EDISON: Feature Extraction for NLP
Mark Sammmons, Christos Christodoulopoulos, Parisa Kordjamshidi, Daniel Khoshabi, Vivek Srikumar, Paul Vijayakumar, Mazin Bokhari, Xinbo Wu, and Dan Roth
1Cognitive Computation Group, University of Illinois
2School of Computing, University of Utah

Use Feature Extractors Programmatically
LBJava and Saul
Edison’s feature extractor classes can be directly incorporated in Java applications using the CCG data structures, and in programmatic learning environments. LBJava is a self-contained extension to Java that supports machine learning. It provides a specification for languages that allows users to rapidly develop prototype machine learning algorithms.
Saul is a new Learning-Based Programming framework in Scala that supports rapid development of machine learning applications that use Structured Prediction methodologies. It generalizes the capabilities of LBJava.

LBJava: https://github.com/IllinoisCogComp/LBJava
Saul: https://github.com/IllinoisCogComp/saul

Example Task: Named Entity Recognition
Named Entity Recognition (NER) is a basic NLP task that identifies proper nouns and their types in English text. It is a useful component for other applications. Features represent information used to support decisions about what to label target elements of the text – in this case, targets are words. Some features that we can use to predict the NER label for a word include the NER labels predicted for previous words, and part-of-speech labels of neighboring words. Figure 3 illustrates these features for a given sentence. The information in this example is called a feature extractor.

For more complex tasks, features may themselves represent structures that span multiple constituents in multiple languages (to use Illinois-core-utilities terminologies), or which use knowledge resources such as gazetteers.

Use Feature File Outputs
SVMLight format
Edison provides support for the use of popular machine learning packages such as Weka, Mallet, etc. by providing classes that transform feature extractor outputs in the SVMLight data format. Edison generates a lexicon mapping feature types to integer values, and SVMLight input files that use these integer values.

#svmlight format

#edison lexicon
1 B-POS
2 I-POS
3 B-NN
4 I-NN
5 B-PER
6 I-PER
7 B-ORG
8 I-ORG

Figure 3: Using Edison feature extractors in Saul

What is Feature Extraction?
Natural Language Processing systems such as Part of Speech taggers, Shallow Parsers, and Named Entity Recognizers are typically built around Machine Learning algorithms ("learners"). These algorithms build statistical models that take some representation of text as input, and predicts labels for elements of that text ("examples") as output. To work well, this representation must express useful abstractions over the text. The process of mapping from raw text to learner inputs is known as Feature Extraction.

Feature Extraction Challenges
- Implementation is time-consuming.
- Replicating other people’s research is hard – their published descriptions of feature extractors may lack important details.
- There is much diversity of effort: many different versions of the same tools; and across different NLP applications, many similar families of feature extractors.
- Even given existing feature extraction code, it may be hard to find the extractor you need.

Project Goals
- Speed up development by centralizing feature extraction and making it easy for developers to find existing implementations for feature extractors they need.
- Clarify individual project code and reduce maintenance overhead by using the same reference implementation of features where appropriate.
- Share reference implementation of feature extraction for specific applications/publications to support duplication of results by other researchers.
- Initiate an open source project that can be improved and used by other researchers.

Use Feature Extraction Library
Using the Edison Feature Extraction Library
Figure 1: Illinois-core-utilities data structures

Edison Feature Extractor Search Interface
Edison users need to know what feature extractors are available, and what they do. Every feature extractor is described according to what Views they use and what features they generate. Each is documented with a clear description of its behavior, and each has a unit test that illustrates its use and specifies a representative output for that extractor.

The search interface allows users to search for NLP terms, keywords, and View names and retrieve matching Feature Extractors, displaying the unit test code for each selected extractor.

Figure 2: Screenshot of feature extractor search interface

Find the source code at:
https://github.com/IllinoisCogComp/Edison