**CS 6320 – Natural Language Processing**

**Fall 2019**

**Dr. Mithun Balakrishna**

**Course Project**

# Project Steps and Deadlines:

* **Project Group Formation**:
  + Due by **Thursday, 10/17/2019, 11:59pm**
  + 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 **Thursday, 10/17/2019** at **8:15pm** 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: **12/6/2019 and 12/7/2019**
  + 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 <https://github.com/takshakpdesai/CS6320.501> or from the project folder in eLearning.

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\_predictions.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 determine the semantic textual similarity (STS) between two pieces of text and produce at similarity score (integer value between 1 (lowest) and 5(highest)):
   * Run the above described deep NLP on the input corpus (train or dev set).
   * Using the train set, implement/apply a machine-learning, statistical, or heuristic (or a combination) based approach to learn a rules/model that can determine the STS between any two pieces of input text and produce at similarity score (integer value between 1 (lowest) and 5(highest)).
   * On the dev set, evaluate your STS system 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 train/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 dev-set-predicted-answers.txt*

1. **Task 4**: The performance of your NLP and STS system will evaluated on the test set.
   * Run the above described deep NLP on the input test set.
   * Run the test set through your STS system using the given evaluation script. The script takes two files as arguments: the test file containing the input text sentence pairs, and the prediction file containing the predicted labels. Note that 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 test-set.txt test-set-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.

# 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
* [scikit-learn](https://scikit-learn.org/stable/): Python API for ML frameworks
* [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

# Links to Related Papers

Some papers that implement STS systems are cited below:

* Han, Lushan, Abhay L. Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. "UMBC\_EBIQUITY-CORE: Semantic textual similarity systems." In Second Joint Conference on Lexical and Computational Semantics (\* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, pp. 44-52. 2013.
* Croce, Danilo, Valerio Storch, and Roberto Basili. "Unitor-core\_typed: Combining text similarity and semantic filters through SV regression." In Second Joint Conference on Lexical and Computational Semantics (\* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, pp. 59-65. 2013.
* Brychcín, Tomáš, and Lukáš Svoboda. "UWB at SemEval-2016 Task 1: Semantic textual similarity using lexical, syntactic, and semantic information." In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 588-594. 2016.

# 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: 5 points
      2. Task 2: 30 points
      3. Task 3: 35 points
      4. Task 4: 20 points
   3. Project Report: 8 points