**CS 6320 – Natural Language Processing**

**Fall 2019**

**Dr. Mithun Balakrishna**

**Course Project**

# Project Steps and Deadlines:

* **Project Group Formation**:
  + Due by **TBD**
  + A maximum of two (2) students per project group
  + The group should decide on an appropriate group name
  + One group member should submit a document containing the group name and the group member information i.e. Group name and Group member names, via eLearning
    - Please name the document following the convention “ProjectGroupInfo-GROUPNAME.pdf”, where GROUPNAME is your project group’s name.
    - Submit the document to the “Group Information Submission” assignment inside the “Final Project” folder listed in the course home page on eLearning.
    - Students that want to work on the project individually should also submit this document
  + Students that need help to form a group should meet the Instructor on **TBD** at **TBD** in the class room (GR 2.302)
    - Students that want to work on the project individually do NOT need to do this
* **Project Demo**:
  + Due date: **TBA**
  + Demo sign-up details: **TBA**
  + Submit your project source code and report via eLearning before your group’s allocated demo session:
    - One group member should submit a single zip file containing the following via eLearning:
      * Project source code/script file(s)
      * A ReadMe file with instructions on how to access the project demo
      * Project report in PDF or MS Word document format.
    - Please name the zip archive document following the convention “ProjectFinalSubmission-GROUPNAME.zip”, where GROUPNAME is your project group’s name.
    - Submit the document to the “Project Final Submission” assignment inside the “Final Project” folder listed in the course home page on eLearning.
  + Please hand over a hard copy of the project report before the start of your group’s demo session with the TA

# Project Report

Please write a project report (5 to 10 pages) with the following details:

* + - Problem description
    - Proposed solution
    - Full implementation details
      * Programming tools (including third party software tools used)
      * Architectural diagram
      * Results and error analysis (with appropriate examples)
      * A summary of the problems encountered during the project and how these issues were resolved
      * Pending issues
      * Potential improvements

# Project Description:

For this project, you will design and implement a model that determines how similar two chunks of text are. The similarity score takes an integer value between 1 and 5 (included). The higher the score, the more similar the two chunks are.

In general, semantic textual similarity (STS) is a challenging problem; as it requires both an understanding of lexical-level similarity, and the semantic composition of the two chunks of text being analyzed. As a reference, here are some motivating examples:

*Sentence 1: Birdie is washing itself in the water basin.*

*Sentence 2: The bird is bathing in the sink.*

*Score: 4*

*Comment: Both sentences convey the message that a bird is taking a bath.*

*Sentence 1: The young lady enjoys listening to the guitar.*

*Sentence 2: The young lady enjoys playing the guitar.*

*Score: 2*

*Comment: Both sentences involve a lady and a guitar, but convey different actions i.e. listening to the guitar and playing the guitar respectively.*

The contents of this project can be downloaded from <INSERT LINK HERE>. The project contains these files:

* A data folder containing train, dev and test files.

The train and dev files are of the form:

<Input\_Id><TAB><Sentence 1><TAB><Sentence 2><TAB><Score>

The test file is of the form:

<Input\_Id><TAB><Sentence 1><TAB><Sentence 2>

* A python evaluation script that provides the evaluation metrics for your model.
* A sample prediction file ‘sample.txt’ of the form:

<Input\_Id><TAB><Predicted Tag>

The predictions made by your model must be output in the same format as that given in the prediction file.

The following are the tasks that need to be performed:

1. **Task 1**: Create a class CorpusReader that is able to read the data files and represent the information in a way such that your model can process it.
2. **Task 2**: Implement a deep NLP pipeline to extract the following NLP based features from the natural language statements:
   * Tokenize the two sentences into words.
   * Lemmatize the words to extract lemmas as features
   * Part-of-speech (POS) tag the words to extract POS tag features
   * Perform dependency parsing or full-syntactic parsing to get parse-tree based patterns as features
   * Using WordNet, extract hypernymns, hyponyms, meronyms, AND holonyms as features
   * Some additional features that you can think of, which may make your representation better.

Note: you are free to implement or use a third-party tool. Some useful resources are provided at the end of this document.

1. **Task 3**: Implement a machine-learning, statistical, or heuristic (or a combination) based approach to extract filled information templates from the corpus of natural language statements:
   * Run the above described deep NLP on the corpus.
   * Implement a machine-learning, statistical, or heuristic (or a combination) based approach to extract filled information templates from the corpus of natural language statements
   * Evaluate the results on the dev set using the given evaluation script. The script takes two files as arguments: the gold file containing the gold labels, and the prediction file containing the predicted labels. Note that the gold file is same as the training/dev sets provided, and the prediction file is the one output by your program, which must have the same format as the attached sample file.

*Sample script call: python evaluation.py dev-set.txt predicted-answers.txt*

Note that the test set does not contain any gold tags. For the test set, you need to create a prediction file. During the demo, the TA will test your answers with the gold key.

# Communications with TA

A dedicated slack channel has been created for communication related to the project. You may also send me an email, but slack will allow you to do both: communicate with each other and with me more quickly; and discuss ideas among each other.

Use this link to join the slack channel:

<https://join.slack.com/t/cs6320501proj/shared_invite/enQtNzU4NDAxODU1MDkyLWQ4YjVkMzBkMmFlYTUxNzBiNTY2MTUyMTVmYjFlOTA4MmU1YTQyNjc1MjIwODBhZjJjN2Q1ZTExYWIxMDYzNTU>

# Useful resources

Some resources that you may find useful for this project are listed below:

* [TextBlob](https://textblob.readthedocs.io/en/dev/): Python API for common NLP tasks
* [spaCy](https://github.com/explosion/spaCy): Python API commonly used in the industry
* [NLTK](https://www.nltk.org/): Python API for common NLP tasks
* [PyTorch](https://pytorch.org/docs/stable/index.html): Python library for deep learning
* [TensorFlow](https://www.tensorflow.org/api_docs/python/): Another more common Python library for deep learning
* [Stanford NLP](https://nlp.stanford.edu/software/index.shtml): Java tool for common NLP tasks
* [OpenNLP](https://opennlp.apache.org/): Java tool that provides machine learning libraries for NLP tasks
* [MIT-IE toolkit](https://github.com/mit-nlp/MITIE): C, C++ and Python tools for Information Extraction
* [Charniak Parser](https://github.com/BLLIP/bllip-parser): C++ implementation of the Charniak parser

# Project Point Distribution

1. Max points available: 100 points
2. Division of points:
   1. Group information: 2 points
   2. Project implementation and demo: 90 points
      1. Task 1: TBD points
      2. Task 2: TBD points
      3. Task 3: TBD points
   3. Project Report: 8 points