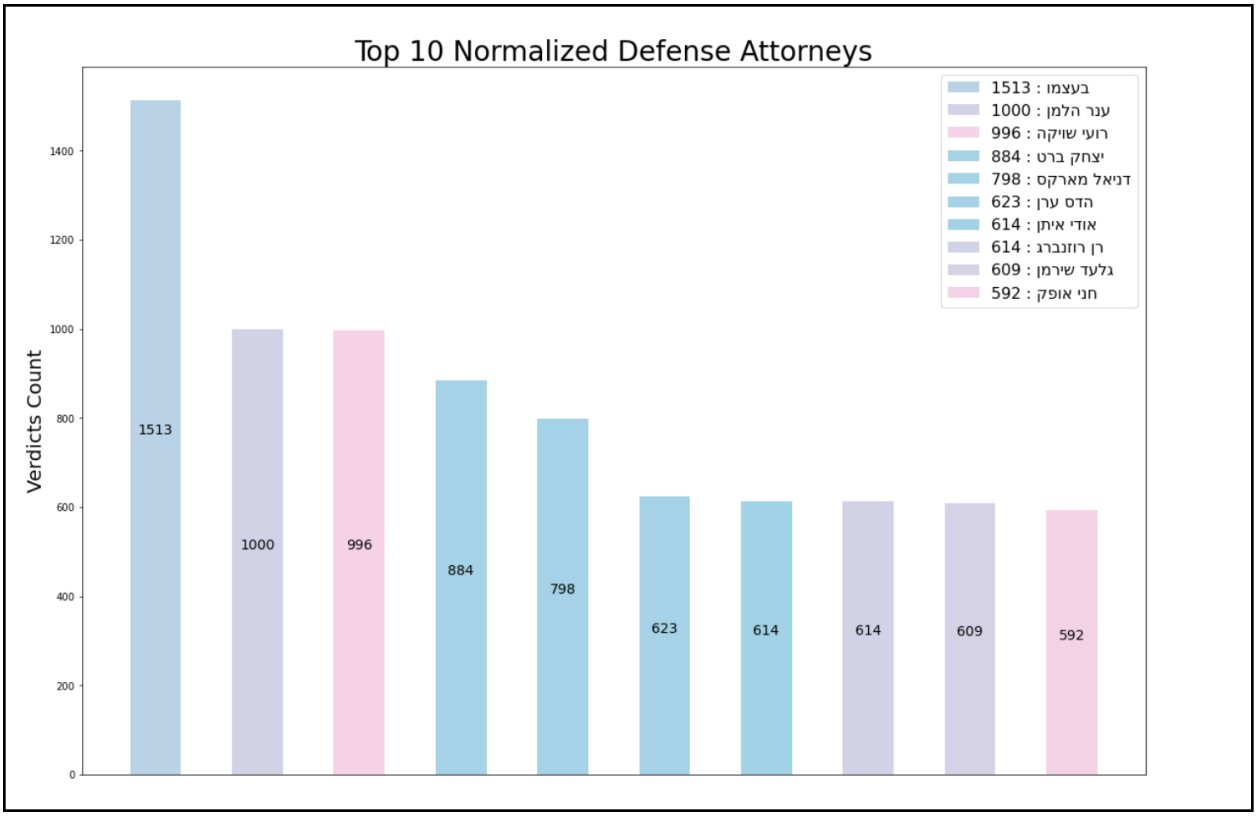


Table 6. Normalized Defense Attorneys

1. ‘בעצמו’ - Himself - Defending himself, without representation, anonymously
2. ‘ענר הלמן’ - Aner Helman
3. ‘רועי שויקה’ - Roei Shauke
4. ‘יצחק ברט’ - Yizhak Bert
5. ‘דניאל מארקס’ - Daniel Marks
6. ‘הדס ערן’ - Hadas Eran
7. ‘אודי איתן’ - Udi Eitan
8. ‘רן רוזנברג’ - Ran Rozenberg
9. ‘גלעד שירמן’ - Gilad Shirman
10. ‘חני אופק’ - Hanny Ofek

Classified Data

The classified data has been examined to look for correlation between the normalized names and the classified categories. We have used a Pearson based correlation matrix that will suggest which name is likely to be associated with which category.

Pearson's formula is :

$$P_{x,y} = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

Figure 32. Pearson's Correlation Formula

When:

- Nominator is the covariance matrix between x and y.
- Denominator is the product of x and y standard deviations

The equation is basically the tendency of two variables to act similarly, divided by the product of how dispersed each of the values are between the same category. The values normally are between -1 to 1, the histograms below will show -.3 to .3 in order to highlight the correlated and uncorrelated values. 1 to -1 sensitivity seems to obscure the abnormal behaviour, most of the values seem to be around 0 which indicate that there is not a specific association between a specific name to a specific category overall. In the following sections the correlation between the categories and the normalized names will be demonstrated using a diagonal Pearson's correlation matrix when the upper diagonal is trimmed due to a similar mirrored value with the bottom one. The histogram of the correlation matrix will be presented alongside a "Top 3 Positively Correlated" and "Top 3 Negatively Correlated" conclusion section. The table in the conclusion section will present three values:
Value 1 : Name/Category
Value 2 : Name/Category
Value 3: Actual numerical value between 1 to -1

Notes:

- No category-category will be shown in the table - each verdict/discussion has been categorized to a single category.
- Values above 0.5 are highlighted.

- For convenience reasons only values higher or equal to 0.1 or only values smaller or equal to -0.1 are being annotated.

Categories Distribution

The classification procedure led to the following category count:

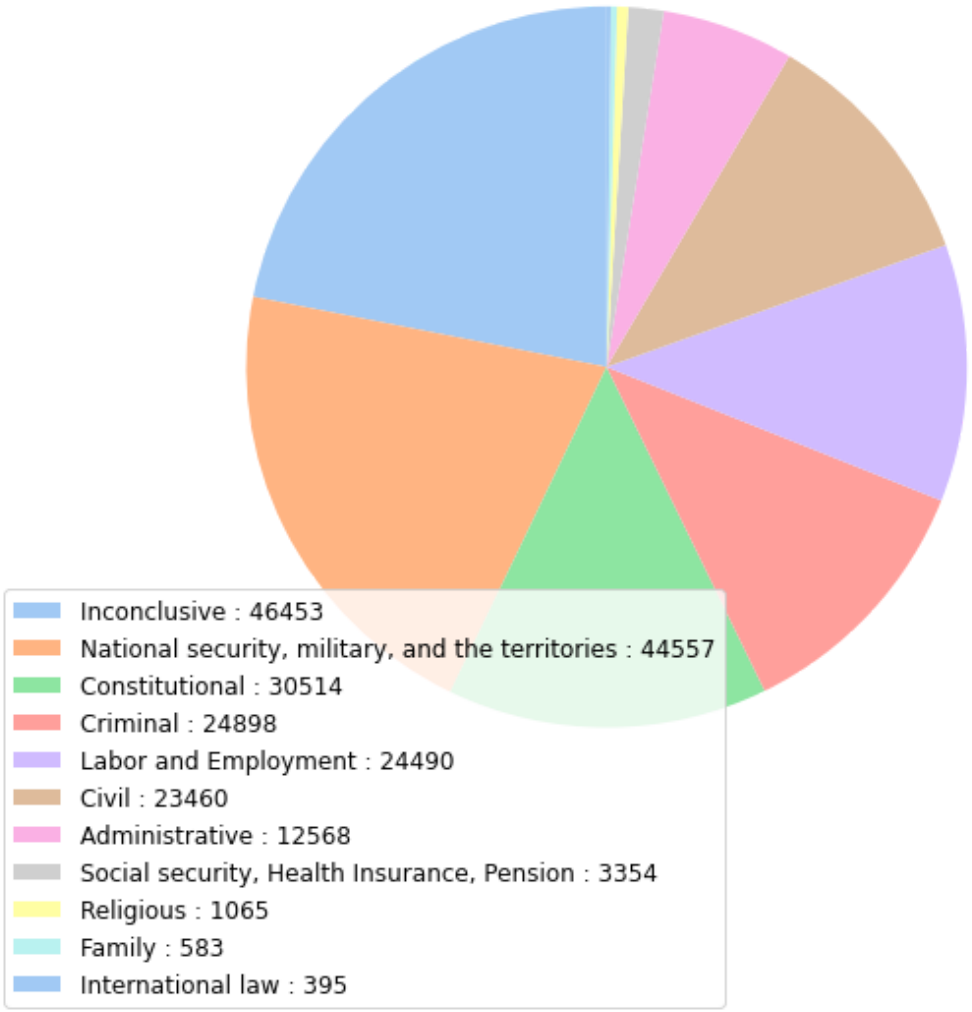
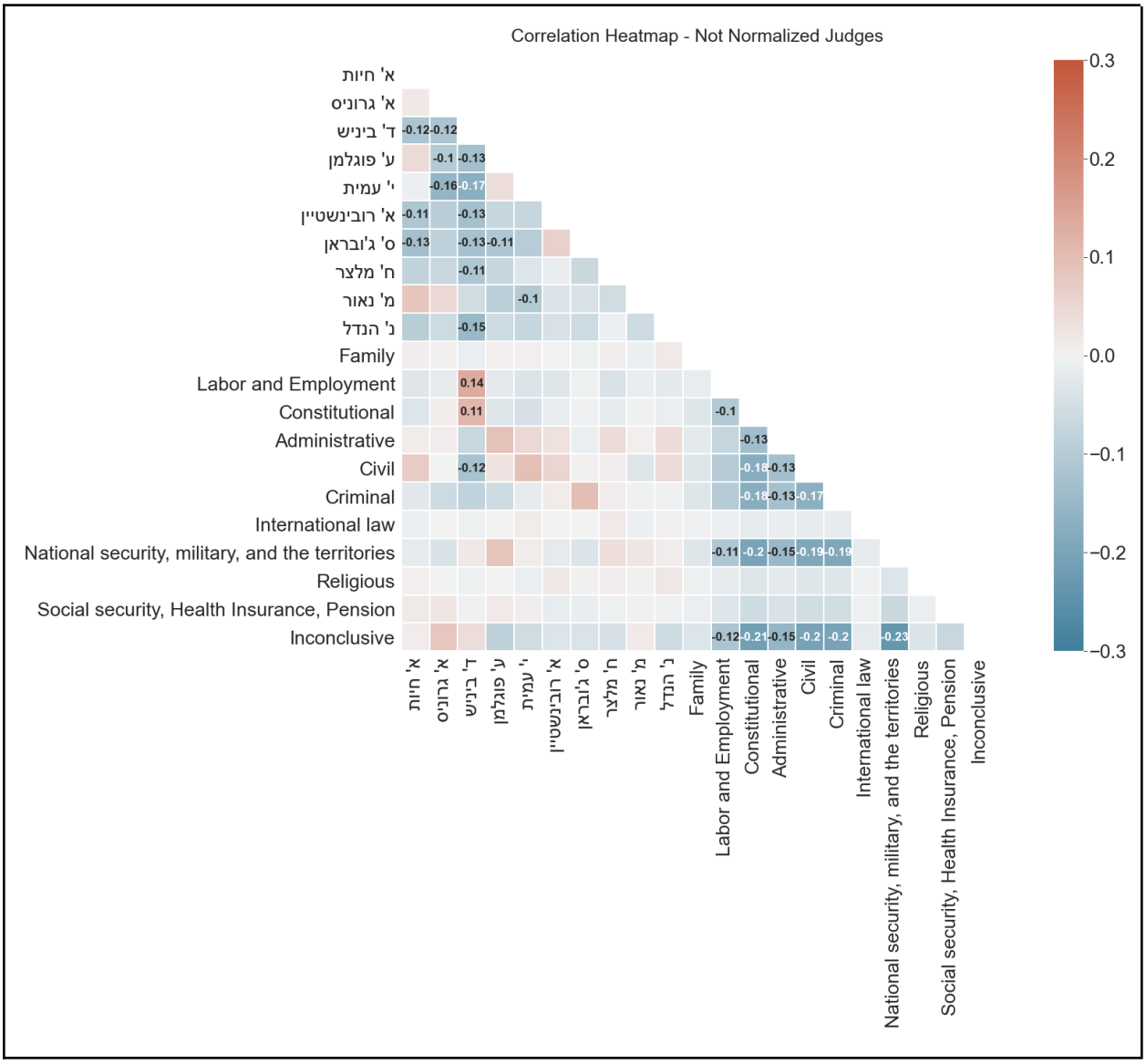


Figure 33. Categories Distribution

Judges

Histogram



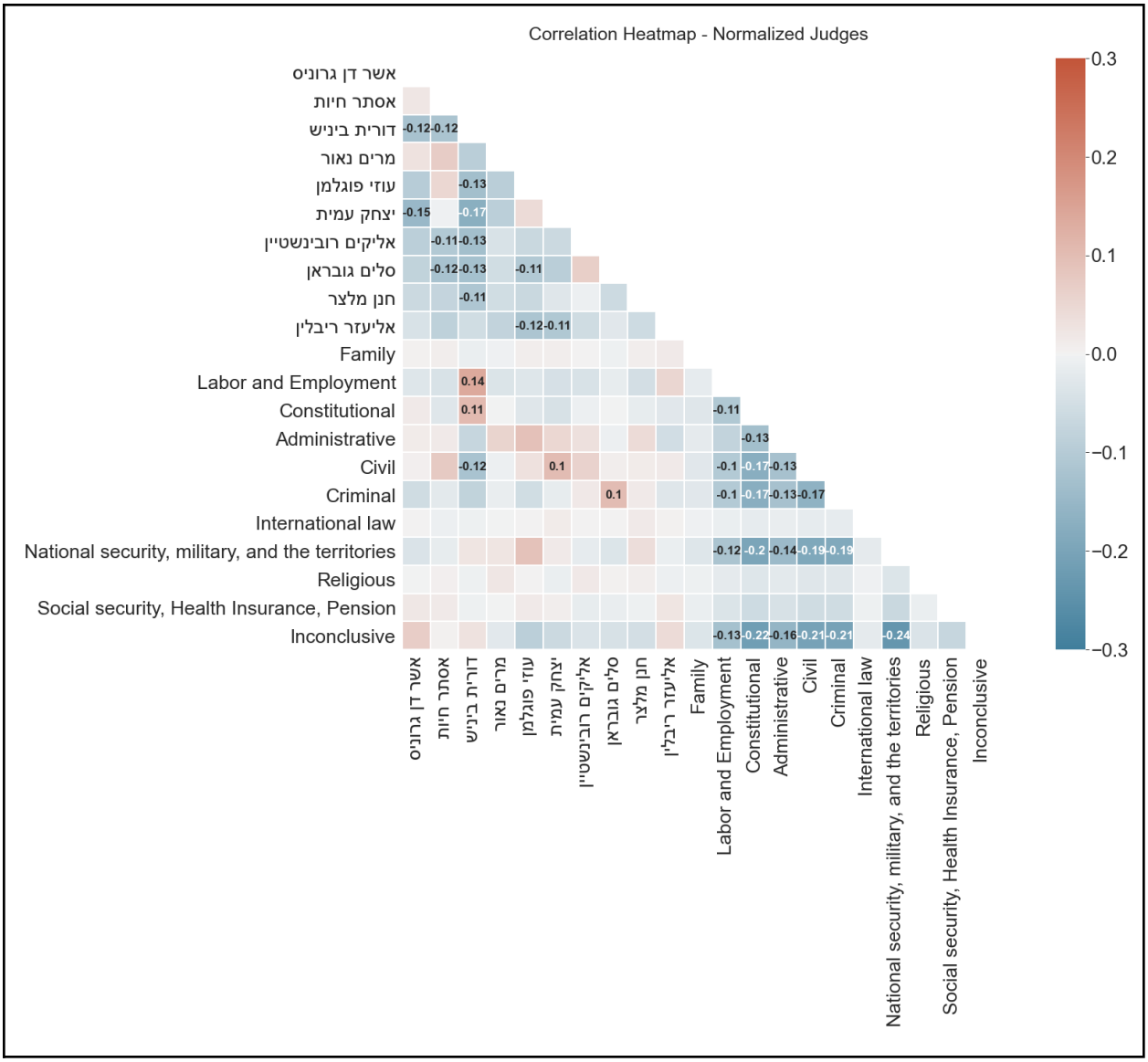


Table 7. Correlation Matrix - Judges

Conclusion

The results suggest that the categories are spread between the judges in a way that no specific judge corresponds to a specific category.

Not Normalized Positively Correlated:

ד' ביניש	Labor and Employment	0.137009
ס' ג'ובראן	Criminal	0.099464

י' עמית	Civil	0.098333

Table 8. Judges Not Normalized Positively Correlated

Not Normalized Negatively Correlated:

ד' ביניש	י' עמית	-0.172220
א' גרוניס	י' עמית	-0.155199
ד' ביניש	נ' הנדל	-0.145378

Table 9. Judges Not Normalized Negatively Correlated

Normalized Positively Correlated:

דורית ביניש	Labor and Employment	0.140247
סלים גובראן	Criminal	0.103024
יצחק עמית	Civil	0.102679

Table 10. Judges Normalized Positively Correlated

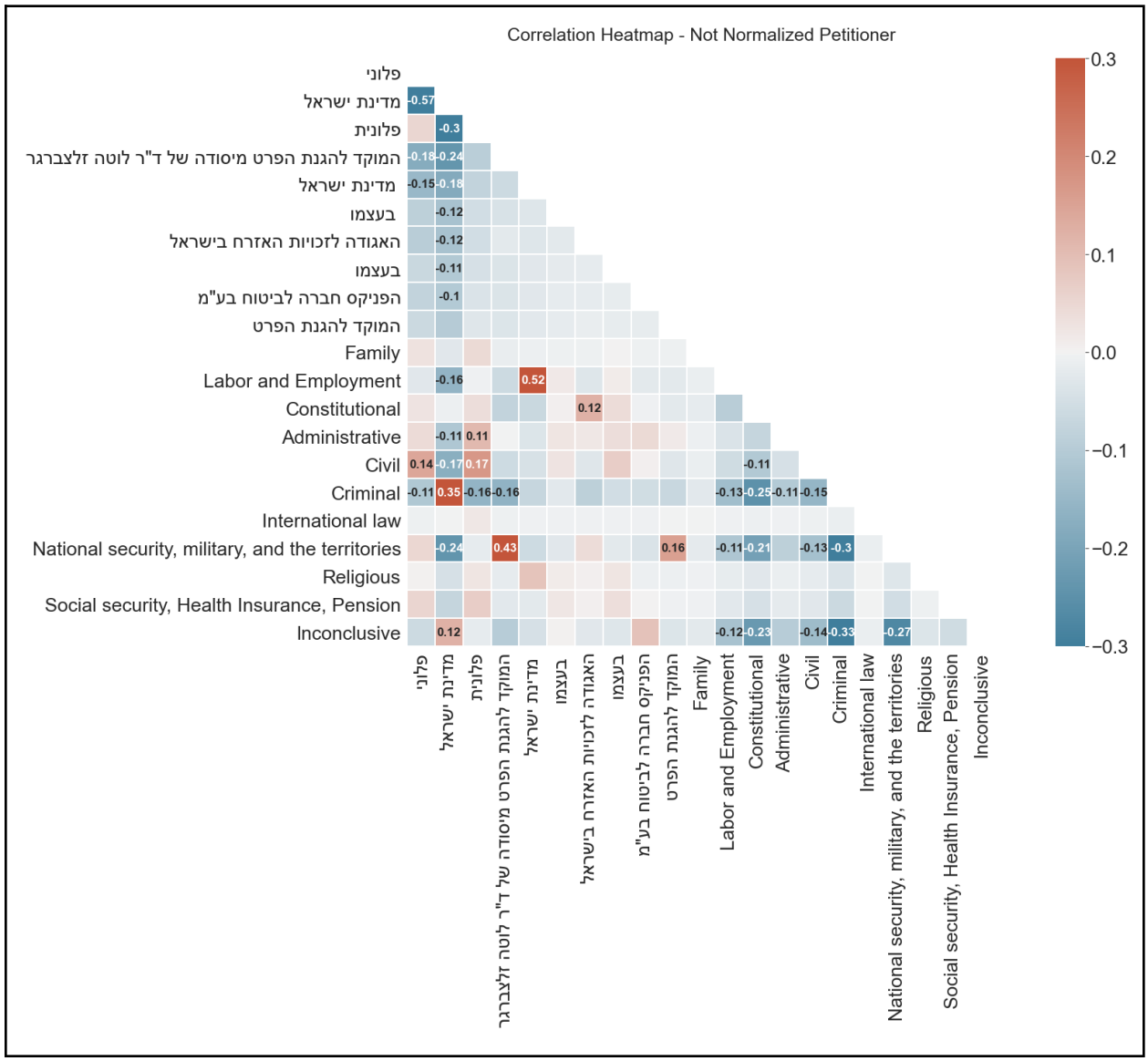
Normalized Negatively Correlated:

דורית ביניש	יצחק עמית	-0.172963
אשר דן גרוניס	יצחק עמית	-0.149797
דורית ביניש	אליקים רובינשטיין	-0.132570

Table 11. Judges Normalized Negatively Correlated

Petitioner

Petitioner - Histogram



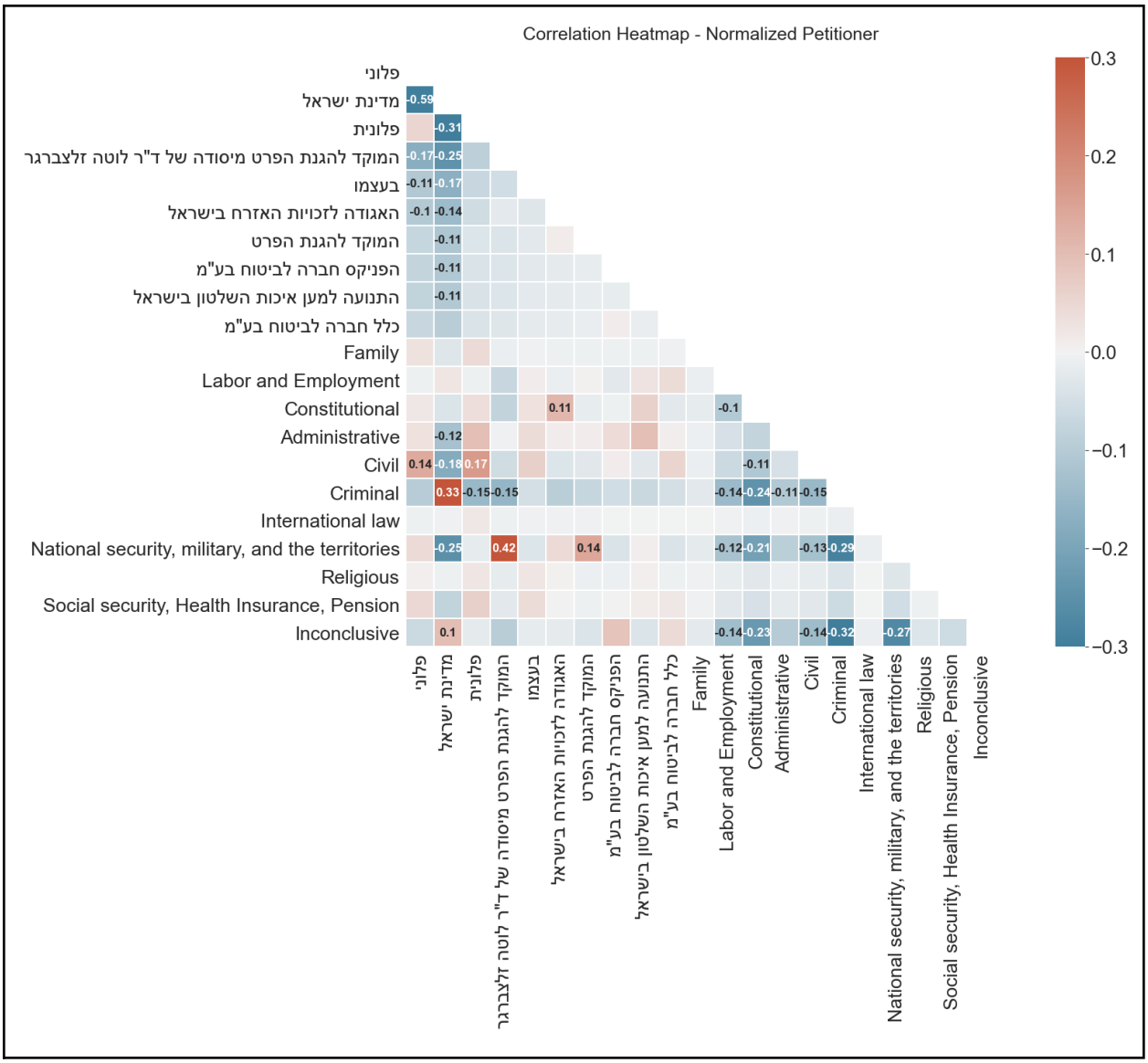


Table 12. Correlation Matrix - Petitioner

Conclusion

Not Normalized Positively Correlated:

מדינת ישראל	Labor and Employment	0.521314
המוקד להגנת הפרט מיסודה של ד"ר לוטה זלצברגר	National security, military, and the territories	0.430355
מדינת ישראל	Criminal	0.349556

Table 13. Petitioner Not Normalized Positively Correlated

Not Normalized Negatively Correlated:

פלוני	מדינת ישראל	-0.567313
מדינת ישראל	פלונית	-0.298727
מדינת ישראל	National security, military, and the territories	-0.240407

Table 14. Petitioner Not Normalized Negatively Correlated

Normalized Positively Correlated:

המוקד להגנת הפרט מיסודה של ד"ר לוטה זלצברגר	National security, military, and the territories	0.421376
מדינת ישראל	Criminal	0.325338
פלונית	Civil	0.167632

Table 15. Petitioner Normalized Positively Correlated

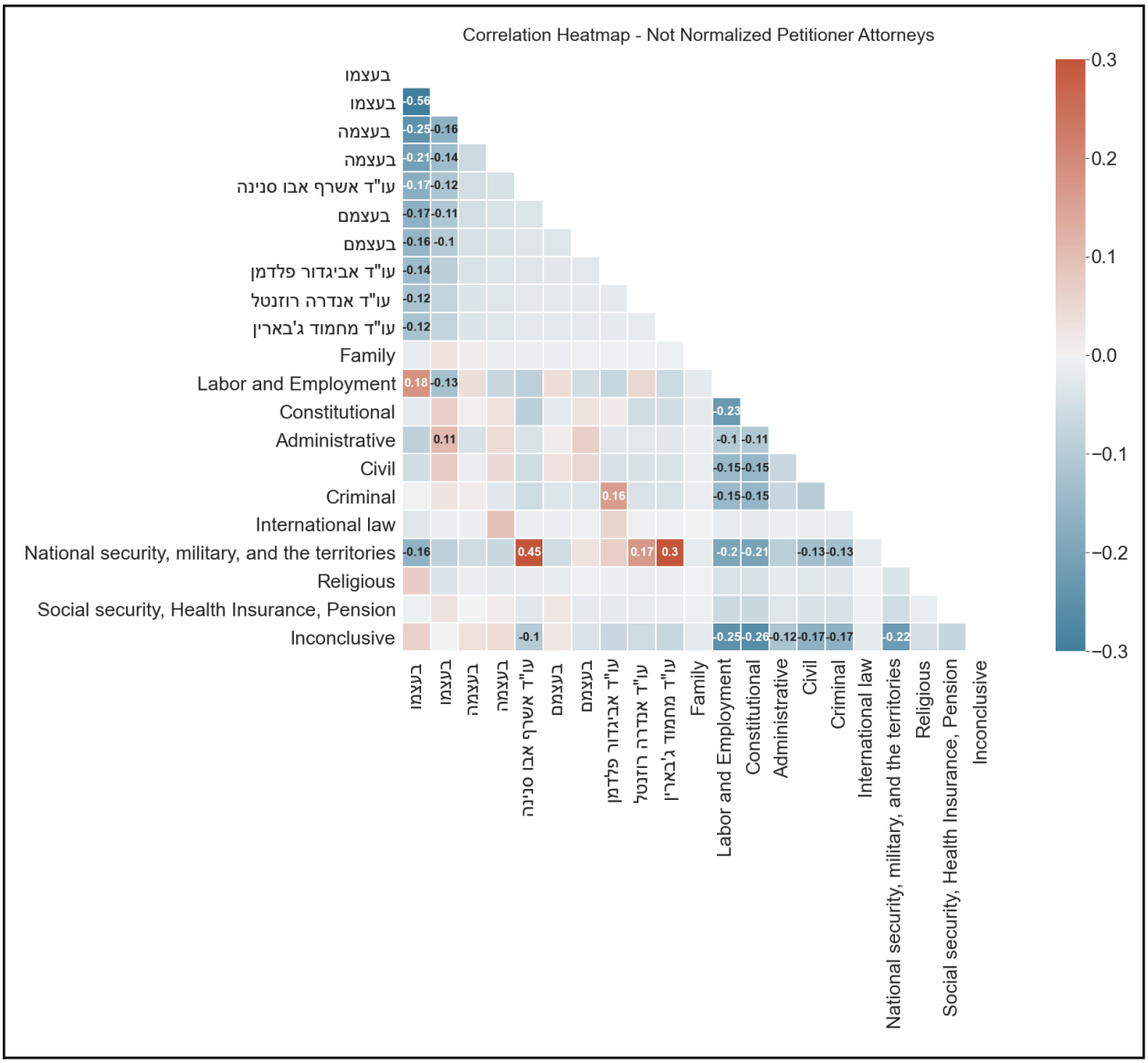
Normalized Negatively Correlated:

פלוני	מדינת ישראל	-0.591682
מדינת ישראל	פלונית	-0.311619
פלוני	המוקד להגנת הפרט מיסודה של ד"ר לוטה זלצברגר	-0.173628

Table 16. Petitioner Normalized Negatively Correlated

Petitioner Attorneys

Histogram



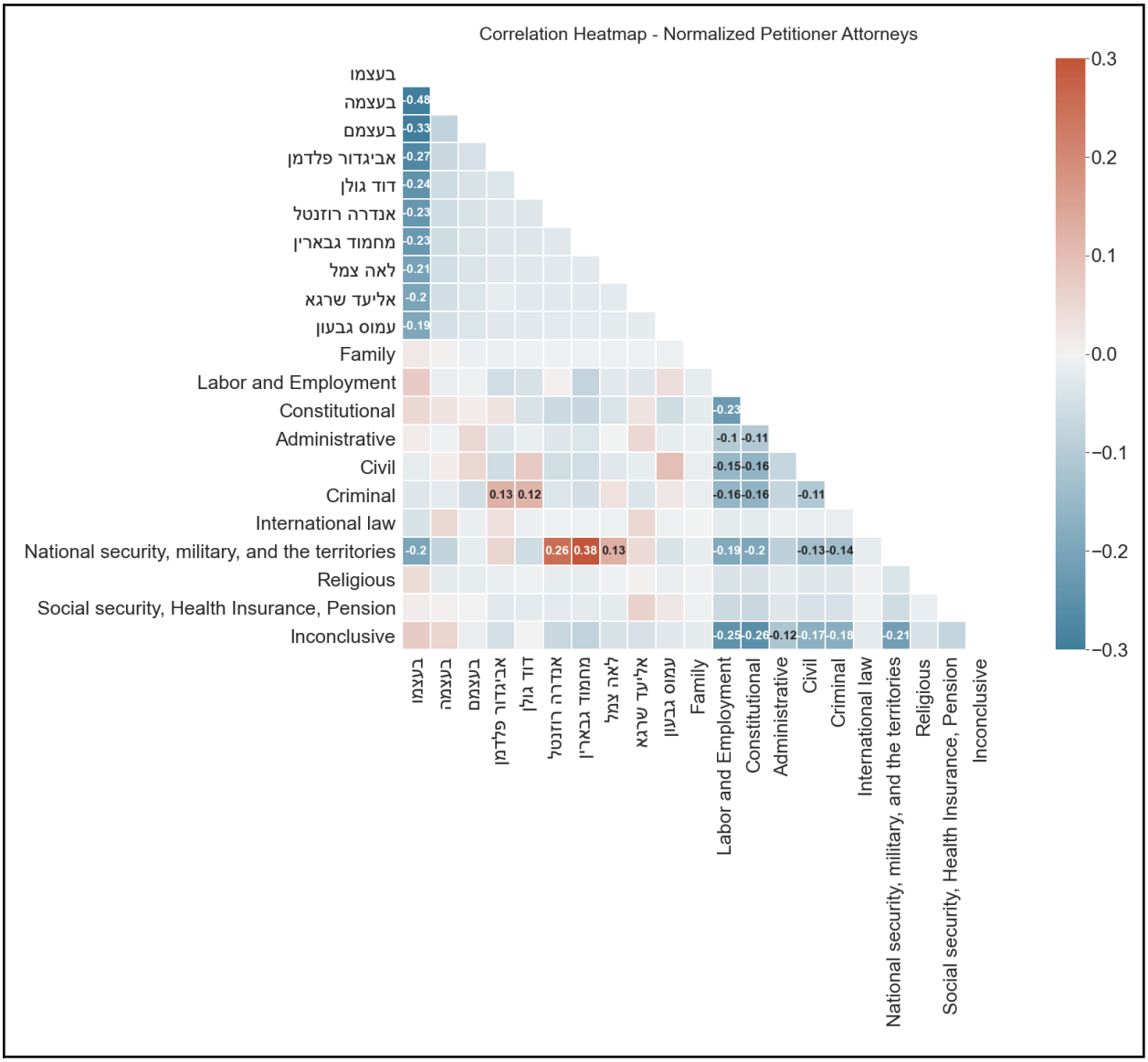


Table 17. Correlation Matrix - Petitioner Attorneys

Conclusion

Not Normalized Positively Correlated:

עו"ד אשרף אבו סנינה	National security, military, and the territories	0.446524
עו"ד מחמוד ג'בארין	National security, military, and the territories	0.302148

בעצמו	Labor and Employment	0.182712

Table 18. Petitioner Attorneys Not Normalized Positively Correlated

Not Normalized Negatively Correlated:

בעצמו	National security, military, and the territories	-0.162303
עו"ד אשרף אבו סנינה	Inconclusive	-0101114
עו"ד אשרף אבו סנינה	Constitutional	-0.089391

Table 19. Petitioner Attorneys Not Normalized Negatively Correlated

Normalized Positively Correlated:

מחמוד גבארין	National security, military, and the territories	0.382448
אנדרה רוזנטל	National security, military, and the territories	0.255299
לאה צמל	National security, military, and the territories	0.127943

Table 20. Petitioner Attorneys Normalized Positively Correlated

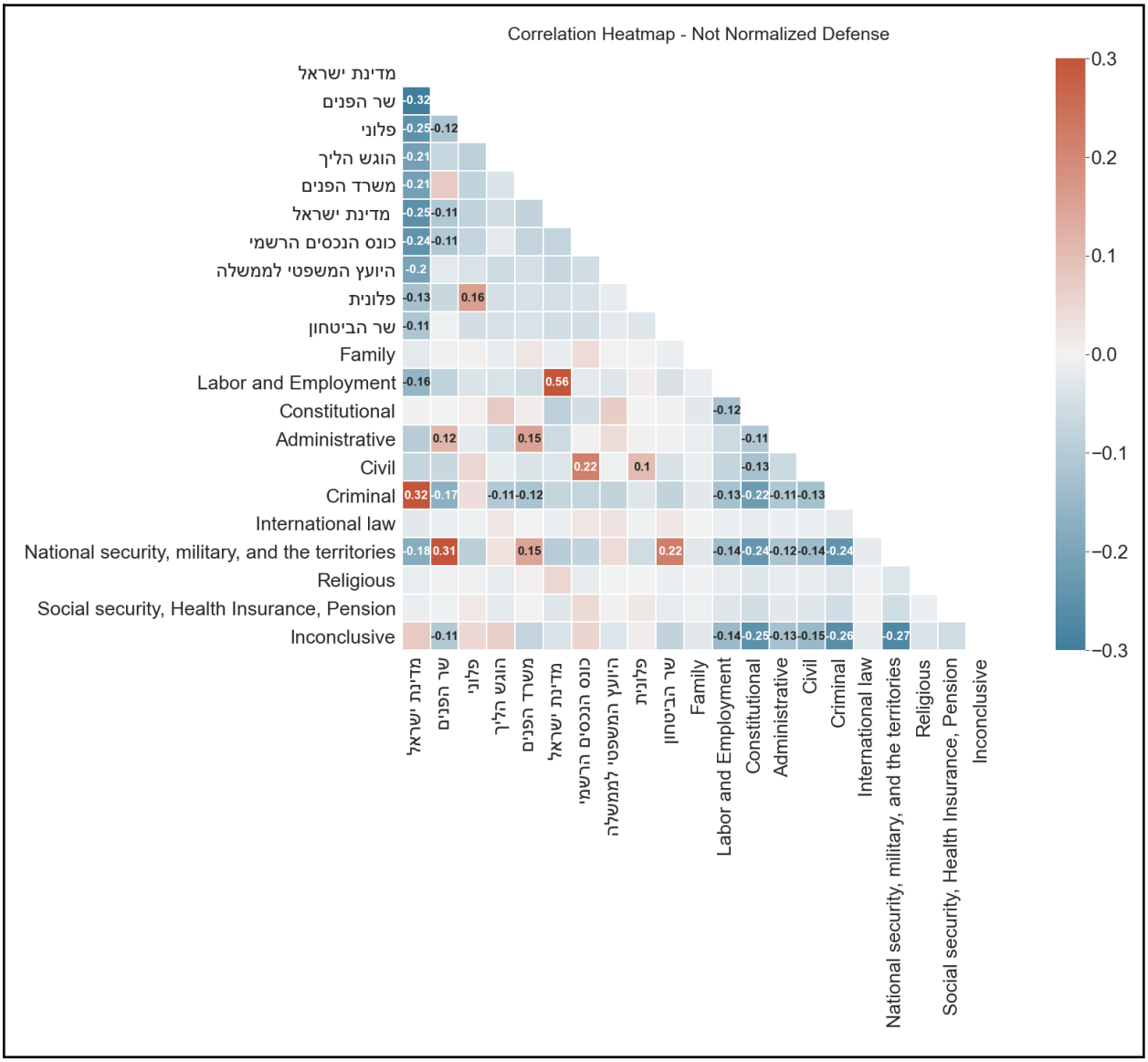
Normalized Negatively Correlated:

מחמוד גבארין	Labor and Employment	-0.076572
מחמוד גבארין	Constitutional	-0.068113
אנדרה רוזנטל	Inconclusive	-0.067513

Table 21. Petitioner Attorneys Normalized Negatively Correlated

Defense

Histogram



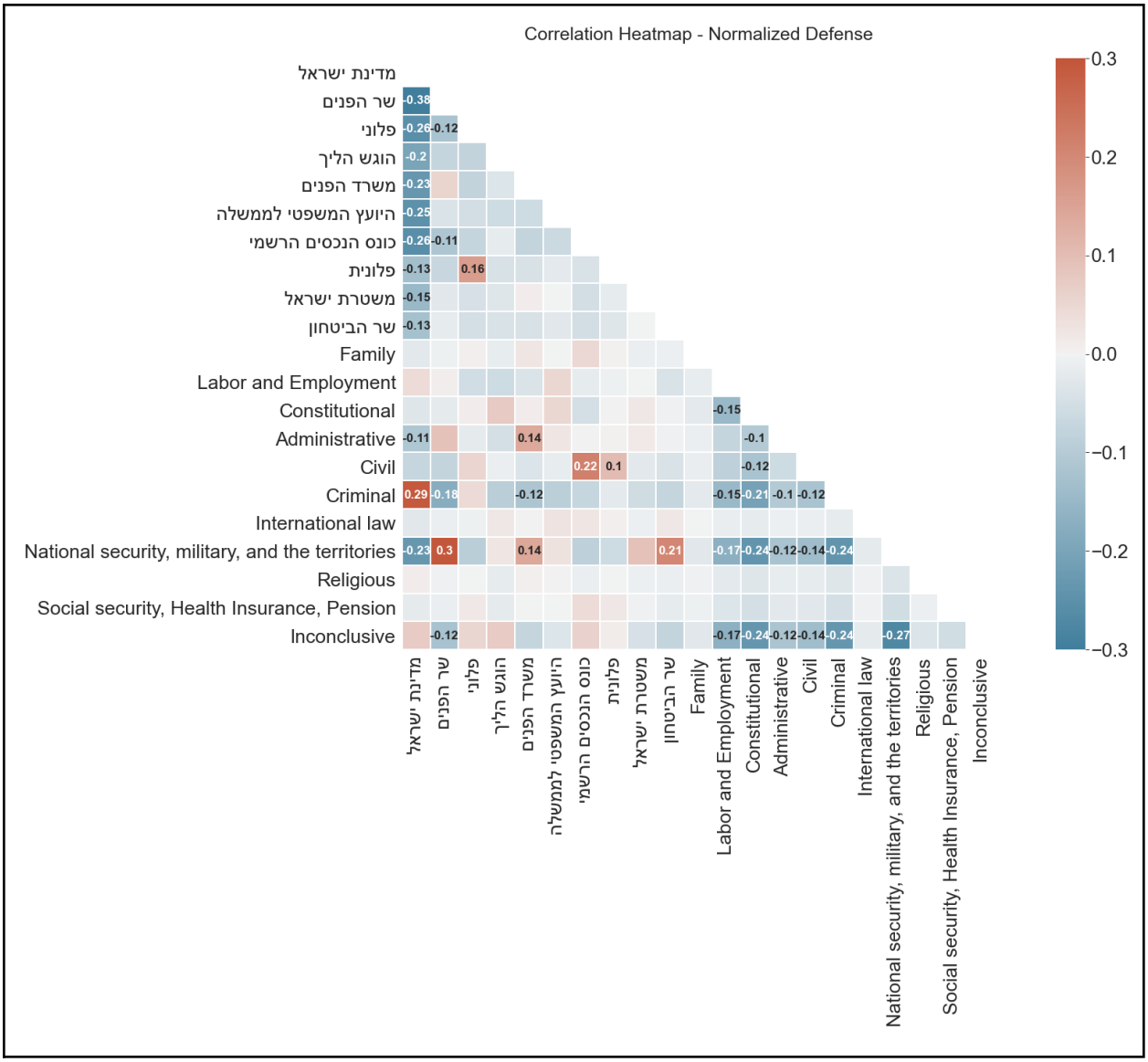


Table 22. Correlation Matrix - Defense

Conclusion

Not Normalized Positively Correlated:

מדינת ישראל	Labor and Employment	0.563491
מדינת ישראל	Criminal	0.317608
שר הפנים	National security, military, and the territories	0.307228

Table 23. Defense Not Normalized Positively Correlated

Not Normalized Negatively Correlated:

מדינת ישראל	שר הפנים	-0.323492
שר הפנים	Criminal	-0.169035
מדינת ישראל	Labor and Employment	-0.156580

Table 24. Defense Not Normalized Negatively Correlated

Normalized Positively Correlated:

שר הפנים	National security, military, and the territories	0.297262
מדינת ישראל	Criminal	0.290903
כונס הנכסים הרשמי	Civil	0.217648

Table 25. Defense Normalized Positively Correlated

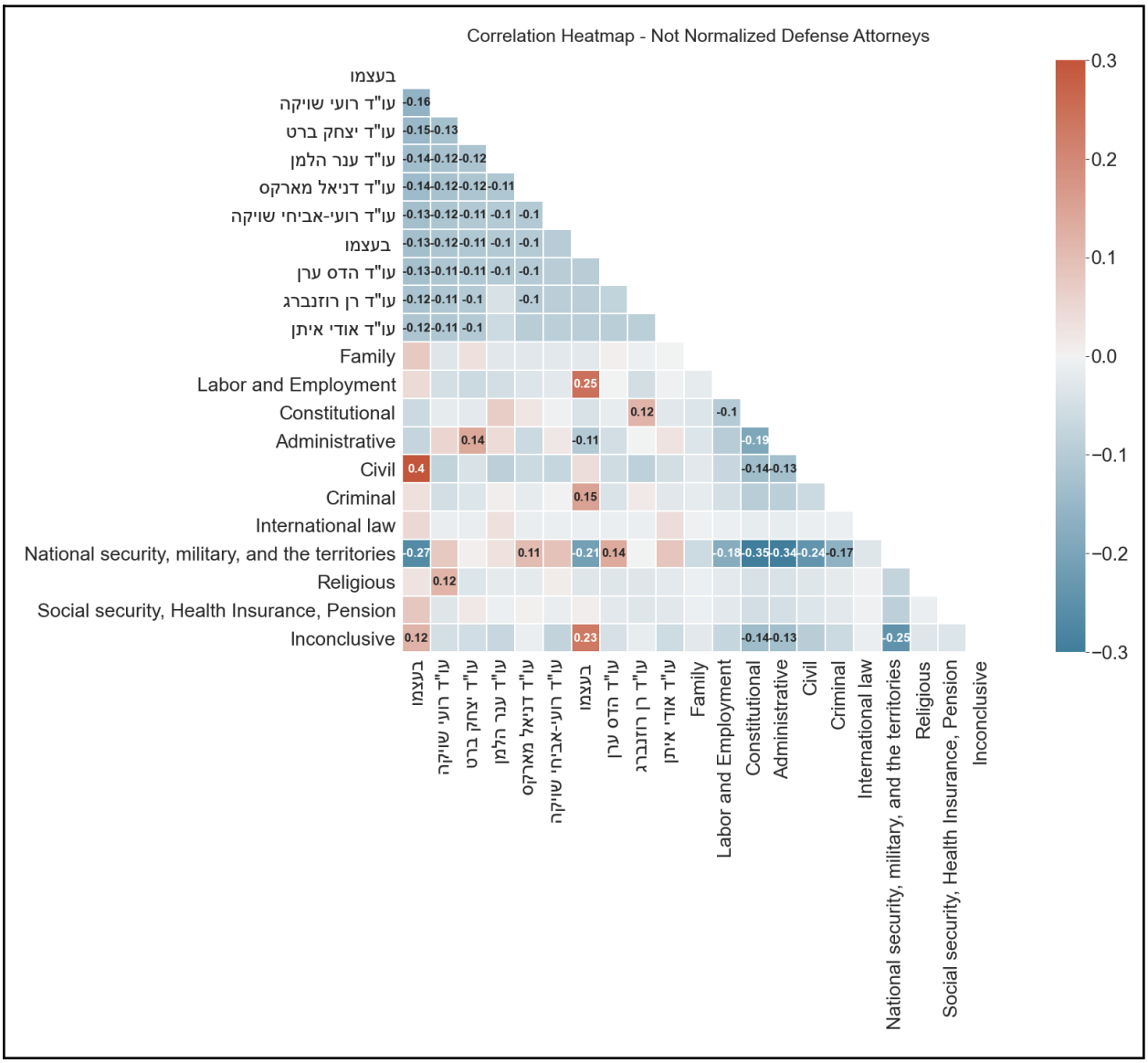
Normalized Negatively Correlated:

מדינת ישראל	שר הפנים	-0.381947
מדינת ישראל	National security, military, and the territories	-0.232758
שר הפנים	Criminal	-0.176312

Table 26. Defense Normalized Negatively Correlated

Defense Attorneys

Histogram



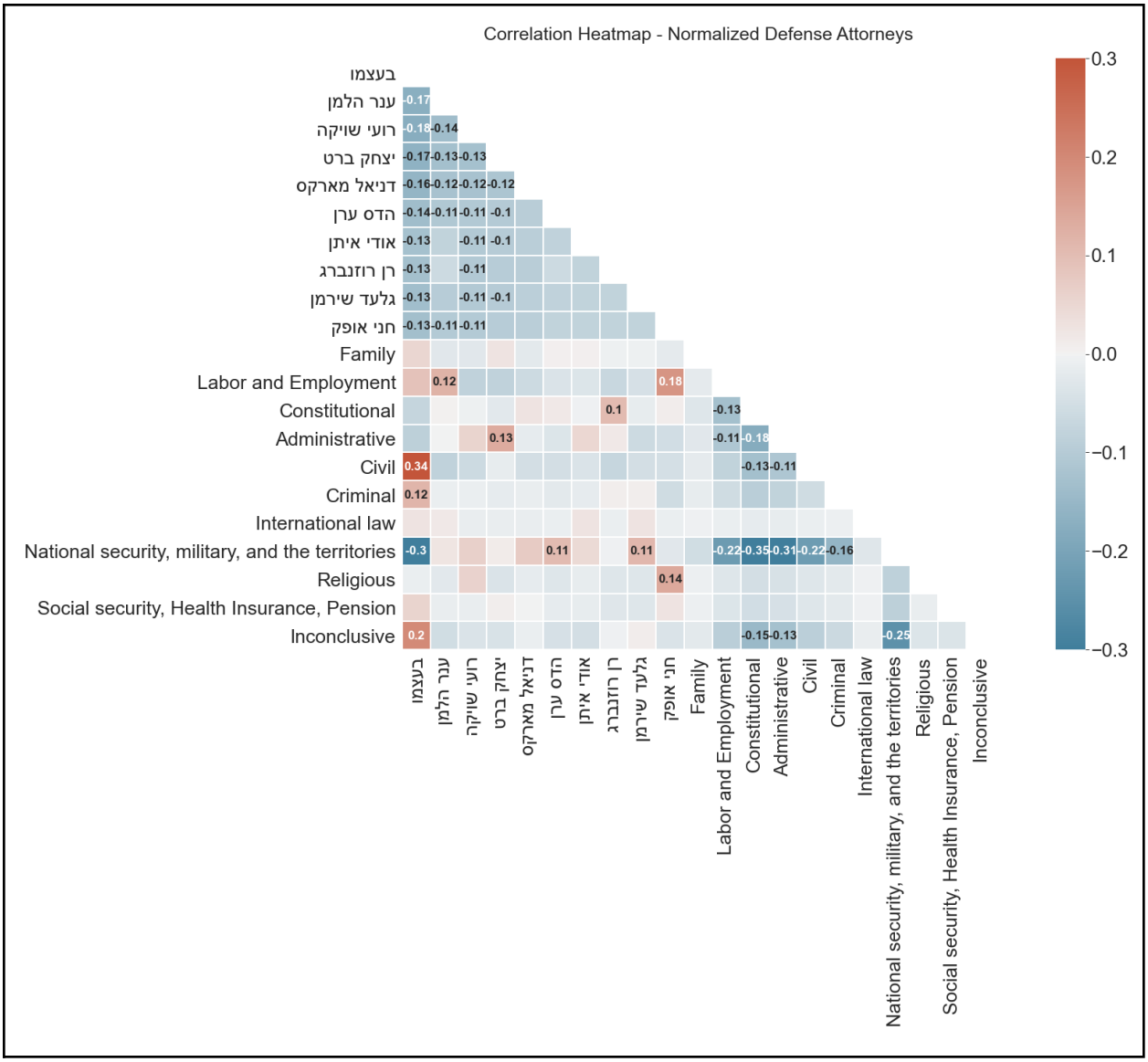


Table 27. Correlation Matrix - Defense Attorneys

Conclusion

Not Normalized Positively Correlated:

בעצמו	Civil	0.397564
בעצמו	Labor and Employment	0.245378
עו"ד יצחק ברט	Administrative	0.142669

Table 28. Defense Attorneys Not Normalized Positively Correlated

Not Normalized Negatively Correlated:

בעצמו	National security, military, and the territories	-0.266935
עו"ד רועי שויקה	עו"ד יצחק ברט	-0.128356
עו"ד ענר הלמן	עו"ד יצחק ברט	-0.117280

Table 29. Defense Attorneys Not Normalized Negatively Correlated

Normalized Positively Correlated:

בעצמו	Civil	0.344158
חני אופק	Labor and Employment	0.176065
יצחק ברט	Administrative	0.133833

Table 30. Defense Attorneys Normalized Positively Correlated

Normalized Negatively Correlated:

בעצמו	National security, military, and the territories	-0.302386
ענר הלמן	רועי שויקה	-0.136850
ענר הלמן	יצחק ברט	-0.131131

Table 31. Defense Attorneys Normalized Negatively Correlated

Further Research

HCVA-env-set-up

Remove unused services

The current version of HESU includes the full bundle of services included in the docker-elk original repository. Not all services are necessarily needed. These services can perhaps be removed and the whole repository can be cleaned up.

Add support for logging in kibana/logstash

Currently we write all our logs to the filesystem. This makes tracking the logs a challenging and inconveniente task. This can be improved drastically if we could use the power of Kibana with Logstash. We have all the needed framework, we just need to add logging support. This will allow us to improve our capability to pinpoint errors and cut down on debugging issues.

Support updated versions of elasticsearch

Elasticsearch v7.10.2 is set to be deprecated in mid 2022. This means there might be a need for future iterations of the project to update the plugin to allow for the upgrade of the Elasticsearch version. Further research and discussion is needed in order to decide if this is a viable option. Otherwise, all future work will be limited to v7.10.2.

Add newer versions of Hspell dictionary to improve the Hebmorph

Because the upgrade of the plugin version was a challenging task, we did not have time to directly improve the plugin engine itself. Newer versions of the Hspell dictionary have been released since the latest official release of the plugin (v1.2). We suggest updating the plugin with one of those versions, either v1.3 or v1.4. The upgrade might improve the capabilities of the Hebmorph engine.

HebrewCourtVerdictsAnalyzer

Scraper

Update chrome webdriver if needed / auto download latest version
Currently we are using a tightly coupled webdriver version to run the scraper. This is true for both the Chrome and the Firefox webdriver. This might cause issues in the future while developing or when relying on the Firefox version on the dedicated server. The current versions used in the project may soon be deprecated as newer versions of Chrome and Firefox are released often. We suggest creating a mechanism to support dynamically updated webdriver that will remove the need to be dependent on a certain version of the webdriver. This will certainly cut down on development time by removing the need for downloading a new version of the webdriver for each release of Chrome or Firefox.

Parser

Consider rewriting the Parser or combining it with the Enricher
Most of the work added to the Parser was done externally, without a dependency on the Parser itself. This fact, combined with the current state of the Parser leads us to suggest refactoring the Parser, either by including the Enricher or alternatively merging the Parser into the Enricher, will be much more development friendly and will make more sense architecturally.

Elastic

Adding additional dashboards
So far, all data saved to Elasticsearch was used only by us for researching purposes. In the future, more people, most likely non-developers, might use the system for their own research. We suggest adding additional custom dashboards to Kibana to allow anybody to use the system comfortably.

Opening access for remote users
This leads us to the next suggestion. Adding support for external users. This will include authorizing and creating new users in the system, both on the OS and in Kibana. Kibana supports multiple user access and can be modified to provide custom dashboards to each user (or user type).

Add simple UI to allow the average user to use the power of Elasticsearch
If the first option proves to be too challenging, consider creating a simple app that provides an easy UI to allow external users to use the system.

Project Architecture

Consider splitting to micro-services

Finally, an idea that arose during development was to perhaps change the architecture of the project from a monolith into micro-services. This will allow us to remove the need to communicate between components using the filesystem. This will also improve the development by removing dependencies between the components. We could work and update only one component without shutting down the whole system each time.

Add CI/CD

Implement a basic CI/CD integration using github actions. Although we have not faced too many issues directly while trying to install the project, adding a simple pipeline is a good start to check all parts are initialized correctly. This is the right approach moving forward as more features will be added to the project.

JSON- SCHEME

A change in the scheme might improve Kibana's enumerating capabilities. The suggested change is to separate the names to different attributes instead of a list of strings as one.

```
'המשיב' : ['one','two']
```

Table 32. Old Names JSON Attribute

```
{
      "המשיב": [
            {
                  "name": "one"
            },
            {
                  "name": "two"
            }
      ]
}
```

Table 33. New Names JSON Attribute

The change will improve the data holding capabilities for every node. The more data it holds - the more data to be examined as a whole. Such examples are “court activity” for a lawyer and “justice role” for a judge.

HCVA - ML

Formatting Case Results - Who Won The Case?

Currently, the case's results are not a part of the research. This means that we are not taking into consideration the fact that in a certain case the petitioner or the defense won. By correlating these values with the legal personal and the categories we can get a glimpse into the actual verdict result based on who are the players involved in it.

Normalization

Lawyer Database

The list of judges is found on the web, thus, an injective function between the pre-processed legal name and its full name has been successfully made. In order to achieve best normalization results, a list of all Israeli valid lawyer names should be searched for and created. The current logic must be taken into account because the key-value approach has already been implemented in the normalizer. Right now adding the defense and petitioner attorneys to the object's legal dictionary is possible and had been forbidden in order to allow high resolution debugging on judges names only.

Names Fix Mechanism

A naming mechanism for all names is suggested. The mechanism should include a way of fixing the input name. Some utilities are already included in the normalizer.py class skeleton: jellyfish library (Jellyfish) which has the Lewenstein distance (Levenshtein distance) method. That method checks for the differences between two strings. By mixing it with a frequency mechanism, the normalization results can be improved to near perfection. With that being said, a trimming mechanism must be implemented as part of the solution. Note 'Petitioner' normalized values has two similar objects represented differently - 'המוקד להגנת הפרט מיסודה של ד"ר לוטה זלצברג' and 'המוקד להגנת הפרט'.

Leaks

Check why 'הוגש הליך' is being passed as defense - leak or special labeling. May be solved by "consider re-writing the parser or combining with enricher".

Classification

Model Alternatives

The current approach is using a Multinomial Naive Bayes model in order to classify the verdict. There are multiple other choices that could achieve other results. SVM Classifier is one of the available solutions.

Multi Category Classification

Right now the predicted category with the highest value is being selected as the verdict's category. This algorithm may support multi classification by adding a threshold value. This may allow verdicts to be classified into various categories. Keep in mind that low threshold will cause unwanted behaviour. The procedure must be hyper-parametrized - a number of thresholds need to be examined before selecting one. Another thing to be aware of is to check whether the highest value is greater than the threshold. If it is - check for more values, whereas if it does not - the highest valued category is the only one.

Improving Verdict Category Classifiers Separately

Despite being hyptermatrized - a deeper analysis of each category will ultimately lead to better prediction. Examining the transformer vocabulary and looking for crossed categorical keywords will empower single classifier abilities.

Decrease Maltransformed Cases Summaries

Some of the cases summaries led into a 50%-50% classifier prediction. The approach we use is to take the maximal value - hence the maximal categorical prediction regardless of its value. That approach can lead into a scenario when multiple categories have the same prediction values. The max() python function does not resolve this issue in a nicely fashion - it just takes the first object that has the maximal value. Needless to say - it leads to bad results from a researcher perspective. The current project iteration solves this issue by addressing it with a threshold approach - any value less or equal to 0.5 will be considered as an "Inconclusive" category.

Appendix

A. Elasticsearch Schema

```
{
        "$schema": "http://json-schema.org/draft-07/schema#",
        "description": "validate json object",
        "title": "supreme_court_rulings",
        "type": "object",
        "additionalProperties": true,
        "required": [
                "Doc Details"
        ],
        "properties": {
                "Doc Details": {
                        "type": "object",
                        "required": [
                                ","לפני
                                ,"העותר"
                                ,"המשיב"
                                ,"מספר הליך"
                                ,"סוג מסמך"
                                ,"סיכום"
                                ,"עמודים"
                                ,"תאריך"
                                ,"לפני מנורמל"
                                ,"העותר מנורמל"
                                ,"המשיב מנורמל"
                                ,"בשם המשיב מנורמל"
                                ,"בשם העותר מנורמל"
                                ,"בשם המשיב"
                                "בשם העותר""
                        ],
                        "items": {
                                "properties": {
                                        "עמודים": {
                                                "type": "string"
                                        },
                                        "תאריך": {
                                                "type": "integer"
```

```
},
"מספר הליך": {
        "type": "string"
},
"לפני": {
        "type": "array"
},
"העותר": {
        "type": "array"
},
"המשיב": {
        "type": "array"
},
"סוג מסמך": {
        "type": "string"
},
"סיכום": {
        "type": "string"
},
"לפני מנורמל": {
        "type": "string"
},
"העותר מנורמל": {
        "type": "string"
},
"המשיב מנורמל": {
        "type": "string"
},
"בשם העותר מנורמל": {
        "type": "string"
},
"בשם המשיב מנורמל": {
        "type": "string"
},
"בשם העותר": {
        "type": "string"
},
"בשם המשיב": {
        "type": "string"
}
```

```
                        }
                    }
                }
            }
}
```

Table 34. ElasticSearch Schema

B. Initial Installation

1. **Update Aptitude:**
   $ sudo apt-get update
2. **Git**
   $ sudo apt-get install git
3. **Python3**
   $ sudo apt install python3.9
4. **Pip3**
   $ sudo apt-get -y install python3-pip
5. **Docker** (from the official documentation)
   Installing the repository
   $ sudo apt-get remove docker docker-engine docker.io containerd runc
   $ sudo apt-get update
   $ sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \ gnupg \ lsb-release
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
   echo \ "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
   Installing Docker Engine
   $ sudo apt-get update
   $ sudo apt-get install docker-ce docker-ce-cli containerd.io
   Test
   $ sudo docker run hello-world
   Finally, add privileges to users: sudo chmod 666 /var/run/docker.sock

C. HCVA-ML Set up

Using the Pipe-Lined Classifier

Check and confirm that the utils file has all the correct path routes and match the class requirements:
utils.NGRAM_JSON

Check and confirm that the settings file has all the correct path routes and match the class requirements:
settings.CLASSIFIERS

Using the Classifier as an isolated component

- Check all the required libraries are installed.
- Check that you have the 'models' folder of the pre-trained models and vectorizers.
- Check that the path routes are updated - as mentioned above - "Using the Pipe-Lined Classifier"
- Instantiate the object in your favorite environment.
- Use the method "classify" to predict your desired verdict summary string category.

Using the pipe-lined Normalizer

Check and confirm that the utils file has all the correct path routes and match the class requirements:
utils.COMMON_TITLES_CSV
utils.SW_AFTER_FILTER_CSV
utils.COURT_JUDGES

This module uses the settings file for further development compatibility.

Using the Normalizer as an isolated component

- Check all the required libraries are installed.
- Check all of the path routes are updated - as mentioned above - "Using the pipe-lined Normalizer"
- Instantiate the object in your favorite environment.
- Use the method pre_process_legal to pre_process legal system related names.

- Use the method pre_process_not_legal to pre_process non-legal system related names.
- Pay attention if you want a fixed judge name, if so, you will need to use the "fix_judge_names" method and pass the list of judge names and the path of the judges names text file.

Notebooks

A number of Jupyter notebooks have been written as the project progressed. Those notebooks can be found in the Google Drive listed below as well as the Github repository. Keep in mind that the parts that are fully commented on are the pipe-lined parts - Enricher.py, Normalizer.py, Classifier.py. Comments can be found on specific notebooks but it is not mandatory, these notebooks are the pre-production workspace.

Important Note: NEVER RUN ANY OF THE NOTEBOOKS BEFORE EXAMINING IT ALL. DO SO, AT YOUR OWN RISK - IT WILL POTENTIALLY RESULT IN UNINTENDED BEHAVIOR.

Normalization.ipynb
Classification.ipynb
Utilities.ipynb
Enrichment Results.ipynb
Verdict json manipulation

Models and Vectorizers

In order for the program to run, we need to place the 'models' folder in our root execution folder. The 'models' folder is too big for Github to store, as there is a 100mb limit. Therefore, we've added it to the project's dedicated Google Drive and all files can be found there.

Google Drive

The Google Drive folder tree has three folders - Jupyter Notebooks, models and Verdicts. Jupyter Notebook has the notebook specified above. The 'models' folder has the Multinomial Naive Bayes model files and tf-idf vectorization files. Verdicts folder contains the extracted verdicts in order to work locally on your machine - H.I.T infrastructure, this year, was unstable. You can contact us directly for permission grant.
Link -
https://drive.google.com/drive/folders/1qL6reXzC3MDDG8_LTxdJJb8MFgq9zXOl?usp=sharing

D. Valid output of full project run

```
project2021@court1-virtual-machine:~/HebrewCourtVerdictsAnalyzer$ ./init_app.sh
project2021@court1-virtual-machine:~/HebrewCourtVerdictsAnalyzer$
```

Figure 34. Valid Output of Full Project Run

E. Compared index differences

Old version of the index - v5.5.3:

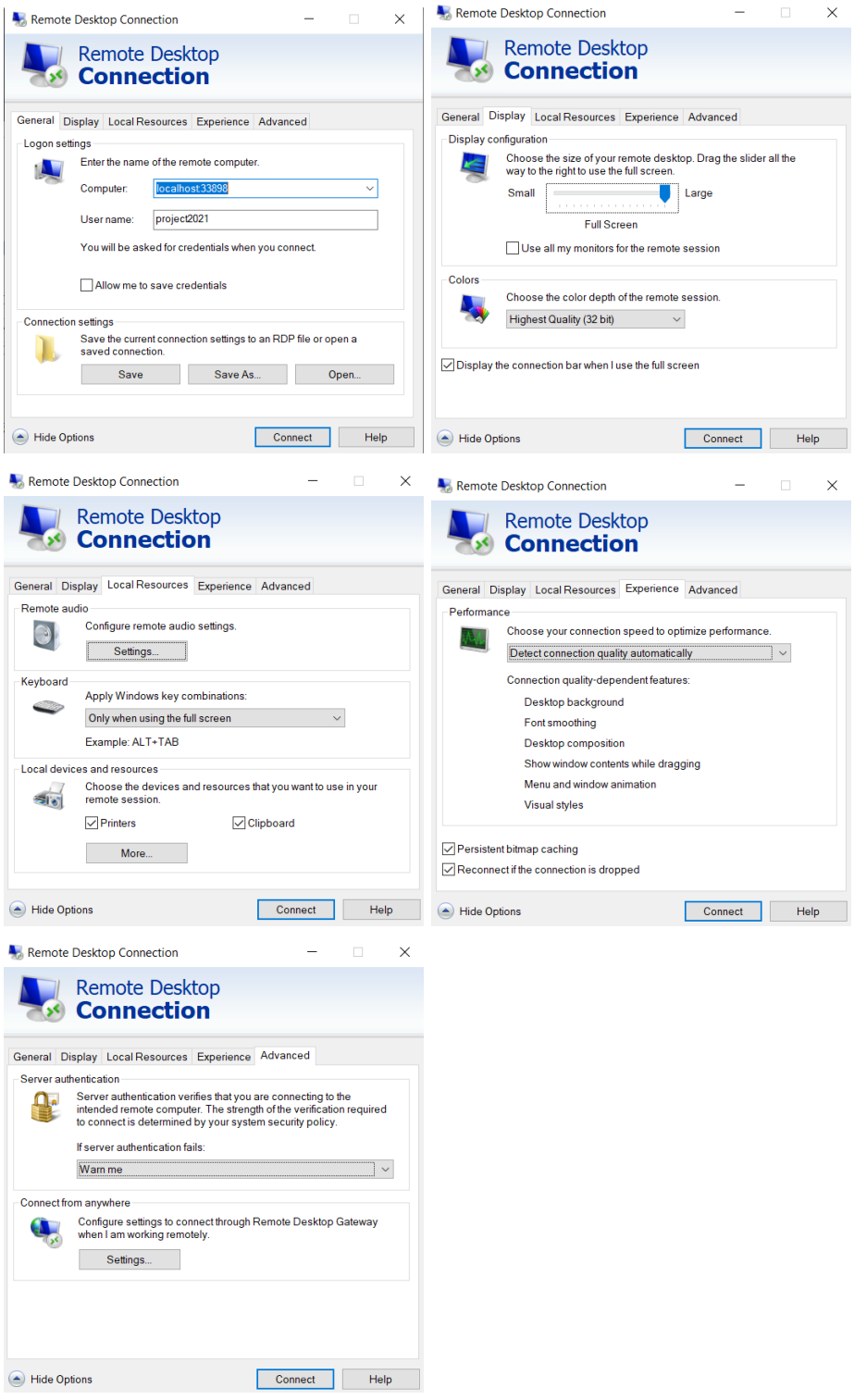
Figure 35. Old Index 5.5.3

The updated version - v7.10.2:

Figure 36. New Index 7.10.2

F. Remote Connection

In order to connect remotely type the following command in Windows PowerShell:
ssh -L 33898:localhost:3389 project2021@192.114.5.109 -p3022
Then launch an RDP program, for windows - type mstsc in the “Run” window.



The windows file is on the google drive.

Drawings

List Of Images

List Of Tables

Bibliography

Chromium. *Downloads*. ChromeDriver. https://chromedriver.chromium.org/downloads

Conseil de L'Europe. (2020). Evaluation of the judicial systems (2018 - 2020). *The European Commission for the Efficiency of Justice*, 104.

deviantony. *docker-elk*. Github. https://github.com/deviantony/docker-elk

Docker. *Docker Desktop*. https://www.docker.com/products/docker-desktop

Elastic. *Elastic product end of life dates*. elastic.co. https://www.elastic.co/support/eol

Elastic. *Language Analyzers*. Elasticsearch Guide [7.10]. https://www.elastic.co/guide/en/elasticsearch/reference/7.10/analysis-lang-analyzer.html#analysis-lang-analyzer

Elastic. *Removal of Types*. Elasticsearch Guide [7.10]. https://www.elastic.co/guide/en/elasticsearch/reference/7.10/removal-of-types.html#removal-of-types

Elastic. *Upgrade Elasticsearch*. Elasticsearch Guide [7.10]. https://www.elastic.co/guide/en/elasticsearch/reference/7.10/setup-upgrade.html

IBM. *nohup Command*. IBM AIX. https://www.ibm.com/docs/zh/aix/7.2?topic=n-nohup-command

Immanuelbh. *HCVA-env-set-up*. Github. https://github.com/Immanuelbh/HCVA-env-set-up

Israel Ministry of Foreign Affairs. *The Judiciary: The Court System*. mfa.gov.il. Retrieved 10 9, 2021, from https://www.mfa.gov.il/mfa/aboutisrael/state/democracy/pages/the%20judiciary-%20the%20court%20system.aspx

Jellyfish. Jellyfish. https://pypi.org/project/jellyfish/

Levenshtein distance. Levenshtein distance. https://en.wikipedia.org/wiki/Levenshtein_distance

Multinomial Naive Bayes.

https://scikit-learn.org/stable/modules/generated/sklearn.naivebayes.MultinomialNB.html

pandas.DataFrame. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

pickle. https://docs.python.org/3/library/pickle.htm

Revah, L., & Shalumov, B. *AccessibleCourtData*. Github.

https://github.com/liron7722/AccessibleCourtData

synhershko. *elasticsearch-analysis-hebrew*. Github.

https://github.com/synhershko/elasticsearch-analysis-hebrew

synhershko. (2019). *HebMorph*. Github. https://github.com/synhershko/HebMorph

Tfidf Vectorizer.

https://scikit-learn.org/stable/modules/generated/sklearn.featureextraction.text.TfidfVectorizer.html

Train Test Split.

https://scikit-learn.org/stable/modules/generated/sklearn.modelselection.traintestsplit.html.