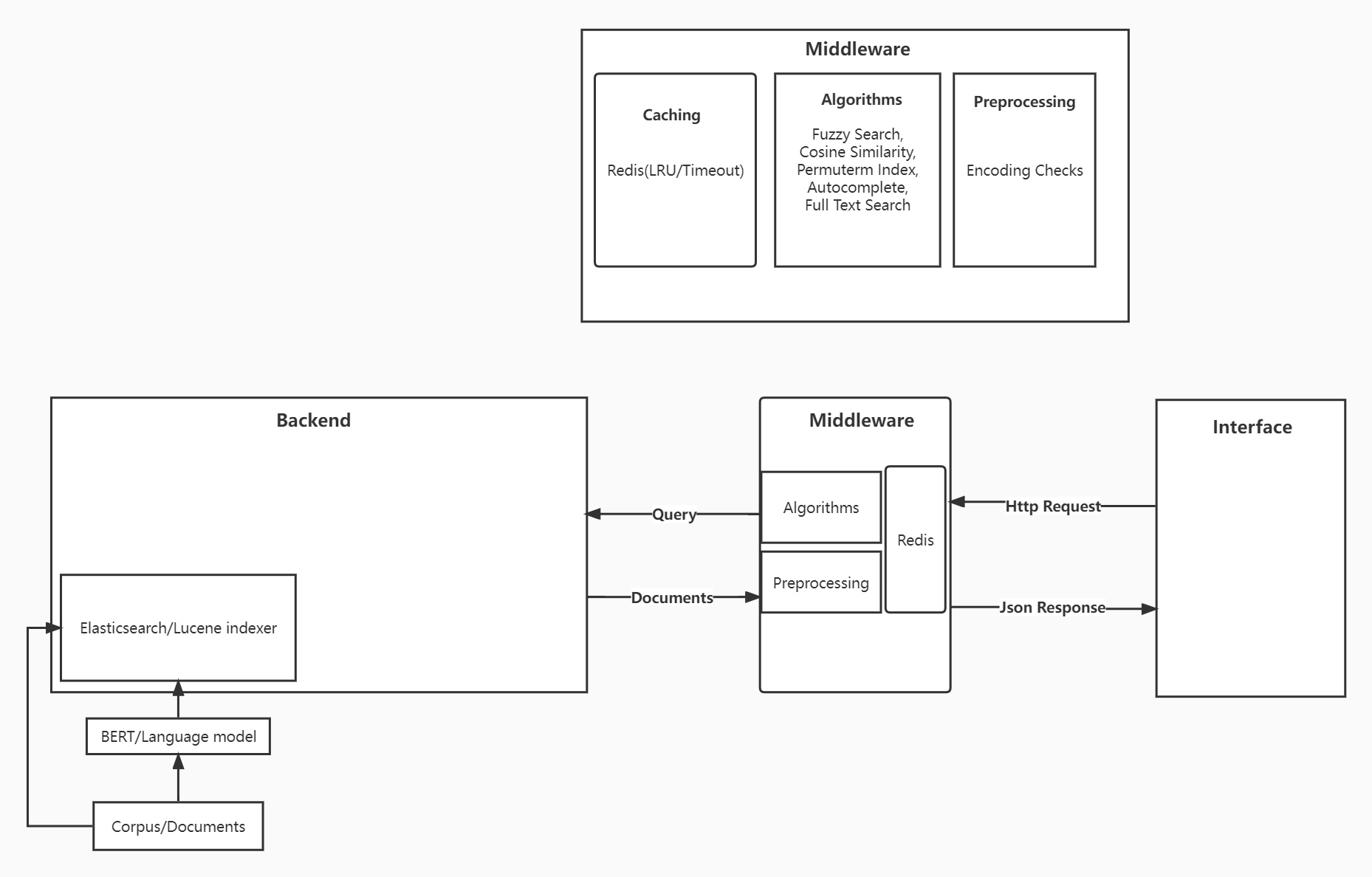
**COMP631 Project Task 2**

****

**Team Members**: Yuan Chen yc149@rice.edu Haodong Qi hq9@rice.edu Kun Lai

**Summary**

For our project, our group choose to use Elasticsearch as our primary backend instead of Solr because Elasticsearch is a more scalable and new frameworks compared to Solr. During task2, we did additional text preprocessing, text distill, feature engineering, and more. Also, we decided to add more supplmary documents in addition to the reddit posts.

**Comparison between solr and elasticsearch**

Elasticsearch is also an open-source search engine built on top of Apache Lucene, as the rest of the ELK Stack, including Logstash and Kibana. It extends Lucene’s powerful indexing and search functionalities using RESTful APIs, and it archives the distribution of data on multiple servers using the index and shards concept. Elasticsearch is completely based on JSON and is suitable for time series and NoSQL data. Some of its primary features include distributed full-text distributed search, high availability, powerful query DSL, multitenancy, Geo Search, and horizontal scaling.

Apache Solr is an open-source search server built on top of Lucene that provides all of Lucene’s search capabilities through HTTP requests. It has been around for almost a decade and a half, making it a mature product with a broad user community. Solr offers powerful features such as distributed full-text search, faceting, near real-time indexing, high availability, NoSQL features, integrations with big data tools such as Hadoop, and the ability to handle rich-text documents such as Word and PDF.

However, base on our research, solr is only limited to small to medium sized project, while Elasticsearch can support project at any size. Thus, we chose Elasticsearch as our backend.

**Text Preprocessing and Text Distill**

During task2, we took advantage of google and retrieved relevant URL links and crawled those links to enrich our corpus base, because we realized that most of the times reddit posts contain low quality information.

First of all, we removed all pages with little information or non-asccii encoding characters. Then, we feeds all the documents through a summarization model (Using the package from here [github.com/summanlp/textrank](https://github.com/summanlp/textrank)). The summarization model will perform a textrank extract keywords and keysentences for later usage. Lastly, we feeds all the key sentences to the base version of BERT(<https://github.com/google-research/bert>), to extract 768 feacture vector and store it in to our backend.

**Mapping Engineering**

Although Elasticsearch supports dynamic type mapping, it will consume extra disk storage. Thus, we engineered the mapping for search data types for our documents, which is shown at the last page.

**Demo:**

<https://youtu.be/BtLjMINgk5E>

