1 Objective

In this assignment, you will build a search engine for sentiment analysis. In particular, you will 1) crawl a text corpus of your interest, 2) build a search engine to query over the corpus, and 3) perform sentiment analysis over it. The system can be about any domain and data you like. For example, it could be a system for social media marketing, political forecasting, healthcare, financial forecasting, or a recommendation system. For more ideas, you can check our project page at sentic.net/projects.

Your final score will depend not only on how you developed your system but also on its novelty and your creativity: in other words, to get a high score you do not only need to implement a system that works, but also a system that is kind of useful and user-friendly. If you are not sure whether your idea for the assignment is good, you can discuss it with the course coordinator anytime after class or via email.

2 Deadline

The assignment constitutes 35% of your total grade for the course. Assignments are to be submitted via Blackboard (email submissions will not be considered) by Sunday 9th April at 11:59pm SGT. 5% points will be deducted for each rounded-off day after the deadline. Blackboard allows you to do multiple submissions but only the last uploaded version will be marked (please try not to make too many submissions: ideally only one).

3 Grouping

The assignment will be done in groups of 6 or 5 people. Members will be randomly assigned by the system for fairness. The main tasks of the assignment are crawling (20 points), indexing (40 points) and classification (40 points). If you like, you can split your group into up to three subgroups taking care of each of these tasks and specify who did what in your final report so that each member will be graded accordingly. If this information is not specified, a unique grade will be given to the whole project and this will be shared among all members of the group (recommended option).

Some overlap between projects from different groups is allowed but beware that, if we find out that a project has more than 30% overlap with another project from this year or past years, your group will be disqualified (and get zero points as final grade for the assignment). Hence, it is OK to share the general idea of your assignment with other groups but not the implementation details.

4 Tasks and Questions

4.1 Crawling (20 points)

Crawl text data from any sources which you are interested in and permitted to access, e.g., Twitter API or Reddit API. The crawled corpus should have at least 10,000 records and at least 100,000 words. It is OK to use available datasets for training (e.g., popular sentiment benchmarks), but you still have to at least crawl and label data for testing. For your own evaluation dataset, use the same format as those sentiment benchmarks (using a different format may result in demerit points). Also, make sure your dataset does not contain duplicates and try your best to make it balanced (e.g., equal number of positive and negative entries). Before crawling any data, carefully consider the questions in this material, e.g., check whether the data have enough details to answer the questions. You can use any third party libraries for the crawling task, e.g.:

- Jsoup: https://jsoup.org
- Twitter4j: https://twitter4j.org/en/index.html
- Facebook marketing: https://developers.facebook.com/docs/marketing-apis
- Instagram: https://instagram.com/developer
- Amazon: https://github.com/ivanpgs/amazon-crawler
- Tinder: https://gist.github.com/rtt/10403467
- Tik Tok: https://developers.tiktok.com

Question 1: Explain and provide the following:

1. How you crawled the corpus (e.g., source, keywords, API, library) and stored it
2. What kind of information users might like to retrieve from your crawled corpus (i.e., applications), with sample queries
3. The numbers of records, words, and types (i.e., unique words) in the corpus

4.2 Indexing (40 points)

Indexing: You can do this from scratch or use some combination of available tools, e.g., Solr+Lucene+Jetty. Solr runs as a standalone full-text search server within a servlet container such as Jetty. Solr uses the Lucene search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. Useful documentations include:

- Solr project: https://lucene.apache.org/solr
- Solr wiki: https://wiki.apache.org/solr/FrontPage
- Lucene tutorial: https://lucene.apache.org/core/quickstart.html
- Solr with Jetty: https://wiki.apache.org/solr/SolrJetty
- Jetty tutorial: https://wiki.eclipse.org/Jetty/Tutorial

You can also choose other inverted-index text search engine open projects, e.g., Sphinx, Nutch, and Lemur. However, you should NOT simply adopt SQL-based solutions for text search (for example, you CANNOT solve text search simply using Microsoft Sqlserver or MySql).

Querying: You need to provide a simple but friendly user interface (UI) for querying. It could be either a web-based or mobile app based UI. You could use JSP in Java or Django in Python to develop your UI website. Since Solr provides REST-like APIs to access indexes, one extra JSon or RESTFul library would be enough. Otherwise, you may use any third party library. The UI must be kept simple. A sophisticated UI is not necessary nor encouraged (as it is not the focus of this course). Detailed information besides text is allowed to be shown for the query results, e.g., product images on Amazon, ratings on Amazon, and pictures on Instagram. The details should be designed to solve specific problems.

Question 2: Perform the following tasks:

- Build a simple web interface for the search engine (e.g., Google)
- Design a simple UI for crawling and incremental indexing of new data
- Write five queries, get their results, and measure the speed of the querying

Question 3: Explore some innovations for enhancing the indexing and ranking. Explain why they are important to solve specific problems, illustrated with examples. You can list anything that has helped improving your system from the first version to the last one, plus queries that did not work earlier but now work because of the improvements you made. Possible innovations include (but are not limited to) the following:

- Enhanced search (e.g., add histograms, timelines, pie charts, or word clouds)
- Interactive search (e.g., refine search results based on users' relevance feedback)
- Geo-spatial search (e.g., use map information to refine query results and improve visualization)
- Multimodal search (e.g., implement image or video retrieval)
- Multilingual search (e.g., enable your system to retrieve data in multiple languages)
- Multifaceted search (e.g., visualize information according to different categories)

4.3 Classification (40 points)

Although often defined as a binary categorization problem, sentiment analysis is actually a complex task, or suitcase research problem, as it requires tackling many other subtasks. Choose at least two subtasks to perform information extraction on your crawled data. Unless you are sure that your data does not contain any neutral content, you should always cover at least subjectivity detection and polarity detection. Namely, you should first categorize your data as *neutral* versus *opinionated* and then classify the resulting opinionated data as *positive* versus *negative*. Different classification approaches can be applied, including:

- knowledge based, e.g., SenticNet
- rule based, e.g., linguistic patterns
- machine learning based, e.g., deep neural networks
- hybrid (a combination of any of the above)

You can tap into any resource or toolkit you like, as long as you motivate your choices and you are able to critically analyze obtained results. Some possible choices include:

- Weka: https://cs.waikato.ac.nz/ml/weka
- Hadoop: https://hadoop.apache.org
- Pylearn2: https://deeplearning.net/software/pylearn2
- SciKit: https://scikit-learn.org
- NLTK: https://nltk.org
- Theano: https://github.com/Theano
- Keras: https://github.com/fchollet/keras
- Tensorflow: https://github.com/tensorflow/tensorflow
- PyTorch: https://pytorch.org
- Huggingface: https://huggingface.co/
- AllenNLP https://github.com/allenai/allennlp

Question 4: Perform the following tasks:

- Motivate the choice of your classification approach in relation with the state of the art
- Discuss whether you had to preprocess data (e.g., microtext normalization) and why
- Build an evaluation dataset by manually labeling 10% of the collected data (at least 1,000 records) with an inter-annotator agreement of at least 80% (it is recommended to have 3 annotators, but 2 is also OK)
- Provide evaluation metrics such as precision, recall, and F-measure on such dataset
- Perform a random accuracy test on the rest of the data and discuss results
- Discuss performance metrics, e.g., records classified per second, and scalability of the system
- Design a simple UI for visualizing classified data

Question 5: Explore some innovations for enhancing classification. If you introduce more than one, perform an ablation study to show the contribution of each innovation. For example, if you perform word sense disambiguation (WSD) and named entity recognition (NER) to enhance sentiment analysis, show the increase in accuracy when adding only WSD, the increase in accuracy when adding only NER, and the increase in accuracy when adding both WSD and NER to your system. Explain why they are important to solve specific problems, illustrated with examples. Possible innovations include (but are not limited to) the following:

- Enhanced classification (add a sentiment analysis subtask, e.g., sarcasm detection)
- Fine-grained classification (e.g., perform aspect-based sentiment analysis)

- Hybrid classification (e.g., apply both symbolic and subsymbolic AI)
- Cognitive classification (e.g., use brain-inspired algorithms)
- Multitask classification (e.g., perform two sentiment analysis tasks jointly)
- Ensemble classification (e.g., use stacked ensemble)

5 Submission

Select one person of the group in charge of submitting the final assignment report. Submission has to be done via Blackboard (do not email your report). If you search CZ4034 on YouTube, you can view some examples from past years. You can take inspiration from them but doing exactly the same project will result in the annulment of your assignment. The submission shall consist of one single PDF file named after your group number, e.g., if you are group 10, your file should be titled simply 10.pdf. Failing to name the file correctly or sending it in the wrong format, e.g., zip or MS Word, may result in demerit points. Do not use the old NTU logo and add some pictures to make your report clearer and easier to read. No page limit nor special formatting required. The file shall contain the following five key items:

1. The names of the group members + matriculation number in the first page
2. Your answers for all the above questions
3. A YouTube link to a video presentation of up to 5 minutes: in the video, introduce your group members and their roles, explain the applications and the impact of your work and highlight, if any, the creative parts of your work (note that you do not have to give all the answers in the video presentation)
4. A Dropbox (or Google Drive) link to a compressed (e.g., zip) file with crawled text data, queries and their results, evaluation dataset, automatic classification results, and any other data for Questions 3 and 5
5. A Dropbox (or Google Drive) link to a compressed (e.g., zip) file with all your source codes and libraries, with a README file that explains how to compile and run the source codes

Good luck! :)