Clairlib: A Toolkit for Natural Language Processing, Information Retrieval, and Network Analysis

Amjad Abu-Jbara
EECS Department
University of Michigan
Ann Arbor, MI, USA
amjbara@umich.edu

Dragomir Radev
EECS Department and
School of Information
University of Michigan
Ann Arbor, MI, USA
radev@umich.edu

Abstract

In this paper we present Clairlib, an open-source toolkit for Natural Language Processing, Information Retrieval, and Network Analysis. Clairlib provides an integrated framework intended to simplify a number of generic tasks within and across those three areas. It has a command-line interface, a graphical interface, and a documented API. Clairlib is compatible with all the common platforms and operating systems. In addition to its own functionality, it provides interfaces to external software and corpora. Clairlib comes with a comprehensive documentation and a rich set of tutorials and visual demos.

1 Introduction

The development of software packages and code libraries that implement algorithms and perform tasks in scientific areas is of great advantage for both researchers and educators. The availability of these tools saves the researchers a lot of the time and the effort needed to implement the new approaches they propose and conduct experiments to verify their hypotheses. Educators also find these tools useful in class demonstrations and for setting up practical programming assignments and projects for their students.

A large number of systems have been developed over the years to solve problems and perform tasks in Natural Language Processing, Information Retrieval, or Network Analysis. Many of these systems perform specific tasks such as parsing, Graph Partitioning, co-reference resolution, web crawling etc. Some other systems are frameworks for performing generic tasks in one area of focus such as NLTK (Bird and Loper, 2004) and GATE (Cunningham et al., 2002) for Natural Language Processing; Pajek (Batagelj and Mrvar, 2003) and GUESS (Adar, 2006) for Network Analysis and Visualization; and Lemur[1] for Language Modeling and Information Retrieval.

This paper presents Clairlib, an open-source toolkit that contains a suit of modules for generic tasks in Natural Language Processing (NLP), Information Retrieval (IR), and Network Analysis (NA). While many systems have been developed to address tasks or subtasks in one of these areas as we have just mentioned, Clairlib provides one integrated environment that addresses tasks in the three areas. This makes it useful for a wide range of applications within and across the three domains.

Clairlib is designed to meet the needs of researchers and educators with varying purposes and backgrounds. For this purpose, Clairlib provides three different interfaces to its functionality: a graphical interface, a command-line interface, and an application programming interface (API).

Clairlib is developed and maintained by the Computational Linguistics and Information Retrieval (CLAIR) group at the University of Michigan. The first version of Clairlib was released in the year 2007. It has been heavily developed since then until it witnessed a qualitative leap by adding the Graphical Interface and many new features to the latest version that we are presenting here.

Clairlib core modules are written in Perl. The GUI was written in Java. The Perl back-end and the Java front-end are efficiently tied together through a communication module. Clairlib is compatible with

[1] http://www.lemurproject.org/

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 121–126,
Portland, Oregon, USA, 21 June 2011.

all the common platforms and operating systems. The only requirements are a Perl interpreter and Java Runtime Environment (JRE).

Clairlib has been used in several research projects to implement systems and conduct experiments. It also has been used in several academic courses.

The rest of this paper is organized as follows. In Section 2, we describe the structure of Clairlib. In Section 3, we present its functionality. Section 4 presents some usage examples. We conclude in Section 5.

2 System Overview

Clairlib consists of three main components: the core library, the command-line interface, and the graphical user interface. The three components were designed and connected together in a manner that aims to achieve simplicity, integration, and ease of use. In the following subsections, we briefly describe each of the three components.

2.1 Modules

The core of Clairlib is a collection of more than 100 modules organized in a shallow hierarchy, each of which performs a specific task or implements a certain algorithm. A set of core modules define the data structures and perform the basic processing tasks. For example, Clair::Document defines a data structure for holding textual data in various formats, and performs the basic text processing tasks such as tokenization, stemming, tag stripping, etc.

Another set of modules perform more specific tasks in the three areas of focus (NLP, IR, and NA). For example, Clair::Bio::GIN::Interaction is devoted to protein-protein interaction extraction from biomedical text.

A third set contains modules that interface Clairlib to external tools. For example, Clair::Utils::Parse provides an interface to Charniak parser (Charniak, 2000), Stanford parser (Klein and Manning, 2003), and Chunklink[2].

Each module has a well-defined API. The API is oriented to developers to help them write applications and build systems on top of Clairlib modules; and to researchers to help them write applications and setup custom experiments for their research.

[2]http://ilk.uvt.nl/team/sabine/chunklink/README.html

2.2 Command-line Interface

The command-line interface provides an easy access to many of the tasks that Clairlib modules implement. It provides more than 50 different commands. Each command is documented and demonstrated in one or more tutorials. The function of each command can be customized by passing arguments with the command. For example, the command

partition.pl -graph graph.net -method GirvanNewman -n 4

uses the GrivanNewman algorithm to divide a given graph into 4 partitions.

2.3 Graphical User Interface

The graphical user interface (GUI) is an important feature that has been recently added to Clairlib and constituted a quantum leap in its development. The main purpose of the GUI is to make the rich set of Clairlib functionalities easier to access by a larger number of users from various levels and backgrounds especially students and users with limited or no programming experience.

It is also intended to help students do their assignments, projects, and research experiments in an interactive environment. We believe that visual tools facilitate understanding and make learning a more enjoyable experience for many students. Focusing on this purpose, the GUI is tuned for simplicity and ease of use more than high computational efficiency. Therefore, while it is suitable for small and medium scale projects, it is not guaranteed to work efficiently for large projects that involve large datasets and require heavy processing. The command-line interface is a better choice for large projects.

The GUI consists of three components: the Network Editor/Visualizer/Analyzer, the Text Processor, and the Corpus Processor. The Network component allows the user to 1) build a new network using a set of drawing and editing tools, 2) open existing networks stored in files in several different formats, 3) visualize a network and interact with it, 4) compute different statistics for a network such as diameter, clustering coefficient, degree distribution, etc., and 5) perform several operations on a network such as random walk, label propagation, partitioning, etc. This component uses the open source library, JUNG[3] to visualize networks. Figure 1 shows

[3]http://jung.sourceforge.net/

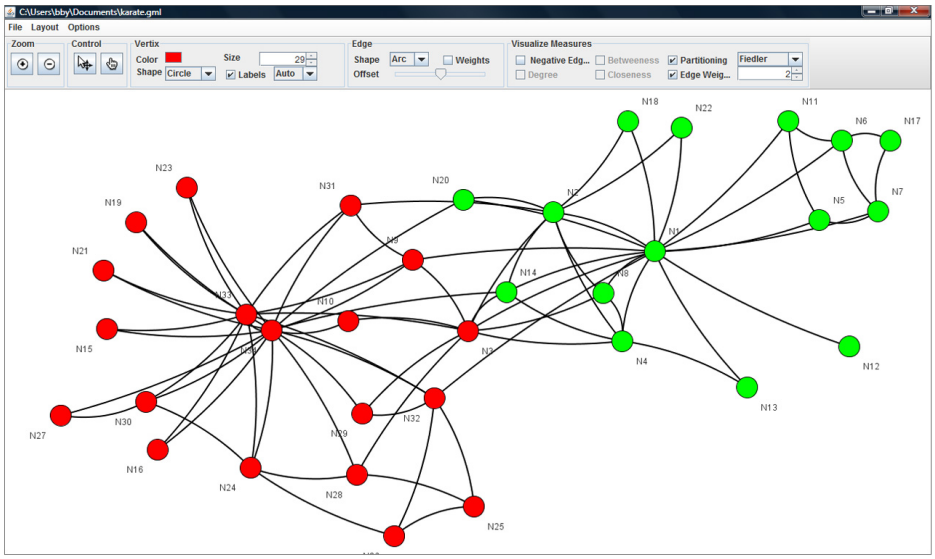

Figure 1: A screenshot for the network visualization component of Clairlib

a screenshot for the Network Visualizer.

The Text Processing component allows users to process textual data published on the internet or imported from a file stored on the disk. It can process data in plain, html, or PDF format. Most of the text processing capabilities implemented in Clairlib core library are available through this component. Figure 2 shows a screenshot of the text processing component.

The Corpus Processing component allows users to build a corpus of textual data out of a collection of files in plain, HTML, or PDF format; or by crawling a website. Several tasks could be performed on a corpus such as indexing, querying, summarization, information extraction, hyperlink network construction, etc.

Although these components can be run independently, they are very integrated and designed to easily interact with each other. For example, a user can crawl a website using the Corpus component, then switch to the Text Processing component to extract the text from the web documents and stem all the words, then switch back to the Corpus component to build a document similarity graph. The graph can then be taken to the Network component to be visualized and analyzed.

2.4 Documentation

Clairlib comes with an extensive documentation. The documentation contains the installation information for different platforms, a description of all Clairlib components and modules, and a lot of usage examples. In addition to this documentation, Clairlib provides three other resources:

API Reference

The API Reference provides a complete description of each module in the library. It describes each subroutine, the task it performs, the arguments it takes, the value it returns, etc. This reference is useful for developers who want to use Clairlib modules in their own applications and systems. The API Reference is published on the internet.

Tutorials

Tutorials teach users how to use Clairlib by examples. Each tutorial addresses a specific task and provides a set of instructions to complete the task using Clairlib command-line tools or its API.

Visual Demos

Visual demos target the users of the graphical interface. The demos visually show how to start the GUI and how to use its components to perform several tasks.

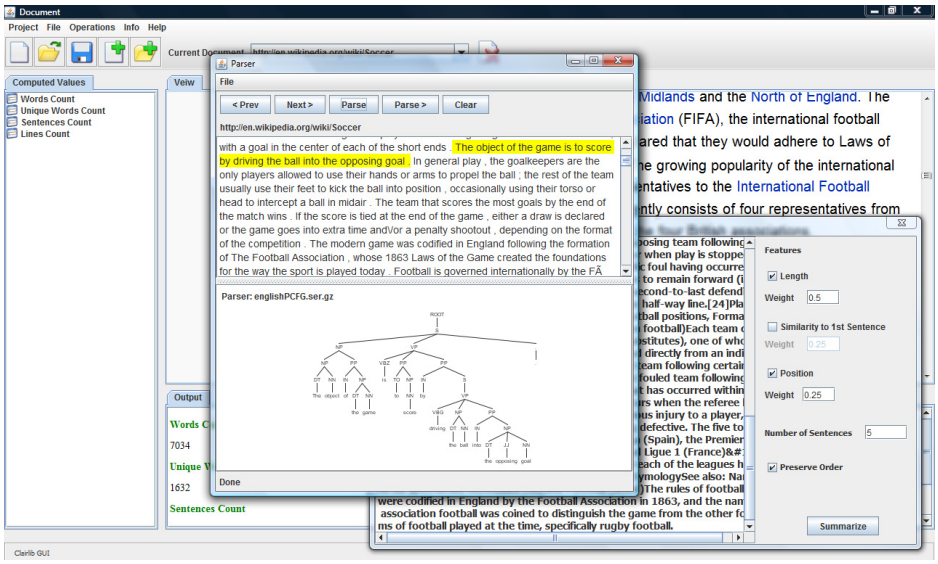

Figure 2: A screenshot for the text processing component of Clairlib

3 Functionality

Clairlib provides modules and tools for a broad spectrum of tasks. Most of the functionalities are native to Clairlib. Some functionalities, however, are imported from other open-source packages or external software. This section lists the main functionalities categorized by their areas.

3.1 Natural Language Processing

NLP functionalities include Tokenization, Sentence Segmentation, Stemming, HTML Tags Stripping, Syntactic Parsing, Dependency Parsing, Part-of-Speech Tagging, Document Classification, LexRank, Summarization, Synthetic Corpus Generation, N-grams Extraction, XML Parsing, XML Tree Building, Text Similarity, Political Text Analysis, and Protein Name Tagging.

3.2 Information Retrieval

IR functionalities include Web Crawling, Indexing, TF-IDF, PageRank, Phrase Based Retrieval, Fuzzy OR Queries, Latent Semantic Indexing, Web Search, Automatic Link Extraction, and Protein-Protein Interaction Extraction.

3.3 Network Analysis

Network Analysis functionalities include Network Statistics, Random Network Generation, Network Visualization, Network Partitioning, Community Finding, Random Walks, Flow Networks, Signed Networks, and Semi-supervised Graph-based Classification. Network Statistics include Centralities, Clustering Coefficient, Shortest Paths, Diameter, Triangles, Triplets, etc.

Some of these functionalities are implemented using several approaches. For example, Clairlib have implementations for 5 graph partitioning algorithms. This makes Clairlib a useful tool for conducting experiments for comparative studies.

4 Uses of Clairlib

The diverse set of domains that Clairlib covers and the different types of interfaces it provides make it suitable for use in many contexts. In this section, we highlight some of its uses.

Education

Clairlib contains visual tools that instructors can use to do class demonstrations to help their students understand the basic concepts and the algorithms they face during their study. For example, the random walk simulator can be used to teach the students how random walk works by showing a sample network and then walk randomly step-by-step through it and show the students how the probabilities change after each step.

It can also be used to create assignments of varying levels of difficulty and different scopes. Instruc-

tors may ask their students to do experiments with a dataset using Clairlib, write applications that use the API, extend an existing module, or contribute new modules to Clairlib. One example could be to ask the students to a build a simple information retrieval system that indexes a collection of documents and executes search queries on it.

Clairlib has been used to create assignments and projects in NLP and IR classes at the University of Michigan and Columbia University. The experience was positive for both the instructors and the students. The instructors were able to design assignments that cover several aspects of the course and can be done in a reasonable amount of time. The students used the API to accomplish their assignments and projects. This helped them focus on the important concepts rather than diving into fine programming details.

Research

Clairlib contains implementations for many algorithms and approaches that solve common problems. It also comes with a number of corpora and annotated datasets. This makes it a good resource for researchers to build systems and conduct experiments.

Clairlib was successfully used in several research projects. Examples include Political Text Analysis (Hassan et al., 2008), Scientific Paper Summarization (Qazvinian and Radev, 2009), Blog Networks Analysis (Hassan et al., 2009), Protein Interaction Extraction (Ozgur and Radev, 2009), and Citation-Based Summarization (Abu-Jbara and Radev, 2011).

4.1 Examples

In this subsection, we present some examples where Clairlib has been used.

Example: Protein-Protein Interaction Extraction

This is an example of a project that builds an information extraction system and uses Clairlib as its main processing component (Ozgur and Radev, 2009). This system is now part of a larger bioinformatics project, NCIBI.

The system uses Clairlib to process a biomedical article: 1) splits it into sentences using the segmentation module, 2) parses each sentence using the interface to the Stanford Dependency Parser, 3) tags the protein names, 4) extracts protein-protein interactions using a specific Clairlib module devoted to this task, and then 5) it builds a protein interaction network in which nodes are proteins and edges represent interaction relations. Figure 3 shows an example protein interaction network extracted from the abstracts of a collection of biomedical articles from PubMed. This network is then analyzed to compute node centralities and the basic network statistics.

Example: Scientific Paper Summarization Using Citation Networks

This is an example of a research work that used Clairlib to implement an approach and conduct experiments to support the research hypothesis. Qazvinian and Radev (2009) used Clairlib to implement their method for citation-based summarization. Given a set of sentences that cite a paper, they use Clairlib to 1) construct a cosine similarity network out of these sentences, 2) find communities of similar sentences using Clairlib community finding module, 3) run Clairlib LexRank module to rank the sentences, 4) extract the sentence with the highest rank from each community, and finally 5) return the set of extracted sentences as a summary paragraph.

Example: Text Classification

This is an example of a teaching assignment that was used in an introductory course on information retrieval at the University of Michigan. Students were given the 20-newsgroups corpus (a large set of news articles labeled by their topic and split into training and testing sets) and were asked to use Clairlib API to: 1) stem the text of the documents, 2) convert each document into a feature vector based on word frequencies, 2) train a multi-class Perceptron or Naive Bayes classifier on the documents in the training set, and finally 3) classify the documents in the testing set using the trained classifier.

5 Conclusions

Clairlib is a broad-coverage toolkit for Natural Language Processing, Information Retrieval, and Network Analysis. It provides a simple, integrated, interactive, and extensible framework for education and research uses. It provides an API, a command-

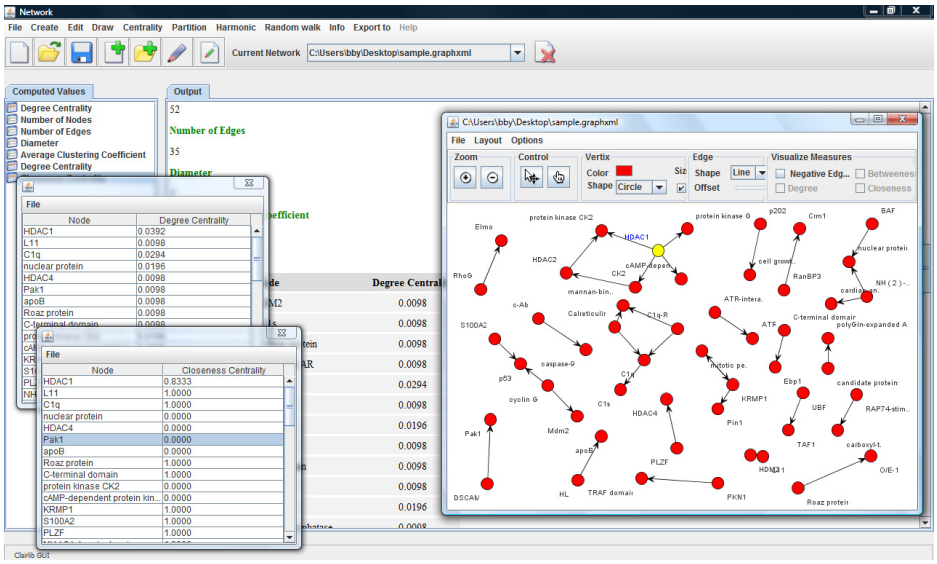

Figure 3: Clairlib used to construct and analyze a protein network extracted from biomedical articles

line interface, and graphical user interface for the convenience of users with varying purposes and backgrounds. Clairlib is well-documented, easy to learn, and simple to use. It has been tested for various types of tasks in various environments.

Clairlib is an open source project and we welcome all the contributions. Readers who are interested in contributing to Clairlib are encouraged to contact the authors.

Acknowledgements

We would like to thank Mark Hodges, Anthony Fader, Mark Joseph, Joshua Gerrish, Mark Schaller, Jonathan dePeri, Bryan Gibson, Chen Huang, Arzucan Ozgur, and Prem Ganeshkumar who contributed to the development of Clairlib.

This work was supported in part by grants R01-LM008106 and U54-DA021519 from the US National Institutes of Health, U54 DA021519, IDM 0329043, DHB 0527513, 0534323, and 0527513 from the National Science Foundation, and W911NF-09-C-0141 from IARPA.

References

R. Gaizauskas, P. J. Rodgers and K. Humphreys 2001. Visual Tools for Natural Language Processing. *Journal of Visual Languages and Computing, Volume 12, Issue 4, Pages 375-412.*

Arzucan Ozgor and Dragomir Radev 2009. Supervised classification for extracting biomedical events. *Proceedings of the BioNLP'09 Workshop Shared Task on Event Extraction at NAACL-HLT, Boulder, Colorado, USA, pages 111-114*

Ahmed Hassan, Dragomir R. Radev, Junghoo Cho, Amruta Joshi. 2009. Content Based Recommendation and Summarization in the Blogosphere. *ICWSM-2009.*

Vahed Qazvinian, Dragomir Radev. 2008. Scientific Paper Summarization Using Citation Summary Networks. *COLING 2008.*

Ahmed Hassan, Anthony Fader, Michael Crespin, Kevin Quinn, Burt Monroe, Michael Colaresi and Dragomir Radev. 2008. Tracking the Dynamic Evolution of Participants Salience in a Discussion. *COLING 2008.*

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. *Proceedings of NAACL-2000.*

Dan Klein and Christopher Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of ACL-2003.*

Amjad Abu-Jbara and Dragomir Radev 2011. Coherent Citation-based Summarization of Scientific Papers *Proceedings of ACL-2011.*

H. Cunningham and D. Maynard and K. Bontcheva and V. Tablan 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications *Proceedings of ACL-2002, Philadelphia.*

Steven Bird and Edward Loper. 2004. NLTK: The Natural Language Toolkit *Proceedings of ACL-2004.*

V. Batagelj and A. Mrvar 2003. Pajek - Analysis and Visualization of Large Networks *Springer, Berlin.*

Eytan Adar. 2006. GUESS: A Language and Interface for Graph Exploration *CHI 2006.*