Expert Systems with Applications 42 (2015) 6864–6877

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Evolutionary fine-tuning of automated semantic annotation systems

John Cuzzola [a,*], Jelena Jovanović [b], Ebrahim Bagheri [a], Dragan Gašević [c]

[a] Laboratory for Systems, Software and Semantics (LS[3]),[1] Ryerson University, Ontario, Canada
[b] Faculty of Organizational Sciences (FOS),[2] University of Belgrade, Belgrade, Serbia
[c] Schools of Education and Informatics,[3] University of Edinburgh, Scotland, United Kingdom

ARTICLE INFO

Article history:
Available online 9 May 2015

Keywords:
Semantic annotation
Automated configuration
Genetic algorithm
Parameter learning

ABSTRACT

Considering the ever-increasing speed at which new textual content is generated, an efficient and effective use of large text corpora requires automated natural language processing and text analysis tools. A subset of such tools, namely automated semantic annotation tools, are capable of interlinking syntactical forms of text with their underlying semantic concepts. The optimal performance of automated semantic annotation tools often depends on tuning the values of the tools' adjustable parameters to the specificities of the annotation task, and particularly to the characteristics of the text to be annotated. Such characteristics include the text domain, terseness or verbosity level, text length, structure and style. Since the default configuration of annotation tools is not suitable for the large variety of input texts that different combinations of these attributes can produce, users often need to adjust the annotators' tunable parameters in order to get the best results. However, the configuration of semantic annotators is presently a tedious and time consuming task as it is primarily based on a manual trial-and-error process. In this paper, we propose a Parameter Tuning Architecture (PTA) for automating the task of configuring parameter values of semantic annotation tools. We describe the core fitness functions of PTA that operate on the quality of the annotations produced, and offer a solution, based on a genetic algorithm, for searching the space of possible parameter values. Our experiments demonstrate that PTA enables effective configuration of parameter values of many semantic annotation tools.

1. Introduction

The quantity and variety of unstructured textual content has rapidly increased over the last few years, leading large and small organizations towards seeking solutions that enable effective and efficient use of both the internally produced textual content, and the content originating from the Web.[4] Considering the amount of textual content and the speed at which it has to be processed, it is gradually becoming evident that automated machine comprehension of text is a necessity, if the objectives of efficiency and effectiveness were to be reached. This has led to an increased research focus, both in academia and industry, on text mining, natural language processing and other related Artificial Intelligence fields (Hovy, Navigli, & Ponzetto, 2013), and resulted in numerous proposals and specific software solutions for addressing some aspects of text comprehension through, for example, named entity extraction (Ratinov & Roth, 2009; S. Atdağ & Labatut, 2013), relation extraction (Weston, Bordes, Yakhnenko, & Usunier, 2013; Yan, Okazaki, Matsuo, Yang, & Ishizuka, 2009), and sentiment analysis (Liu, 2012).

Automated semantic annotation of textual content addresses an important aspect of text comprehension, namely, the extraction and disambiguation of entities and topics mentioned in or related to a given piece of text (Uren et al., 2005). Each identified entity is disambiguated, i.e., unambiguously defined, by establishing a link to an appropriate entry (concept or instance) in a knowledge base that uniquely identifies the entity and provides further information about it. This task, also known as *entity linking* (Hachey, Radford, Nothman, Honnibal, & Curran, 2013), typically relies on large, general-purpose, Web-based knowledge bases, such as Wikipedia and other more structured knowledge bases such as DBpedia (http://dbpedia.org), YAGO (http://www.mpi-inf.mpg.de/yago-naga/yago/), and Wikidata (http://wikidata.org).

Tools and services for automated semantic annotation of text are offered by a constantly increasing number of companies and research groups (Jovanovic et al., 2014). Major Internet players are also very active in this area. For instance, to fulfill its well

* Corresponding author.
E-mail addresses: jcuzzola@ryerson.ca (J. Cuzzola), jeljov@gmail.com (J. Jovanović), bagheri@ryerson.ca (E. Bagheri), dgasevic@acm.org (D. Gašević).

[1] http://ls3.rnet.ryerson.ca.
[2] http://www.fon.bg.ac.rs.
[3] http://www.inf.ed.ac.uk.
[4] http://blog.digitalinsights.in/social-media-users-2014-stats-numbers/05205287.html.

http://dx.doi.org/10.1016/j.eswa.2015.04.054
0957-4174/

known mission of "organizing the world's information", Google is continuously evolving its proprietary knowledge base – the Knowledge Graph – and according to one Google executive, "every piece of information that we [Google] crawl, index, or search is analyzed in the context of Knowledge Graph".[5] In addition, Google has been working on a probabilistic knowledge base, named Knowledge Vault, that combines automated extraction of facts from the Web and prior knowledge derived from existing knowledge bases (Dong et al., 2012). Similarly, Microsoft is developing its own knowledge repository called Satori and using it to semantically index content and thus improve both its search engine Bing and the applications running on Windows.[6]

In (Jovanovic et al., 2014), we have provided a comprehensive descriptive comparison of the state-of-the-art semantic annotation tools by considering numerous features, especially those that could be relevant for selecting the right tool(s) to use in a specific application case. One common characteristic of all the reviewed tools is that they need to be optimally configured in order to give their best results when working with different kinds of texts – such as texts of diverse level of formality, length, domain-specificity, and use of jargon. While the examined annotators provide default configuration of their parameters suitable for some annotation tasks, to our knowledge, no single annotator can reach its best performance on all kinds of text with one single configuration. Furthermore, the quality of an annotator's output is not a category that could be assessed in absolute terms; instead, it depends on the application case, i.e., on the specificities of the requirements that stem from a particular context of use (Maynard, 2008). For instance, in some cases a very detailed annotation would be required and highly valued, whereas in other cases a terse annotation of only the most relevant entities would be considered the best output. This indicates that in order to get the best from a semantic annotation tool, one should configure it according to the specificities of the intended context of use, including both the characteristics of the text to be annotated and the requirements of the annotation task (e.g., precision/recall trade-off).

Configuration of semantic annotators is not an easy task, for at least two reasons. First, since an annotator's configuration parameters are closely tied to the tool's internal functioning, it is difficult to expose them in a manner that would enable users to effectively and efficiently use the tool without having to know the details of the tool's inner logic. In other words, the first challenge is in enabling users to tune the annotator with respect to the key issues such as specificity and comprehensiveness of annotations, without them being concerned with the details of the tool's parameters. The second challenge stems from the fact that those configuration parameters are not mutually independent but interact with one another, so that one has to find an optimal combination of parameter values for a specific application case. Moreover, annotators may have many parameters, and some of those parameters are continuous variables, thus making the tuning task very time consuming. As the state-of-the-art annotators do not provide support for finding an optimal parameter combination for a specific annotation task, it is often done manually, through a trial-and-error process. For example, consider the commercial semantic annotator *TextRazor* whose best practices state the following:

> "Experiment with different confidence score thresholds...If you prefer to avoid false-positives in your application you may want to ignore results below a certain threshold. The best way to find an appropriate threshold is to run a sample set of your documents through the system and then manually inspect the results." [7].

[5] http://goo.gl/mZ7a9H.
[6] http://goo.gl/iDwP2x.
[7] https://www.textrazor.com/docs/rest#optimization.

To our knowledge, no solution to the above stated problem of parameter configuration has been reported in the literature. Therefore, in this paper, we make the following contributions:

- Parameter Tuning Architecture (PTA) to automate the task of parameter value selection for a user-supplied testing set; thus resulting in performance that is better or at least equal to the tool's performance with its default parameter values.
- Five variations of the fitness function that emphasize different aspects of annotation quality (namely most annotations produced, most known correct annotations, least unknown annotations, best recall/precision), and a means to identify which variation performed the best for a particular testing set.
- A method to efficiently search the solution space of possible parameter values using a Genetic algorithm.

The proposed Parameter Tuning Architecture (PTA) is applicable to a variety of automated semantic annotators, since its core component of the fitness function is not concerned with any textual or annotator-specific features but rather metrics based on known correct, known incorrect, or unknown annotations produced. To search the space of possible solutions, i.e., possible configurations of parameter values, we rely on a Genetic algorithm (for the reasons given in Section 4), although PTA can also be applied with other methods for searching a large solution space (e.g., evolutionary algorithms or probabilistic methods). Our experiments with PTA have demonstrated that PTA can be used as an effective configurator for automated semantic annotators.

After more precisely defining and illustrating the problem of parameter tuning in the context of semantic annotation tools (Section 2), and the associated challenges (Section 3), in Section 4, we present PTA in detail. Section 5 reports on the experiments that we performed in order to evaluate the PTA's ability to find a set of parameter values that provides an adequate level of the annotator's output while minimizing annotation errors. The experimental results and the overall proposal are further critically discussed and summarized in Section 6, while Section 7 positions the contributions of our work with respect to related research work. Lastly, we acknowledge the limitations of our solution and propose future experiments before we conclude our paper (Sections 8 and 9).

2. Problem definition

As indicated in the Introduction, today's automated semantic annotators offer a variety of tunable parameters in order to produce results accordant with the desired level of granularity, precision, and recall. There is no single "best" configuration as this is a function of various factors: is the text we are annotating restricted to a specific topic or domain such as history, food, or politics? Is the input text descriptive with verbose and meaningful wording or is it terse with numerous empty (stop) words? What is the length, style, and structure of the input text: paragraph, single sentence, or tweet? Therefore, we must decide on a *gold-standard*, a manually labelled training or testing set, that contains such factor-specific target questions that the annotator will be exposed to. Observe, as in the TextRazor introduction example, that an annotator would already be trained with default parameter values; thus, PTA uses the gold standard as an evaluation/testing set to tailor the annotator's parameters to the kind of input text represented by the gold standard.

Further difficulties arise when a parameter configuration comprises many individual or continuous floating point parameters resulting in an exponential number of possible combinations that make a complete search of this space unrealistic. To illustrate this, consider Table 1 showing how the *TagME* semantic annotator