The Excitement Open Platform for Textual Inferences

Bernardo Magnini*, Roberto Zanoli*, Ido Dagan°, Kathrin Eichler†, Günter Neumann†, Tae-Gil Noh‡, Sebastian Pado‡, Asher Stern°, Omer Levy°

*FBK (magnini|zanoli@fbk.eu) ‡Heidelberg, Stuttgart Univ. (pado|noh@cl.uni-heidelberg.de)

†DFKI (neumann|eichler@dfki.de) °Bar Ilan University (dagan|sterna3|omerlevy@cs.biu.ac.il)

Abstract

This paper presents the Excitement Open Platform (EOP), a generic architecture and a comprehensive implementation for textual inference in multiple languages. The platform includes state-of-art algorithms, a large number of knowledge resources, and facilities for experimenting and testing innovative approaches. The EOP is distributed as an open source software.

1 Introduction

In the last decade textual entailment (Dagan et al., 2009) has been a very active topic in Computational Linguistics, providing a unifying framework for textual inference. Several evaluation exercises have been organized around Recognizing Textual Entailment (RTE) challenges and many methodologies, algorithms and knowledge resources have been proposed to address the task. However, research in textual entailment is still fragmented and there is no unifying algorithmic framework nor software architecture.

In this paper, we present the Excitement Open Platform (EOP), a generic architecture and a comprehensive implementation for multilingual textual inference which we make available to the scientific and technological communities. To a large extent, the idea is to follow the successful experience of the Moses open source platform (Koehn et al., 2007) in Machine Translation, which has made a substantial impact on research in that field. The EOP is the result of a two-year coordinated work under the international project EXCITEMENT.[1] A consortium of four academic partners has defined the EOP architectural specifications, implemented the functional interfaces of the EOP components, imported existing entailment engines into the EOP and finally designed and implemented a rich environment to support open source distribution.

The goal of the platform is to provide functionality for the automatic identification of entailment relations among texts. The EOP is based on a modular architecture with a particular focus on *language-independent* algorithms. It allows developers and users to combine linguistic pipelines, entailment algorithms and linguistic resources within and across languages with as little effort as possible. For example, different entailment decision approaches can share the same resources and the same subcomponents in the platform. A classification-based algorithm can use the distance component of an edit-distance based entailment decision approach, and two different approaches can use the same set of knowledge resources. Moreover, the platform has various multilingual components for languages like English, German and Italian. The result is an ideal software environment for experimenting and testing innovative approaches for textual inferences. The EOP is distributed as an open source software[2] and its use is open both to users interested in using inference in applications and to developers willing to extend the current functionalities.

The paper is structured as follows. Section 2 presents the platform architecture, highlighting how the EOP component-based approach favors interoperability. Section 3 provides a picture of the current population of the EOP in terms of both entailment algorithms and knowledge resources. Section 4 introduces expected use cases of the platform. Finally, Section 5 presents the main features of the open source package.

2 Architecture

The EOP platform takes as input two text portions, the first called the Text (abbreviated with T), the second called the Hypothesis (abbreviated with H).

[1] http://www.excitement-project.eu

[2] http://hltfbk.github.io/Excitement-Open-Platform/

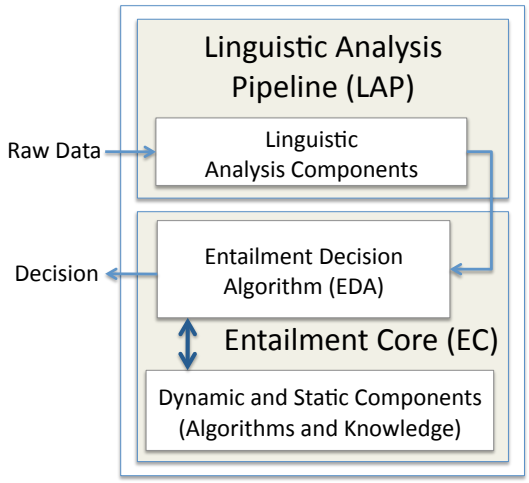

Figure 1: EOP architecture

The output is an entailment judgement, either "Entailment" if T entails H, or "NonEntailment" if the relation does not hold. A confidence score for the decision is also returned in both cases.

The EOP architecture (Padó et al., 2014) is based on the concept of modularization with pluggable and replaceable components to enable extension and customization. The overall structure is shown in Figure 1 and consists of two main parts. The Linguistic Analysis Pipeline (LAP) is a series of linguistic annotation components. The Entailment Core (EC) performs the actual entailment recognition. This separation ensures that (a) the components in the EC only rely on linguistic analysis in well-defined ways and (b) the LAP and EC can be run independently of each other. Configuration files are the principal means of configuring the EOP. In the rest of this section we first provide an introduction to the LAP, then we move to the EC and finally describe the configuration files.

2.1 Linguistic Analysis Pipeline (LAP)

The Linguistic Analysis Pipeline is a collection of annotation components for Natural Language Processing (NLP) based on the Apache UIMA framework.[3] Annotations range from tokenization to part of speech tagging, chunking, Named Entity Recognition and parsing. The adoption of UIMA enables interoperability among components (e.g., substitution of one parser by another one) while ensuring language independence. Input and output of the components are represented in an extended version of the DKPro type system based on UIMA Common Analysis Structure (CAS) (Gurevych et al., 2007; Noh and Padó, 2013).

[3] http://uima.apache.org/

2.2 Entailment Core (EC)

The Entailment Core performs the actual entailment recognition based on the preprocessed text made by the Linguistic Analysis Pipeline. It consists of one or more Entailment Decision Algorithms (EDAs) and zero or more subordinate components. An EDA takes an entailment decision (i.e., "entailment" or "no entailment") while components provide static and dynamic information for the EDA.

Entailment Decision Algorithms are at the top level in the EC. They compute an entailment decision for a given Text/Hypothesis (T/H) pair, and can use components that provide standardized algorithms or knowledge resources. The EOP ships with several EDAs (cf. Section 3).

Scoring Components accept a Text/Hypothesis pair as an input, and return a vector of scores. Their output can be used directly to build minimal classifier-based EDAs forming complete RTE systems. An extended version of these components are the *Distance Components* that can produce normalized and unnormalized distance/similarity values in addition to the score vector.

Annotation Components can be used to add different annotations to the Text/Hypothesis pairs. An example of such a type of component is one that produces word or phrase alignments between the Text and the Hypothesis.

Lexical Knowledge Components describe semantic relationships between words. In the EOP, this knowledge is represented as directed rules made up of two word–POS pairs, where the LHS (left-hand side) entails the RHS (right-hand side), e.g., $(shooting\_star, Noun) \implies (meteorite, Noun)$. Lexical Knowledge Components provide an interface that allows for (a) listing all RHS for a given LHS; (b) listing all LHS for a given RHS; and (c) checking for an entailment relation for a given LHS–RHS pair. The interface also wraps all major lexical knowledge sources currently used in RTE research, including manually constructed ontologies like WordNet, and encyclopedic resources like Wikipedia.

Syntactic Knowledge Components capture entailment relationships between syntactic and

lexical-syntactic expressions. We represent such relationships by entailment rules that link (optionally lexicalized) dependency tree fragments that can contain variables as nodes. For example, the rule *fall of X* $\Longrightarrow$ *X falls*, or *X sells Y to Z* $\Longrightarrow$ *Z buys Y from X* express general paraphrasing patterns at the predicate-argument level that cannot be captured by purely lexical rules. Formally, each syntactic rule consists of two dependency tree fragments plus a mapping from the variables of the LHS tree to the variables of the RHS tree.[4]

2.3 Configuration Files

The EC components can be combined into actual *inference engines* through configuration files which contain information to build a complete inference engine. A configuration file completely describes an experiment. For example, it specifies the resources that the selected EDA has to use and the data set to be analysed. The LAP needed for data set preprocessing is another parameter that can be configured too. The platform ships with a set of predefined configuration files accompanied by supporting documentation.

3 Entailment Algorithms and Resources

This section provides a description of the Entailment Algorithms and Knowledge Resources that are distributed with the EOP.

3.1 Entailment Algorithms

The current version of the EOP platform ships with three EDAs corresponding to three different approaches to RTE: an EDA based on transformations between T and H, an EDA based on edit distance algorithms, and a classification based EDA using features extracted from T and H.

Transformation-based EDA applies a sequence of transformations on T with the goal of making it identical to H. If each transformation preserves (fully or partially) the meaning of the original text, then it can be concluded that the modified text (which is actually the Hypothesis) can be inferred from the original one. Consider the following simple example where the text is "The boy was located by the police" and the Hypothesis is "The child was found by the police". Two transformations for "boy" $\rightarrow$ "child" and "located" $\rightarrow$ "found" do the job.

[4] Variables of the LHS may also map to null, when material of the LHS must be present but is deleted in the inference step.

In the EOP we include a transformation based inference system that adopts the knowledge based transformations of Bar-Haim et al. (2007), while incorporating a probabilistic model to estimate transformation confidences. In addition, it includes a search algorithm which finds an optimal sequence of transformations for any given T/H pair (Stern et al., 2012).

Edit distance EDA involves using algorithms casting textual entailment as the problem of mapping the whole content of T into the content of H. Mappings are performed as sequences of editing operations (i.e., insertion, deletion and substitution) on text portions needed to transform T into H, where each edit operation has a cost associated with it. The underlying intuition is that the probability of an entailment relation between T and H is related to the distance between them; see Kouylekov and Magnini (2005) for a comprehensive experimental study.

Classification based EDA uses a Maximum Entropy classifier to combine the outcomes of several scoring functions and to learn a classification model for recognizing entailment. The scoring functions extract a number of features at various linguistic levels (bag-of-words, syntactic dependencies, semantic dependencies, named entities). The approach was thoroughly described in Wang and Neumann (2007).

3.2 Knowledge Resources

As described in Section 2.2, knowledge resources are crucial to recognize cases where T and H use different textual expressions (words, phrases) while preserving entailment. The EOP platform includes a wide range of knowledge resources, including lexical and syntactic resources, where some of them are grabbed from manual resources, like dictionaries, while others are learned automatically. Many EOP resources are inherited from pre-existing RTE systems migrated into the EOP platform, but now use the same interfaces, which makes them accessible in a uniform fashion.

There are about two dozen lexical (e.g. wordnets) and syntactic resources for three languages (i.e. English, Italian and German). However, since there is still a clear predominance of English resources, the platform includes lexical and syntactic knowledge mining tools to bootstrap resources from corpora, both for other languages and

EDA	Accuracy / F1
Transformation-based English RTE-3	67.13%
Transformation-based English RTE-6	49.55%
Edit-Distance English RTE-3	64.38%
Edit-Distance German RTE-3	59.88%
Edit-Distance Italian RTE-3	63.50%
Classification-based English RTE-3	65.25%
Classification-based German RTE-3	63.75%
Median of RTE-3 (English) submissions	61.75%
Median of RTE-6 (English) submissions	33.72%

Table 1: EDAs results

for specific domains. Particularly, the EOP platform includes a language independent tool to build Wikipedia resources (Shnarch et al., 2009), as well as a language-independent framework for building distributional similarity resources like DIRT (Lin and Pantel, 2002) and Lin similarity(Lin, 1998).

3.3 EOP Evaluation

Results for the three EDAs included in the EOP platform are reported in Table 1. Each line represents an EDA, the language and the dataset on which the EDA was evaluated. For brevity, we omit here the knowledge resources used for each EDA, even though knowledge configuration clearly affects performance. The evaluations were performed on RTE-3 dataset (Giampiccolo et al., 2007), where the goal is to maximize accuracy. We (manually) translated it to German and Italian for evaluations: in both cases the results fix a reference for the two languages. The two new datasets for German and English are available both as part of the EOP distribution and independently[5]. The transformation-based EDA was also evaluated on RTE-6 dataset (Bentivogli et al., 2010), in which the goal is to maximize the F1 measure.

The results of the included EDAs are higher than median values of participated systems in RTE-3, and they are competing with state-of-the-arts in RTE-6 results. To the best of our knowledge, the results of the EDAs as provided in the platform are the highest among those available as open source systems for the community.

4 Use Cases

We see four primary use cases for the EOP. Their requirements were reflected in our design choices.

Use Case 1: Applied Textual Entailment. This category covers users who are not interested in the details of RTE but who are interested in an NLP task in which textual entailment can take over part of or all of the semantic processing, such as Question Answering or Intelligent Tutoring. Such users require a system that is as easy to deploy as possible, which motivates our offer of the EOP platform as a library. They also require a system that provides good quality at a reasonable efficiency as well as guidance as to the best choice of parameters. The latter point is realized through our results archive in the official EOP Wiki on the EOP site.

Use Case 2: Textual Entailment Development. This category covers researchers who are interested in Recognizing Textual Entailment itself, for example with the goal of developing novel algorithms for detecting entailment. In contrast to the first category, this group need to look "under the hood" of the EOP platform and access the source code of the EOP. For this reason, we have spent substantial effort to provide the code in a well-structured and well-documented form.

A subclass of this group is formed by researchers who want to set up a RTE infrastructure for languages in which it does not yet exist (that is, almost all languages). The requirements of this class of users comprises clearly specified procedures to replace the Linguistic Analysis Pipeline, which are covered in our documentation, and simple methods to acquire knowledge resources for these languages (assuming that the EDAs themselves are largely language-independent). These are provided by the language-independent knowledge acquisition tools which we offer alongside the platform (cf. Section 3.2).

Use Case 3: Lexical Semantics Evaluation. A third category consists of researchers whose primary interest is in (lexical) semantics.

As long as their scientific results can be phrased in terms of semantic similarities or inference rules, the EOP platform can be used as a simple and standardized workbench for these results that indicates the impact that the semantic knowledge under consideration has on deciding textual entailment. The main requirement for this user group is the simple integration of new knowledge resources into the EOP platform. This is catered for through the definition of the generic knowledge component interfaces (cf. Section 2.2) and detailed documentation on how to implement these interfaces.

[5] http://www.excitement-project.eu/index.php/results

Use Case 4: Educational Use. The fourth and final use case is as an educational tool to support academic courses and projects on Recognizing Textual Entailment and inference more generally. This use case calls, in common with the others, for easy usability and flexibility. Specifically for this use case, we have also developed a series of tutorials aimed at acquainting new users with the EOP platform through a series of increasingly complexity exercises that cover all areas of the EOP. We are also posting proposals for projects to extend the EOP on the EOP Wiki.

5 EOP Distribution

The EOP infrastructure follows state-of-the-art software engineering standards to support both users and developers with a flexible, scalable and easy to use software environment. In addition to communication channels, like the mailing list and the issue tracking system, the EOP infrastructure comprises the following set of facilities.

Version Control System: We use GitHub,[6] a web-based hosting service for code and documentation storage, development, and issue tracking.

Web Site: The GitHub Automatic Page Generator was used to build the EOP web site and Wiki, containing a general introduction to the software platform, the terms of its license, mailing lists to contact the EOP members and links to the code releases.

Documentation: Both user and developer documentation is available from Wiki pages; the pages are written with the GitHub Wiki Editor and hosted on the GitHub repository. The documentation includes a Quick Start guide to start using the EOP platform right away, and a detailed step by step tutorial.

Results Archive: As a new feature for community building, EOP users can, and are encouraged to, share their results: the platform configuration files used to produce results as well as contact information can be saved and archived into a dedicated page on the EOP GitHub repository. That allows other EOP users to replicate experiments under the same condition and/or avoid doing experiments that have already been done.

[6] https://github.com/

Build Automation Tool: The EOP has been developed as a Maven[7] multi-modules project, with all modules sharing the same Maven standard structure, making it easier to find files in the project once one is used to Maven.

Maven Artifacts Repository: Using a Maven repository has a twofold goal: (i) to serve as an internal private repository of all software libraries used within the project (libraries are binary files and should not be stored under version control systems, which are intended to be used with text files); (ii) to make the produced EOP Maven artifacts available (i.e., for users who want to use the EOP as a library in their own code). We use Artifactory[8] repository manager to store produced artifacts.

Continuous Integration: The EOP uses Jenkins[9] for Continuous Integration, a software development practice where developers of a team integrate their work frequently (e.g., daily).

Code Quality Tool: Ensuring the quality of the produced software is one of the most important aspects of software engineering. The EOP uses tools like PMD[10] that can automatically be run during development to help the developers check the quality of their software.

5.1 Project Repository

The EOP Java source code is hosted on the EOP Github repository and managed using Git. The repository consists of three main branches: the *release branch* contains the code that is supposed to be in a production-ready state, whereas the *master branch* contains the code to be incorporated into the next release. When the source code in the master branch reaches a stable point and is ready to be released, all of the changes are merged back into release. Finally, the *gh-pages branch* contains the web site pages.

5.2 Licensing

The software of the platform is released under the terms of General Public License (GPL) version 3.[11] The platform contains both components and resources designed by the EOP developers, as well as others that are well known and freely available

[7] http://maven.apache.org/
[8] http://www.jfrog.com/
[9] http://jenkins-ci.org/
[10] http://pmd.sourceforge.net
[11] http://www.gnu.org/licenses/gpl.html

in the NLP research community. Additional components and resources whose license is not compatible with the EOP license have to be downloaded and installed separately by the user.

6 Conclusion

This paper has presented the main characteristics of Excitement Open Platform platform, a rich environment for experimenting and evaluating textual entailment systems. On the software side, the EOP is a complex endeavor to integrate tools and resources in Computational Linguistics, including pipelines for three languages, three pre-existing entailment engines, and about two dozens of lexical and syntactic resources. The EOP assumes a clear and modular separation between linguistic annotations, entailment algorithms and knowledge resources which are used by the algorithms. A relevant benefit of the architectural design is that a high level of interoperability is reached, providing a stimulating environment for new research in textual inferences.

The EOP platform has been already tested in several pilot research projects and educational courses, and it is currently distributed as open source software under the GPL-3 license. To the best of our knowledge, the entailment systems and their configurations provided in the platform are the best systems available as open source for the community. As for the future, we are planning several initiatives for the promotion of the platform in the research community, as well as its active experimentation in real application scenarios.

Acknowledgments

This work was partially supported by the EC-funded project EXCITEMENT (FP7ICT-287923).

References

Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*, pages 871–876, Vancouver, BC.

Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2010. The Sixth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of TAC*, Gaithersburg, MD.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Journal of Natural Language Engineering*, 15(4):i–xvii.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The Third PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Prague, Czech Republic.

Iryna Gurevych, Max Mühlhäuser, Christof Müller, Jürgen Steimle, Markus Weimer, and Torsten Zesch. 2007. Darmstadt knowledge processing repository based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture (UIMA@GSCL 2007)*, Tübingen, Germany.

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, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL demo session*, pages 177–180, Prague, Czech Republic.

Milen Kouylekov and Bernardo Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the First PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 17–20, Southampton, UK.

Dekang Lin and Patrick Pantel. 2002. Discovery of Inference Rules for Question Answering. *Journal of Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of ACL/COLING*, pages 768–774, Montréal, Canada.

Tae-Gil Noh and Sebastian Padó. 2013. Using UIMA to structure an open platform for textual entailment. In *Proceedings of the 3rd Workshop on Unstructured Information Management Architecture (UIMA@GSCL 2013)*.

Sebastian Padó, Tae-Gil Noh, Asher Stern, Rui Wang, and Roberto Zanoli. 2014. Design and realization of a modular architecture for textual entailment. *Journal of Natural Language Engineering*. doi: 10.1017/S1351324913000351.

Eyal Shnarch, Libby Barak, and Ido Dagan. 2009. Extracting lexical reference rules from Wikipedia. In *Proceedings of ACL-IJCNLP*, pages 450–458, Singapore.

Asher Stern, Roni Stern, Ido Dagan, and Ariel Felner. 2012. Efficient search for transformation-based inference. In *Proceedings of ACL*, pages 283–291, Jeju Island, South Korea.

Rui Wang and Günter Neumann. 2007. Recognizing textual entailment using a subsequence kernel method. In *Proceedings of AAAI*, pages 937–945, Vancouver, BC.