IJCNLP2013

THE 6TH INTERNATIONAL JOINT CONFERENCE ON
NATURAL LANGUAGE PROCESSING

OCTOBER 14–18, 2013

NAGOYA CONGRESS CENTER, NAGOYA, JAPAN

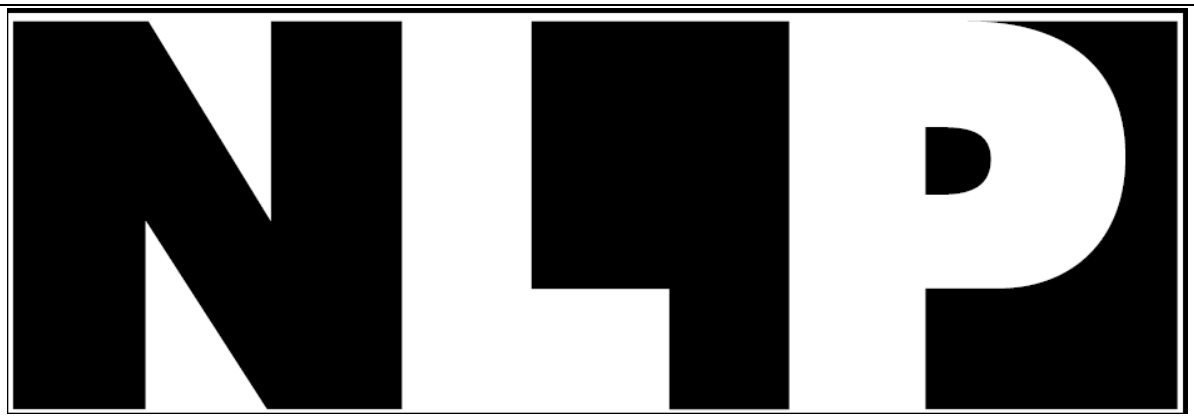
PROCEEDINGS

Sixth International Joint Conference on
Natural Language Processing

The Companion Volume of the Proceedings: System Demonstrations

We wish to thank our sponsors and supporters!

Platinum Sponsors



www.anlp.jp

Silver Sponsors

www.google.com

Bronze Sponsors

www.rakuten.com

Supporters

NAGOYA CONVENTION & VISITORS BUREAU

Nagoya Convention & Visitors Bureau

We wish to thank our organizers!

Organizers

Asian Federation of Natural Language Processing (AFNLP)

Toyohashi University of Technology

ISBN 978-4-9907348-1-7

Preface

Welcome to the companion volume of the proceedings of IJCNLP 2013. This companion volume contains the papers of the system demonstrations presented at the 6th International Joint Conference on Natural Language Processing, held in Nagoya, Japan, on October 14-18, 2013.

The system demonstrations program offers presentations of early research prototypes as well as interesting mature systems. The system demonstration co-chairs and the members of the program committee received 16 submissions, 12 of which were selected for inclusion in the program after reviews by three members of the program committee.

We would like to thank the members of the program committee for their excellent job in reviewing the submissions and providing their support for the final decision.

Demonstration Co-chairs
Kentaro Torisawa (NICT, Japan)
Hang Li (Huawei Technologies, China)

October 14-18, 2013
Nagoya, Japan

Co-chairs:

Kentaro Torisawa, NICT, Japan
Hang Li, Huawei Technologies, China

Program Committee:

Eiji Aramaki, University of Tokyo, Japan
Timothy Baldwin, The University of Melbourne, Australia
Francis Bond, Nanyang Technological University, Singapore
Qingcai Chen, Harbin Institute of Technology Shenzhen Graduate School, China
Wenliang Chen, I2R, Singapore
Atsushi Fujita, Future University Hakodate, Japan
Guoping Hu, iFlyTEK Research, China
Yunhua Hu, Taobao, China
Jung-Jae Kim, Nanyang Technological University, Singapore
Jungi Kim, Technische Universität Darmstadt, Germany
Mamoru Komachi, Tokyo Metropolitan University, Japan
Cane Wing-Ki Leung, Huawei Noah's Ark Lab, Hong Kong
Diana McCarthy, University of Cambridge (DTAL), United Kingdom
Manabu Sassano, Yahoo Japan Corporation, Japan
Young-In Song, NHN Corporation, Korea
Yoshimasa Tsuruoka, University of Tokyo, Japan
Naoki Yoshinaga, University of Tokyo, Japan
Yibo Zhang, Huawei Technologies, China

Table of Contents

Program of System Demonstrations

Wednesday October 16, 2013 (15:30 - 17:00)

A Web-based Annotation Framework For Large-Scale Text Correction
Ossama Obeid, Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Kemal Oflazer and Nadi Tomeh

An English Reading Tool as a NLP Showcase
Mahmoud Azab, Ahmed Salama, Kemal Oflazer, Hideki Shima, Jun Araki and Teruko Mitamura

Dashboard: A Tool for Integration, Validation, and Visualization of Distributed NLP Systems on Heterogeneous Platforms
Pawan Kumar, Rashid Ahmad, Banshi Chaudhary and Mukul Sinha

DIRA: Dialectal Arabic Information Retrieval Assistant
Arfath Pasha, Mohammad Al-Badrashiny, Mohamed Altantawy, Nizar Habash, Manoj Pooleery, Owen Rambow, Ryan M. Roth and Mona Diab

Keyphrase-Driven Document Visualization Tool
Gabor Berend and Richárd Farkas

Making Headlines in Hindi: Automatic English to Hindi News Headline Translation
Aditya Joshi, Kashyap Popat, Shubham Gautam and Pushpak Bhattacharyya

MaltDiver: A Transition-Based Parser Visualizer
Miguel Ballesteros and Roberto Carlini

NICT Disaster Information Analysis System
Kiyonori Ohtake, Jun Goto, Stijn De Saeger, Kentaro Torisawa, Junta Mizuno and Kentaro Inui

SINNET: Social Interaction Network Extractor from Text
Apoorv Agarwal, Anup Kotalwar, Jiehan Zheng and Owen Rambow

SmartNews: Towards content-sensitive ranking of comments
Marina Litvak and Leon Matz

Tmuse: Lexical Network Exploration
Yannick Chudy, Yann Desalle, Benoit Gaillard, Bruno Gaume, Pierre Magistry and Emmanuel Navarro

WISDOM2013: A Large-scale Web Information Analysis System
Masahiro Tanaka, Stijn De Saeger, Kiyonori Ohtake, Chikara Hashimoto, Makoto Hijiya, Hideaki Fujii and Kentaro Torisawa

A Web-based Annotation Framework for Large-scale Text Correction

Ossama Obeid[1] **Wajdi Zaghouani**[1] **Behrang Mohit**[1]
Nizar Habash[2] **Kemal Oflazer**[1] **Nadi Tomeh**[2]

[1]Carnegie Mellon University in Qatar
{oobeid@,wajdiz@,behrang@,ko@cs.}cmu.edu

[2]Center for Computational Learning Systems, Columbia University
{habash,nadi}@ccls.columbia.edu

Abstract

We demonstrate a web-based, language-independent annotation framework used for manual correction of a large Arabic corpus. Our framework provides intuitive interfaces for annotating text and managing the annotation process. We describe the details of both the annotation and the administration interfaces as well as the back-end engine. We also show how this framework is able to speed up the annotation process by employing automated annotators to fix basic Arabic spelling errors.

1 Introduction

Errors in natural language text, such as incorrect spelling, word choice, or grammar, are problematic for natural language processing (NLP) systems: they contribute to data sparseness and limit the effectiveness of NLP models. Automatic correction of these errors have been studied for different languages (Kukich, 1992). QALB (Qatar Arabic Language Bank)[1] is a project on automatic correction of errors in Arabic text. Our approach has two components: (a) large scale manual annotation (correction) of Arabic errors, and (b) statistical modeling of the text correction.

In this paper we focus on the first task and describe the design and implementation of our language-independent, web-based annotation system. Our framework provides intuitive interfaces for both managing the annotation process and performing the annotations. Additionally, we show how our framework employs automatic annotators to correct basic Arabic spelling mistakes to speed up the annotation process.

Our framework consists of two major interfaces: (a) an Admin interface, which enables the lead annotator to create, assign, monitor, evaluate and export annotation tasks in large scale; and (b) an Annotation interface, which enables annotators to conduct and review annotation tasks for different types of text. The interface is flexible for handling monolingual annotation (e.g. Arabic text) and also bilingual annotation (e.g. post-editing of MT output). The interface provides the required annotation functionality for moving, merging, replacing and editing words within a paragraph along with the undo and redo actions.

In addition to the Admin tools, the framework provides the lead annotator with components for automatic correction of basic errors and also quality control. The annotation framework is currently deployed and is expected to be used by up to twenty users, annotating an aggregated corpus of two million words.

2 Related Work

Traditionally, manual text correction is performed under the context of post-editing machine translation (MT) output. The goal of post-editing is to evaluate MT systems rather than building corpora of edits.

Tools like PET (Aziz et al., 2012) and BLAST (Stymne, 2011) provide annotators a text-editor-like interface to identify, record, and correct errors. Text-editor-like interfaces are very flexible and allow all forms of corrections to be performed, but, they are not capable of accurately tracking token alignment, if at all.

Frameworks such as EXMARaLDA (Schmidt, 2010) and GATE (Cunningham et al., 2011) facilitate multi-layer and multi-round annotations. An example of such approach is the work of Dickinson and Ledbetter (2012) who annotated errors in Hungarian students essays using multiple annotation layers from phonology to syntax in different stages.

TCTool (Llitjós and Carbonell, 2004) provides a token-based correction interface. It allows for tokens to be moved around, deleted, or added, but, does not allow for tokens to be merged or split.

[1]http://nlp.qatar.cmu.edu/qalb

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 1–4,
Nagoya, Japan, 14-18 October 2013.

This is because it assumes all input text to be outputs of MT systems, which do not produce merged or split words. Since our source text contains a majority of manually written text, this assumption does not hold. Furthermore, TCTool is designed to deal with short sentences while we aim to annotate larger documents in order to benefit from wider context.

Above all, these tools are not designed for large-scale, distributed annotation projects. They do not provide facilities for managing thousands of documents, distributing tasks to tens of annotators and evaluating inter-annotator agreement (IAA). Our system draws on the advantages of the above tools while adding the required facilities to manage a large annotation project. In this aspect, our system is similar to the COLABA project annotation tool, a web application for dialectal Arabic text annotation (Benajiba and Diab, 2010; Diab et al., 2010).

3 System Requirements and Constraints

The QALB project aims to produce a large corpus (two million words) of manually corrected Arabic text. Our system must allow for quick annotation of texts without sacrificing annotation correctness and consistency. This section describes the requirements and constraints that our annotation system needs to fulfill.

Text Correction The QALB corpus will contain corrections of errors produced by native speakers of various dialects, non-native speakers and machine translation systems in a variety of contexts including news, Wikipedia articles, forum posts, and student essays. Therefore, our annotation system should not just account for spelling mistakes. Our annotation interface allows annotators to perform different types of *actions* which correspond to the following types of corrections: (a) *Edit* actions: words that are misspelled or mistyped should be modified. (b) *Move* actions: Words that are not in the right location should be moved to the right location. (c) *Add* actions: words that are missing need to be added. (d) *Delete* actions: extraneous words should be deleted. (e) *Merge* actions: words that have been split by mistake should be merged. (f) *Split* actions: words that have been merged by mistake should be split.

Token Alignment Since we allow a large range of corrections, we need to be able to track the alignment of corrected tokens to the original text. In addition to token alignment, we want to track the list of actions performed by each annotator in the hope that we may learn from the human correction process.

Efficiency Due to the large amount of text that needs to be corrected, our Annotation interface should allow annotators to concurrently log into the system and perform their annotation tasks very quickly.

Quality Control To ensure the quality of corrections in our corpus we should be able to monitor the performance of each individual annotator and the consistency of annotators among each other. Therefore, our system should allow us to perform inter-annotator agreement (IAA) evaluation. We also need to ensure that all source documents are of reasonable quality. For this we need a mechanism for annotators to flag low quality (e.g. highly dialectal) text.

4 Annotation Web Interface

In this section we present our framework, the Annotation Web Interface, which will be used to carry out the annotation process. Our main contribution is the Annotation interface (Figure 1) which provides an intuitive drag-and-drop interface to manipulate tokens in a document. We describe the design and implementation of each component in further details.

4.1 Architecture

Our framework has three core components: the Annotation interface, the Admin interface, and the application programming interface (API) server.

The Annotation interface is used by annotators to correct assigned documents. The Admin interface is used by the lead annotator to manage annotators and documents, assign tasks, evaluate IAA, and monitor the overall progress of the annotation process. Figure 2 illustrates how these components interact with each other.

Both Admin and Annotation interfaces are web pages that complete their respective tasks by sending HTTP requests to the API server. The API server handles these requests and performs the necessary operations using the local file system and a database.

In addition to the three core components, there are automated annotators. Automated annotators are scripts that interface with the API server to perform automatic corrections. We discuss how we use an automated annotator to speed up the annotation process.

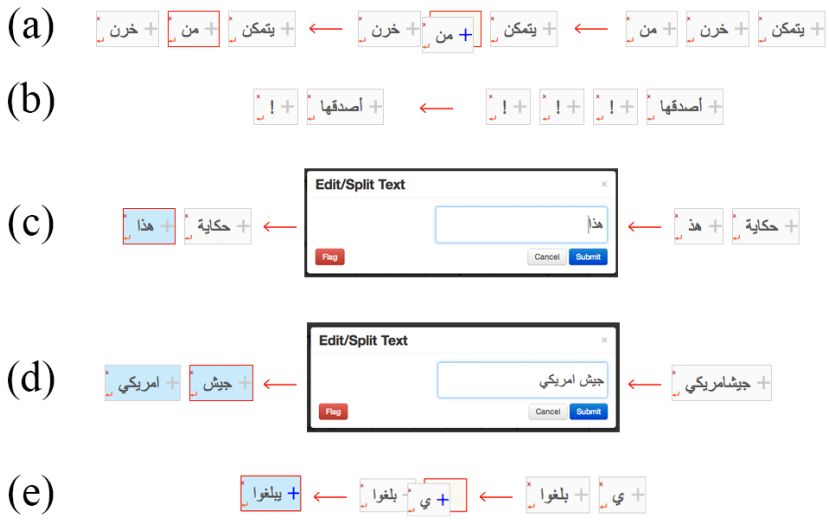


Figure 1: Sample of corrections of a token: (a) Moving (b) Deleting (c) Editing (d) Splitting (e) Merging

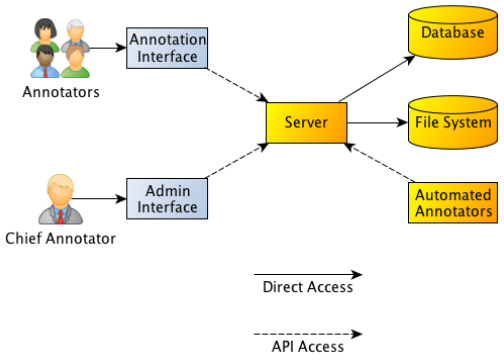


Figure 2: General Architecture Diagram.

Figure 3: A single token box.

4.2 Annotation Interface

Annotators need to be able to view their assigned tasks, perform corrections, and submit their final corrections. This is done through the Annotation Interface. The Annotation Interface first displays a list of tasks assigned to an annotator. The annotator selects a task and then displays the Annotation Window, where corrections are performed.

The Annotation Window displays tokens in separate boxes. Each box can either be dragged or double-clicked. Tokens can be moved by dragging and dropping tokens to the desired location. Tokens can be merged by dragging a token slightly to the left or to the right to be merged with the previous or the next token, respectively. Double clicking on a token opens a dialog box with a text input which contains the current value of the clicked token and can be used to modify the token's text. Adding a space between two characters of a token performs a split. Annotators cannot modify a token and split it at the same time. This allows us to track individual changes so that we have a consistent action history.

As illustrated in Figure 3, each box has additional buttons that can be used to either delete a token, add a new token before the selected token, or add a new line after the selected token. Figure 1 shows screen-shots of each action performed in sequence.

The Annotation Interface also has few other features to help with the annotation task. *Undo* and *Redo* buttons are provided to allow annotators to go back and fix mistakes. The interface also provides access to both the original Arabic text, and, if the text was the output of a machine translation system, the original English text. This additional information help annotators in determining how much their corrections alter the meaning of the original text. If a document has poor quality of writing or translation, the interface provides the annotator with a *Flag* button, which alerts the lead annotator about the issue.

4.3 Admin Interface

The lead annotator will be managing a team of about twenty annotators who can use the system remotely and concurrently. The Admin Interface contains: (a) a user management tool for creating new annotator accounts and viewing annotator progress; (b) a document management tool for uploading new documents, assigning them for annotation, and viewing submitted annotations; and (c) a monitoring tool for viewing overall annotation progress and evaluating IAA.

4.4 API Server and Automated Annotators

The API server lies at the heart of our framework. It is a Python server that provides a web API through HTTP requests for retrieving, creating, and modifying content such as user records, source documents, and annotation submissions. All responses by the API server are JSON objects. This allows us to easily create dynamic web pages for the Annotation and Admin interfaces as well as automated annotators. One automated annotator we deployed to automatically correct ة (*Ta Marbuta*) versus ه (*Ha*) errors and ء (*Hamza*) placement errors by running each document through the MADA system (Habash et al., 2009). All documents are corrected by our automated annotator before being assigned to annotators to cut down annotation time.

5 Demonstration Script

During the demonstration, we will present the use of the Admin and Annotator interfaces using simple and complex examples of various kinds of edits as discussed above. In particular, we will show how the Annotation tool can be used for correcting a sample piece of text using the various allowed operations.

6 Conclusion and Future Work

We presented a detailed overview of our web-based annotation framework for correcting writing errors. Deployment for error correction in other languages is a natural extension of this work since almost all functionalities of our system are language independent. In the future, we plan to include new functionalities for increasing annotators' search and lookup power and a web-based component for training new annotators. We also plan to make use of the created annotations to develop automatic error detection and correction systems.

7 Acknowledgements

We thank anonymous reviewers for their valuable comments and suggestions. This publication was made possible by grants NPRP-4-1058-1-168 and YSREP-1-018-1-004 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Wilker Aziz, Sheila Castilho Monteiro de Sousa, and Lucia Specia. 2012. PET: a tool for post-editing and assessing machine translation. In *Proceedings of the LREC'2012*.

Yassine Benajiba and Mona Diab. 2010. A web application for dialectal Arabic text annotation. *Proceedings of the LREC Workshop for Language Resources (LRs) and Human Language Technologies (HLT) for Semitic Languages: Status, Updates, and Prospects*.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6)*. University of Sheffield.

Mona Diab, Nizar Habash, Owen Rambow, Mohamed Altantawy, and Yassine Benajiba. 2010. Colaba: Arabic dialect annotation and processing. *LREC Workshop on Semitic Language Processing*, pages 66–74.

Markus Dickinson and Scott Ledbetter. 2012. Annotating errors in a Hungarian learner corpus. In *Proceedings of the LREC'2012*.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439, December.

Ariadna Font Llitjós and Jaime G. Carbonell. 2004. The translation correction tool: English-Spanish user studies. In *Prceedings of the LREC'04*.

Thomas Schmidt. 2010. Linguistic tool development between community practices and technology standards. In *Proceedings of the LREC Workshop Language Resource and Language Technology Standards*.

Sara Stymne. 2011. Blast: a tool for error analysis of machine translation output. In *Proceedings of the ACL'2011: Systems Demonstrations*, pages 56–61.

An English Reading Tool as a NLP Showcase

Mahmoud Azab Ahmed Salama Kemal Oflazer
Carnegie Mellon University-Qatar
Doha, Qatar
{mazab,ahmedsaa,ko}@qatar.cmu.edu

Hideki Shima Jun Araki Teruko Mitamura
Carnegie Mellon University,
Pittsburgh, PA, USA
{hideki,junaraki,teruko}@cs.cmu.edu

Abstract

We introduce -*SmartReader*- an English reading tool for non-native English readers to overcome language related hindrances while reading a text. It makes extensive use of widely-available NLP tools and resources. *SmartReader* is a web-based application that can be accessed from standard browsers running on PCs or tablets. A user can choose a text document from the system's library they want to read or can upload a new document of their own and the system will display an interactive version of such text, that provides the reader with an intelligent e-book functionality.

1 Introduction

Reading texts in a second language presents the language readers with a number of comprehension problems, especially when the reader does not have access to aids that would enable her to get over them including the problem of unknown words interpretation, unrecognized and forgotten names, difficult and hard-to-understand sentences, and lack of or forgetting the prior context in a former session of reading. There are many NLP-based tools, that offer various kinds of aids, to non-native English readers to help them in understanding a document. Many tools focus on assisting the reader in understanding of a specific word which may lead to better comprehension and vocabulary acquisition such as (Nerbonne et al., 1997) and (Eom et al., 2012). Some other tools focus on assisting the reader and second language learner with highlighting different patterns in the documents and providing the learner a visually enhanced version of the document (Meurers et al.,).

SmartReader is an implementation of a NLP-powered tool to aid in reading texts in English by non-native readers of the language which aims to make reading an active and interactive experience. In this paper, we present the underlying client-server architecture of *SmartReader*; for a detailed presentation of the user functionality provided by *SmartReader*, we refer the reader to Azab et al. (2013).

The main contribution of *SmartReader* is the integration of NLP tools and resources under the UIMA framework within a client-server architecture. The resulting web-accessible reading application can run on various browser platforms to help secondary language learners of English overcome language hindrances. Although currently *SmartReader* has currently been developed for English, it is *language independent*; it can easily be adapted to another language provided the relevant annotation tools and resources are available.

2 System Overview

Our system is based on client-server architecture as shown in Figure 1. The server is responsible for annotating plain text with NLP-related annotations and retrieving them based on the user's interactions. The client is a standard web browser running on PCs or touch tablets and interacts with a server running under a Tomcat web server. It passes user queries to the server and presents menu options and responses to the user. All annotations that are needed to respond to user requests (except for summarization), are produced, by preprocessing the text documents through a series of document annotators and storing their outputs in a UIMA Document Library file accessible to the server

SmartReader is based on significant preprocessing and annotation of texts using many publicly and fairly mature available NLP components for English, integrated in a UIMA (Unstructured Information Management Architecture)

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 5–8,
Nagoya, Japan, 14-18 October 2013.

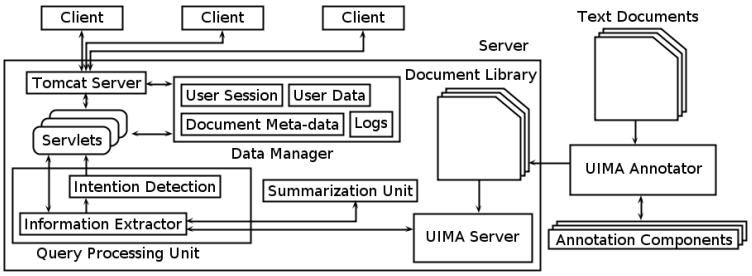


Figure 1: Client-Server Architecture

based server. UIMA is an architecture for the management and analysis of unstructured information such as text and voice, which is based on referential annotation (Ferrucci and Lally, 2004). It facilitates developing and integrating different text analysis engines and annotators. The input can be monotonically enriched while passing from one NLP analysis engine to the next, using a common data repository to all components (Götz and Suhre, 2004). UIMA also supports a flexible combination of individual NLP components into larger processing pipelines. Thus, we can re-use the same annotations (e.g., segmentation, tokenization, POS tagging) for all the next NLP components. It also provides a very powerful querying and search mechanisms for retrieving the annotations from the annotated documents.

The *SmartReader* server has two major functionalities: (1) annotating documents through preprocessing with UIMA annotators, (2) query processing in response to user requests.

Documents Annotation: During annotation, the input plain text is passed to a preprocessor to validate and normalize its orthography. Using Stanford CoreNLP tools, we segment the text into sentences, and then tokenize and perform POS tagging.[1] We then use the following NLP components to annotate the text:

- The **Stanford Dependency Parser** (De Marneffe et al., 2006), provides grammatical relation annotations for each word within the sentence.
- The **Stanford Named Entity Recognizer** (Finkel et al., 2005) and then the **Stanford Co-reference Resolution** (Lee et al., 2013; Lee et al., 2011; Raghunathan et al., 2010) are used to determine the entities in the text and the relationships between them.
- A simple **Word Sense Annotator** based on the Princeton WordNet (Fellbaum, 1998) is used as a broad-coverage machine-readable dictionary of English. As many words in WordNet have more than one sense, we narrow down the available senses by incorporate morphological analysis and part-of-speech filtering then annotate words with the most frequent WordNet sense under the selected part-of-speech. We are currently working on integrating a word sense disambiguation annotator.
- A **Compound Annotator** identifies and looks up the meaning of the phrasal verbs and the compound nouns in the text from WordNet.
- An **In-text Question Answering Annotator** assigns the questions to the related named entities, and ranks them. This is done in two phases. In the first phase, questions are generated using Heilman's question generator tool (Heilman and Smith, 2010). This tool generates a list of questions on every sentence by performing a set of syntactic and semantic transformations. Then, it ranks the generated questions for each sentences according to certain features. In the second phase, we go through the previously annotated named entities and the coreference chains they belong to and assign to every single mention a set of related questions that are generated in the first phase.

Once annotated, a document is loaded into the library. Figure 2, shows the annotation components that each document goes through. Figure 3 shows a logical view of a subset of the annotations for one sentence.

Query Processing: All queries from the user client application are translated into a character offset in the text. Thus when this character offset is passed to UIMA, it returns efficiently all the

[1] `http://nlp.stanford.edu/software/corenlp.shtml`

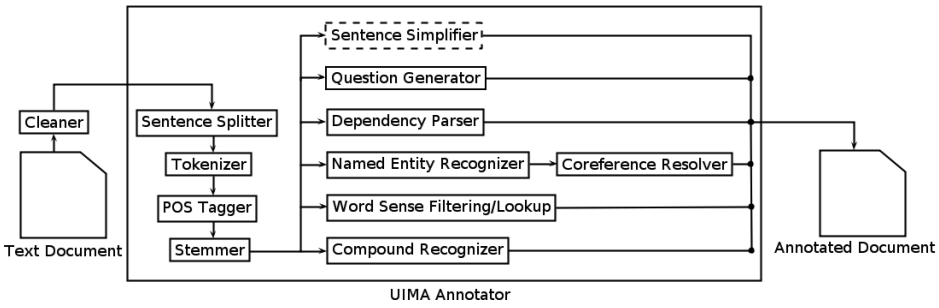


Figure 2: Document Annotation (modules with dotted lines are under development)

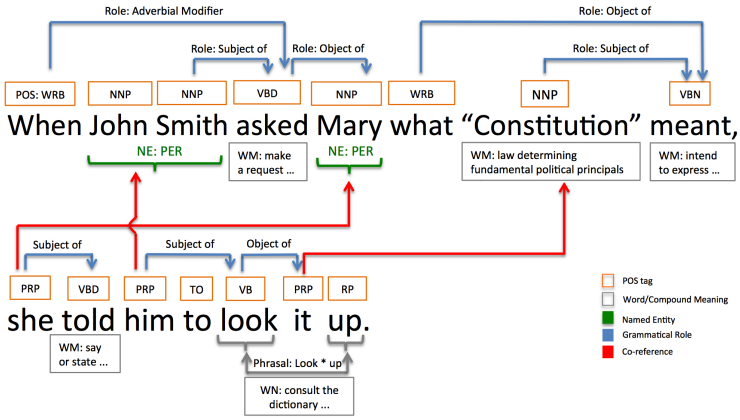


Figure 3: UIMA annotations for one sentence

annotations associated with the word overlapping with that position. These are which are then interpreted by the query processing unit as described earlier.

This step is responsible for processing the user queries and returning the required information. It consists of the following modules:

- **Information Extractor module:** This module efficiently searches a loaded document, then extracts and prepares the relevant information in response to the user's query.
- **Intention Detection module:** This module receives the annotations from the Information Extractor module then orders the set of response options to be presented to the user. It assumes that the annotations available for word/selection and its context indicates the intent of the user making the query.

3 Summary of User Functionality

From a reader's perspective, *SmartReader* is a web-based browser application, that runs on many browsers running on PCs or touch tablets; so on the reader side no additional software is needed. It has a simple and intuitive web interface to sign up/in, browse available texts to the system's library and to upload user's texts. Users also have the option to either upload their input text or try available preprocessed documents in the system library. After uploading/opening a text, *SmartReader* then loads an interactive version of the text into a distraction-free tab, and then the reader can start interacting with the text either by clicking on a word or selecting any segment of text.

The system in turn queries the server, which takes into account all the annotations around the clicked/selected word's/segment's and based on these annotations, highlights a segment of the text depending on the selection context, and presents a response, which most likely fits the reader's intent at the click position.[2] For instance:

- If the reader clicks on a content word, *SmartReader* will present the word meaning, along with word type, sentence examples including the inquired word, as the default response. In case the clicked word is a part of a (possibly discontinuous) compound verb/noun, the tool highlights the whole compound structure and provides its meaning.

[2]For a much more detailed overview of user functionality, please refer to Azab et al. (2013).

- If the reader clicks on a pronoun, the system will inform the reader to whom this pronoun refers by highlighting both the pronoun and the antecedent in context. It will also provides the reader with the ability to navigate through all previous and future mentions in the text.
- The reader can also inquire about the grammatical role of a word within the sentence. *SmartReader* provides the reader with the grammatical role in a user-friendly fashion by mapping dependency labels to more descriptive and meaningful labels.
- The reader can explore beyond the default response by using the additional menu items provided: for instance she may select from a set of questions that *SmartReader* can generate involving a selected named-entity and get the response.
- Beyond these sentences/words interactions, *SmartReader* provides the reader with different levels of text summarization such as multi-section and whole document summarization. For this purpose, we use the Mead toolkit for English to provide the summarization functionality. [3]

4 Conclusion

We presented the implementation and architecture of a tool for helping non-native readers of English text to overcome language related hindrances while reading text. Our tool dubbed *SmartReader* can also be seen as a showcase of English NLP tools and resources that have been built by the NLP community, integrated into an e-book reader application. Our system architecture is general so that *SmartReader* can be adapted to more languages provided annotations resources are available for use in the UIMA framework. We are currently completing our implementation and are in the process of planning a test deployment for students for experimentation.

Acknowledgments

This publication was made possible by grant NPRP-09-873-1-129 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

[3] Available at `http://www.summarization.com/mead/`

References

Mahmoud Azab, Ahmed Salama, Kemal Oflazer, Hideki Shima, Jun Araki, and Teruko Mitamura. 2013. An NLP-based reading tool for aiding non-native English readers. In *Proceedings of RANLP*, Hissar, Bulgaria.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Soojeong Eom, Markus Dickinson, and Rebecca Sachs. 2012. Sense-specific lexical information for reading assistance. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*.

Christiane Fellbaum. 1998. WordNet: An electronic lexical database. *The MIT Press*.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of ACL'05*.

T. Götz and O. Suhre. 2004. Design and implementation of the UIMA common analysis system. *IBM Syst. J.*, 43(3):476–489.

Michael Heilman and Noah Smith. 2010. Extracting simplified statements for factual question generation. In *Proceedings of the 3rd Workshop on Question Generation*.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CONLL shared task. In *Proceedings CONLL'11*, pages 28–34.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, pages 1–54.

Detmar Meurers, Ramon Ziai, Luiz Amaral, Adriane Boyd, Ar Dimitrov, Vanessa Metcalf, Niels Ott, and UniversitŁt Tbingen. Enhancing authentic web pages for language learners.

John Nerbonne, Lauri Karttunen, Elena Paskaleva, Gabor Proszeky, and Tiit Roosmaa. 1997. Reading more into foreign languages. In *Proceedings of ANLP'97*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of EMNLP'10*, pages 492–501.

Dashboard: A Tool for Integration, Validation, and Visualization of Distributed NLP Systems on Heterogeneous Platforms

Pawan Kumar
Expert Software Consultants Ltd
New Delhi, India
hawahawai@gmail.com

Rashid Ahmad
LTRC, IIIT
Hyderabad, India
rashid.ahmed@research.iiit.ac.in

B. D. Chaudhary
CSED, MNNIT
Allahabad, India
bdc@mnnit.ac.in

Mukul K. Sinha
Expert Software Consultants Ltd
New Delhi, India
mukulksinha@gmail.com

Abstract

Dashboard is a tool for integration, validation, and visualization of Natural Language Processing (NLP) systems. It provides infrastructural facilities using which individual NLP modules may be evaluated and refined, and multiple NLP modules may be combined to build a large end-user NLP system. It helps system integration team to integrate and validate NLP systems. The tool provides a visualization interface that helps developers to profile (time and memory) for each module. It helps researchers to evaluate and compare their module with the earlier versions of same module. The tool promotes reuse of existing modules to build new NLP systems. Dashboard supports execution of modules that are distributed on heterogeneous platforms. It provides a powerful notation to specify runtime properties of NLP modules. It provides an easy-to-use graphical interface that is developed using Eclipse RCP. Users can choose an I/O perspective (view) that allows him better visualization of intermediate outputs. Additionally, Eclipse RCP provides plugin architecture; hence customized end-user specific functionality can be easily added to the tool.

1 Introduction

Dashboard is a tool for integration, validation, and visualization of Natural Language Processing (NLP) systems on heterogeneous platforms. Dashboard tool has been designed to build NLP systems reusing multiple heterogeneous (or homogeneous) NLP components.

Dashboard provides a common representation called *Shakti Standard Format* (SSF) (Akshar Bharati et al. 2007), a tree data structure, for storing linguistic information (analysis) produced by an NLP module in *attribute/value* pairs called *feature structure*. The SSF format has both *in-memory* representation as well as *stream* representation. They are inter-convertible using a *Reader* (stream to memory) and *Printer* (memory to stream). The in-memory representation is good in speed of processing, while the stream representation is good for *portability*, *heterogeneous platforms*, and *flexibility*, in general.

The tool provides SSF API for integrating NLP modules; modules use APIs for accessing in-memory data in multiple programming languages. For integration of existing modules SSF adapters are used. These adapters transform varying input/output representations to SSF format.

Dashboard has a frontend called *Dashboard* and a backend called *Dashboard Runtime*. Dashboard provides a graphical interface for setting up and configuring NLP systems. Dashboard Runtime performs the tasks of *coordination* and *communication* between the NLP modules.

A. Dashboard – the frontend

The Dashboard frontend provides a mechanism to setup and configure the NLP Systems that are based on Blackboard Architecture (Hayes-Roth 1985, Sangal 2004) runs on the Dashboard platform.

B. Dashboard Runtime

Dashboard Runtime performs the tasks of coordination and communications between the modules.

C. Dashboard as a Visualization Tool

Dashboard provides visualization and debugging functionality to researchers and developers.

This tool is now released and is being used by consortium members at 11 research institutions. Eighteen machine translation systems which translate between 9 pairs of Indian languages have been integrated and tested using this tool

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 9–12,
Nagoya, Japan, 14-18 October 2013.

(Sampark MT 2013, Anthem 2010). A comprehensive description of Dashboard tool is available in (Pawan et al. 2010, 2013).

2 Dashboard Features

Distinguishing features of Dashboard tool are:

A) NLP System can be partitioned, and different partitions (comprised of one or more modules) can reside on heterogeneous platforms on distributed machines.

B) Dashboard Tool provides powerful notation for describing runtime properties of individual modules as well as complete NLP system.
   - New modules can be added/removed either using user-interface or using declarative notation in specs file,
   - Switch runtime modes (*speed*, *debug* or *stepwise*) by the user/developer
   - *Conditional-run* for each module, i.e., at runtime user can skip execution of a module depending on (presence or absence of a *feature structure* in SSF data) input data,
   - *Triggers* for modules to promote robustness, e.g., set timeout for module that goes into an infinite loop.

C) Visualization interface is developed using Eclipse Rich Client Platform (RCP). RCP provides plugin architecture (McAffer et al. 2010). Developers can build user-defined plugins leveraging this plugin architecture.

D) Unlike other frameworks like GATE, UIMA and OpenNLP, Dashboard linguistic pipeline is created by specifying module properties in attribute/value pairs, which is compiled to generate an efficient runtime.

E) Similar to GATE, UIMA and OpenNLP, Dashboard can output linguistic analysis in SSFXML format that can be transformed to various formats using XSLT.

3 Sampark MT System: An Application Using Dashboard

We provide here screen shots of Punjabi-Hindi Sampark MT System (*sampark-pan2hin*). Figure 1 shows the normal view of the Dashboard. For reasons of clarity, we have labeled the icons in Figure 1, with numbers in red. On the Tool bar (labeled-2) we have *Run* button (labeled-1), and *View* button (labeled-6) on extreme right.

Below the Tool bar on extreme left is the *Application Explorer* pane (labeled-8). It shows the list of Dashboard applications (*sampark-pan2hin* system and *sampark-hin2pan* system). Just below that, i.e., *sampark-pan2hin* system all the modules of *sampark-pan2hin* system is listed. On the right of this pane we have a tabbed window where input text (labeled-3), in Punjabi language, *punjabi.txt* (visible), and output text (labeled-4), in Hindi language, *punjabi-output.txt* (this tab is invisible) are shown. Below this pane input/output from a module, *lexical-transfer* is shown (labeled-9 and labeled-10). Label-11 shows the error log if any from the system. Label-12 shows the selected module name along with selected sentence number. Label-13 shows the time profile for the selected module. Label -15 and 16 show tool bar for switching views of input and output data useful to computational linguists. The user can select views from a list of views i.e., SSF or XML or Native.

Figure 2, shows complete translation of Punjabi text into Hindi. Left pane shows the input text composed of 10 sentences, written in source language Punjabi, and the right pane shows translated output text, as the output from Sampark system written in target language Hindi.

Once the system is executed completely, the session can be saved for future analysis. In case the user wants to analyze the intermediate output of any specific module for a specific sentence, he can do so through Dashboard. For this he has to first expand the module list in the Application Explorer pane. Select the module whose intermediate output he wants to analyze. Once the user selects the module, he gets a Dashboard view as shown in Figure 3.

Figure 3, shows input to lexical-transfer module in the left pane, and output from lexical-transfer module into the right pane for sentence number 5 in SSF format. Time taken to execution of lexical-transfer module is shown at the bottom of the pane earlier labeled-13 in Figure 1.

To start any application/project system has to be configured/setup, which is done chronologically earlier but is being explained now. Figure 4 shows how a Sampark system is setup and configured for a new application/project. In the Application Explorer pane user has to add an application first. Once he adds an application, *DashboardSpec.xml* icon appears in the Application Explorer pane. When the user selects *DashboardSpecs.xml* icon, a tab window, earlier la-

beled-5 becomes visible, 'Applications Module Specification' pane is opened. On the bottom of this pane there are 5 tabs, viz., *Overview*, *Global Prop.*, *Runtime*, *Modules*, and *DashboardSpec.xml*. *Overview* tabs gives an overview about system configuration and setup. '*Global Prop*' tab configures the global properties for Dashboard Runtime, like, location of SSF API for various languages. Currently SSF API is available for C/C++, Perl, Python, and Java. *Runtime* tab allows for adding runtime arguments to the Dashboard (like UNIX command line parameters). *Modules* tab allows defining runtime properties of each module. Figure 4 shows the runtime properties of Punjabi Morph Analyzer. In the figure, only standard properties are defined. In the *DashboardSpec.xml* tab, user can manually edit the *DashboardSpec.xml* file.

Acknowledgments

We sincerely thank TDIL group, Dept. of IT, Govt. of India in granting and supporting such a challenging project. All NLP researchers and software engineers of the participating institutions (viz., IIIT Hyderabad, IIT Mumbai, IIT Kharagpur, Central University Hyderabad, AU-KBC Research Center Chennai, C-DAC NOIDA, Jadavpur University, Kolkata, IIIT Allahabad, and IISc Bangalore) need special thanks who have been using this tool and have helped in identifying new requirements for its improvement. Their continuous feedback has not only improved its functionality, but also stabilized it as a product.

We thank our colleagues, Rambabu, Phani, Avinash, and Sanket without whose tireless effort current version of the tool would not be possible.

We sincerely thank Prof. Rajeev Sangal, who allowed us to work on such an innovative project, because 'Dashboard as NLP Tool' was primarily his idea.

References

Akshar Bharati, Rajeev Sangal and Dipti M. Sharma. 2007. *SSF: Shakti Standard Format Guide*, LTRC, IIIT, Hyderabad, Report No: TR-LTRC-33.

Sampark MT. *Machine Translation System among Indian languages*. http://sampark.org.in, last accessed on 15-Feb-2013.

G. Anthes. 2010. *Automated Translation of Indian Languages*. CACM, vol. 53 (1), pp. 24-26.

B. Hayes-Roth. 1985. *Blackboard Architecture for Control*, Art. Intelligence.

R. Sangal. 2004. *Architecture of Shakti Machine Translation System*. IIIT Hyderabad.

Pawan Kumar, et al. 2010. *Dashboard: An Integration and Testing Platform based on Blackboard Architecture for NLP Applications*. Proc. of IEEE 6th Intl. Conf. on Natural Language Processing and Knowledge Engineering, Beijing, China. Report No: IIIT/TR/2010/84.

Jeff McAffer. 2010. *Eclipse Rich Client Platform*, Second Edition, Addison Wesley.

Pawan Kumar, et al. 2013. *Enriched Dashboard: An Integration and Visualization Tool for Distributed NLP Systems on Heterogeneous Platforms*. Proc. of IEEE 13th Intl. Conf. on Computational Science and Applications, Ho Chi Minh City, Vietnam.

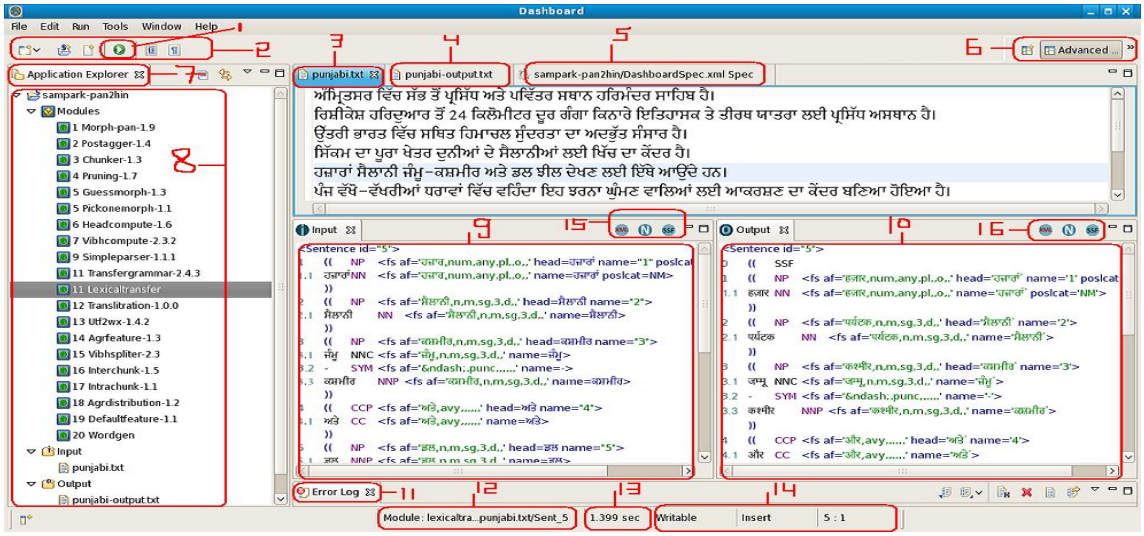


Figure 1: Normal view of Dashboard Tool, icons/panes/buttons are labeled to describe the tool's functionality.

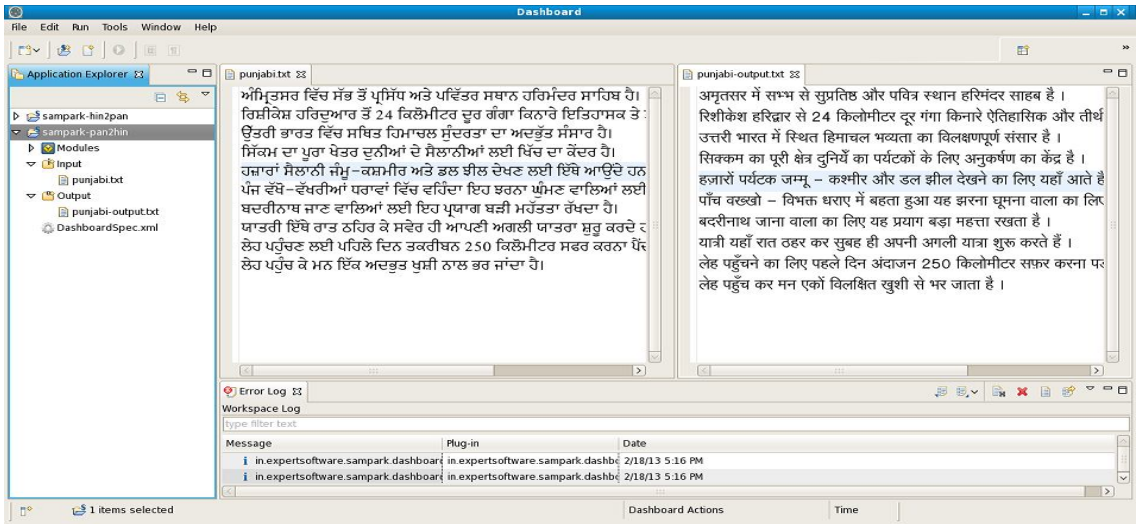


Figure 2: Shows Punjab-Hindi MT system, shows input and output text for 10 sentences.

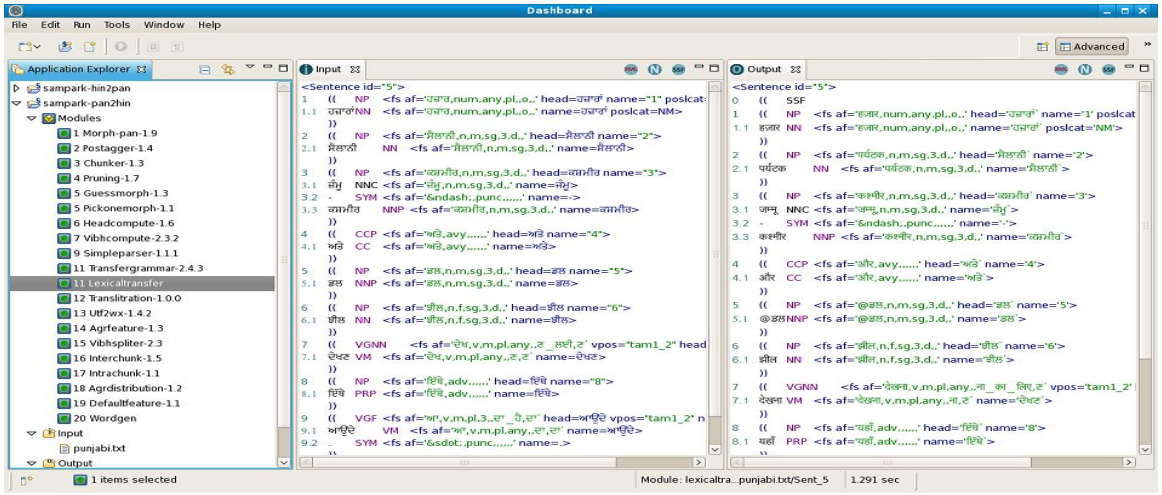


Figure 3: Module level input and output for *lexical-transfer* module, data in SSF format for sentence number 5.

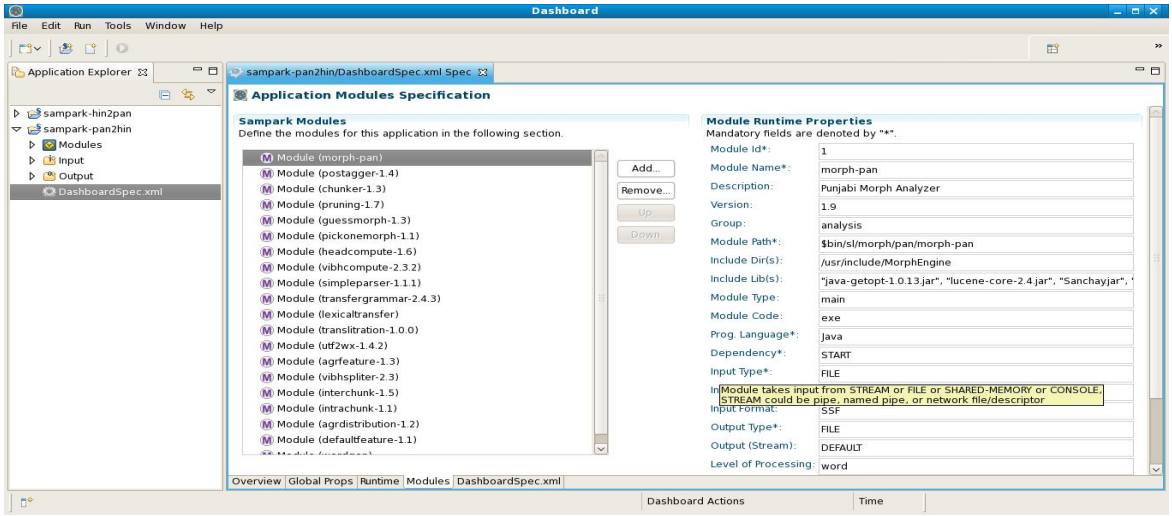


Figure 4: Application Setup and Configuration view. Right pane shows runtime properties for *Punjabi Morph Analyzer*. Also shows help text, if the mouse is hovered on the text labels.

DIRA: Dialectal Arabic Information Retrieval Assistant

Arfath Pasha, Mohammad Al-Badrashiny, Mohamed Altantawy, Nizar Habash, Manoj Pooleery, Owen Rambow and Ryan M. Roth
Center for Computational Learning Systems
Columbia University, New York, NY
Mona Diab
Department of Computer Science
The George Washington University
dira@ccls.columbia.edu

Abstract

DIRA is a query expansion tool that generates search terms in Standard Arabic and/or its dialects when provided with queries in English or Standard Arabic. The retrieval of dialectal Arabic text has recently become necessary due to the increase of dialectal content on social media. DIRA addresses the challenges of retrieving information in Arabic dialects, which have significant linguistic differences from Standard Arabic. To our knowledge, DIRA is the only tool in existence that automatically generates dialect search terms with relevant morphological variations from English or Standard Arabic query terms.

1 Introduction

The Arabic language poses two problems for information retrieval (IR). First, Arabic is morphologically rich, which increases the likelihood of mismatch between words used in queries and words in documents. Much work has been done on addressing this issue in the context of Modern Standard Arabic (MSA), primarily using different methods of stemming and query reformulation (Al-Kharashi and Evens, 1999; Darwish et al., 2005; Habash et al., 2006; Larkey et al., 2007).[1]

Secondly, the Arabic-speaking world displays diglossia, meaning that a standard language, MSA, co-exists with dialects, such as Egyptian Arabic (EGY). The dialects differ from MSA in many dimensions, which limits the effectiveness of using MSA tools to handle the dialects. Relevant to IR are lexical and morphological differences. Lexically, different words may be used to convey the same meaning in different dialects and MSA. Table 1 presents the same set of four words in English, MSA, Egyptian Arabic and Levantine Arabic.[2]

English	MSA	Egyptian	Levantine
to see	رأى *rÂý*	شاف *šAf*	شاف *šAf*
only	فقط *faqaT*	بس *bas*	بس *bas*
table	طاولة *TAwilaħ*	طربيزة *tarabayzaħ*	طاولة *TAwliħ*
wife [of]	زوجة *zawjaħ*	مرات *mirAt*	مرت *mart*
these	هؤلاء *hawˆlA'*	دول *dawl*	هدول *hadawl*

Table 1: Four examples showing lexical variation among Arabic dialects and MSA.

Morphologically, the dialects may use different forms from MSA, e.g., the short phrase 'he writes' appears as يكتب *yaktubu* in MSA , but as بيكتب *biyiktib* in EGY, ديكتب *dayiktib* in Iraqi Arabic and كيكتب *kayiktib* in Moroccan Arabic. The differences between MSA and dialect morphology can be rather large: Habash et al. (2012a) report that over one-third of EGY words cannot be analyzed using an MSA morphological analyzer; and Habash and Rambow (2006) report similar figures for Levantine verbs.

Furthermore, while MSA has a standard orthography, the dialects are not orthographically standardized, which leads to the coexistence of multiple spellings for the same word, e.g., the future marker in EGY may be written as ه *h* or ح *H*. We address this problem in the context of natural language processing of Arabic dialect by proposing a conventional orthography for representing dialec-

[1] For more information on Arabic natural language processing issues, see (Habash, 2010).

[2] Arabic transliteration throughout the paper is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007): (in alphabetical order) *AbtθjHxdðrzsšSDTĎςγfqklmnhwy* and the additional symbols: ' ء, Â أ, Ǎ إ, Ā آ, ŵ ؤ , ŷ ئ , ى ħ ة, ý ى.

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 13–16,
Nagoya, Japan, 14-18 October 2013.

tal Arabic elsewhere (Habash et al., 2012b).

Traditionally, almost all written Arabic was in MSA and not in the dialects. The retrieval of dialectal Arabic text has recently become necessary due to the increase of dialectal content on social media that is not "curated" (i.e., not chosen or edited by professionals). Our tool, DIRA (Dialectal [Arabic] Information Retrieval Assistant), is a query expansion tool that generates search terms, comprising both lexical and morphological variants, in MSA and EGY when provided with queries in English or MSA. No stemming decisions are made as part of DIRA in order to allow its output to be usable by a variety of IR systems with different stemming decisions. While the problem of morphological richness in IR has been addressed before, our DIRA system is, to our knowledge, the only system that addresses the problem of dialectal variation.

In the next three sections, we discuss DIRA's functionality, some of DIRA's implementation details, and two use scenarios.

2 DIRA's Functionality

DIRA is designed to be used as a component in a cross-lingual information retrieval system (Gey and Oard, 2001). Its purpose is to allow English and Arabic speakers to search for MSA and dialectal content using English or MSA queries. For instance, teachers and language learners may use English queries in DIRA to search the web for sentences containing certain MSA or EGY inflected word forms. An Arabic speaker may use MSA queries in DIRA to search for online EGY content.

The interface accepts English or MSA lemmas (citation forms) as input. MSA lemmas can be undiacritized or (partially) diacritized. Depending on user choice, DIRA outputs a set of MSA or EGY inflected forms for each lemma. The expansions are scored and ranked based on their frequency of use in large MSA and EGY corpora. Advanced settings give the users of DIRA the ability to specify weights for different inflectional feature values such as singular number, imperfective aspect, masculine gender, etc. This allows the system to prefer certain feature values that may be used more often in certain types of content. For instance, 1st and 2nd person may be used more often than 3rd person in blog articles while the converse may be true for news related articles. The weight of specific feature-value pairs can also be set to zero, thus eliminating their corresponding inflected forms from the expanded query.

We demonstrate DIRA's utility in a web application that uses Google search as the IR system. In this application, DIRA first translates (if needed) user queries (in English or MSA) and then morphologically expands the lemmas in the target language or dialect. Google's boolean search operators are used to concatenate a user-selected subset of the generated search terms to build the final search query. This final search query is used to perform a Google search for related online material. The demo web application shows the generated search terms as well as the Google search results. See Figures 1 and 2.

The online demo is available at **http://nlp.ldeo.columbia.edu/dira**.

3 DIRA's Implementation

DIRA expansion consists of three stages: lemma translation, morphological generation, and output ranking. First, DIRA translates each input lemma into a set of target lemmas using a trilingual English-MSA-EGY dictionary containing about 70,000 entries (Diab et al., in preparation). Second, DIRA morphologically expands the target lemmas into sets of inflected word forms using a target-language morphological generator (Habash, 2007). For MSA, the generator uses the databases of the BAMA/SAMA morphological analyzer (Buckwalter, 2004; Graff et al., 2009). For EGY, it uses the databases of the CALIMA-ARZ analyzer (Habash et al., 2012a). Since the CALIMA-ARZ analyzer maps a set of common spelling variations to the conventional orthography we use for EGY (Habash et al., 2012b), in generation mode, different spelling variants are produced. This is a desirable feature as it allows us to match more terms. In order to speed up the expansion process, DIRA utilizes a lookup cache created from large MSA and EGY corpora and extended online with new generated forms. Third, DIRA ranks the expansions using a weighted combination of (a) lemma-feature probabilities estimated from large MSA and EGY annotated corpora, and (b) user-provided weights for various feature-value pairs.

The DIRA framework has been designed to be easily extended to other dialects. At a minimum, a dialectal-MSA-English dictionary and a databases for morphological generation are required. Additional optional resources include corpora for the new dialects that can be used to estimate different probabilities.

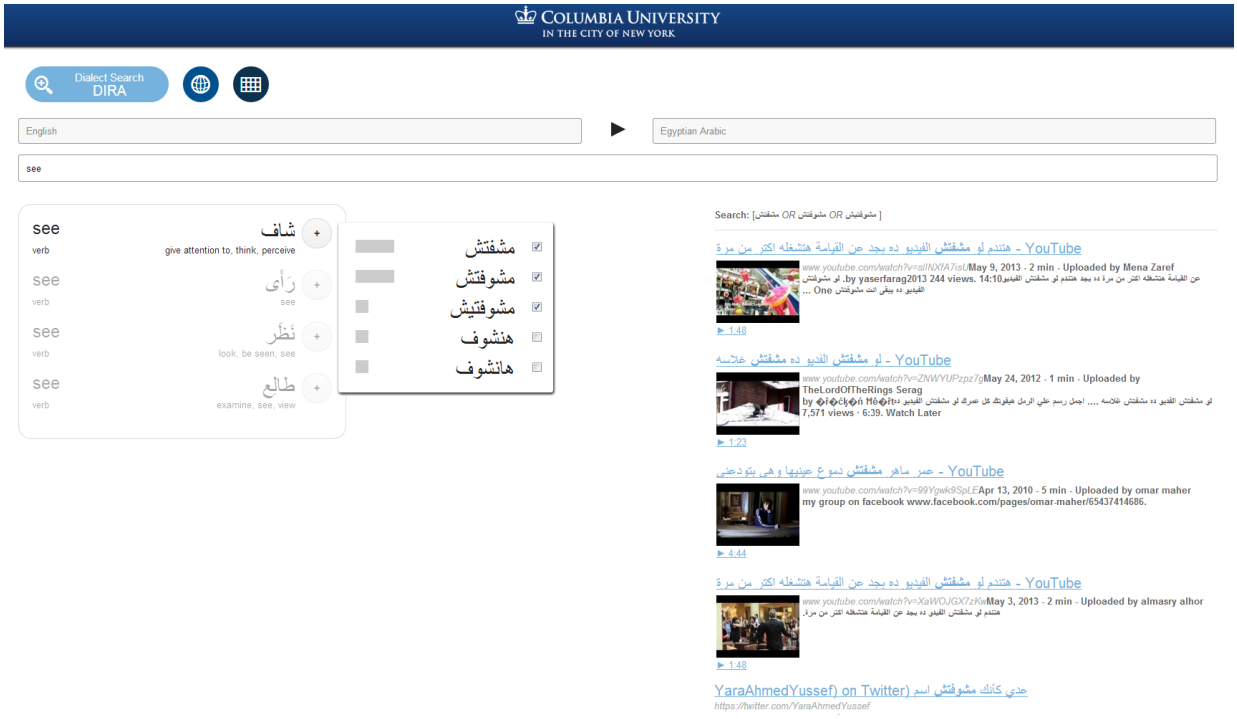


Figure 1: Screenshot of the DIRA demo web application. In this example, the user entered the English query ‘see’ and requested that the translation and expansion target Egyptian Arabic.

4 Two Use Scenarios

We discuss next two use scenarios. In the first scenario, a teacher of Arabic as a foreign language wishes to use real materials to teach the negation forms in Egyptian Arabic. This scenario is illustrated in Figure 1. She selects English as the input language and Egyptian Arabic as the output language. She chooses to search for the verb “see”. The system provides an English gloss for each lemma to help semantically distinguish different lemmas. The teacher can change the lemma choice, but she doesn’t because the first lemma is strongly dialectal. To see the available inflected forms for this lemma, she clicks on the plus sign next to the lemma. The online system proposes a maximum of five inflected forms per lemma. The first two are automatically selected by the system and both happen to express morphological negation. She additionally selects the third term, which is also negated.

As soon as the English search query is entered, the system immediatelty returns four lemmas, uses two inflected forms of the top-ranked lemma to construct the search query, and diisplays the results of the search with that seach query. The query and the results of the Google search are shown on the right hand side of the interface. As the user modifies the choice of lemma or inflected forms on the left-hand side of the interface, the query and search results are immediately updated to the right. In Figure 1, we see the search query is the disjunction of the three inflected forms our user selected.

In the second scenario, a native speaker of Arabic who may not know Egyptian Arabic wishes to conduct a search in Egyptian Arabic. This scenario is illustrated in Figure 2. He selects Standard Arabic as the input language and Egyptian Arabic as the output language. He enters the MSA question ‘أين مرسي’ *Âyn mrsy* ‘where is Morsi’ as his base query. DIRA expands identifies two possible lemma matches for each term. For the first word, it generates the verbal lemma أيّن *Âyn* ‘ionize’ and the interrogative particle lemma فين *fyn* ‘where’. For the second word, it generates the noun lemma مرسى *mrsý* ‘harbor’ and the proper noun lemma مرسي *mrsy* ‘Morsi’. For both terms, the first lemma is automatically selected. The user deselects the system’s automatic choices and clicks on the second reading for each term as these choices fit his intended query. As in the first scenario, after each choice is made, the query terms are adjusted and the search results presented immediately.

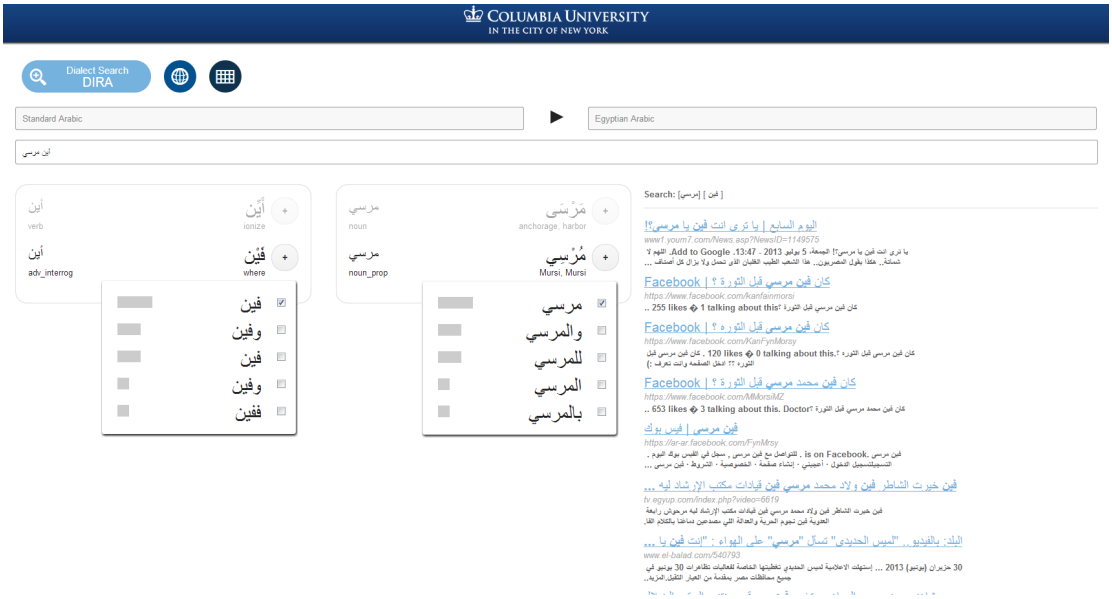


Figure 2: Screenshot of the DIRA demo web application. In this example, the user entered the MSA query '*أين مرسي* *Âyn mrsy* 'where is Morsi' and requested that the translation and expansion target Egyptian Arabic.

References

Ibrahim A Al-Kharashi and Martha W Evens. 1999. Comparing words, stems, and roots as index terms in an Arabic information retrieval system. *Journal of the American Society for Information Science*, 45(8):548–560.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Kareem Darwish, Hany Hassan, and Ossama Emam. 2005. Examining the effect of improved context sensitive morphology on Arabic information retrieval. *Computational Approaches to Semitic Languages*, 100:25.

Mona Diab, Abdelati Hawwari, Heba Elfardy, Pradeep Dasigi, Mohammad Al-Badrashiny, Ramy Eskander, and Nizar Habash. in preparation. Tharwa: A multi-dialectal multi-lingual machine readable dictionary.

F. Gey and D. Oard. 2001. The TREC-2001 Cross-Language Information Retrieval Track: Searching Arabic Using English, French or Arabic Queries. In *The 10th Text Retrieval Conference (TREC-10)*.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of ACL'06*, Sydney, Australia.

Nizar Habash, Clinton Mah, Sabiha Imran, Randy Calistri-Yeh, and Páraic Sheridan. 2006. Design, Construction and Validation of an Arabic-English Conceptual Interlingua for Cross-lingual Information Retrieval. In *LREC-2006*, Genoa, Italy.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

N. Habash, R. Eskander, and A. Hawwari. 2012a. A Morphological Analyzer for Egyptian Arabic. In *NAACL-HLT 2012 Workshop on Computational Morphology and Phonology (SIGMORPHON2012)*, pages 1–9.

Nizar Habash, Mona Diab, and Owen Rabmow. 2012b. Conventional Orthography for Dialectal Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Istanbul.

Nizar Habash. 2007. Arabic Morphological Representations for Machine Translation. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Leah S. Larkey, Lisa Ballesteros, and Margaret E. Connell, 2007. *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, chapter Light Stemming for Arabic Information Retrieval. Springer Netherlands, Kluwer/Springer edition.

Keyphrase-Driven Document Visualization Tool

Gábor Berend and Richárd Farkas
Department of Informatics,
University of Szeged
Árpád tér 2., Szeged, 6720, Hungary
{berendg,rfarkas}@inf.u-szeged.hu

Abstract

The need to navigate through massive document sets is getting common due to the abundant data available around us. To alleviate navigation, tools that are able to grasp the most relevant aspects of document subsets and their relations to other parts of the corpus can be highly beneficial. In this paper, we shall introduce an application[1] that processes and visualizes corpora to reveal the main topics and their relative roles to each other. Our suggested solution combines natural language processing and graph theoretic techniques for the visualization of documents based on their automatically detected keyphrases. Furthermore keyphrases that describe thematically related subcorpora are also extracted based on information-theoretic grounds. As for demonstration purposes our application currently deals with papers published at ACL workshops.

1 Introduction

The abundance of textual data that surrounds us often poses difficulties when we are looking for relevant documents in some field. For instance, when a researcher faces a new problem to be solved, she often has to process large amounts of academic papers that are not necessarily directly related to her field of expertise. Difficulties can arise simply from the amount of data to be processed as well as from the absence of knowledge about which articles to regard as relevant. Various solutions exist that try to alleviate data management, such as assigning keyphrases or generating summaries to documents and document subsets, but probably the most useful way of doing so is to support these approaches with some kind of visualization.

Our application constructs a similarity graph of documents and performs a force-directed layout implementation on that graph. Document similarity can be measured based on multiple criteria, i.e. bibliographic similarity – based on co-authorships and citations – or contextual similarity – based on the shared vocabulary or proper keyphrases that documents have in common. In our demonstration – which provides a visualization framework for ACL workshop papers – we present a contextual similarity-based visualization which is based on **keyphrases**.

In order to determine the kepyhrases of articles our state-of-the-art keyphrase extraction module was utilized. Then similarity between pairs of documents is calculated based on the extracted keyphrases and a similarity graph of the documents is formed.

As a subsequent step document communities, i.e. subcorpora of thematically related articles are formed. The most representative keyphrases are then assigned to the identified document subsets which are determined relying on information theoretic grounds. Using a few keyphrases to describe a cluster can help the users in identifying topics they are interested in.

Representing documents in a bag-of-keyphrases fashion – instead of a bag-of-word one – had multiple benefits, i.e.

1. our representation is less influenced by the problem of measuring similarities in high dimensional spaces and
2. we naturally enjoyed computational benefits by representing documents with their most relevant terms only.

For some empirical support regarding these observations see our previous studies (2013).

[1] available at http://www.inf.u-szeged.hu/~berendg/keyphraseViz

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 17–20,
Nagoya, Japan, 14-18 October 2013.

2 Related work

Recently, several methods have been suggested for a more effective handling of large document sets.

Quazvinian et al. (2013) proposed a graph-based approach – utilizing the so-called *Citation Summary Network* built from sentences citing a particular paper – to create extractive summaries of scientific articles. They employed their approach not only for single documents but for scientific topics, i.e. multiple documents from the same area as well. Even though their suggested methodology is appealing, it treated the topics to be summarized and the assignment of documents to those topics to be known in advance. Also, summaries can be beneficial in getting to know a topic from a glance, however, it can hardly be utilized to reveal the intra-topic document relations, nor the relatedness of different topics to each other.

Topic models such as Latent Dirichlet Allocation (Blei et al., 2003) provide an efficient way to analyse document sets. In their model, documents are treated as a mixture of topics where each topic has a distribution over the vocabulary of words. Although topic models are able to reveal general trends and identify topics based on word usage of documents, it does not really make it clear how documents are organized within each topic and it is also unclear how different topics connect to each other.

Eisenstein et al. (2012) introduced *TopicViz*, an LDA-based document visualization system, which can be regarded as a visually-aided information retrieval system. There are two basic differences between their approach and ours. First, they relied on topic models, whereas we employed graph partitioning in order to automatically determine document subtopics. Second, in their work they manually identified the topics determined by LDA whereas we let the automatically detected communities "speak for themselves", i.e. the most informative sets of keyphrases of size 3 were determined based on information theoretic grounds. Our proposed method did not need to know the number of topics to be identified in advance and its time requirements are also more favourable compared to the training of topic models, i.e. it can be performed on the fly during the initialization of our application. A further possible advantage of our approach compared to other LDA models as topic models tend to be trained on the single tokens level, whereas our approach can easily extract informative noun phrases and multi-word expressions as it operates on n-gram level.

In scientific document set visualization, citation analysis is often taken into consideration. The explicit relations among documents like citations can be naturally taken into consideration in our graph-based approach (which is not straightforward in LDA-based solutions). However, citation-based methods have the limitation that they are mostly useful for scientific document sets where citations exist.

3 Document set representation

Our representation used for the visualization of document collections is based on a weighted, undirected graph having individual documents as its nodes. In our case study – when our purpose was to visualize a document set that is clearly interpretable for computational linguists and which is comprised of easily distinguishable, thematically related subcorpora – we relied on the workshop papers present in the ACL Anthology Corpus (Schäfer et al., 2012).

3.1 Single-Document Keyphrase Extraction System

To have an efficient representation of the documents we first used our single-document keyphrase extraction system. Keyphrase extraction was treated as a supervised learning task where successive n-grams extracted from a document (i.e. keyphrase candidates) have to be classified as proper and improper keyphrases.

We utilized the NUS Keyphrase Corpus (2007) and the database of the SemEval-2 shared task on scientific keyphrase extraction (Kim et al., 2010) as training data for our supervised keyphrase candidate ranker. Our keyphrase ranking solution was based on the posterior probability of a "keyphrase or not" binary MaxEnt model trained within the MALLET (2002) framework and using a combination of our feature sets from our previous works on keyphrase extraction as described in (2010) and (2011).

3.2 Visualizing and partitioning the document set

Keyphrases extracted from the individual papers were used next as an input for the construction of a similarity graph which served as the basis of visualization.

3.2.1 Similarity graph

$G_{n,t} = (V, E_n, w_t)$ was defined as a weighted graph of documents, where $E_n = \{(u, v) : v \in neigh(u, n) \vee u \in neigh(v, n)\}$ and $neigh(u, n)$ is a function which returns the set of the n vertices that are closest to vertex u based on the similarity measure w_t.

The similarity measure $w_t(u, v)$ assigns a positive similarity score to documents u and v comparing the overlap between their top-t keyphrases that best describe them. Values n and t are thus hyperparameters that can be adjusted in our application to see their effects on the connectedness of the document graph.

Since a pair of documents can have multiple keyphrases in common, the weight assigned to a pair of nodes can be determined in multiple ways (which can be adjusted in the applet). For a similarity graph $G_{n,t}$, the similarity of documents u and v is 0 if the two documents have no keyphrases in common, otherwise it is aggregated due to one of the following strategies, via calculating

1. the Jaccard or Dice similarity between them, accordingly to the formulae $\frac{A \cap B}{A \cup B}$ and $\frac{2|A \cap B|}{|A|+|B|}$,
2. the cosine similarity of the two documents based on their top-t ranked keyphrases
3. $\sum_{k \in A \cap B} p(k, u)p(k, v)$
4. $\min_{k \in A \cap B}(p(k, u), p(k, v))$
5. $\max_{k \in A \cap B}(p(k, u), p(k, v))$

where sets A and B consist of the top-t ranked keyphrases of documents u and v, respectively and $p(k, u)$ is the probability that is assigned to the event that phrase k is a proper keyphrase of document u.

3.2.2 Visualization of the similarity graph

A force-directed layout visualization is employed based on the publicly available Java implementation of TouchGraph.[2] Their source code was extended and modified to our special needs, e.g. the awt windowing scheme was replaced by the more standardized Swing technology and various input fields for user interaction were added to the user interface. User interactions supported by the current version of the program are

1. Filter documents for keyphrases
2. Filter documents for some kind of metadata (such as the date or authors of a publication)
3. Hide/unhide entire document communities.

To illustrate the importance of nodes within the document graph, PageRank values are determined for each vertex. Since the similarity graph created is designed to be sparse – influenced by the parameter of maximal neighbours n – these calculations can be efficiently calculated with sparse matrix multiplication during the initialization phase of the programme.

Our application supports two kinds of partitioning of the document set to be visualized. If the user has a reliable partitioning of the document set in advance, it can be employed directly during the visualization, otherwise an automatic community detection is to be performed.

3.2.3 Modularity-driven community detection

The community detection employed here maximizes Newman's modularity (2004) in order to obtain a partitioning of the documents. Intuitively, what modularity measures for a given partitioning of a graph is the difference between the fraction of intra-community edges and the expected fraction of intra-community edges in the graph with the same number of vertices and edges but with its edges rewired randomly.

As our intention was to be able to deal with possibly massive data collections, it was a key aspect to keep computation requirements relatively low. Blondel et al. (2008) introduced a method which greedily approximates that partitioning of a graph which has the highest modularity. The proposed iterative method works in a bottom-up manner, starting from the state when all the vertices of the graph form a separate community. In the following steps vertices are moved into a community in such a way that their replacement should yield a best increase locally in the modularity.

3.3 Multi-Document Keyphrase Extraction System

Keyphrases are not only useful in automatically determining thematically related subgroups in document sets, but can be also applied to characterize those subgroups found in some corpus. For this reason our application assigns representative phrases to each community determined according to Section 3.2.3.

[2] http://sourceforge.net/projects/touchgraph/

The keyphrases of a cluster are those top-ranked keyphrases of the individual documents comprising the cluster which had the highest information gain metric, i.e. using the numbers of occurrences of a cluster-level keyphrase candidate inside and outside the particular cluster.

Then, for a subset of the document collection, the top-3 highest ranked candidates based on their information gain – which had at least a high relative frequency within the documents in the particular cluster as the relative frequency of the phrase outside the cluster – were treated as the keyphrases of the given cluster.

4 Conclusions

Since it is crucial in information-rich environments to be able to navigate quickly and effectively within document sets, we created a framework which alleviates it by implementing a visualization tool. Based on the keyphrases that can be automatically determined for individual documents of a document collection, useful and computationally efficient visualization can be built.

The fact that the visualization module only waits for a plain text input makes it possible to easily visualize datasets other than the one comprising of ACL workshop papers. Even though we favored keyphrase-based calculation of document similarities, the simple structure of the input file makes it also possible to perform visualizations based on other (e.g. bibliographic) criteria as well.

Besides providing a visualization tool for document sets it is also made easy to obtain more information from it via the automatic detection of document subgroups and the multi-document keyphrases assigned to each of them which makes the identification of topics possible.

Acknowledgments

This work was in part supported by the European Union and the European Social Fund through the project FuturICT.hu (TÁMOP-4.2.2.C-11/1/KONV-2012-0013) and "Hungarian National Excellence Program" (TÁMOP 4.2.4.A/2-11-1-2012-0001).

References

Gábor Berend and Richárd Farkas. 2010. Sztergak: Feature engineering for keyphrase extraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 186–189, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gábor Berend and Richárd Farkas. 2013. Extracción de palabras clave de documentos individuales para extracción de palabras clave de documentos múltiples. *Computación y Sistemas*, 17(2).

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1162–1170, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008+, July.

Jacob Eisenstein, Duen Horng Chau, Aniket Kittur, and Eric Xing. 2012. Topicviz: interactive topic exploration in document collections. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 2177–2182, New York, NY, USA. ACM.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 21–26, Morristown, NJ, USA. ACL.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113+, February.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the 10th international conference on Asian digital libraries: looking back 10 years and forging new frontiers*, ICADL'07, pages 317–326, Berlin, Heidelberg. Springer-Verlag.

Vahed Qazvinian, Dragomir R. Radev, S. M. Mohammad, Bonnie J. Dorr, David M. Zajic, M. Whidby, and T. Moon. 2013. Generating extractive summaries of scientific paradigms. *J. Artif. Intell. Res. (JAIR)*, 46:165–201.

Ulrich Schäfer, Jonathon, and Stephan Oepen. 2012. Towards an acl anthology corpus with logical document structure. an overview of the acl 2012 contributed task. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 88–97, Jeju Island, Korea, July. Association for Computational Linguistics.

Making Headlines in Hindi: Automatic English to Hindi News Headline Translation

Aditya Joshi[1,2] Kashyap Popat[2] Shubham Gautam[2] Pushpak Bhattacharyya[2]
[1]IITB-Monash Research Academy, IIT Bombay
[2]Dept. of Computer Science and Engineering, IIT Bombay
{adityaj,kashyap,shubhamg,pb}@cse.iitb.ac.in

Abstract

News headlines exhibit stylistic peculiarities. The goal of our translation engine '*Making Headlines in Hindi*' is to achieve automatic translation of English news headlines to Hindi while retaining the Hindi news headline styles. There are two central modules of our engine: the modified translation unit based on Moses and a co-occurrence-based post-processing unit. The modified translation unit provides two machine translation (MT) models: phrase-based and factor-based (both using in-domain data). In addition, a co-occurrence-based post-processing option may be turned on by a user. Our evaluation shows that this engine handles some linguistic phenomena observed in Hindi news headlines.

1 Introduction

'*Making Headlines in Hindi*' is a web-based translation engine for English to Hindi news headline translation. Hindi[1] is a widely spoken Indian language and has several news publications. The aim of our translation engine is *to translate English news headlines to Hindi preserving the content as well as Hindi news headline structure to the extent possible*. The engine is based on Moses[2] and has two central parts: modified translation unit and a co-occurrence based post-processing unit. The modified translation unit consists of phrase-based MT (Koehn et al., 2003)) and factor-based MT (Koehn et al., 2007). The automatic post-processing module performs co-occurrence-based replacement for correct sense translation of words by replacing translation of a word with the most frequently co-occurring translation candidate. This paper is organized as follows. Section 2 presents challenges of translating news headlines. Section 3 describes the UI layout. Section 4 discusses technical details of the modified translation unit while section 5 describes the post-processing module that uses co-occurrence-based replacement of words. Finally, Section 6 presents an evaluation of the engine while section 7 concludes our work.

[1]https://en.wikipedia.org/wiki/Hindi
[2]http://www.statmt.org/moses/

2 Challenges of News Headline Translation

Hindi news headlines have stylistic features that pose challenges to translation as follows:

1. **S-V-O order**: Hindi news headlines often follow the S-V-O order as opposed to S-O-V as commonly seen in Hindi sentences. A common news headline is '*अब तिहाड़ जेल में बिस्कुट बनाएंगे चौटाला (ab tihaaD jel mein biskooT banayenge chauTala; Now Chautala will make biscuits in Tihar jail)*' where the verb '*बनाएंगे (banayenge; will make)*' preceeds the object '*चौटाला (chauTala; Chautala)*'.

2. **Numbers for people**: Use of numbers to indicate a group of people, like in the case of English news headlines, is also common in Hindi news headlines. For example, the word '*Five*' in '*Five held for molesting woman*' stands for five people.

3. **Preferred choice of words**: Words that are commonly used in news headlines are often different from accurate translations. For example, '*RBI*' (abbreviation for 'Reserve Bank of India') is common in English news headlines - however, instead of using its transliterated form, news headlines tend to

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 21–24,
Nagoya, Japan, 14-18 October 2013.

translate it to '*रिज़र्व बैंक (rizarv bank; Reserve Bank)*' in Hindi news headlines.

4. **Missing verbs**: Often, verbs are also dropped as in the case of '*महाकुंभ में अजब-गजब संतो की भीड़ (mahakumbh mein ajab-gajab santon kii bheeD; Herds of fascinating saints in Mahakumbh (fair))*' where a form of the word 'be' has been dropped.

Figure 1: Making Headlines in Hindi: Snapshot of Output

3 UI Layout

The interface of the engine is divided into two vertical blocks for clarity: one for input and another for output. The input to the translation engine consists of:

(a) Text area for English news headline(s),

(b) Option to select Phrase-based v/s Factor-based model,

(c) Checkboxes for co-occurrence based replacement, transliteration for OOVs and displaying alignment table for the output: Each of these options can be turned on/off.

While one out of the two options in (b) must be selected, check-boxes in (c) are optional. Each of the components stated above are described in Section 4.

The output consists of:

(a) The *best five translations* obtained in Hindi

(b) A *color-coded alignment table* in case the option to display the alignment table : This helps to understand how each word got translated and then reordered.

(c) *Time taken* for translation

Figure 1 shows a snapshot of the UI. Moses-Baseline indicates the naive translation engine while Moses-MLM-Dict is the modified phrase model.

4 Modified Translation Unit

We implemented two translation models: phrase-based and factor-based. The training corpus consisted of parallel corpus obtained from (a) Gyan-nidhi[3] consisting of 2,27,123 sentences and (b) Mahashabdkosh[4] consisting of 46,825 judicial sentences. To transliterate out-of-vocabulary words, we modified transliteration engine provided by Chinnakotla et al. (2010). The original transliteration was trained for Hindi to English transliteration. For the purpose of our engine, we re-trained this model for English to Hindi transliteration. This section describes each of these components.

4.1 Phrase-based Model

The **Phrase-based MT** model was trained using Moses by (Koehn et al., 2007). In order to improve the quality of translation, we modify different components of the model in two ways. To preserve sentence order, we use a **modified language model** - a language model trained using in-domain data consisting of 20,220 news headlines from BBC Hindi website[5] and 2,02,335 news headlines from Dainik Bhaskar[6] archives of 2010 and 2011. The fact that this modified language model is a better fit to the target data is highlighted by the perplexity value obtained using SRILM toolkit by (Stolcke, 2002). For bi-grams, the perplexity of the Dainik Bhaskar corpus with a test news headline corpus was 434.06 while the perplexity of corpus consisting of tourism documents was 1205.58. Similar trend was observed in case of tri-grams. To enrich the translation mapping table available, we added a **bilingual dictionary** to the parallel corpus used for training the translation

[3]http://www.cdacnoida.in/snlp/digital_library/gyan_nidhi.asp
[4]http://www.e-mahashabdkosh.cdac.in/
[5]http://www.bbc.co.uk/hindi/
[6]http://www.bhaskar.com/

model. This bilingual dictionary was downloaded from CFILT, IIT Bombay[7]. This dictionary contains a total of 1,28,240 mappings and includes words as well as phrases. The fact that this dictionary enriches translations is observed in the case of a news headline containing the word '*catch-22*'. This word does not occur in the parallel news headlines. However, it gets correctly translated to '*जटिल (jaTil)*' according to the entry in the dictionary.

4.2 Factor-based Model

Our **Factor-based MT** model uses a set of factors along with words for translation. The factors used on source and target side are as follows.
1) On the source side, we use POS, lemma, tense and number. The POS tags are obtained from Stanford POS tagger[8] while the lemma are obtained from MIT Wordnet stemmer[9]. Tense and number are derived from POS tags.
2) On the target side, we use CFILT hybrid POS tagger[10] to obtain POS tags.
The factors are combined using options available in Moses. The lemma, tense and number on the source side generate the translated word on the target side. On the target side, words generate POS features. By generating best possible translations using a POS-based target language model, we hope to obtain translations in a POS order best suited to the news headline domain.

5 Post-processing: Co-occurrence-based Replacement

The engine provides an optional **co-occurrence based replacement** strategy to post-process the output. A manual evaluation showed that 14 out of 50 headlines were incorrect because of incorrect sense of one or more words. To overcome this problem, we implemented a post-processing strategy that automatically edits output obtained from the MT model using co-occurrence statistics as found in the in-domain news headline corpus. To elaborate how this works, consider the English news headline '*crpf jawan held on molestation charge*'. The translation obtained was '*सीआरपीएफ़ जवान पर आयोजित उत्पीड़न चार्ज (crpf jawaan par aayojit utpiDan chaarj; molestation charge organized on crpf jawan)*'. The word '*held*' gets translated to '*आयोजित (aayojit; organized/conducted)*' as opposed to '*गिरफ़्तार (giraftar; arrested)*'. The language model relies on n-grams and hence, does not take into account the correct sense of words in cases where the words do not occur together. For this purpose, we implemented a post-processing strategy that considers co-occurrence statistics of a target word with all other words in the sentence to find the best sense translation. In case of the above example, using the co-occurrences in a news headline corpus, we select the sense of '*held*' in Hindi which occurs most frequently with other words and replace the word with this translation. We do not consider co-occurrence statistics for function words. We understand that the above strategy does not work in the case of inflected forms of words in Hindi.

6 Evaluation

We evaluated the engine using a test set of 787 headlines downloaded from the website of a popular English daily, The Hindu[11] and manually translated into Hindi by native speakers. A BLEU score of 13.40 is obtained for phrase-based MT and 5.73 for factor-based MT. In order to understand how the engine performs for different kinds of linguistic phenomena, we also performed a qualititative evaluation of the output. The following are examples of output from our engine. They handle different linguistic phenomena as follows:

1. **Ambiguity**:
   Input: Industrialist remembered
   Output: उद्योगपति को याद किया
   (udyogpati ko yaad kiyaa;
   Industrialist remembered)
   The input sentence in domains other than news headlines is ambiguous as it could mean that either an industrialist was remembered or an industrialist remembered something. In case of news headline, however, the former holds true. This is correctly reflected in the Hindi translation.

2. **S-V-O order**:
   Input: Now Jaganmohan will make biscuits in jail
   Output: अब जगमोहन करेंगे बिस्कुट जेल में
   (ab jaganmohan karenge biskoot jel mein;
   Now Jaganmohan will do biscuits in jail)

[7] http://www.cfilt.iitb.ac.in
[8] http://www-nlp.stanford.edu/software/tagger.shtml
[9] http://projects.csail.mit.edu/jwi/api/edu/mit/jwi/morph/WordnetStemmer.html
[10] http://www.cfilt.iitb.ac.in/Tools.html
[11] www.thehindu.com

The verb '*will do*' gets placed correctly in the target sentence thus preserving the verb order. However, the translation '*karenge (will do)*' is incorrect and must be '*banaange (will make)*'.

3. **Numbers for people**:
   Input: Five killed in bomb blast
   Output: पांच बम विस्फोट में मारे गये
   (paanch bum visfot mein maare gaye;
   Five killed in bomb blast)
   The output sentence is a perfect translation and correctly translates 'five' as 'paanch'. However, the news headline order is not retained in this case.

4. **Missing verbs**:
   Input: Veteran journalist dead
   Output: दिग्गज पत्रकार मृत
   (diggaj patrakaar mrut;
   Veteran journalist dead)
   The output sentence is a perfect translation although a form of '*be*' is absent in the source sentence.

5. **Translation of idioms**:
   Input: Croatia and Serbia bury the hatchet
   Output: क्रोटिया और सर्बिया जगड़ा ख़तम करना
   (kroatia aur serbia jhagDa khatam karna;
   Croatia and Serbia do-end-quarrel)
   The idiom '*bury the hatchet*' gets correctly translated to '*जगड़ा ख़तम करना; jhagDa khatam karna; to end a quarrel*' as a complete entity. This is a direct mapping from the bilingual dictionary and does not have the correct inflection.

6. **Sense correction due to co-occurrence based replacement**:
   Input: No hike in AMU tuition fees
   Moses-MLM-Dict: amu अध्यापन फ़ीस में कोई वृद्धि नहीं
   (amu adhyaapan fees mein koi pad-yaatra;
   hike (trek) in AMU tuition fees)
   Moses-CoOcc: amu शिक्षण क्षेत्र में कोई वृद्धि नहीं
   (amu shikshan fees mein koi vriddhi;
   hike (increase) in AMU tuition fees)
   We observe that our post-processing unit improves the output in some cases. The original output translates '*hike*' as '*पदयात्रा (pad-yaatra ; hike)*'. The co-occurrence-based replacement unit identifies and corrects the sense to '*वृद्धि (vriddhi; increase)*'. We understand that the '*no*' gets missed out in the translation.

7 Conclusion & Future Work

We presented '*Making headlines in Hindi*', a translation engine that aims to translate English news headlines to Hindi while preserving news headline styles in the target language. Our engine includes a phrase-based model and a factor-based model. The phrase-based model uses an in-domain language model and a bilingual dictionary. The factor-based model uses factors like POS, lemma, tense and number. In addition, we also described our post-processing strategy that performs co-occurrence-based replacement of words to obtain correct sense of target language words. An evaluation of the output of our translation engine shows that it performs well for many linguistic styles used in Hindi news headlines.

The co-occurrence-based strategy is naive. As a future work, co-occurrence-based strategy can be improved to incorporate inflections of words. Also, other approaches to improve translation quality may be considered.

References

Manoj Kumar Chinnakotla, Om P. Damani and Avijit Satoskar. 2010. Transliteration for Resource-Scarce Languages. *Proc. ACM Trans. Asian Lang. Inf. Process.*,

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation *Proc. of ACL 2007, demonstration session*, Prague, Czech Republic

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. *Proc. of EMNLP-CoNLL 2007*, Prague, Czech Republic

Philipp Koehn and Franz Josef Och and Daniel Marcu,. 2003. Statistical phrase-based translation *Proc. of NAACL 2003*, Edmonton, Canada

A. Stolcke. 2002. SRILM - An extensible language modeling toolkit. *Proc. International Conference on Spoken Language Processing, vol. 2*

MaltDiver: A Transition-Based Parser Visualizer

Miguel Ballesteros **Roberto Carlini**
Natural Language Processing Group
Pompeu Fabra University
Barcelona, Spain
{miguel.ballesteros, roberto.carlini}@upf.edu

Abstract

Transition-based dependency parsers are widely used in the Natural Language Processing community but they are normally treated as black boxes, assuming that they provide the dependency parsing of a set of examples. We present *MaltDiver*, a tool developed to visualize the transitions performed by the transition-based parsers included in MaltParser and to show how the parsers interact with the sentences and the data structures within. During the demo session, we will run MaltDiver on several sentences and we will explain the potentialities of such a system.[1]

1 Introduction

Natural language processing researchers apply transition-based parsers frequently, these parsers are implemented in MaltParser (Nivre and Hall, 2005; Nivre et al., 2007b). Most of the application developers make use of the parsers without knowing how these parsers actually work, treating them as black boxes.

In order to have a system that could help to understand how a transition-based parser works, we present *MaltDiver*. MaltDiver is a tool developed to visualize the transitions performed by the transition-based parsers included in MaltParser and to show how they traverse the transition-system. We believe that there are mainly two different target researchers, that belong to different knowledge levels: (i) expert users who are willing to see how the parser behaves with a new set of features or with a different parsing constraint, and (ii) non-expert users who are willing to understand how the parsers work with the sentences that they are interested to parse, helping them to find out errors during the parsing process or inconsistencies in the annotation.

In the rest of the paper, we explain how a transition-based parser works (Section 2), we describe how we have implemented MaltDiver (Section 3), we present related work (Section 4), we show some ideas for further work (Section 5) and we conclude (Section 6).

2 Transition-based parsing - MaltParser

A transition-based parser learns parsing models that are trained to predict the next state of a state machine. To this end, it uses features that are annotated in the input sentence and dependency structure features that are dynamically generated. A typical transition-based parser state, see Figure 1, consists in two data structures (a stack and a buffer), and the partially built dependency structure. The parser starts in an initial state and produces transitions in order to reach new states by using the predictions of the trained model. This kind of parsing is very efficient, normally linear, O(n), in the sentence length and it provides the possibility of using features based on the partially built dependency structure. However, in a transition-based parsing strategy, in which there is a lack of backtracking, it is difficult to avoid an error propagation when it occurs (McDonald and Nivre, 2007). This may serve also as an evidence about why we are interested in the existence of a system as the one that we are presenting in this paper.

MaltParser (Nivre and Hall, 2005; Nivre et al., 2007b) is a transition-based dependency parser generator that provides high results. In the CoNLL Shared Tasks in 2006

[1]The system is available for download at `http://taln.upf.edu/pages/MaltDiver/`. It includes examples and a complete readme file that explains how to use the tool.

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 25–28,
Nagoya, Japan, 14-18 October 2013.

Nivre's transition system:

Initial configuration → Terminal configuration:

Transitions:

SHIFT: $\langle \Sigma, i|B, H, D\rangle \Rightarrow \langle \Sigma|i, B, H, D\rangle$

REDUCE: $\langle \Sigma|i, B, H, D\rangle \Rightarrow \langle \Sigma, B, H, D\rangle$

LEFT-ARC (r): $\langle \Sigma|i, j|B, H, D\rangle \Rightarrow \langle \Sigma, j|B, H[i \rightarrow j], D[i \rightarrow \mathrm{r}]\rangle$
if h(i) $\neq 0$.

RIGHT-ARC (r): $\langle \Sigma|i, j|B, H, D\rangle \Rightarrow \langle \Sigma|i|j, B, H[j \rightarrow i], D[j \rightarrow \mathrm{r}]\rangle$
if h(j) $= 0$.

Figure 1: Transition System for arc-eager algorithm.

and 2007 (Buchholz and Marsi, 2006; Nivre et al., 2007a), it was one of the best parsers. MaltParser contains four different families of transition-based parsers, the current version of MaltDiver only handles arc-eager parsing algorithm. These parsers mainly differ in the attachment of right-dependents, being the arc-eager greedier when right attachments have to be generated (Ballesteros and Nivre, 2013).

Figure 1 shows the parsing transitions for Nivre arc-eager with reduce transition: (i) SHIFT, (ii) REDUCE, (iii) LEFT-ARC and (iv) RIGHT-ARC. Nivre's arc-eager parsing algorithm makes use of two data structures in order to handle the input words: a *buffer*, which keeps the words that have to be read, and a *stack*, containing words that have already been processed but they are still available to producing a dependency arc. The SHIFT transition removes the first word in the buffer, and puts it on the top of the stack. The REDUCE transition removes the word that is on the top of the stack because there are no more arcs that have this word as a dependent or as a head. The LEFT-ARC and RIGHT-ARC transitions create either and arc from right to left or left to right, and stores the new arc in the dependency structure H and list D of dependency labels for each word.

3 MaltDiver

MaltDiver is a system implemented in Java that *dives* into the transition-based system with the intention of showing the states that the parser performs for a given sentence. At this writing, our MaltDiver implementation only allows the visualization of the arc-eager parsing algorithm (Nivre, 2003) – but it would not be difficult to include new transition systems (Nivre, 2008) as we also mention in Section 5.

MaltDiver processes the outcome of the *diagnostic feature* of MaltParser,[2] which prints the transition sequence for each sentence of the test corpus. It basically shows the list of transitions and the dependency label selected (if available). Besides that, MaltDiver also makes use of the dependency tree produced in order to ensure the reliability of the transition sequences inferred in the MaltDiver processes.

We included an extra option in MaltDiver which is the one that corresponds with the *allow_root* option in MaltParser.[3] This option decides whether there is a dummy root node included in the first parsing state on the stack. As in MaltParser, the *allow_root* option is set to *true* in default settings.

Therefore, MaltDiver takes the following inputs: (i) input sentence, (ii) a sequence of transitions provided by the MaltParser diagnostic feature and (iii) a dependency tree produced by MaltParser for the input sentence. After that, it processes the list of transitions from left to right and it reconstructs the parser configurations off line.

MaltDiver includes a console version of the system, that prints the parsing states and the dependency structure that is being produced in each state in a pretty-print way. In order to ensure the usefulness of the system, MaltDiver produces a *pdf* file for each state of the parsing process by using the TikZ-dependency tool,[4] which provides a LaTeX interface that we use for the production of the different states and partially built dependency structures. Therefore, the pdf file allows to go backward and forward and save the current state in a separate *pdf* file. Besides the pdf file, the user could also access the LaTeX format file. Figure 2 shows an intermediate state of the pdf file generated within MaltDiver transitions by processing a sentence written in Spanish. The structure to the left of the picture is the **stack**,

[2] This can be achieved by using the following setting in MaltParser: *-di true -dif filename.log*

[3] See `www.maltparser.org/userguide.html`

[4] TikZ-dependency tool is available for downloading through `https://sourceforge.net/projects/tikz-dependency/`

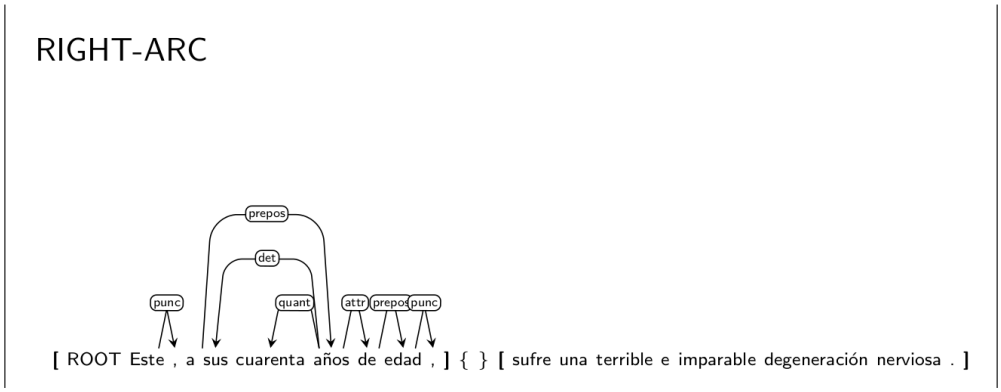


Figure 2: A print-screen of the system for the following sentence written in Spanish: *Este, a sus cuarenta años de edad, sufre una terrible e imparable degeneración nerviosa.* The structure to the left of the picture is the **stack**, and the one to the right is the **buffer**.

and the one to the right is the **buffer**.

4 Related Work

The importance of visualization systems has been evidenced during the last years in the NLP community. In the parsing and generation area we can find systems, such as MaltEval (Nilsson and Nivre, 2008), the Mate Tools (Bohnet et al., 2000), XLDD (Culy et al., 2011) or more recently, TreeExplorer (Thiele et al., 2013), which are, among other things, systems that visualize parse trees for evaluation and to provide the option of exploring dependency structures.

We also consider relevant and motivated in a similar way the work developed by Christopher Collins et al. about visualization of linguistic data in the Computer Graphics area (Collins et al., 2009a; Collins et al., 2009b), in which they present interactive visualization systems for NLP in discourse analysis, document content and even machine translation parse trees.

5 Future Work

A tool like MaltDiver provides several future directions and applications in different scenarios. The first idea would be to include other parsers in the system, such as the ones included in MaltParser that are not treated with MaltDiver. Some of them would be very easy to include, because they share with Nivre's parsers the transition system. However, there are some parsers that are a challenge, because we would have to include additional data structures in the visualization.

We could also provide an implementation of the pseudo-projective transformation of Nivre and Nilsson (2005) in the system process. We believe that the implementation of this step is rather straightforward, because we would only have to trace the projective parsing process –as we have already done– resulting in the pseudo-projective tree before post-processing. By comparing this to the final tree output by the system, we can then infer which arcs were moved due to pseudo-projective parsing. In fact, this is something that an user could do manually in the current version of MaltDiver.

A great idea would be to integrate MaltDiver with MaltOptimizer (Ballesteros and Nivre, 2012) in order to understand how the parser changes its behavior by updating the features selected.

6 Conclusions

We have presented MaltDiver, a tool that may serve as support for people interested in parsing research. This kind of tool would allow to understand the parsing processing by preparing resources about transition-based parsing in short time; researchers with deep knowledge about transition-based parsing could find it useful in order to understand the outcomes that the parsers may produce for a given sentence. For instance, a possible MaltDiver use could be an automatic comparison between different parsing behaviors for experiments about parsing root positions (Ballesteros and Nivre, 2013) or parsing directions modifications (Attardi and Dell'Orletta, 2009) and (Hall et al., 2007).

Acknowledgments

Thanks to Joakim Nivre, Johan Hall and Leo Wanner for their kind support and useful comments.

References

Giuseppe Attardi and Felice Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 261–264.

Miguel Ballesteros and Joakim Nivre. 2012. MaltOptimizer: A System for MaltParser Optimization. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*.

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1):5–13.

Bernd Bohnet, Andreas Langjahr, and Leo Wanner. 2000. A development environment for an mtt-based sentence generator. In *Proceedings of the First International Natural Language Generation Conference*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164.

Christopher Collins, M. Sheelagh T. Carpendale, and Gerald Penn. 2009a. Docuburst: Visualizing document content using language structure. *Comput. Graph. Forum*, 28(3):1039–1046.

Christopher Collins, Gerald Penn, and M. Sheelagh T. Carpendale. 2009b. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1009–1016.

Chris Culy, Verena Lyding, and Henrik Dittmann. 2011. xldd: Extended linguistic dependency diagrams. In *Proceedings of the 2011 15th International Conference on Information Visualisation*, IV '11, pages 164–169, Washington, DC, USA. IEEE Computer Society.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 933–939.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

Jens Nilsson and Joakim Nivre. 2008. Malteval: an evaluation and visualization tool for dependency parsing. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA).

Joakim Nivre and Johan Hall. 2005. MaltParser: A language-independent system for data-driven dependency parsing. In *Proceedings of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 137–148.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 915–932.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Gregor Thiele, Markus Gärtner, Wolfgang Seeker, Anders Björkelund, and Jonas Kuhn. 2013. Treeexplorer – an extensible graphical search tool for dependency treebanks. In *Proceedings of the Demonstrations of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*.

NICT Disaster Information Analysis System

Kiyonori Ohtake Jun Goto Stijn De Saeger* Kentaro Torisawa
Universal Communication Research Institute, NICT / Kyoto, Japan.
{kiyonori.ohtake, goto-j, stijn, torisawa} (at) nict.go.jp

Junta Mizuno
Resilient ICT Research Center,
NICT / Miyagi, Japan.
junta-m (at) nict.go.jp

Kentaro Inui
Graduate School of Information Sciences,
Tohoku University / Miyagi, Japan.
inui (at) ecei.tohoku.ac.jp

Abstract

Immediately after the 2011 Great East Japan Earthquake, the Internet was flooded by a huge amount of information concerning the damage and problems caused by the earthquake, the tsunami, and the nuclear disaster. Many reports about aid efforts and advice to victims were also transmitted into cyberspace. However, since most people were overwhelmed by the massive amounts of information, they could not make proper decisions, and much confusion was caused. Furthermore, false rumors spread on the Internet and fanned such confusion. We demonstrate NICT's prototype disaster information analysis system, which was designed to properly organize such a large amount of disaster-related information on social media during future large-scale disasters to help people understand the situation and make correct decisions. We are going to deploy it using a large-scale computer cluster in fiscal year 2014.

1 Introduction

It is widely recognized that Twitter and other social media played a significant role during the aftermath of the 2011 Great East Japan Earthquake by providing a huge amount of information concerning damages, problems, and aid efforts. But since this information exploded without any system to organize and disseminate it, most of the posted information was not effectively utilized for helping people (Varga et al., 2013).

We demonstrate NICT's prototype disaster information analysis system that organizes a large amount of disaster-related information and supports victims and rescue workers during future large-scale disasters. Its core is a question-answering (QA) system that lists answers to such disaster-related questions as "What is in short supply in Tokyo?" from the 50 million tweets transmitted within a month after the Great East Japan Earthquake. We designed our QA system to provide a wide range of answers including unpredictable ones, unlike the single answers given by IBM's Watson (Ferrucci et al., 2010). With our system, we can actually find much unpredictable information that is useful in aid efforts, including such diverse topics as *allergy friendly food for children*, *psychotropic medicine*, *dialyzers*, and *women's underwear*, all of which were scarce in the earthquake and tsunami areas. One lesson from the Great East Japan Earthquake was that a large-scale disaster can destroy a wide range of infrastructure in society, disrupt daily lives, and cause many unpredictable situations. We expect that QA systems, which can automatically process huge bodies of text to extract a wide range of answers to a wide range of questions, will be indispensable for dealing with such unpredictable situations.

Also, our system can map answers to Google Maps and help local governments and NPOs recognize the big picture of the damage caused by disasters as well as the gaps in aid efforts. Another of its functionalities helps people recognize false rumors spread on social media like Twitter. One well-known false rumor just after the earthquake was that Povidone-iodine provides protection from radioactivity. Using the methodologies of the STATEMENT MAP (Mizuno et al., 2012), our system would have identified that this rumor had been refuted by tweets just after it started to spread. If many people had found such tweets, the spread of such rumors might have been stopped or mitigated.

*Current address: Nuance Communications, Inc., Germany. stijn.desaeger (at) nuance.com

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 29–32,
Nagoya, Japan, 14-18 October 2013.

In this demonstration, we use as an information source the more than 50 million disaster-related tweets that were posted from March 9, 2011 to April 4, 2011. We show that our system provides valuable answers that are hard to predict and anticipate. We also demonstrate how its results support the decision-making processes of local governments or humanitarian organizations during large-scale disaster situations and how to deal with the credibility issues of tweets.

2 Overview of NICT's disaster information analysis system

Our system consists of the following components: a QA module, a web-based interface, a large-scale pattern entailment database[1] obtained from the web, and an indexing module for Twitter data.

The QA module is an extension of a pattern-based relation extraction method (De Saeger et al., 2009). Basically, it converts such input questions as "What causes deflation?", into lexico-syntactic pattern "X causes Y" and automatically computes its entailing patterns with the database, such as "X triggers Y" and "Y is a cause of X". X and Y are variables that correspond to the topic and interrogative pronoun of the question. These patterns are then matched against the index constructed from Twitter data after one of the variables is filled with the corresponding noun in the original question (Y = "deflation" in the above example). The nouns matching the unfilled variable (X) are provided as answers. This is the basic algorithm, which was extended in several aspects to deal with a wide range of questions.

Figure 1 shows the system's interface on web browsers that accept any simple natural language question. The system provides two modes for displaying the answers. One is the *semantic map* mode that categorizes the answers in semantic clusters with different colors to help users quickly survey all the answers for interesting and surprising answers (Figure 2). The other is Google Maps mode, which locates answers on Google Maps (Figure 3).

The system, which we demonstrate at IJCNLP 2013, runs on a single server. We are now developing a system that can work on large-scale computer clusters that can work with on-line indexing in real-time and simultaneously respond in real-time to many questions.

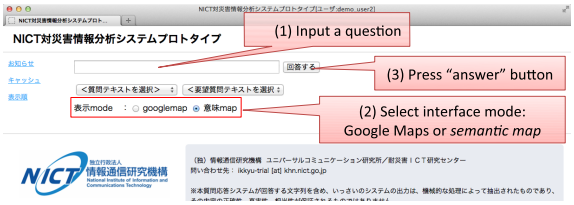


Figure 1: System's interface.

In our evaluation we obtained an average of 1,900 answers per question with 76% recall and 56% precision. We used 300 useful and important questions for disaster situations and 22,000 answers, which were manually collected using a full text search engine, and checked the top 1,000 results of each search.

Our system's target domain is not limited to disasters. We can apply it general events. At IJCNLP 2013, NICT also demonstrates WISDOM2013 (Tanaka et al., 2013), which shares the same QA module and targets a very large-scale web archive without limitation of the target domain.

3 Outline of demonstration

The following four steps outline our demonstration:

1. Our system accepts such questions as "What is in short supply in Tokyo?" We can choose a user interface for the system's response when we input a question.
2. Our system returns in the selected interface the results that were discovered from the 50 million tweets.
3. Our system can show a pop-up window that indicates the original tweets if we want to see the original texts from which the answer was extracted.
4. We can use the STATEMENT MAP developed by Tohoku University to confirm the credibility of the original tweets.

Since our system uses Japanese tweets and its results are in Japanese, we provide English translations.

Below, we describe the details of our system's results, and a method to check a given answer's credibility to a user's question. We also describe a smartphone version of the QA interface, which is also shown during our demonstration.

[1]This database includes more than six billion pattern pairs.

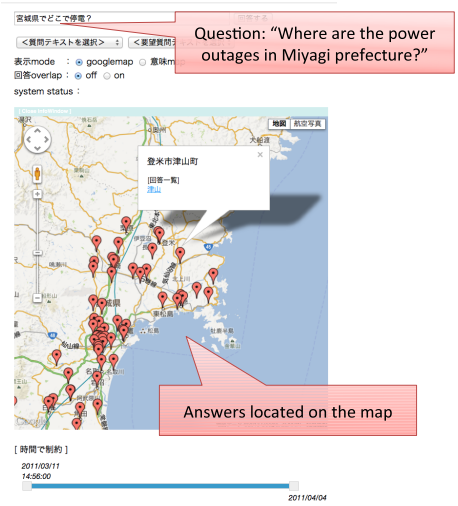


Figure 3: System’s result on Google Maps for the question:Where are the power outages in Miyage prefecture?

3.1 System results in selected interface

After a question is input, the system returns answers in the selected interface mode. The results of question “What shortages are there in Miyagi prefecture?” in the *semantic map* mode are shown in Figure 2. Many unpredictable answers are given.

During a large-scale disaster, we must grasp the locations of events that answer such questions as “Where are the power outages in Tokyo?” To understand the geological relations of events, our system locates the results on Google Maps. An example of our system’s results in the Google Maps mode is shown in Figure 3. We didn’t employ geotags in the tweets, because less than 1%[2] of them were geotagged. Instead, we prepared a huge location dictionary that contains location names and their addresses. The system uses this dictionary to detect location names and processes them for the Google Gecoding API[3]. By locating the results on a map, we can easily create a bird’s-eye view for the focus area that enables us to send relief to heavily damaged areas.

We also integrated into our scheme an information extraction system that was designed to extract problem reports and aid messages related to a disaster from tweets (Varga et al., 2013). If a question like “What problems have been reported in Fukushima prefecture?” is given, the problem reports, aid messages, and tweet pairs, which are problem-aid tweet matches, are provided by the information extraction system. These answers consist of the reports of the problems related to disasters along with aid messages, i.e., tweets describing efforts to solve problems. Such information is particularly useful for grasping the big picture of the damage and corresponding rescue efforts.

[2] http://semiocast.com/publications/2010_03_31_only_thirty_percent_of_tweets_are_from_the_us

[3] https://developers.google.com/maps/documentation/geocoding/

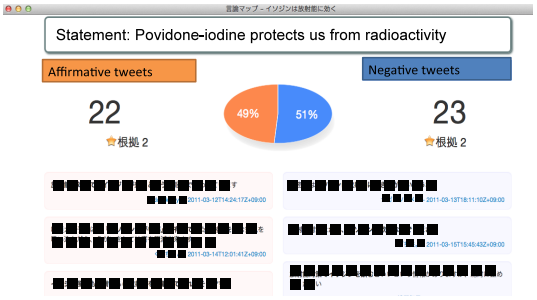


Figure 4: STATEMENT MAP results for statement: “Povidone-iodine protects us from radioactivity.”

3.2 Checking credibility issues

Due to the unreliable nature of information obtained by social media, someone may want to verify an answer’s credibility. For example, the results for the question, “What is effective against radiation?” include such unreliable ones as *gargling*, *soft seaweed*, *beer*, and *soybeans*. Our system provides a support method that evaluates the credibility of information sources by presenting a comprehensive survey of opinions on a topic.

Figure 4 shows the results[4] produced with a STATEMENT MAP of the query: “Povidone-iodine protects us from radioactivity.” Both opinions affirming and contradicting this statement are arranged to highlight their contrast. If a statement is a false rumor, many contradicting tweets will probably be presented.

3.3 Smartphone applications

An application that provides almost all of the functions of our system is available for iPhones. Figure 5 shows some screenshots of the iPhone application.

[4] The tweets are blacked out due to copyright issues.

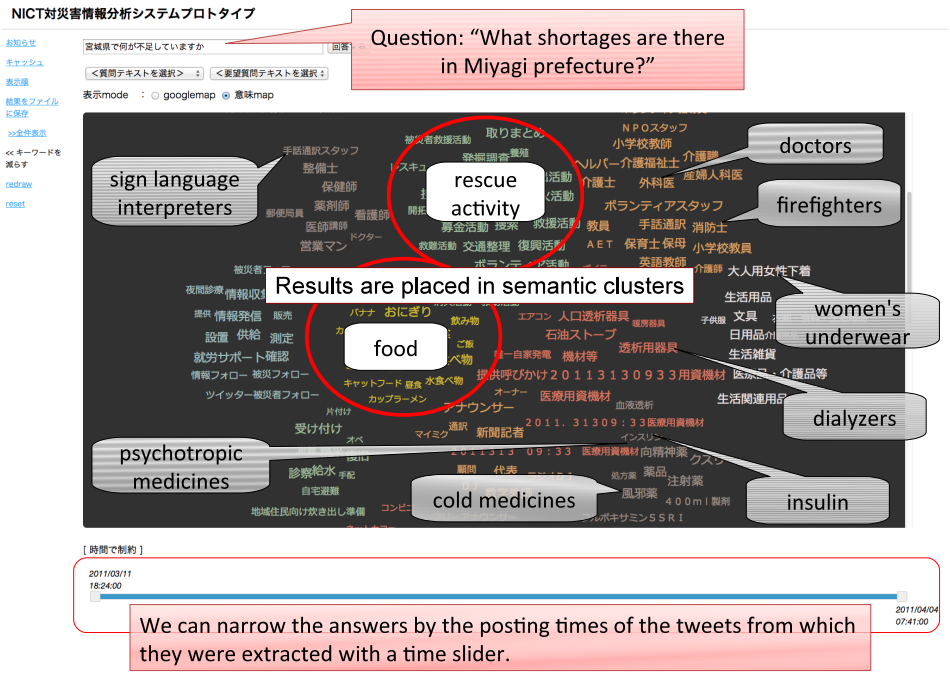


Figure 2: Example of system’s answer in *semantic map* mode

Figure 5: Screenshots of iPhone application

4 Conclusion

This paper briefly introduced NICT’s prototype disaster information analysis system, and our demonstration of it at IJCNLP 2013. We will make the system available to the public in fiscal year 2014. Future work will introduce such new functionalities as Why-Question Answering (Oh et al., 2013).

References

Stijn De Saeger, Kentaro Torisawa, Jun’ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *Proceedings of the IEEE International Conference on Data Mining (ICDM’09)*, pages 764–769.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An overview of the deepQA project. *AI Magazine*, 31(3):59–79.

Junta Mizuno, Eric Nichols, Yotaro Watanabe, and Kentaro Inui. 2012. Organizing information on the web through agreement-conflict relation classification. In *Proceedings of the 8th Asia Information Retrieval Societies Conference (AIRS2012)*, pages 126–137.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-question answering using intra- and inter-sentential causal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1733–1743.

Masahiro Tanaka, Stijn De Saeger, Kiyonori Ohtake, Chikara Hashimoto, Makoto Hijiya, Hideaki Fujii, and Kentaro Torisawa. 2013. WISDOM2013: A large-scale web information analysis system. In *Proceedings of the IJCNLP 2013 System Demonstrations*.

István Varga, Motoki Sano, Kentaro Torisawa, Chikara Hashimoto, Kiyonori Ohtake, Takao Kawai, Jong-Hoon Oh, and Stijn De Saeger. 2013. Aid is out there: Looking for help from tweets during a large scale disaster. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1619–1629.

SINNET: Social Interaction Network Extractor from Text

Apoorv Agarwal
Computer Science, Columbia University
New York, NY, USA
apoorv@cs.columbia.edu

Anup Kotalwar
Microsoft, Inc.
Redmond, WA, USA
ankotalw@microsoft.com

Jiehan Zheng
Peddie School
Hightstown, NJ, USA
jzheng-14@peddie.org

Owen Rambow
CCLS, Columbia University
New York, NY, USA
rambow@ccls.columbia.edu

Abstract

In this paper we present a demo of our system: Social Interaction Network Extractor from Text (SINNET). SINNET is able to extract a social network from unstructured text. Nodes in the network are people and links are *social events*.

1 Introduction

Language is the primary tool that people use for establishing, maintaining and expressing social relations. This makes language the real carrier of social networks. In this paper, we present a demo of our system that automatically extracts a social network from raw texts such as literary texts, emails, blog comments and news articles.[1] We take a "social network" to be a network consisting of individual human beings and groups of human beings who are connected to each other through various relationships by the virtue of participating in *social events*. We define social events to be events that occur between people where at least one person is aware of the other and of the event taking place. For example, in the sentence *John talks to Mary*, entities John and Mary are aware of each other and of the talking event. In the sentence *John thinks Mary is great*, only John is aware of Mary and the event is the thinking event.

There has been recent work on extracting social networks from literary text (Elson et al., 2010; He et al., 2013). However, both these works focus on extracting only conversational links between people, signaled in text by quotation marks. They do not extract *social event* links from other parts of text such as reported speech and other non-dialogue text. Our system overcomes this limitation.

The rest of the paper is structured as follows: In section 2, we briefly describe the research that has gone into building the system. In section **??**, we present the technical details of SINNET and describe our web demo.

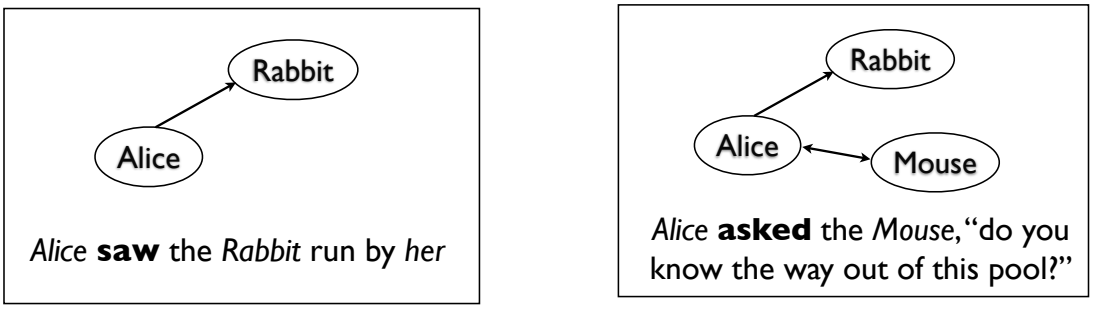
2 Research

The SINNET system is the result of several years of research (Agarwal et al., 2010; Agarwal and Rambow, 2010; Agarwal et al., 2012; Agarwal et al., 2013). In Agarwal et al. (2010), we introduced the notion of *social events*. A *social event* is a *happening* between two people, at least one of whom is cognizant of the other and of the event taking place. At a broad level, there are two types of social events: interaction (**INR**) and observation (**OBS**). INR is a bi-directional event in which both parties are mutually aware of each other. Examples of INR are a meeting or a dinner. OBS is a one-directional event in which only one party is aware of the other. Examples of OBS are thinking about someone, or missing someone.

In Agarwal and Rambow (2010), we presented a preliminary system that uses tree kernels and Support Vector Machines (SVMs) to extract social events from news articles. In Agarwal et al. (2012), we presented a case study on a manually extracted network from *Alice in Wonderland*, showing that analyzing networks based on these social events gives us insight into the roles of characters in the story. Also, static network analysis has limitations which become apparent from our analysis. We propose the use of dynamic network analysis to overcome these limitations. In Agarwal et al. (2013), we introduce two baselines for

[1] A web demo is available at http://nlp.ldeo.columbia.edu/sinnet/

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 33–36,
Nagoya, Japan, 14-18 October 2013.



(a) (b)

Figure 1: Two figures exemplifying the meaning of social events and social network.

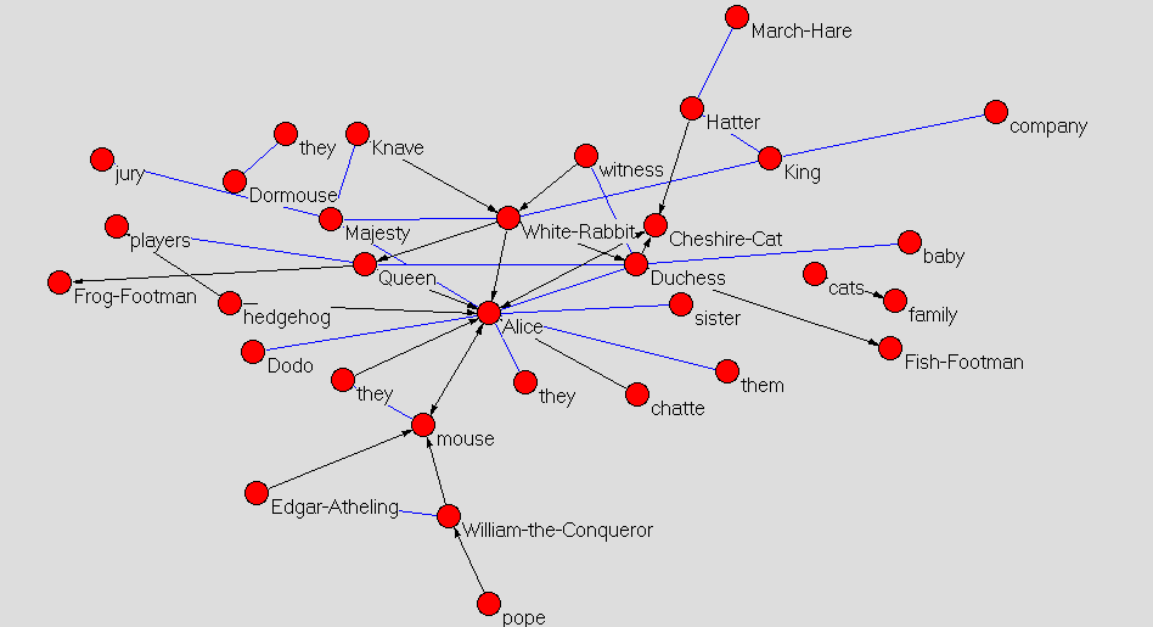
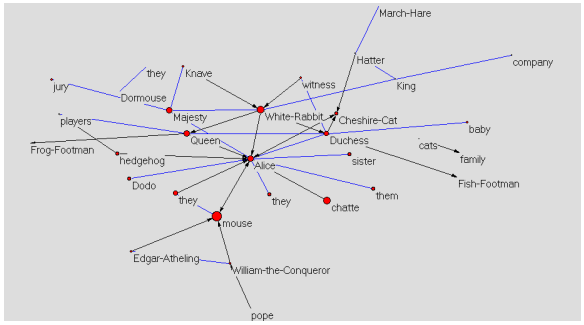


Figure 2: Social network of the entire *Alice in Wonderland*.

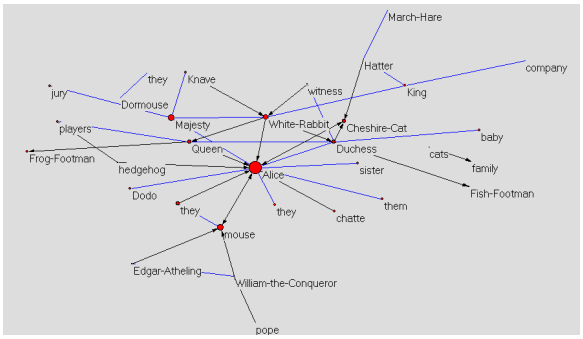
the social event extraction task and show that our system trained on a news corpus using tree kernels and support vector machines beats the baseline systems by a statistically significant margin. We also show that while the performance of our system on detecting social events in *Alice in Wonderland* achieves an F-measure of 61%, the unweighted network built using these detected social events is not statistically distinguishable from the un-weighted gold network according to popularly used network measures.

Figure 1 shows two figures exemplifying the meaning of social events and social networks. In the first figure, there are three entity mentions: Alice, Rabbit and her (co-referential with Alice). There is an OBS event between Alice and Rabbit triggered by the word in bold – *saw*. The direction of the event is from the observer to the one being observed. In the second figure there are two entity mentions: *Alice* and *Mouse*. There is a bi-directional interaction link between the Alice and Mouse triggered by the word *asked*.

Figure 2 shows the network extracted from an abridged version of *Alice in Wonderland* (Agarwal et al., 2012). Figure 3 shows the output of running the Hubs and Authority algorithm (Kleinberg, 1998) on the network. In information retrieval, an *authority* is a webpage that many *hubs* point to and a *hub* is a webpage that points to many *authorities*. In our network, webpages are synonymous to characters. Figure 3a shows the hubs in decreasing order of hub weights. Figure 3b shows the authorities in decreasing order of authority weights. We see that the main character of the story, *Alice*, is the main authority but not the main hub. This network may be used for other

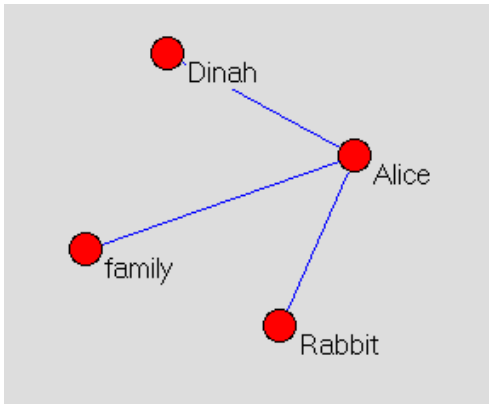


(a) Hubs in order of decreasing hub weight: Mouse, White Rabbit, Alice

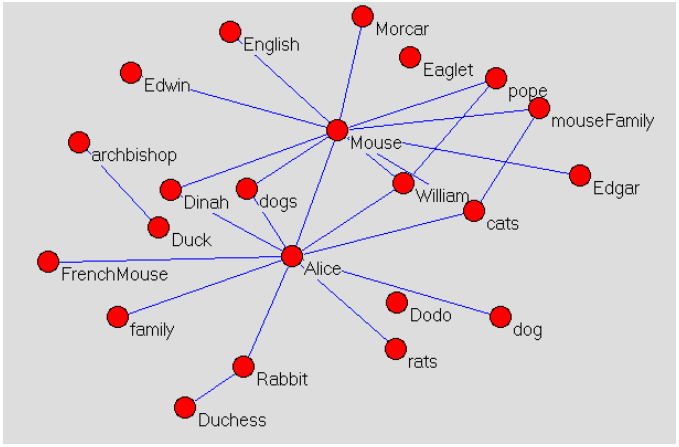


(b) Authorities in order of decreasing authority weights: Alice, Majesty (King), Mouse, White Rabbit

Figure 3: Hub and authority weights of characters. Larger the node, higher its weight.



(a) Network at the end of Chapter 1



(b) Network at the end of Chapter 3

Figure 4: Dynamic network plots of *Alice in Wonderland*

types of social network analyses such as finding communities.

In Agarwal et al. (2012), we argued that a static network does not bring out the true nature of a network. For example, even though the centrality of the *Mouse* in a static network is high, a dynamic network analysis shows that the mouse is central only in one chapter of the novel (Chapter 3 – *The drying ceremony*). Figure 4 shows the the network at the end of chapter 1 and chapter 3.

3 System details and Web demo

SINNET is fully implemented in Java. Following is a list of external off-the-shelf tools used by our current pipeline: Jet sentence splitter, Jet NER (Grishman et al., 2005), Stanford parser (Klein and Manning, 2003), SVM-Light-TK (Moschitti, 2006),

Input to SINNET may be provided in two formats: as raw text or text with entity annotations.

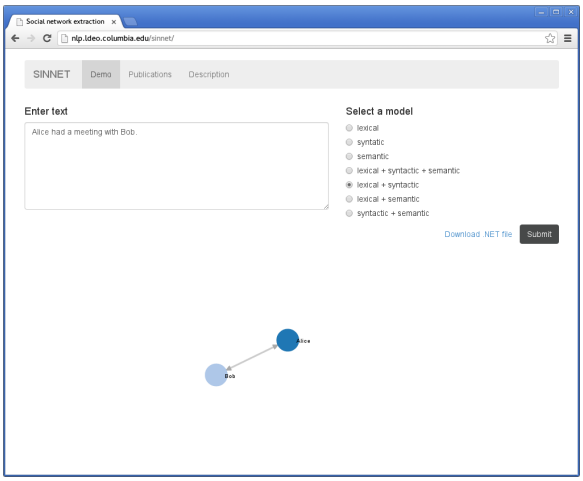


Figure 5: Image of our web demo

If the text is input as raw text without any entity annotations, SINNET first runs an off-the-shelf named entity recognizer and co-reference resolution (NER) tool. Currently, we run Jet (Grish-

man et al., 2005), but an interface makes it easy to plug-in any other NER tool. Once the text is annotated with entity mentions, for each sentence, for each entity mention pair per sentence, we create *test* examples in the format that our models accept. We use tree kernels with Support Vector Machines (SVM) for our models. Details of our system may be found in Agarwal and Rambow (2010). Any sentence splitter may be plugged in. Currently, we are using Jet's sentence splitter. Finally, the examples are fed to the models for prediction. The output is stored as a list of entities and their relations in a standard graph format. Currently, the output formats include graph modeling language (gml) and Pajek's .net format (Batagelj and Mrvar, 1998).

In many situations, the input text may already have entity mentions annotated and co-referenced. In these situations, SINNET will accept these gold entity mention annotations instead of running the NER tool. The rest of the processing remains the same as above.

Figure 5 shows an image of our web demo.[2] The demo has a text box for entering text. We have various models that use features from three levels of natural language abstractions: lexical, syntactic and semantic. Users of the web demo are given the option of selecting the type of model used for making predictions. We have seven models in place: lexical, syntactic, semantic and all combinations of these three types. Once the user inputs a text and selects the type of model, we display the extracted network and make the file with the extracted network (which is in a standard graph format such as .gml/.net) available for download. Our web demo has two other tabs: one listing the publications relevant to SINNET and the other mentioning technical details and capabilities of our web demo.

Acknowledgments

This paper is based upon work supported in part by the DARPA DEFT Program. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Kotalwar participated in the work described in this paper while at Columbia University.

[2] Available at http://nlp.ldeo.columbia.edu/sinnet/

References

Apoorv Agarwal and Owen Rambow. 2010. Automatic detection and classification of social events. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1024–1034, Cambridge, MA, October. Association for Computational Linguistics.

Apoorv Agarwal, Owen C. Rambow, and Rebecca J. Passonneau. 2010. Annotation scheme for social network extraction from text. In *Proceedings of the Fourth Linguistic Annotation Workshop*.

Apoorv Agarwal, Augusto Corvalan, Jacob Jensen, and Owen Rambow. 2012. Social network analysis of alice in wonderland. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 88–96, Montréal, Canada, June. Association for Computational Linguistics.

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. *In the Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)*.

Vladimir Batagelj and Andrej Mrvar. 1998. Pajek-program for large network analysis. *Connections*, 21(2):47–57.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting social networks from literary fiction. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147.

Ralph Grishman, David Westbrook, and Adam Meyers Proc. 2005. NYU's english ace 2005 system description. In *ACE Evaluation Workshop*.

Hua He, Denilson Barbosa, and Grzegorz Kondrak. 2013. Identification of speakers in novels. *The 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430.

Jon Kleinberg. 1998. Authoritative sources in a hyperlinked environment. *In Proc 9th ACMSIAM Symposium on Discrete Algorithms*, pages 668–677.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proceedings of European chapter of Association for Computational Linguistics*.

SmartNews: Towards content-sensitive ranking of comments

Marina Litvak
Sami Shamoon College of Engineering
Beer Sheva, Israel
marinal@sce.ac.il

Leon Matz
Sami Shamoon College of Engineering
Beer Sheva, Israel
leonm@sce.ac.il

Abstract

Various news sites exist today where internet users can read the most recent news and people's opinions about. However, usually these sites do not organize comments well and do not filter irrelevant content. Due to this limitation, readers who wonder about people's opinion regarding some specific topic, have to manually follow relevant comments, reading and filtering a lot of irrelevant text. In this work[1], we introduce a publicly available software implementing our approach, previously introduced in (Litvak and Matz, 2013), for retrieving and ranking the relevant comments for a given paragraph of news article and vice versa. We use Topic-Sensitive PageRank for ranking comments/paragraphs relevant for a user-specified paragraph/comment.

1 Introduction

Almost all modern news sites allow people to share their opinions by commenting a read article. However, usually comments are not organized well and appear under one long thread in chronological order. Some commenting systems include a rating component, but it is usually based on explicit feedback of users and does not relate to any specific content. In such conditions, the only way a reader can follow people's opinion about some *specific aspect* mentioned in the article is to scan manually a huge amount of comments.

Ranking comments on the web is a one of the central directions of IR in recent years (Dalal et al., 2012; Hsu et al., 2009). However, none of the works focused on the topic-sensitive ranking of comments. Since in many web domains like news different comments may refer to different aspects of the same article, resolving this problem is very important for structuring and better retrieval of user-contributed content.

In this paper we introduce an application for ranking comments in news websites relative to a given content[2]. The application provides ranked comments to the user-specified paragraph of a news item and, vice versa, ranked paragraphs that are relevant to a given comment. Our approach, that was previously introduced in (Litvak and Matz, 2013), is unsupervised and does not require training on an annotated data. It reduces the problem of topic-sensitive ranking of comments to the calculating of eigenvector centrality by adapting Topic Sensitive PageRank algorithm. The introduced application is implemented as a Chrome Extension to Yahoo! News site and is publicly available at author's homepage[3].

2 SmartNews

2.1 Problem Setting

We are given a set of comments $C_1, ..., C_m$ referring to an article describing some event and speaking on several related subjects. An article consists of a set of paragraphs $P_1, ..., P_n$ speaking on different related subjects. Our goal is, given paragraph P_i, to find a subset $C_{i_1}, ..., C_{i_r}$ of comments such that[4] (1) These are the most relevant to P_i comments that refer to topics described in P_i itself or comments about it; (2) The comments are ordered by the "relevancy" rank; (3) There are at most M comments.

Our method is based on eigenvector centrality

[1]This work was partially funded by the U.S. Department of Navy, Office of Naval Research.

[2]Here and further we refer to a paragraph as an independent text unit describing one of the article's aspects

[3]http://www.cs.bgu.ac.il/~litvakm/research/

[4]Here and further, we focus on comments ranking problem, while, generally, our method can be applied to the inverse problem – ranking paragraphs given a comment. Our plugin implements both directions.

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 37–40,
Nagoya, Japan, 14-18 October 2013.

principle (PageRank, as its variant), that already has been successfully applied for ranking and extracting (Mihalcea and Tarau, 2004; Erkan and Radev, 2004) text units. Our approach consists of two main stages: (1) graph constructing and (2) computing the eigenvector centrality. Since this paper is focused on the *application* and not approach, the next two subsections briefly summarize both stages. The details can be found in (Litvak and Matz, 2013).

2.2 Graph Representation Model

In order to represent our textual data as a graph, we relay on the known factors influencing PageRank described in (Sobek, 2003). They are also enumerated and discussed in details in (Litvak and Matz, 2013). We organize comments as nodes in a graph (denoted by a **comments graph**), linked by edges weighted with text similarity score calculated between nodes. Formally speaking, we build a graph $G(E,V)$, where $N_i \in V$ stands for a comment C_i, and $e_k \in E$ between two nodes C_i and C_j stands for similarity relationship between texts of the two comments.[5] Each edge e_k is labeled by a weight w_k equal to the similarity score (we use cosine similarity (Salton et al., 1975)) between the linked text units. Edges with a weight lower then a pre-defined threshold are removed. By weighing links we diminish the influence of links between thematically unrelated text units and, conversely, increase the influence of links between strongly related ones. An example of resulted comments graph is demonstrated in Figure 1(a).

We treat a paragraph as a query that must to influent the resulted ranks of comments. We add an additonal node (denoted by a **query node**) for the paragraph with respect to which the comments should be ranked. The query node is also linked to the comments nodes by similarity relations, with weighted edges directed from a query node to comment nodes. Adding weighed inbound links from the query node to thematically related comment nodes must increase their PageRank relative to other nodes. Here and further, we call the resulted graph **extended graph**. This stage is demonstrated in Figure 1(b).

The situation, where extended graph has groups of strongly connected nodes, mostly thematically irrelevant to a query node, is created when we have comments "talking" to each other and deviate from the main (query) topic. It is enough that only one node from a group will be linked to a query node for "grabbing" a query's rank to a group and, at each iteration, enlarging the PageRank of strongly connected nodes. In order to avoid (1) PageRank increasing in unrelated nodes linked with related ones in a closed system and (2) "leakage" of PageRank in a query node, we add outbound links from comment nodes to a query node. For uniform impact on all comment nodes, we give all edges the same weights of 1. Comment nodes that are strongly related to a query, will gain their PageRank back in each iteration due to a high weight assigned to inbound links from a query node, while irrelevant nodes will "loose" their PageRank irretrievably. The described update applied to a graph from Figure 1(b) will result in a new structure depicted in Figure 1(c).

In order to obtain similarity scores between nodes standing for text units, we calculate cosine similarity between vectors representing related texts, according to the Vector Space Model (Salton et al., 1975). Formally speaking, each text unit–paragraph or comment–is represented by a real vector V of size n, where $V[i]$ stands for *tf-idf* (Salton et al., 1975) of term i and n is a vocabulary size. Since we treat each text unit as a *document*, we adapt tf-idf to *tf-ipf* (term frequency inverse *paragraph* frequency) and *tf-icf* (term frequency inverse *comment* frequency) when applied on a paragraph or comment, respectively. The details and exact formulas can be retrieved from (Litvak and Matz, 2013).

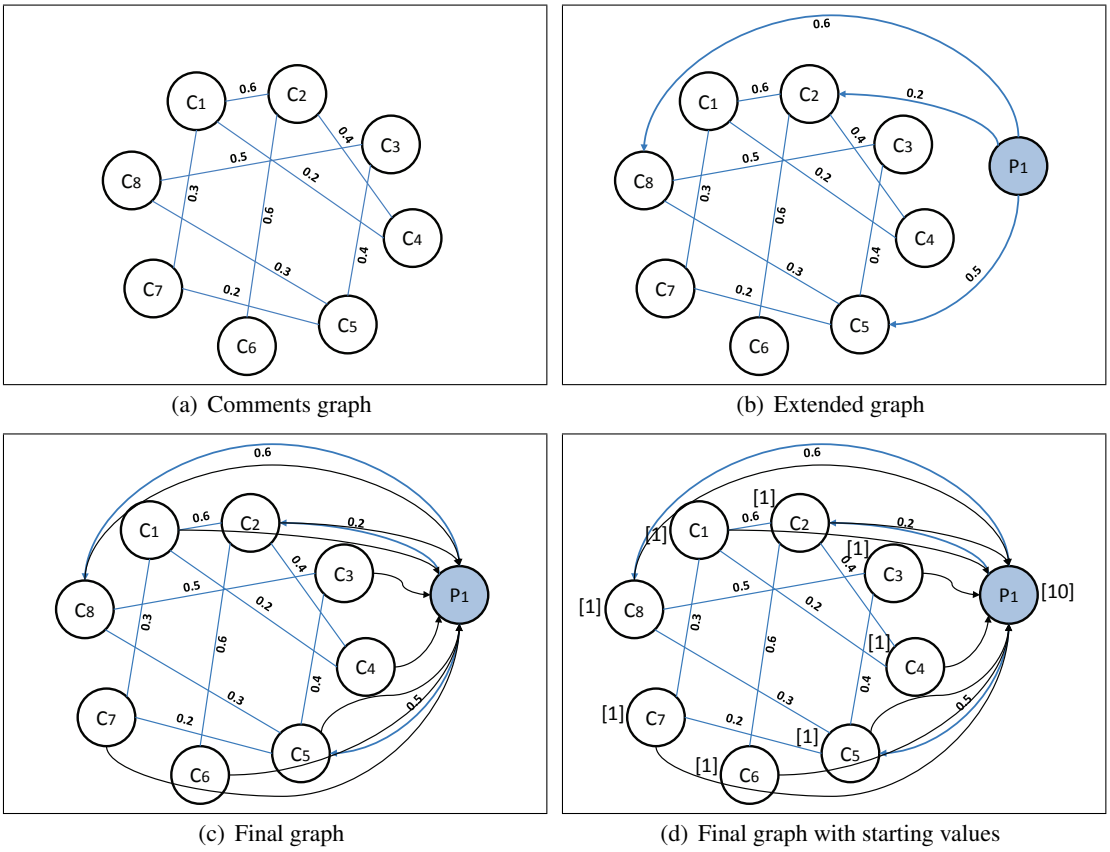
2.3 Computing the eigenvector centrality

For ranking and retrieving comments, we compute their eigenvector centrality by applying PageRank algorithm (Brin and Page, 1998) to an extended graph.

In order to influence node's rank by a query node for topic-sensitive retrieval, we relay on the known factors influenting PageRank score which are enumerated and described in (Litvak and Matz, 2013).

First, we give a high starting value to a query node before the iterative computation of PageRank begins. Adding outbound links from comment nodes to a query node (described in 2.2) helps to keep high PageRank in the query node through successive iterations. The final graph structure

[5] For the inverse problem, we represent a document as a graph of paragraphs (aka **paragraphs graph**) linked by a similarity relationship (Salton et al., 1997).



(a) Comments graph (b) Extended graph

(c) Final graph (d) Final graph with starting values

Figure 1: Graph representation: four steps.

including initial starting values is shown in Figure 1(d).

Second, in order to implement a theme-based retrieval, we adapt the idea of Topic Sensitive PageRank[6], where the thematically relevant comments get higher damping factor d. The final formula for ranking comments looks as follows.

$$PR(a) = E(a)d + (1-d) \sum_{x \in adj(a)} \frac{PR(x)w(a,x)}{\sum_{y \in adj(x)} w(y,x)},$$

where $E(a) = \frac{w(a,q)}{\sum_{i \in V} w(i,q)}$, q is a query node, $w(x,y)$ is a similarity score between nodes x and y.

We treat a PageRank score as a final rank of items. In a greedy manner, we extract and present at most M most ranked comments ordered by their rank to the end user. In our settings, $M = 5$.

3 Implementation Details

We implemented the introduced approach as a Chrome Extension (plugin) for the Yahoo! News[7] website. The plugin contains two sides: (1) **client** responsible for a data collecting and the results representation, and (2) **server** calculating ranks in a background.

Client performs the following: (1) collects the necessary textual and meta data and transfers it to the server, (2) visualizes the output (ranked comments and paragraphs, etc.) to the end user. The initial filtering of textual data is performed before transferring it to the server. The comments containing no words (considering synonyms) in common with the article are discarded. We used the following technologies for client's implementation: Javascript and jQuery Folder for scanning the article and collecting the relevant data, and JSON object as a data structure.

Server performs the following: (1) gets the textual data, (2) applies standard preprocessing including: HTML parsing, paragraph and sentence splitting, tokenization, stopwords removal, stemming, and synonyms resolving[8] for handling texts expressing the same issues with different vocab-

[6]Topic-Sensitive PageRank is a very intuitive choice in our setting, since we retrieve comments *with respect to* a given paragraph representing a topic an actual user is interested in.

[7]http://news.yahoo.com/

[8]with Synonym Map http://lucene.apache.org/core/old_versioned_docs/versions/2_9_1/api/all/org/apache/lucene/index/memory/SynonymMap.html

ulary, (3) builds VSM representation, then (4) builds graph representation, (5) calculates ranks of comments given a specified paragraph as a query or vice versa, (6) converts the processed data into Json object, and (7) transfers it to the client. We used the following technologies for server's implementation: Java EE, Tomcat server, Spring Environment.

In order to apply a Topic-Sensitive PageRank for a specific paragraph[9] we identify the actual paragraph a user is interested in by sending the position of the user's mouse to the server.

Figure 2 demonstrates the infrastructure of the plugin including interconnection between client (front end) and server (back end) sides.

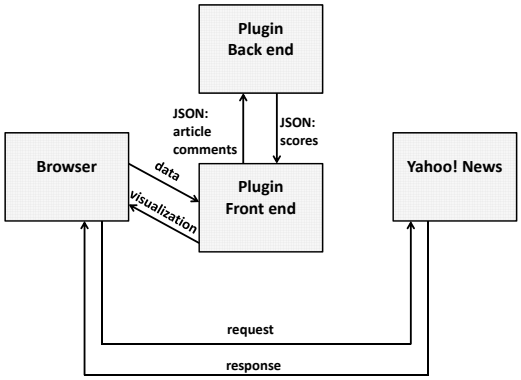


Figure 2: Application infrastructure.

The computational complexity of our approach depends on graph construction time, that is quadratic in number of comments/paragraphs in a given article. In practice, it takes about two seconds to perform precomputation–graph construction and ranks calculation on all article-related data–when user opens an article page, and then the results for any user-specified paragraph/comment are provided immediately.

4 Conclusions

The examples of article texts and the most ranked comments, per paragraph, can be seen in `http://goo.gl/7idNw`. It can be seen that the comments are very related to the paragraphs content and, moreover, they relates the **subject** of a paragraph as well as a **discussion** and **opinions** it arises, beyond the text overlapping. Such performance is provided by a recursive nature of PageRank, where the relationships between comments are iteratively elaborated. Unlike this approach, ranking comments by a (text) similarity to a given paragraph would not retrieve related comments with a different vocabulary.

The plugin implementing our approach is publicly available from `http://goo.gl/To4Rd`.[10] In future, we intend to evaluate our system by comparing it to the other state-of-the-art ranking techniques.

Acknowledgments

Authors thank project students: M. Magaziner, A. Shpilgerman and S. Pinsky for implementing the introduced approach, and I. Vinokur for a technical support of the software. Especial thanks to Dr. Amin Mantrach from Yahoo! Labs, Barcelona, for very constructive and helpful comments.

REFERENCES

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

Dalal, O., Sengemedu, S. H., and Sanyal, S. (2012). Multi-objective ranking of comments on web. In *Proceedings of the 21st international conference on World Wide Web*, pages 419–428.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Hsu, C.-F., Khabiri, E., and Caverlee, J. (2009). Ranking comments on the social web. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, pages 90–97.

Litvak, M. and Matz, L. (2013). Smartnews: Bringing order into comments chaos. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, KDIR '13.

Mihalcea, R. and Tarau, P. (2004). Textrank – bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Salton, G., Singhal, A., Mitra, M., and Buckley, C. (1997). Automatic text structuring and summarization. *Information Processing and Management*, 33(2):193–207.

Salton, G., Yang, C., and Wong, A. (1975). A vector-space model for information retrieval. *Communications of the ACM*, 18.

Sobek, M. (2003). A Survey of Google's PageRank. `http://pr.efactory.de/`.

[9]The original idea of a Topic-Sensitive PageRank was to calculate PageRank for several topics simultaneously, but we don't need to do that until a user is interested in all paragraphs of a given article.

[10]Unzip the archive, press "Load unpacked extension" in "Developer mode" of chrome "Extensions" tool, and choose the unzipped plugin folder.

Tmuse: Lexical Network Exploration

Yannick Chudy[†], **Yann Desalle**[⋆]
Benoît Gaillard[⋆], **Bruno Gaume**[⋆], **Pierre Magistry**[‡], **Emmanuel Navarro**[†]
[⋆] : CLLE-ERSS, University of Toulouse,
[†] : IRIT, University of Toulouse,
[‡] : INRIA, University of Paris 7

Abstract

We demonstrate an online application to explore lexical networks. Tmuse displays a 3D interactive graph of similar words, whose layout is based on the *proxemy* between vertices of synonymy and translation networks. Semantic themes of words related to a query are outlined, and projected across languages. The application is useful as, for example, a writing assistance. It is available, online, for Mandarin Chinese, English and French, as well as the corresponding language pairs, and can easily be fitted to new resources.

1 Introduction

Although Natural Language Processing applications can not fully replace human abilities to write, read and understand texts, they have proven to be a great assistance for many linguistic tasks. For example, if state of the art Machine Translation (MT) productions can not be considered as accomplished texts, a great variety of Computer Assisted Translation (CAT) software (Trados, OmegaT) help translators work faster, more accurately and consistently. Many writing and reading situations require the extensive use of dictionaries to find or confirm the exact meaning of words to be used in a specific context with specific connotations.

The issue is multiplied when writers manipulate a language for which they are not native. Online dictionaries and thesauri provide the necessary assistance (Linguee, Wordreference, WordNet, Merriam-Webster...), but they can be difficult to make sense of, because, although they provide definitions, subsenses, usage and lists of synonyms, the relations between these informations (semantic similarity of the various synonyms, subsenses) are not directly presented to the user.

Tmuse displays the relations between words that have a meaning similar to that of a query, as shown in Fig. 1. Rather than mere lists, as most dictionaries do, or flat networks of relations [1], Tmuse displays emergent clusters, as semantic fields, and lays out the closest words according to their relative semantic similarity, in a 3D, visually ergonomic presentation. As explained in Navarro et al. (2011), displaying results as a few clusters rather than as long lists is ergonomic, because users find zones of interest at a glance. Beyond the monolingual usage, Tmuse can also help in the cross-lingual case, for instance when the source language is not the user's native language. Tmuse displays the various semantic fields associated with a query word in the source language. Since semantic fields associated with words are not necessarily similar across languages, users might not be familiar with this local semantic structure. Each source semantic field is translated into the target language (user's native language in this example) by a set of target words that are semantically consistent with the source semantic field. This process is called *Proxlation*, as it uses both translation and proxemy in the target language. So, users can grasp, through a set of native language words, the actual meaning of each displayed semantic field, even if it does not constitute a semantic unit of their native language.

The Tmuse exploration tool is based on synonymy graphs and translation bigraphs. The prototype is available online [2] for general English, Mandarin and French, as well as the corresponding language pairs. It can readily be extended to more languages or more specific terminologies, provided the necessary resources.

1. For example : `homepages.inf.ed.ac.uk/adubey/software/wnbrowser/index.html`
2. `www.naviprox.net/tmuse`

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 41–44,
Nagoya, Japan, 14-18 October 2013.

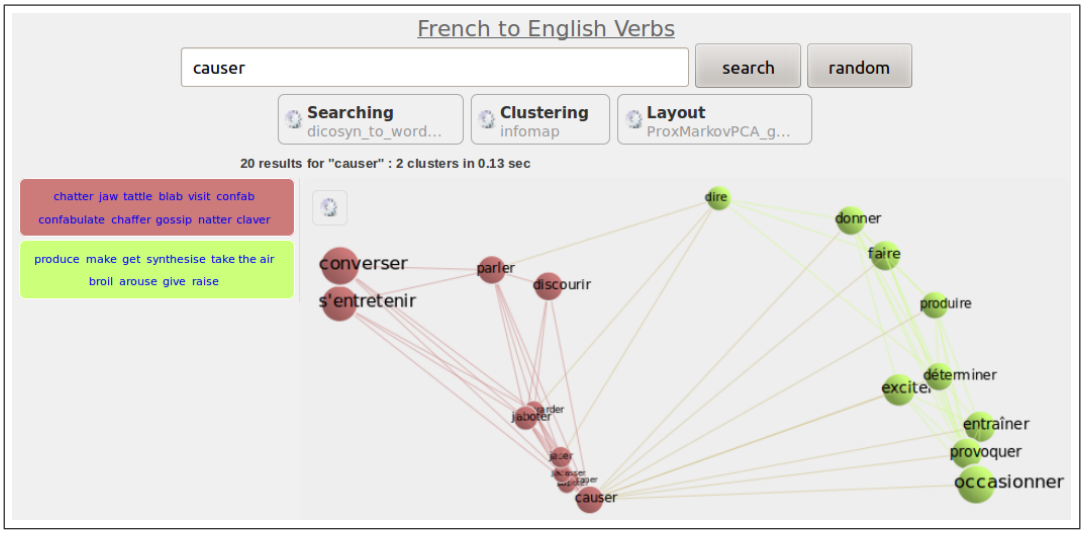


FIGURE 1 – Example of Tmuse translated Semantic Fields

2 Demonstrated system components

This section overviews the chain of semantic processing components that constitute the backbone of Tmuse, and details resource modelling and theoretical principles underlying each step. Tmuse processes query words to provide a topological description of their semantic landscape. On the basis of a synonymy resource, it first finds a number of *proxemes*, i.e. a set of words that are semantically close to the query. This set of proxemes is then represented as a graph, in 3D. The graph's layout respects the semantic proximity of its vertices, and communities of specifically close words are highlighted. In the bilingual case, proxemy-based sets of translations of these communities are presented in relation with the graph clusters.

2.1 Resource modelling

Synonymy Resources are modelled as graphs $G = (V, E)$, where V is the set of vertices. It corresponds to the resource's lemmas, which are unique. Indeed, the several subsenses of a form are not represented by several vertices, but with the various synonymy connections of the single vertex. A pair of vertices (a, b) defines an edge ($(a, b) \in E$) if and only if a is declared synonymous with b in the resource. The resulting graph is then made reflexive and symmetric.

Wordnet and thesaurus type resources are modelled like the synonymy resources, but edges are drawn within synsets : two lemmas are linked if they belong to at least one common synset. Instead of synsets, the leaves of the thesaurus tree of classes are used.

Translation Resources Translation resources are modelled as bigraphs $B = ((V_1, V_2), E)$, where $V_1 \cup V_2 = V$ is the set of vertices of the graph, V_1 representing the source language lemmas, V_2 the target language lemmas ($V_1 \cap V_2 = \emptyset$). $E \subset V_1 \times V_2$ is the set of edges, which only link lemmas of V_1 to lemmas of V_2 if V_2 is declared translation of V_1 in the resource.

2.2 Word query processing pipeline

2.2.1 Subgraph extraction by random walk

Tmuse uses the Prox algorithm to fetch the N closest words, in the synonymy network, of the query : the *proxemes*.

The Prox algorithm : In a graph $G = (V, E)$, the *Proxemy* of a word to the query is the probability of reaching it by a short random walk of t time steps (Gaume, 2004). Such a random walk can be defined by a Markov chain on V with a $|V| \times |V|$ transition matrix $[G]$ (Bollobas, 2002) :

$$[G] = (g_{u,v})_{u,v \in V},$$

with

$$g_{u,v} = \begin{cases} \frac{1}{|N_G^u|} & \textbf{if } \{u, v\} \in E, \\ 0 & \textbf{else.} \end{cases}$$

$|N_G^u|$ is the degree of vertex u in G. Let $P_G^t(u \rightsquigarrow v)$ be the probability of a walker starting on vertex u to reach a vertex v after t steps :

$$P_G^t(u \rightsquigarrow v) = ([G]^t)_{u,v}$$

The starting point of a random walk can be generalised to a probability distribution P_0. In that case :

$$P_G^t(P_0 \rightsquigarrow v) = (P_0.[G]^t)_v$$

We call *proxemes* of an initial probability distribution P_0, the vertices of the graph associated with their proxemy. The best proxemes are the ones with the highest proxemy. As shown in Gaume and Mathieu (2007), the "PageRank" approach, biased with a damping factor to the starting point (sometimes called "personalised PageRank"), results in dynamics similar to such short random walks. Its computational cost is however much higher, as it necessitates the knowledge of the whole graph, whereas short random walks only require knowledge of immediate neighbours, at each time-step.

Subgraph Tmuse fetches the N best proxemes of the query. The subgraph induced by this set in the synonymy graph is displayed. In other words the displayed subgraph is made of these proxemes and all the synonymy links they have between themselves.

2.2.2 Graph clustering

State of the art community detection algorithms (Lancichinetti and Fortunato, 2009) are used to partition the extracted subgraph into several semantic zones, materialized on the interface by several colours. We use for instance the Infomap clustering algorithm (Rosvall and Bergstrom, 2008).

2.2.3 Layout

The extracted subgraph, with colour-coded clusters, is displayed in an interactive 3D representation. Vertices are labeled with their lemmas. Their relative positions respect their semantic proximity, thanks to the following algorithm (Gaume, 2008) :

Each vertex u_0 of the subgraph is associated with a proxemy vector P_{u_0} of $|V|$ dimensions : the v coordinate of P_{u_0} is the proxemy between u_0 and the v vertex of the graph : $P_G^t(u_0 \rightsquigarrow v)$.

This models a set of N location in an $|V|$-dimensional space. Two semantically similar words will have similar proxemy vectors and will therefore lie close to each other.

Principal Components Analysis projects this $N \times |V|$- dimensional data set onto $N \times 3$- dimensional data set, that optimally represents its structure.

Clusters computed by the clustering component 2.2.2 are materialised in the layout by different vertex colours. Vertex labels are listed, cluster by cluster, alongside the 3D representation.

2.2.4 Bilingual exploration by *Proxlation*

Like in the monolingual case, the 3D representation describes the semantic topology around the query, with source language words as vertex labels, and the corresponding clusters.

However, the side lists are labelled with the K best translations of the source language clusters, called *proxlations*, and chosen in two steps.

First, Tmuse lists all the translations of all the vertices of the source cluster. Each (target language) translation is weighted according to the number of words of the cluster it translates. This constitutes P_0, a probability distribution vector from which a random walk is launched, on the target language synonymy graph.

The K best proxemes of P_0 are selected as the proxlations of the source language cluster, and appear in the list of the corresponding colour. Selecting proxlations instead of direct translations enables Tmuse to filter out words whose meaning is not consistent with the cluster's semantic theme.

3 System functionalities

3.1 Basic usage

The typical use case of Tmuse is similar to an information retrieval scenario : the user queries a word, and the application replies with relevant lexical semantic information. As described in 2, the application displays a 3D interactive subgraph and lists of related words. Users can make the subgraph turn, zoom on zones of interest, focus on one "semantic field", highlight the actual synonymy links of any word. They can also explore specific meanings by double clicking on words, which launches a new query with this new word. What the interface displays depends on several parameters (number of proxemes, synonymy only, clustering algorithm and layout) that the interested or more advanced user can set.

3.2 Bilingual exploration

In a bilingual mode, users query a word in the source language, and the application displays both the semantic landscape of the query in the source

3. www.atilf.fr
4. www.gutenberg.org/ebooks/10681
5. dict.revised.moe.edu.tw/
6. cc-cedict.org/wiki/

Name	Language	Type	Reference
Dicosyn	French	synonyms	ATILF & IBM [3]
Wiktionary	French - English	translations	Sajous et al. (2010)
Princetown Wordnet	English	wordnet	Fellbaum (1998)
Roget	English	thesaurus	Gutenberg Projet [4]
Cilin	Mandarin	thesaurus	Mei et al. (1984)
Chinese Wordnet	Mandarin	wordnet	Huang and Hsieh (2010)
MOE dictionary	Mandarin	synonyms	R.O.C Ministry of Education [5]
CEDict	Mandarin - English	translation	dictionary under C.C licence [6]
Authors data	Mandarin - French	translation	own data, to be released soon

TABLE 1 – Resources for Tmuse exploration

language and the proxlations into the target language of each semantic field. Semantic fields are represented by coloured clusters of the extracted source subgraph, their proxlations are displayed in the side lists, with matching colours. Upon clicking on a target word, a new query is launched with the clicked word, on the reverse language pair.

3.3 Resource variations

Users can change the resource of the monolingual application, and also, independently, the source, target and translation resources of the bilingual application. Resources are detailed in Table 1. Results sometimes greatly vary with resource variation. See Gaillard et al. (2011) for an analysis of the similarity of the semantic structure of lexical graphs. Beyond words, Tmuse could be applied to phrases. The computational cost wouldn't be much higher, but one would not only need a phrase translation dictionary, but also phrase synonymy dictionaries. Building such resources could be done by statistical corpus analysis, which would require significant experimental work.

References

Bela Bollobas. 2002. *Modern Graph Theory*. Springer-Verlag New York Inc.

Christiane Fellbaum, editor. 1998. *WordNet : An Electronic Lexical Database*. MIT Press.

Benoit Gaillard, Bruno Gaume, and Emmanuel Navarro. 2011. Invariant and variability of synonymy networks : Self mediated agreement by confluence. In *Proc. of the The 49th ACL-HLT Annual Meeting : 6th TextGraphs workshop*, Portland, Oregon.

Bruno Gaume and Fabien Mathieu. 2007. PageRank Induced Topology for Real-World Networks. *Complex Systems*, to appear :(on line).

Bruno Gaume. 2004. Balades Aléatoires dans les Petits Mondes Lexicaux. *I3 : Information Interaction Intelligence*, 4(2).

Bruno Gaume. 2008. Mapping the form of meaning in small worlds. *Journal of Intelligent Systems*, 23(7) :848–862.

Chu-Ren Huang and Shu-Kai Hsieh. 2010. Infrastructure for cross-lingual knowledge representation - towards multilingualism in linguistic studies. *Taiwan NSC-granted Research Project (NSC 96-2411-H-003-061-MY3)*.

A. Lancichinetti and S. Fortunato. 2009. Community detection algorithms : A comparative analysis. *Phys. Rev. E*, 80(5) :056117.

Jia-Ju Mei, Yi ming Zheng, Yun-Qi Gao, and Hung-Xian Yin. 1984. *TongYiCi CiLin*. Commercial Press, Shanghai.

Emmanuel Navarro, Yannick Chudy, Bruno Gaume, Guillaume Cabanac, and Karen Pinel-Sauvagnat. 2011. Kodex ou comment organiser les résultats d'une recherche d'information par détection de communautés sur un graphe biparti ? In *CORIA'11, Avignon*, pages 25–40. ARIA, mars.

M. Rosvall and C. T. Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4) :1118–1123.

Franck Sajous, Emmanuel Navarro, Bruno Gaume, Laurent Prévot, and Yannick Chudy. 2010. Semi-automatic endogenous enrichment of collaboratively constructed lexical resources : Piggybacking onto wiktionary. In Hrafn Loftsson, Eiríkur Rögnvaldsson, and Sigrún Helgadóttir, editors, *Advances in NLP*, volume 6233 of *LNCS*, pages 332–344. Springer Berlin / Heidelberg.

Appendix : Technical details

Tmuse is available online [7]. It runs on a server, hosted in Toulouse, with 4Gb RAM and 3.4 Ghz CPU. The client browser only runs the 3D display. The main memory costs stem from the size and number of the graphs involved. The loaded 27 graphs use 700Mb of memory. As walks lengthen, the number of probabilities to compute and store exponentially increases, so we set a limit to $t = 10$. Clustering algorithms are well-optimised, and applied to only small subgraphs.

7. `www.naviprox.net/tmuse`

WISDOM2013: A Large-scale Web Information Analysis System

Masahiro Tanaka Stijn De Saeger* Kiyonori Ohtake Chikara Hashimoto
Makoto Hijiya Hideaki Fujii Kentaro Torisawa
National Institute of Information and Communications Technology (NICT)
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 219-0289, Japan
{mtnk, stijn, kiyonori.ohtake, ch, hijiya, h-fujii, torisawa}@nict.go.jp

Abstract

We demonstrate our large-scale web information analysis system called WISDOM2013, which consists of several deep semantic analysis systems such as a factoid QA, a non-factoid QA and a sentiment analyzer, and a software platform on which its semantic analysis systems can be applied to a billion-page-scale web archive. The software platform has an extendable architecture, and we are planning to enhance WISDOM2013 in the future by adding more semantic analysis systems and inference mechanisms.

1 Introduction

The range of questions is unlimited that humans can pose, and web texts are a valuable information source for finding a comprehensive list of answers, which may include "unknown unknowns" in the infamous words of D. H. Rumsfeld: things that "we don't know we don't know" (Torisawa et al., 2010). However, current commercial search engines are not an effective tool for finding such answers. For instance, even though deforestation is a serious and widely discussed problem, no exhaustive list of answers exists to the question: "What are the consequences if deforestation continues?" We may encounter serious unknown or unexpected consequences in the future. Many documents on the web describe its possible consequences, but only a small portion can be discovered using commercial search engines, because they just provide a huge number of documents that users have to read. Our ultimate goal is to solve such problems by developing deep semantic analysis technologies, which can provide a list of the possible consequences of deforestation, for instance, and a software platform on which semantic analysis technologies can be applied to a billion-page-scale web archive.

We introduce WISDOM2013, our large-scale web information analysis system that consists of deep semantic analysis systems, including a factoid QA and a non-factoid QA, such as a what-happens-if QA, which answers "What happens if deforestation continues?" and sentiment/information sender analysis. We also introduce the underlying architecture of the software platform, which is designed to process/store two billion web documents and works as a common software platform for various semantic analyses.

NICT previously proposed an information analysis system called WISDOM[1](Akamine et al., 2009), which is a predecessor of WISDOM2013. But the source of its analyses were limited to 100 million web pages, and it did not provide QA services. In addition, the depth and the scale of its semantic analysis was quite restricted because it performs the semantic analysis online after receiving user requests. In contrast, most semantic processing that runs on WISDOM2013 is done offline. WISDOM2013 immediately analyzes each web document after it is crawled and can store the basic analysis results for billions of documents owing to its software platform. Therefore, we can drastically improve the breadth and depth of semantic analyses.

2 Script Outline

In this section, we introduce the major features that we will demonstrate. They exploit the common fundamental analysis results, which are produced by the underlying architecture for large-scale analysis as shown in Section 3.

*Current address: Nuance Communications, Inc., Germany. stijn.desaeger (at) nuance.com

[1]http://wisdom-nict.jp/

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 45–48,
Nagoya, Japan, 14-18 October 2013.

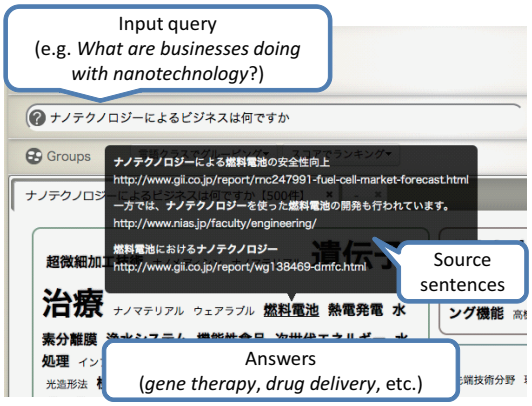


Figure 1: WISDOM2013 interface: factoid QA.

2.1 Factoid QA

Figure 1 shows WISDOM2013's web browser interface. WISDOM2013 takes a question in natural language and returns answers. For example, given question *"What are businesses doing with nanotechnology?"* (ナノテクノロジーによるビジネスは何ですか), WISDOM2013 returns hundreds of answers, such as *gene therapy* (遺伝子治療), *drug delivery* (ドラッグデリバリー), and *artificial joints* (人工関節) and displays them in clusters of semantically related terms. Users can click on each answer to see the original sentence and document from which the answer was extracted.

The system extracts such patterns as "X are businesses doing with Y" in questions and automatically paraphrases the extracted patterns into many synonymous patterns (De Saeger et al., 2009). Those patterns are matched against the web texts using specially designed indexes.

Note that we aim to provide a wide range of answers to user questions, unlike such traditional factoid QAs as IBM's Watson for the Jeopardy! game show (Ferrucci et al., 2010), and suggest unexpected information to users, in other words, "unknown unknowns" (Torisawa et al., 2010). We expect that such unknown unknowns broaden thought and trigger proper decision makings in users.

This technology is an extension of the one used in our voice-activated open domain question answering system (Varga et al., 2011). When it is given a question, "What part of Japan was previously hit by tsunamis?", it found that the Sendai plain, which was devastated by a huge tsunami in the Great East Japan Earthquake in 2011, was also hit 1,000 years ago by a huge tsunami; tsunamis of similar scale are expected to hit again in the future. The system found this answer *from web pages posted before* the Great East Japan Earthquake. For a large number of victims of the 2011 tsunami, this is an example of an "unknown unknown" (or at least relatively unknown facts), and if it had been more widely circulated, lives might have been saved. WISDOM2013 will give chances for many users in the future to discover such *unknown unknowns*.

2.2 What-happens-if QA

When an input question follows the "What happens if X" pattern, WISDOM2013 invokes a special type of QA system, which we call *What-happens-if* QA, and gives the result as a directed graph (Fig. 2). The graph represents the causal chains initiated by the event described in the question. If the given question is *"What happens if deforestation continues?"* (森林破壊が続くとどうなる？), then WISDOM2013 gives a graph that includes the causal chains initiated by the event, "deforestation continues". For instance, the graph contains the following causal chain: "deforestation continues" → "global warming progresses" → "sea temperature rises" → "Vibrio parahaemolyticus swells."[2].

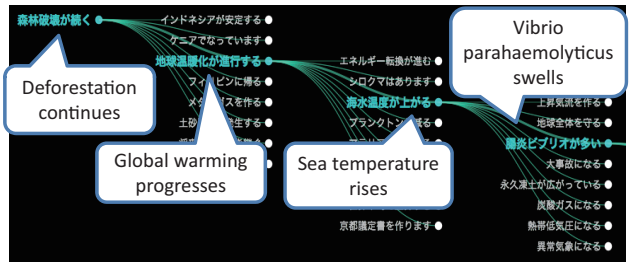


Figure 2: *What-happens-if* QA

Of course, it is debatable whether such causal chains or *future scenarios* will actually happen, and many scenarios are unlikely to become true. Our aim is to provide users the *big picture* of the future concerns of a given question, which is unlikely to be covered by journalism or mass media. We expect that careful examination of such future scenarios will lead to better decision making and preparation for potential and unforeseen risks.

Note that the causalities among nodes are acquired by our previous method (Hashimoto et al.,

[2] An article in *Nature Climate Change* reported that Vibrio infections are caused by global warming in the Baltic Sea (Craig Baker-Austin et al., Nature Climate Change, Vol. 3, pp, 73–77 (2013))

2012) from a large body of web texts. Each single causal relation between two nodes is extracted from a single web page, but a chain of causalities is obtained by combining those extracted causal relations and represents the information scattered over many web pages. In this sense, *what-happens-if* QA involves an certain inference process and enables users to explore possible social scenarios by chaining/combining statements from different documents. In other words, this feature creates awareness of hypothetical future scenarios that are not actually written in any document.

2.3 Sender/Sentiment Analysis

WISDOM2013 can also show the results of sentiment analysis for a given topic or answers for factoid QAs. The amount of positive/negative information based on sentiment analysis is shown in charts to elucidate trends for users (Fig. 3). The results are classified based on the types of senders of the information source page. These functionalities were inherited from WISDOM, WISDOM2013' s predecessor (Akamine et al., 2009). For instance, we can check the reputation of treatments for atopic diseases by applying sentiment analysis to the answers to *"What works for atopy?"* (アトピーに効くのは何ですか) and use the results as clues for determining the treatment' s reliability or uncovering side-effects. In our demonstration, we show that companies post the most positive opinions concerning nutritional supplements that are supposedly effective against atopy. Users might infer that the companies are exaggerating the drug's positive qualities even though much positive information is available about them. In extreme cases, users may question the effectiveness of such supplements or associate side-effects with them.

3 Software Platform

In this section, we describe the architecture of the underlying software platform, which consists of two stages of data processing. Fig. 4 shows the first stage for fundamental analyses and archiving. The fundamental analysis results are designed to be shared by a wide variety of application-oriented analyses in the second stage.

After the crawler collects web documents, fundamental analyses are applied to them, which include document structure analysis, dependency

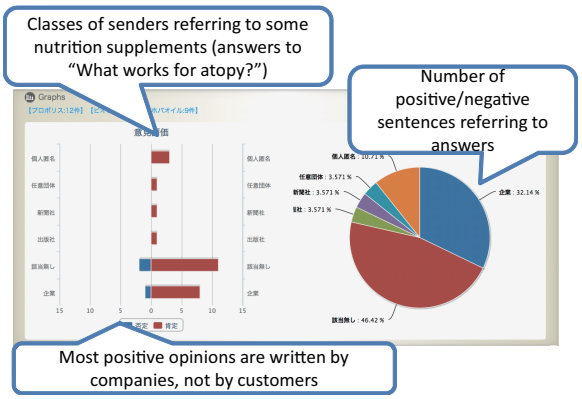


Figure 3: Sender/sentiment analysis

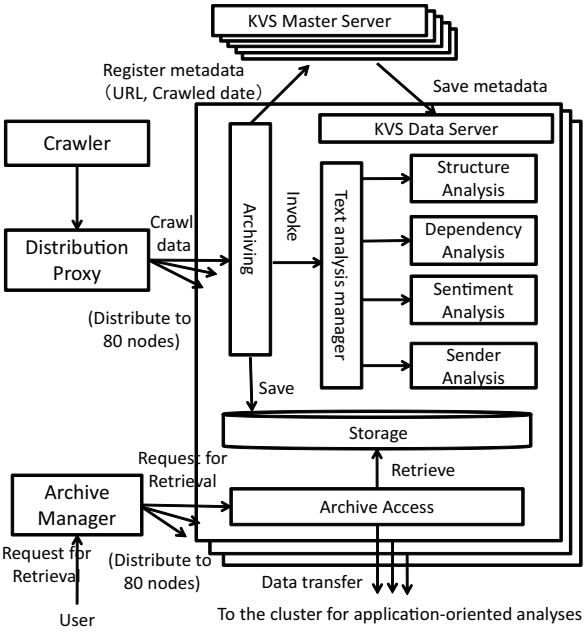


Figure 4: Fundamental analyses and archiving.

analysis, sender analysis, and sentiment analysis. The analyses need to process more than ten million documents daily collected by the crawler. To manage such metadata as URL, the crawled date, and the processing status of each document, we adopted a distributed key-value store (KVS).

In the second stage, more application-oriented analyses are performed based on the fundamental analysis results. For factoid QAs, the preprocessor extracts patterns of phrases that indicate relationships between terms and indexes them. The preprocessor for the what-happens-if QA extracts causal relations and indexes them. Both QAs rely on structure analysis and dependency analysis, both of which are produced in the first stage. Sender/Sentiment are also indexed for the interactive analysis described in Section 2.3. A full text

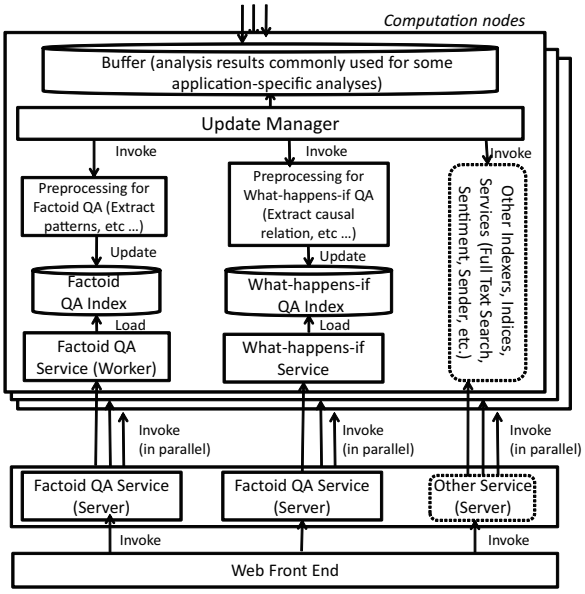


Figure 5: Application-oriented analyses.

search also becomes available based on indexing in this stage. Fig. 5 shows an overview of the process. The update manager transfers the fundamental analysis results to the distributed computation nodes. Due to computational load and data size that exceeds the storage amount of a single node, preprocessing including indexing for some analyses runs on 40 nodes in parallel. The indices for the analyses are used by *access services*, which provide APIs to access the indices. The distributed services are called *worker* services. A *server* service receives a request from the GUI, sends it to all *worker* services in parallel, and aggregates their results. The *server* service also eliminates duplicated results and ranks them. The extensible software platform allows us to add new preprocessors, indices, and services.

4 Conclusion

In this paper, we introduced the major features of WISDOM2013 and described its software platform. We are planning to extend it in the future by adding more semantic analysis systems, such as Why QA (Oh et al., 2013) and inference mechanisms (Tsuchida et al., 2011). We also plan to introduce WISDOM2013 as infrastructure for a counter disaster information analysis system (Ohtake et al., 2013), which we are developing to organize information extracted from tweets after disasters (Varga et al., 2013). WISDOM2013's software and service are scheduled to be made public in 2014.

References

Susumu Akamine, Daisuke Kawahara, Yoshikiyo Kato, Tetsuji Nakagawa, Kentaro Inui, Sadao Kurohashi, and Yutaka Kidawara. 2009. WISDOM: A web information credibility analysis system. In *ACL/AFNLP 2009 (Software Demonstrations)*, pages 1–4.

Stijn De Saeger, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *ICDM'09*, pages 764–769.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong-Hoon Oh, and Jun'ichi Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *EMNLP-CoNLL 2012*, pages 619–630.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-question answering using intra- and inter-sentential causal relations. In *ACL 2013*, pages 1733–1743.

Kiyonori Ohtake, Jun Goto, Stijn De Saeger, Kentaro Torisawa, Junta Mizuno, and Kentaro Inui. 2013. Nict disaster information analysis system. In *IJCNLP 2013 (Demonstration Track)*.

Kentaro Torisawa, Stijn de Saeger, Jun'ichi Kazama, Asuka Sumida, Daisuke Noguchi, Yasunari Kakizawa, Masaki Murata, Kow Kuroda, and Ichiro Yamada. 2010. Organizing the web's information explosion to discover unknown unknowns. *New Generation Computing (Special Issue on Information Explosion)*, 28(3):217–236.

Masaaki Tsuchida, Kentaro Torisawa, Stijn De Saeger, Jong Hoon Oh, Jun'ichi Kazama, Chikara Hashimoto, and Hayato Ohwada. 2011. Toward finding semantic relations not written in a single sentence: An inference method using auto-discovered rules. In *IJCNLP 2011*, pages 902–910.

István Varga, Kiyonori Ohtake, Kentaro Torisawa, Stijn De Saeger, Teruhisa Misu, Shigeki Matsuda, and Jun'ichi Kazama. 2011. Similarity based language model construction for voice activated open-domain question answering. In *IJCNLP 2011*, pages 536–544.

István Varga, Motoki Sano, Kentaro Torisawa, Chikara Hashimoto, Kiyonori Ohtake, Takao Kawai, Jong-Hoon Oh, and Stijn De Saeger. 2013. Aid is out there: Looking for help from tweets during a large scale disaster. In *ACL 2013*, pages 1619–1629.

Author Index