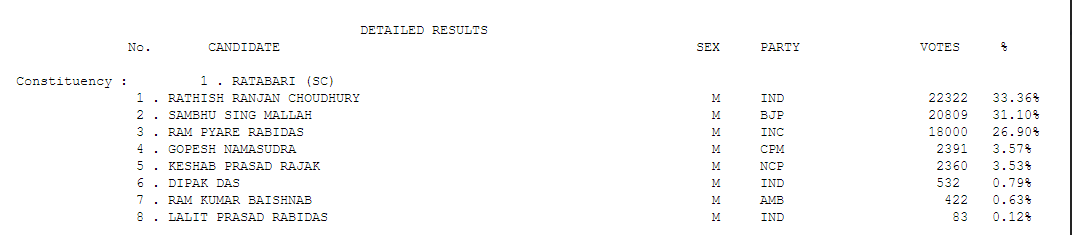
Elections Report Parser

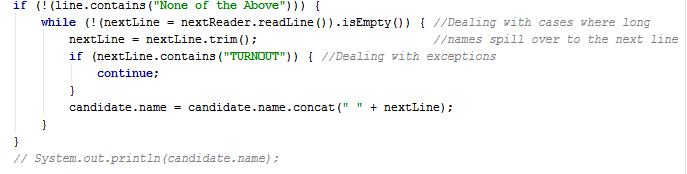
The aim of this project is to create a configurable parser for election results of every state in India over the last 15 years. Each state is divided into constituencies and political leaders of a party represent different constituencies. At the end of each election, information about each constituency is published by the Government. However since this is only available in a PDF format, one cannot extract select data effectively. Our aim is to simplify this process for the user by ensuring that data is categorized based on age, gender, political party etc.

In the past 15 years, one could identify that there were two distinct patterns of statistical reports. While some documents published had nine different categories, others only took the age, gender, political party and total votes won into consideration. Thus the program asks the user to configure their requirements as per their needs. The user can select from two different layouts depending on what file they want to parse.



Since data is read from a notepad file, one may consider using the String Tokenizer in order to extract Strings, convert them into suitable primitive types and store information. On the other hand, one could also consider using the Regular Expressions. The main advantage of using the regex library in Java is that one can create his/her own customized syntax to match words. In the process of categorization, one must be clear of the well defined boundaries. The gender category could be recognized by a simple M or F, whereas the party name could be confused with what category the person belongs to. Hence the regex patterns ensured that while matching a party name: SC, ST and GEN were to be excluded. Similarly, the regex pattern for name of the political leaders specified that a number fallowed by a white space and words are present. Thus each category was clearly distinguishable.

Although regex patterns were customized, sometimes the PDF to notepad conversion led to a lot of errors. Due to notepad errors, one could not differentiate between name of a person and the keyword “TURNOUT”; the matcher would recognize TURNOUT as the name and this would lead to a lot of confusion. Furthermore people often chose not to vote for any of the political parties, in such a situation we had to incorporate the case wherein the file directly reads the Party name as NOTA and move to the next number.



There were also situations where in the name or the political party spilled over to the next line. In such a situation, we must read the name and concatenate it with the name in the line above. As shown in the picture, each case is thus handled to avoid data discrepancy.

The final output is a file in CSV format. According to the choice of the user, the sections with vital information are displayed. The CSV file can be used in analyzing data. Users may choose to use Programming languages such as Matlab, Python and analyze data accordingly. The age of political leaders, the sex ratio or the total number of votes can prove to be vital for parties as well as the Election Commission of India. It gives them an idea about which constituency has a weaker sex ratio or the total number of citizens who chose to vote and the means by which people generally wanted to vote.

The program was tested in for states such as Goa, UP, Assam, Bihar and West Bengal. In the time span of 2000-2016, the files were efficiently read and any kind of error seemed to have been prevented. Thus the program’s layout facility helps making it functional in a large number of files.

One of the important concepts we learnt through this project is that the program must not be hardcoded. In a number of situations, we felt that the best version would be to handle cases on one file differently, however a change made in program due to one file turned out to be inefficient as it is designed to work for all kinds of legislative files between 2000 and 2016. Secondly, we must design the algorithm before starting with the program. It was crucial for us to define the regex patterns of different categories in pen and paper before we could implement it. Similarly we clearly designed the program to have a maximum of 8 fields. Thus ensuring the program skips necessary categories depending on the user’s choice. Lastly, we also learnt that debugging is the most important part of programming as majority of the time was taken to analyze the output and accordingly fixing the various errors observed in those files. It also helped us in understanding the structure of our own program as we realized how switch statements altered, and what the pattern matchers observed in specific lines of notepad that created confusions.

Work Distribution:

Input/ Output and creation of Field Counter, a design of the CSV file and exceptions- Paul

Fixing regex patters, pattern matching and presentation/write up- Shreyash

Finally both of us together spent time on fixing errors and thus debugging.