Candidate Nr: 117, 142 Word count: 627

News graph aggregator

The purpose of this system is to take a stream of small news graphs, identify the similarity of the graphs and merge them to a bigger news graph. The new and bigger graph can be an indicator of what is trending in the news. This information can be used for statistical and scientific purposes, but also in a commercial way. By merging them automatically, it reduces time and effort for the end user. The system should play a part in a bigger, more complex system. Combined with a web crawler, graphs can be added automatically. An analysis tool can save computing time by only focusing on the concatenated graphs. The feedback for the analysis tool can help a news station focus their resources, and also discover new newsworthy topics. By using semantic technologies to represent the data, it gives the data higher quality. Structuring the data in a semantic way, the data can be interpreted by humans as well as machines.

For processing the graph we are using spaCy and Sklearn TfidfVectorizer. SpaCy is a library for NLP, it has pre-trained neural networks for interpreting text. For the first analysis we are sending the entities as a string into the spaCy model to be analyzed. After that we are sending the “ns1:originalText” element to be analyzed by spaCy and the Sklearn model. By using spaCy pre-trained networks, we don't have to train our own model which can be time consuming and maybe not feasible in the time we had.
The TfidfVectorizer model, like the SpaCy model, converts the word into vectors that are being compared. The difference between the two models is that the Tfidf takes into account the word frequency, and calculates the cosine similarity by taking their dot product and divides that by the product of their norms, which gives the cosine of the angle between vectors.

For handling the incoming stream of graphs, we are using a watchdog. Watchdog is a python module. It works by checking for changes in a given file or directory, and reports those changes. When it detects a change in the directory , the program sends the new files on for parsing and processing, and compares all newly added graphs with any potential old graphs in the directory. After all graphs have been compared, and new, merged graphs have been created, the program goes back to monitoring for any incoming graphs again, and so the process repeats.

The program is designed to be implemented into the News Hunter platform to merge the graphs that are found together, and we have used some specific prefixes in the program to make it work with the News Hunter. It would need to be tweaked in order to work for other platforms.

Conclusion

The program completes the task and merges graphs with a somewhat high accuracy, with low false positives but the program is perhaps too dependent on text analysis. If we could have done anything differently, we would have liked to use RDF2Vec, which creates a numeric vector for each node in a graph. This tool is more coherent with the course. We discovered the tool too late in the development process, and did not have enough time to implement it. We are also missing a storage solution as for now the new graphs are only added to the ‘MergingGraph’ directory. A storage solution is something that should be implemented in the future. The learning curve for this project has been steep, and we have encountered problems that haven't been an issue before. Not only to find the right tools for the task, but also programming challenges like recursion depth. The knowledge we have achieved after this task has given us a good foundation for projects in the future.

Contributions

117

Programming: File handling, parsing and filtering the graphs, implementations of process models, storage and graphandling, functions for creation of new graphs and extending existing graphs, implementation of watchdog, terminal interface , making of a requierment.txt and researching.
Written material: Readme file and report writing.

142

Programming: Parsing and filtering graphs, functions for creating new graphs, comparing similarity of graphs, graph merging, storage and graph handling, terminal, stress-testing, error handling.
Written material: Readme file and report writing.