Evolutionary Fine-Tuning of Automated Semantic Annotation Systems

John Cuzzola[a] (jcuzzola@ryerson.ca), Jelena Jovanovic[b] (jeljov@gmail.com), Ebrahim Bagheri[a] (bagheri@ryerson.ca), Dragan Gasevic[c] (dgasevic@acm.org)

[a] *Laboratory for Systems, Software and Semantics (LS^3), http://ls3.rnet.ryerson.ca, Ryerson University, Ontario, Canada*

[b] *Faculty of Organizational Sciences (FOS), http://www.fon.bg.ac.rs, University of Belgrade, Belgrade, Serbia*

[c] *Schools of Education and Informatics, http://www.ed.ac.uk/schools-departments/informatics, University of Edinburgh, Scotland, United Kingdom*

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.

Keywords. semantic annotation, automated configuration, genetic algorithm, parameter learning

I. 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[1]. 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, 2013; Atdağ & Labatut, 2013), relation extraction (Yan, Okazaki, Matsuo, Yang, & Ishizuka, 2009; Weston, Bordes, Yakhnenko, & Usunier, 2013), 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

[1]http://blog.digitalinsights.in/social-media-users-2014-stats-numbers/05205287.html

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 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"[2]. 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., 2014). 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[3].

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

[2] http://goo.gl/mZ7a9H

[3] http://goo.gl/iDwP2x

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." [4]

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 IV), 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 (Sect. II), and the associated challenges (Sect. III), in Section IV, we present PTA in detail. Section V 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 VI, while Section VII 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 VIII / IX).

[4] https://www.textrazor.com/docs/rest#optimization

II. 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 (Ferragina & Scaiella, 2012) choses to disambiguate the same spot ("*field*"), when different values of the tool's two tunable parameters (*epsilon* and *long_text*) are used. As the table indicates, even slight changes to the values of epsilon and long_text can have significant impact on the disambiguation process and lead to absurd results.

Table 1. Disambiguation of the spot "*field*" in the sentence "*Gesture or salute? A soccer star who made the sign on the field says otherwise.*" for different values of TagMe's tunable parameters *epsilon* and *long_text*.

Configuration	**Disambiguation of the spot "field" by TagMe**

epsilon = 0.39
long_text = 0
	Wikipedia Reference: Field (agriculture)
	Abstract: In agriculture, the word field refers generally to an area of land enclosed or otherwise and used for agricultural purposes.

epsilon = 0.30
long_text = 1
	Wikipedia Reference: Field (mathematics)
	Abstract: In abstract algebra, a field is a ring whose nonzero elements form a commutative group under multiplication.

epsilon = 0.52
long_text = 4
	Wikipedia Reference: Academic discipline
	Abstract: An academic discipline, or field of study, is a branch of knowledge that is taught and researched at the college or university level.

epsilon = 0.55
long_text = 6
	Wikipedia Reference: Field (physics)
	Abstract: In physics, a field is a physical quantity associated with each point of spacetime.

It is this difficulty of choosing appropriate parameter values that provides the motivation for the work presented in this paper. To our knowledge, the problem of selecting appropriate values for configuration parameters of semantic text annotators has not yet been reported in the literature. In the following sections, we first further explain the challenges associated with tuning parameters of semantic annotators, and then proceed to introduce our Parameter Tuning Architecture (PTA) and a method to find suitable parameter values in reasonable time.

III. Challenges in Tuning Parameters of Semantic Annotators

Our proposed method involves a fitness function to evaluate the output of an annotator given an arbitrary configuration. PTA's fitness function scores the annotator's output against a testing set containing oracle-identified correct (C) and/or incorrect (E) annotations. Table 2 gives the four possible combinations of labelled annotations within the testing set. NOT(¬) indicates that any correct or incorrect annotations are absent from the testing set.

Table 2. The four kinds of testing sets based on the presence/absence of correct (C) and incorrect (E) labels.

Combination (1,2,3,4)	Correct Annotations (positive labels)	Incorrect Annotations (negative labels)	Testing Model
#1	C	E	two-class supervised
#2	C	¬E	one-class supervised positive labels only
#3	¬C	E	one-class supervised negative labels only
#4	¬C	¬E	unsupervised

Combination #4 assumes a testing set with no labels, and thus is an unsupervised model that PTA cannot take advantage of; hence, it will not be further considered. In contrast, combinations #1, #2, and #3 are supervised models that variants of the PTA fitness function can utilize. However, supervised configuration of semantic annotators have caveats when relying on testing sets. Namely:

caveat (i): A testing set that depends on negative labels (#1, #3) cannot feasibly cover all possible mistakes that an annotator could output for an input text.

caveat (ii): Due to (i), publicly available testing sets with labelled errors are difficult to find, particularly one-class (error-only) testing sets (#3).

caveat (iii): Testing sets with correct labels (#1, #2) often specify only a few correct annotations that the annotator should identify from the input text. However, the output could contain many more correct annotations not mentioned in the testing set.

caveat (iv): An oracle may specify correct but not necessarily ideal annotations. The annotation tool may find an annotation that is better than or of equal semantic quality to that of the oracle-recommended annotation.

caveat (v): Regardless of the form of the testing set used (#1, #2, #3), an annotator will often produce annotations that are not explicitly identified in the testing set, thus forcing any method of evaluation to make assumptions for the unidentified annotations (assume correct, assume incorrect, or ignore). Consequently, the quality of the annotator's output in relation to the testing set provided is questionable.

To illustrate the above given statements, we obtained the *wiki-annot30* dataset of A^3 labs from the *University of Pisa Computer Science Department* available under a Creative Commons License[5]. This set contains 186,000 short text fragments from Wikipedia with correct-only identified annotations (combination #2) and was constructed using the procedure described in Ferragina and Scaiella (2012). Table 3 summarizes the output on one of these text fragments produced by the TagME semantic annotator with default configuration. The output is given alongside the A^3 dataset gold standard. The table gives the Wikipedia pages that are linked to the identified spot. The TagME column also shows whether the linked Wikipedia page for the spot is true correct (C) or a true error (E).

Table 3. Wikipedia entities produced by TagME and compared with the A^3 gold standard for the text fragment "*It is home to the American University Eagles basketball and volleyball teams. The arena, named for Washington DC philanthropists, Howard and Sondra Bender*". TagME links are identified as (C)orrect or (E)rror.

Spot	**A^3 dataset**	**TagME**
volleyball	wikipedia:Volleyball	wikipedia:Volleyball (C)
washington dc	wikipedia:Washington,D.C.	wikipedia:Washington,D.C. (C)
basketball	wikipedia:Basketball	wikipedia:Basketball (C)
the american university (eagles)	wikipedia:American University	wikipedia:American Eagles (C)
arena	wikipedia:Arena	wikipedia:Arena (C)
home	*Not available*	wikipedia:Home(sports) (C)
teams	*Not available*	wikipedia:NHL (E)
howard	*Not available*	wikipedia:Howard University (E)
bender	*Not available*	wikipedia:Gary Bender (E)

From the output we can see that TagME correctly linked more spots than what was listed in the A^3 dataset (caveat *iii*). Specifically, the 'home' spot is a correct entity mention absent from A^3. Further, absent from the A^3 dataset is the correct spot for 'team' that was linked incorrectly by TagME along with incorrect links for 'howard', and 'bender'. Consequently, to assess the accuracy of the output, one must first consider how these unknown spots should be treated (caveat *v*). Lastly, consider the spot 'the

[5] http://acube.di.unipi.it/tagme-dataset

american university (eagles)'. A[3] recommends the Wikipedia link of 'American University' while TagME prefers the link 'American Eagles'. Both suggestions are correct since the official sports team of American University is in fact the American Eagles. However, in the context of the text fragment, it is clear that the link to American Eagles is a better (i.e., more precise) choice than the gold standard (caveat *iv*). A similar situation occurs with the 'arena' spot in which TagME agrees with the wiki-annot30 dataset although the Wikipedia entity for 'Bender Arena' would clearly be superior.

In Section IV we detail our five variations of the Parameter Tuning Architecture (PTA) and how they match to the testing set combinations given in Table 2. We also explain how each variation of PTA addresses the caveats.

IV. The Parameter Tuning Architecture (PTA)

In this section, we derive our Parameter Tuning Architecture as a mathematical model. Let p_i be a tuneable parameter for a semantic annotator. Let v_n be a vector of n tuneable parameters $v_n = [p_1 p_{2,} ..., p_n]$ for a semantic annotator configuration, and let V be the space of all possible configurations. Let T be a testing or validation set of any of the supervised combinations of Table 2 (#1, #2, or #3). We wish to find a vector υ that maximizes the following fitness function:

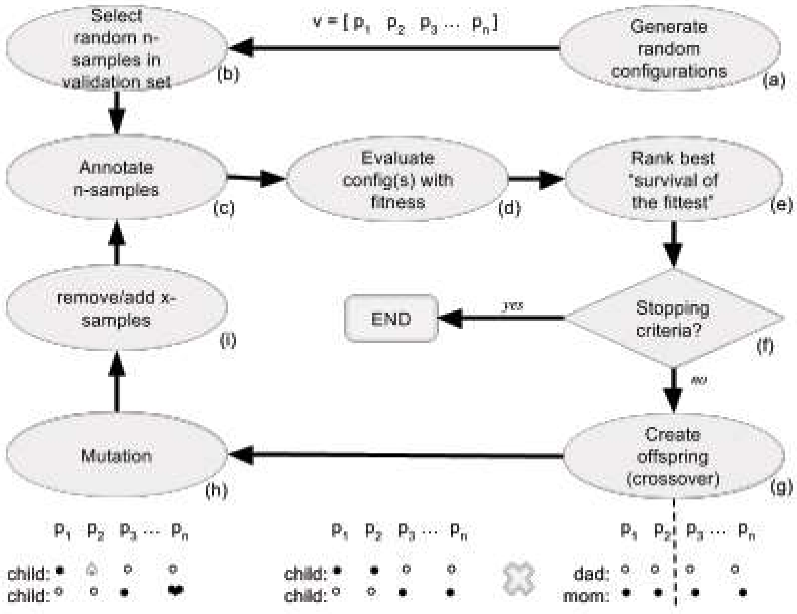
$$FIT(T) = arg\ max_{v \varepsilon V} [f_c(T)]^2 - [f_e(T)]^2 \qquad \text{eq. 1}$$

$$where\ FIT(T)\ in\ [-1, 1]\ and\ f_c(T), f_e(T)\ in\ [0, 1].$$

FIT(T) is a real-valued ranking function in the interval [-1,1] consisting of a reward component $f_c(T)$ and a penalty $f_e(T)$. The reward is a real-valued scaled measure in the interval [0,1] representing the correct annotations discovered by an annotator. Conversely, $f_e(T)$ is a measure of the incorrect annotations within the same interval [0,1]. Consequently, FIT(T) is a trade-off between the correct annotations found by the annotator and the errors the annotator produces. We square f_c and f_e to place more importance on many correct and incorrect annotations over a few correct answers and infrequent mistakes. The exact formula for computation of f_c and f_e is specific to the variant of FIT(T) proposed and is given in sections IV.1 through IV.5.

However, equation 1 faces a key obstacle. Namely, the space of all possible solution vectors V is very large making an exhaustive search of this space intractable. Furthermore, this space may be infinite if any of the tunable parameters are real-valued. Consequently, we need to decide on how to efficiently search a potentially enormous solution space. Our approach is to use a genetic algorithm (GA) for this task. Genetic algorithms are a class of evolutionary algorithms inspired by nature (Chiong & Beng, 2007). They are an easy to implement technique that works well in optimization problems where potential solutions to the given problem can be abstracted to the "chromosomes" model of GA. In our case, the vector of configuration parameters v can be seen as the required chromosome.

Figure 1 outlines our PTA method with GA which begins with a set of randomly created configurations (a). In the next step (b), we randomly select a subset of samples of size *n* from the validation set, then annotate the n-samples (c) using each of the configurations initially created in the step (a). The configurations are evaluated against the samples by our fitness function (d). The highest ranking configurations generate offspring through crossover (g) and gene mutation (h). The initial n-samples from the validation set is updated by removing the oldest x-samples and replacing them with another random set of x-samples (i). We partially re-sample the validation set at each generation as a form of bootstrap aggregation to minimize overfitting and improve generalization. The process repeats with the next generation of configurations until some stopping condition, such as convergence, is satisfied (f).



Select random n-samples in validation set
(b)
v = [p1 p2 p3 ··· pn]
Generate random configurations
(a)
Annotate n-samples
(c)
Evaluate config(s) with fitness
(d)
Rank best "survival of the fittest"
(e)
remove/add x-samples
(i)
END
yes
Stopping criteria?
(f)
no
Mutation
(h)
Create offspring (crossover)
(g)
p1 p2 p3 ··· pn
child:
child:
dad:
mom:

Figure 1. The Genetic Algorithm applied in the Parameter Tuning Architecture (PTA)

Although GA does not guarantee an optimal solution, it often converges to near optimal answers in a relatively short period of time (Szczerbicka, Becker & Syrjakow, 1998). It also has the capability to search multiple regions of the solution space simultaneously since each member of a population occupies a different area than its peers (Grefenstette, 1992). This is particularly useful when there are many global optimal solutions that could satisfy the problem. Finally, GA is an incremental method that allows for starting/stopping the search from its currently best known solutions without beginning from scratch. This trait is useful in reinforcement learning or re-tuning the annotator as more examples become available (Moriarty, Schultz & Grefenstette, 1999).

It is worth noting that our emphasis is on the fitness functions of PTA and less on the choice of evolutionary algorithm for the traversal of the search space. Our future work will investigate the comparative effectiveness of other evolutionary algorithms such as swarm optimization. In order to provide the basis for a comparative analysis and provide GA implementation details, we have made the source code and data available for use under an open source license at: http://ls3.rnet.ryerson.ca/annotator/PTA.

Algorithm 1 outlines the PTA process joining our fitness function (equation 1) with the GA of Figure 1.

```
Input: i) A one-class positive-labeled validation set from the gold standard. ii) A one-class
positively-labeled testing set from the gold standard. iii) A semantic annotator.

Output: A recommended configuration for input annotator (iii).

Algorithm:

1.   For each of these PTA variants: pessimistic (eq. 5), apathetic (eq. 6), delta (eq. 7), optimistic
     (eq. 8) and stochastic (eq. 9)
     1.1.   Find recommended configuration on the validation set using GA of Figure 1.
2.   For each recommended configuration from 1.1:
     2.1.   Annotate the testing set using the recommended configuration.
     2.2.   Calculate $F_1$ area under the curve ($F_1$-AUC) (figures 2,3,4,5).
3.   Best recommended solution is the configuration with highest $F_1$-AUC.
```

Algorithm 1: PTA algorithm for automated evolutionary fine-tuning of semantic annotators.

We now derive various forms of our PTA fitness function, for use within step 1 of Algorithm 1, We focus on the one-class positive labels testing set (Table 2, combination #2) since this is the most prevalent type among the available gold-standard datasets for the entity linking task.

To begin, we define A(X) as the output of a semantic annotator using text fragments from set X as input. Let function C(X) return the set of correct annotations from the set of input text X. Similarly, let E(X) return incorrect annotations from the input text set X. We use the generic term 'annotation' broadly to refer to any kind of output produced by a semantic annotation system (an entity link, a related topic, a keyword) as per the examples provided within the validation or testing set. We define function $f_c(X)$ as a numerical score for the correct annotations of set X, while $f_e(X)$ does the same for the errors of X. With these definitions in place, we can identify subsets such as: (i) annotations that match the gold standard A(T)∩C(T), (ii) gold standard annotations the annotator failed to identify C(T)\A(T), and (iii) additional annotations with unknown label A(T)\C(T). In the following subsections, we present five adaptations of PTA (pessimistic, apathetic, delta, optimistic, stochastic) that pair different combinations of these subsets to meet the challenge of constructing a suitable fitness function in the presence of uncertainty A(T)\C(T). In section V, we examine the effectiveness of these five PTA forms in tuning a variety of annotators.

IV.1 Pessimistic PTA

The basic form of our fitness function is:

$$FIT(T) = f_c(C(A(T)))^2 - f_e(E(A(T)))^2 \quad \text{eq. 2}$$

This basic form is suitable for a two-class supervised testing set (Table 2 #1) because it assumes that we are provided with known correct and incorrect annotations that can be exploited:

$$FIT(T) = f_c(C(A(T)))^2 - f_e(E(A(T)))^2 = f_c(A(T) \cap C(T))^2 - f_e(A(T) \cap E(T))^2 \quad \text{eq. 3}$$

Specifically, we find the correct annotations $C(A(T))$ by considering only those annotations that are explicitly known to us as correct within the testing set $A(T) \cap C(T)$. We do the same for the error component of equation 2 with $A(T) \cap E(T)$. In this variant of PTA the annotations that are not part of the testing set (unrecognized) are ignored (Section II caveat *v*). Although this form operates for a two-class labeled testing set, it cannot be used for the commonly encountered one-class positive labels testing set (Table 2 #2) due to the absence of any defined errors. To compensate, we modify f_e to assume that unrecognized annotations are most likely errors by default:

$$FIT_P(T) = f_c(A(T) \cap C(T))^2 - f_e(A(T) \setminus C(T))^2 \quad \text{eq. 4}$$

We call equation 4 pessimistic because of the assumption that annotations not explicitly labeled as correct should be considered erroneous. The final step in the formulation is to define f_c and f_e as:

$$f_c(A(T) \cap C(T)) = \frac{|A(T) \cap C(T)|}{|A(T) \cap C(T)|_{max}} \text{ and } f_e(A(T) \setminus C(T)) = \frac{|A(T) \setminus C(T)|}{|A(T) \setminus C(T)|_{max}} \quad \text{eq. 5}$$

where the denominator $|Y|_{max}$ of $f(\frac{|Y|}{|Y|_{max}})$ is the maximum observed count from all of the previously encountered sets of Y; it is used to normalize the numerator into a value between 0 and 1 inclusive.

A shortcoming of pessimistic PTA is the possibility that the testing set identifies a smaller number of correct annotations than the annotator might generate (Section III caveat *iii*), as demonstrated in Table 3. Consequently, those unidentified correct annotations would be missed by f_c thus undermining the true quality of the output produced, and would be unfairly counted against the annotator in the error calculation of f_e (equation 5) or ignored entirely (equation 3).

IV.2 Apathetic PTA

In this section, we introduce a version of PTA that allows for ignoring unknown annotations when only a one-class positive testing set is available. Simply, we change the penalty component of PTA f_e to look at the number of missed known gold standard annotations rather than the total number of unrecognized annotations generated. Specifically:

$$FIT_A(T) = f_c(A(T) \cap C(T))^2 - f_e(C(T) \setminus A(T))^2 \quad \text{eq. 6}$$

where f_e is defined as the normalized number of missed gold standard annotations $\frac{|C(T) \setminus A(T)|}{|C(T) \setminus A(T)|_{max}}$ and f_c is the normalized number of matching answers as before $\frac{|A(T) \cap C(T)|}{|A(T) \cap C(T)|_{max}}$. We call this variant *apathetic* because its concern is only with the number of known correct and recognized versus correct but missed annotations, regardless of the number of uncertain annotations.

IV.3 Delta PTA

With apathetic PTA, the reward component of matching annotations from the gold standard is paired with the penalty of missing annotations from the same. In this adaption, we compute Δ as the absolute difference between the total number of annotations and the number of gold-standard annotations desired.

$$\Delta = \big|\, |A(T)| - |C(T)| \,\big|$$

We want Δ to be as close as possible to zero to minimize the number of unrecognized annotations. Consequently, we define the reward component to encourage this result $f_c = 1 - \frac{\Delta}{\Delta_{max}}$. Furthermore, we want A(T) to have many matching C(T) answers thus we penalize missing gold standard annotations using the same f_e as in apathetic PTA.

$$FIT_D(T) = \left[1 - \frac{\Delta}{\Delta_{max}}\right]^2 - \left[\frac{|C(T) \setminus A(T)|}{|C(T) \setminus A(T)|_{max}}\right]^2 \quad \text{eq. 7}$$

IV.4 Optimistic PTA

The former pessimistic, apathetic, and delta versions of PTA reward known correct annotations (equations 3,4,6) while penalizing known errors (equation 3), missed answers (equations 6,7) and uncertainty (equation 4). With optimistic PTA we assume (optimistically) that unknown annotations are more often right than wrong. Consequently, we reward configurations that produce more annotations than those configurations that output less. To this end we redefine the reward component of the function f_c as the normalized ratio of the number of annotations produced :$f_c = \frac{|A(T)|}{|A(T)|_{max}}$

$$FIT_O(T) = \left[\frac{|A(T)|}{|A(T)|_{max}}\right]^2 - f_e(E(A(T)))^2 \quad \text{eq. 8}$$

Equation 8 is available for a two-class testing set and one-class negative label testing set (Table 2 #1, #3). However, for the one-class positive labeled testing set, the f_e term needs to be calculated without assistance from negative testing samples. The f_e term of equation 5 could be used in this circumstance but works against the optimistic assumption that unknown annotations are most likely correct. In section IV.5, we compensate for this rigidness.

IV.5 Stochastic Optimistic PTA

Optimistic PTA in its current form (equation 8) cannot use a one-class positive label testing set without 'borrowing' f_e from its pessimistic counterpart (equation 5). While equation 4 assumes 100% of the unknown (unlabeled) annotations are errors, stochastic optimistic PTA tries to estimate the expected number of errors within this unknown set.

Stochastic Optimistic PTA assumes if oracle-specific annotations are missing from the annotator's output, then this is an indicator of poor performance of the annotation tool. However, it might also happen that a tool produces more correct annotations than what the testing set mentions, with perhaps better semantic quality (Section III caveats iv,v and Table 3). On the other hand, one might assume that although the gold standard might not list all the possible annotations (caveat iii), those annotations that are listed are likely to be the *most important* in the context of the given text fragment. In other words, the annotations that form the gold standard are the results the annotator *should* produce. Consequently, Stochastic Optimistic PTA introduces an adjustment factor (φ) to equation 5 to soften the 100% error assumption while still emphasizing the importance of the gold standard. First, precision (P) and recall (R) are computed:

$P(T) = \frac{|A(T) \cap C(T)|}{|A(T)|}$ and $R(T) = \frac{|A(T) \cap C(T)|}{|C(T)|}$ eq. 9

Then, a *likelihood of error* is calculated using:

$$L_\varphi(T) = \begin{cases} 1 - \left[(1+\beta^2) \times \frac{P(T) \times R(T)}{\beta^2 P(T) + R(T)}\right] & if\ P(T) \neq 0 \\ 1 & if\ P(T) = 0 \end{cases}$$ eq.10

The ratio in brackets [] is the familiar F_β-score metric often used to balance precision and recall. When $\beta<1$, unknown annotations are more likely considered errors, whereas $\beta>1$ puts the emphasis on matching answers to the gold standard. We use $\beta=1$ to equally weight precision and recall. To avoid a division by zero error, we consider the case when precision is zero. In such a circumstance the likelihood of error is 1. Next, the expected number of errors is determined for an estimate of the actual number of errors of A(T).

$\varphi(T) = L_\varphi(T) \times |A(T) \setminus C(T)|$ eq. 11

To illustrate, consider the example of Table 3. TagME produced nine annotations (|A(T)|) of which four matched the testing set ($|A(T) \cap C(T)|$) out of a possible five known correct (|C(T)|). What remained was five annotations of unknown classification {*washington dc, home, teams, howard, bender*} ($|A(T) \setminus C(T)|$). Substituting these values for equations 9, 10 and 11 gives a likelihood of error of $L_\varphi(T) = 0.429$ with an expected error count of $\varphi(T) = 2.15$. The true error count of the unknown set is 3 {*teams, howard, bender*}.

The last step is to compute f_e by normalizing $\varphi(T)$ then substituting into equation 8.

$$FIT_S(T) = \left[\frac{|A(T)|}{|A(T)|_{max}}\right]^2 - \left[\frac{\varphi(T)}{\varphi(T)_{max}}\right]^2 \quad \text{eq. 12}$$

V. Experimentation

In this section, we evaluate the different forms of PTA against our one-class positive labeled gold standard under varying assumptions. Specifically, we examine how each form of PTA performs when the unrecognized annotations are believed to be mostly wrong (pessimistic), mostly right (optimistic), partially correct (stochastic/delta), or unimportant (apathetic).

We used a genetic algorithm (GA) to search the solution space as outlined in Figure 1 of Section IV. The GA parameters were defined as follows: an initial population of 10 random configurations, with a maximum surviving population of 30 configurations per generation and a mutation rate of 0.05. We began with a testing set of 15 randomly selected text fragments from our gold standard set of *wiki-annot30* (Section III), with a sample replacement rate of 5 text fragments per generation. The GA would terminate once the configurations within the surviving population stabilize. The top-ranked configurations for each PTA variant were then evaluated against a random set of 1000 text fragments from the gold standard and compared to the default out-of-the-box configuration of four popular semantic annotators: *TagME, Wikipedia Miner, DBpedia Spotlight,* and *Yahoo Content Analysis*. We also considered numerous other semantic annotators that were ultimately not evaluated due to unavailability of a Web service or restrictive end-user license agreement. Lastly, we focused our tests around Wikipedia-centric annotation tools leaving other knowledge based annotation tools for future work. Table 4 lists all semantic annotators we considered for PTA evaluation.

Table 4. List of tested and considered semantic annotators for PTA evaluation.

Annotator	**Tested**	**URL and notes for considered but not tested annotators**
TagME	*YES*	http://tagme.di.unipi.it/tagme_help.html#tagging
DBPedia Spotlight	*YES*	https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Web-service#Candidates
Wikipedia Miner	*YES*	http://wikipedia-miner.cms.waikato.ac.nz/services/?wikify
Yahoo Content Analysis	*YES*	https://developer.yahoo.com/contentanalysis/
AIDA	*NO*	Authors Web service is for demonstration purpose only. They explicitly request not to use their service for research purposes. http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/aida/webservice/
Denote	*NO*	Denote RESTful Web service is in beta and not available for use at the time of writing this paper. http://inextweb.com/denote_demo
AlchemyAPI	*NO*	Restrictive Terms of Use. Alchemy can not be used for the purpose of benchmarking and/or comparing with other annotators, especially for those

		having a competing annotation service. http://www.alchemyapi.com/api/register.html
Aylien	*NO*	Restrictive Terms of Service. Terms prohibits publication of any results obtained using their annotator without permission. No response from Aylien when contacted to obtain consent. http://aylien.com/text-api-tos
TextRazor	*NO*	Trial account allows for only 500 annotations per day which is insufficient for training and testing, https://www.textrazor.com/plans
OntoText	*NO*	Formerly known as *KIM - The Semantic Annotation Platform*. No accessible Web service without first consultation with the commercial company. http://www.ontotext.com/products/ontotext-semantic-platform/

V.1 PTA with TagME

The TagME annotation service offers fast execution and high accuracy particularly with short text fragments (Chiong and Beng, 2007; Cornolti, Ferragina and Ciaramita, 2013). Table 5 provides summary statistics and the recommended configuration for TagME using each of the five variants of PTA including TagME's default configuration for side-by-side comparison. Summary statistics include the average of the following measures on the 1000 gold standard text fragments: number of annotations A(T), number of annotations matching to the gold standard A(T)∩C(T), number of unmatched gold standard annotations C(T)\A(T), and count of unrecognized annotations A(T)\C(T). Derived from these measures are precision, recall, and F_1-score, also included in Table 5. These measures provide competing dimensions that impact each variant of PTA differently. For instance, TagME's default configuration produces the highest number of annotations with an average of 9.56, but also produces the highest number of unrecognized annotations, 6.22 on average. Comparatively, the pessimistic PTA solution produces the least number of unrecognized annotations (0.228 on average), but does so by allowing only a few annotations (1.63 on average). Table 5 also includes an EMPTY% metric defined as the percentage of text fragments that returned no annotations and/or no gold standard annotations A(T)∩C(T)=∅. Pessimistic appears to do well in the precision dimension, but its scarce annotator output produces a large number of empty annotations at 33.8%.

Figure 2 provides graphs of the recall, precision, and F_1 scores on all individual 1000 text fragments, sorted from lowest to highest score, instead of the average-only values shown in Table 5. Mean values of recall, precision, and F_1 equate to the scaled [0,1] area-under-the-curve (AUC) for Figure 2. Precision and recall (equation 9) were calculated as the ratio of matched gold standard annotations to the total number of annotations produced (precision), and to the total number of gold standard annotations (recall).

The recall graph shows that TagME's default configuration and apathetic PTA performed identically as the best configuration for matching the most number of gold standard annotations followed closely by stochastic, delta, and optimistic PTA. Pessimistic PTA performed the worst with respect to matching gold standard annotations due to its conservative nature of only providing 1.63 annotations on average. In regards to precision, the solutions offered by optimistic, delta, and stochastic PTA outperform the default and apathetic configurations. Finally, when both precision and recall are considered together through F_1 measure we see that stochastic, delta, and optimistic configurations perform the best with average F_1-scores of 0.683, 0.677, and 0.672, respectively, compared with apathetic and default scores of 0.521 and 0.493, respectively.

Table 5. Default and recommended values for TagMe's tunable parameters (first 3 rows), followed by summary statistics (mean values) for comparison metrics computed on 1000 text fragments gold-standard using tunable parameters' default values (1st column) and values recommended by different forms of PTA.

	Default	Pessimistic f_c: ↑ A(T)∩C(T) f_e: ↓ A(T)\C(T)	Apathetic f_c: ↑A(T)∩C(T) f_e: ↓C(T)\A(T)	Optimistic f_c: ↑ A(T) f_e: ↓A(T)\C(T)	Stochastic f_c: ↑ A(T) f_e: ↓ E[A(T)\C(T)]	Delta f_c: ↑ ±A(T)-C(T) f_e: ↓C(T)\A(T)
*epsilon	0.30	0.494	0.282	0.156	0.427	0.357
*long_text	0	6	0	10	10	7
*rho	0.0	0.551	0.0221	0.2662	0.1613	0.2429
C(T)	4.031	4.031	4.031	4.031	4.031	4.031
A(T)	**9.567**	1.631	8.778	3.721	4.915	4.029
A(T)∩C(T)	**3.347**	1.403	3.346	2.757	3.128	2.861
C(T)\A(T)	0.684	2.628	**0.685**	1.274	0.903	1.17
A(T)\C(T)	6.22	**0.228**	5.432	0.964	1.787	1.168
PRECISION	0.375	0.607	0.407	**0.728**	0.659	0.710
RECALL	**0.827**	0.324	**0.827**	0.673	0.768	0.699
F-SCORE	0.493	0.401	0.521	0.672	**0.683**	0.677
EMPTY%	**1.6%**	33.8%	**1.6%**	6.9%	2.8%	5.6%

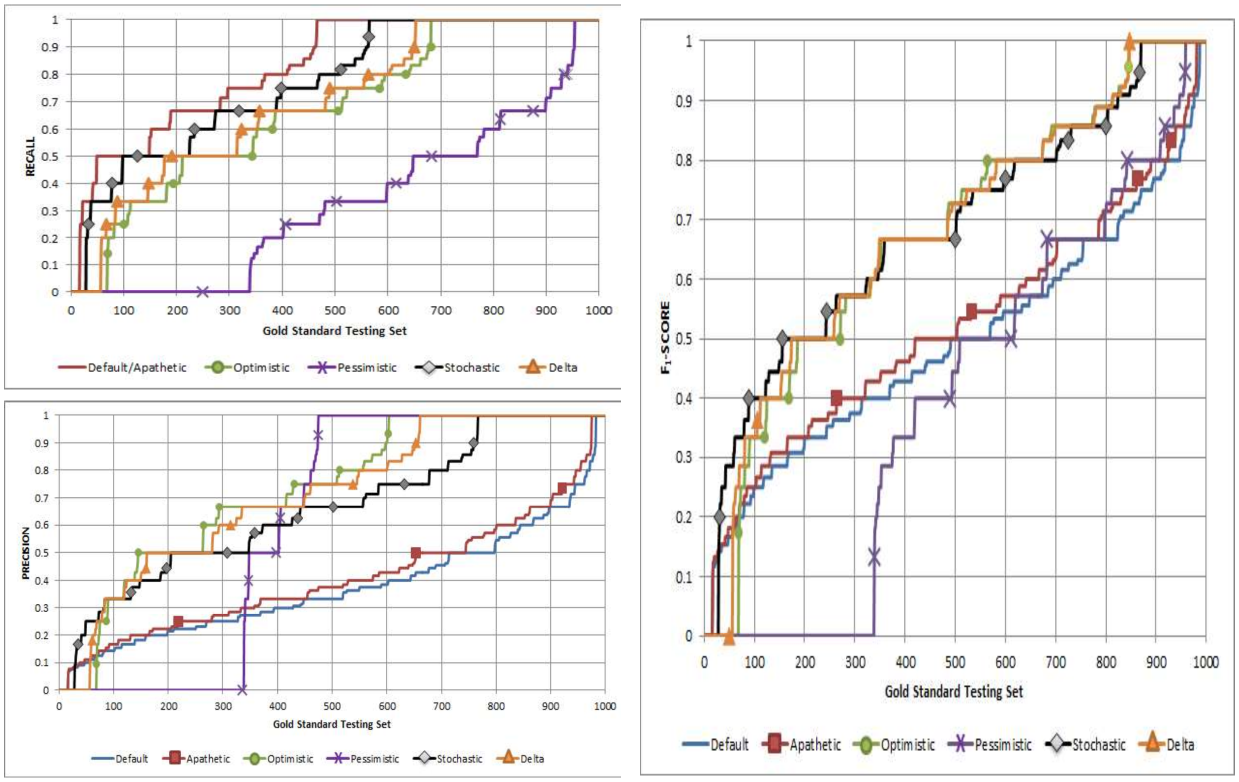


Figure 2. Graphs of recall (top left), precision (bottom left), and F_1-score (right) for each variant of PTA including default configuration used by TagME on 1000 text fragments.

The results indicate that if the objective is to match as many as possible gold standard annotations, without concern for unrecognized annotations, TagME's default or PTA's apathetic solution would be the best option, as they most closely achieve C(T) ⊆ A(T). However, if the goal is to avoid uncertainty, i.e., A(T) ⊆ C(T), then either optimistic, delta, or stochastic PTA would be the best candidates. Finally, for the behaviour that closely resembles the gold standard output, i.e., C(T)=A(T), stochastic PTA would be a good choice.

V.2 PTA with WikipediaMiner

WikipediaMiner is a toolkit that provides semantic services, including semantic annotation through a downloadable software library or Web service. The default configuration along with the solutions recommended by the five variants of PTA are given in Table 6. Pessimistic PTA discovered a solution very similar to WikipediaMiner's default configuration. Apathetic PTA produced the highest number of annotations per text fragment A(T) with an average of 14.23 annotations, but at the expense of the highest number of unknowns A(T)\C(T) (11.2 on average). This resulted in a high recall of 0.75, but low precision of 0.35.

Table 6. Default and recommended values for WikipediaMiner's tunable parameters (first 3 rows), followed by summary statistics (mean values) for comparison metrics computed on 1000 text fragments gold-standard using tunable parameters' default values (1st column) and values recommended by different forms of PTA.

	Default	Pessimistic f_c: ↑ A(T)∩C(T) f_e: ↓ A(T)\C(T)	Apathetic f_c: ↑A(T)∩C(T) f_e: ↓C(T)\A(T)	Optimistic f_c: ↑ A(T) f_e: ↓A(T)\C(T)	Stochastic f_c: ↑ A(T) f_e: ↓ E[A(T)\C(T)]	Delta f_c: ↑ ±A(T)-C(T) f_e: ↓C(T)\A(T)
*minProbability	0.50	0.53	0.023	0.33	0.23	0.46
*disambiguation Policy	strict	strict	loose	strict	loose	strict
*weight	0.0	0.205	0.0427	0.427	0.663	0.199
C(T)	4.031	4.031	4.031	4.031	4.031	4.031
A(T)	3.862	3.862	**14.229**	4.435	6.027	4.185
A(T)∩C(T)	2.099	2.099	**3.029**	2.303	2.718	2.217
C(T)\A(T)	1.932	1.932	**1.002**	1.728	1.313	1.814
A(T)\C(T)	**1.763**	**1.763**	11.2	2.132	3.309	1.968
PRECISION	**0.525**	**0.525**	0.245	0.519	0.468	0.519
RECALL	0.511	0.511	**0.755**	0.566	0.674	0.543
F-SCORE	0.485	0.485	0.350	0.508	**0.525**	0.498
EMPTY%	15.3%	16.8%	**2.9%**	11.6%	5.1%	13.4%

The recall graph of Figure 3 demonstrates that apathetic and stochastic PTA perform the best in this metric, followed by optimistic, delta, default, and pessimistic performing relatively the same. The results were opposite for the precision, with pessimistic, default, delta, and optimistic performing well followed closely by stochastic then lastly apathetic due to its high unknown count. Stochastic PTA gave the best overall solution with comparable precision to the default configuration, but with significantly better recall. In addition, stochastic PTA generated the second-least number of empty annotations with 5.1% compared to the second-worst score of 15.3% from the default configuration.

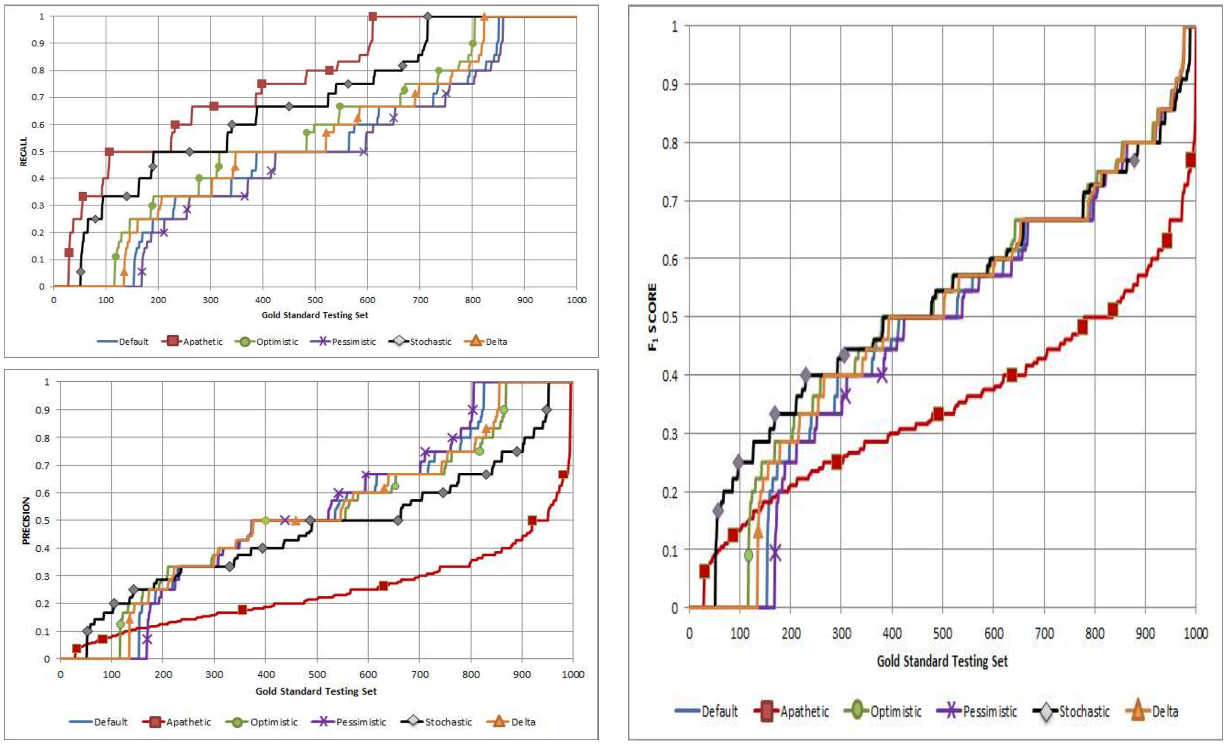


Figure 3. Graphs of recall (top left), precision (bottom left), and F_1-score (right) for each variant of PTA including default configuration used by WikipediaMiner on 1000 text fragments.

V.3 PTA with DBpedia Spotlight

We tested PTA on the DBPedia Spotlight *candidates* Web service. The candidates service is an annotation service that returns a ranked list of annotations per mention. This ranking includes output statistics entitled *contextual score, percentage of second rank,* and *final score* with the intent that the application using the annotator service will prune the ranked list based on the chosen thresholds for these statistics. Consequently, the goal of PTA is to discover what threshold values to use. Therefore, it was not surprising that the default configuration gave the best results for recall (0.52, Table 7) but poor precision (0.19), and the highest number of annotations per text fragment (14.8). Pessimistic PTA had the most difficulty with the lowest F_1-score (0.02) caused by a high number of unrecognized annotations and low number of matching gold standard answers. Stochastic PTA would have provided the best precision if not for the many empty solutions it gave (Figure 4 and Table 7). Overall, optimistic PTA gave the second best recall with the best precision and thus is the recommended solution with the top F_1-score of 0.38.

V.4 PTA with Yahoo Content Analysis (YCA)

The Yahoo Content Analysis API provides named entity linking to Wikipedia entities through its Web service. PTA mean statistics show apathetic PTA leads in seven of the eight metrics (Table 8). However, the difference is statistically inconsequential and did not translate to a significant improvement over the default configuration. The AUC graphs of Figure 5 reveal stochastic, apathetic, and delta forms of PTA perform equally well as YCA's default configuration using different local maxima solutions. Optimistic PTA was a close second while pessimistic struggled.

Table 7. Default and recommended values for Spotlight's tunable parameters (first 5 rows), followed by summary statistics (mean values) for comparison metrics computed on 1000 text fragments gold-standard using tunable parameters' default values (1st column) and values recommended by different forms of PTA.

	Default	Pessimistic f_c: ↑ A(T)∩C(T) f_e: ↓ A(T)\C(T)	Apathetic f_c: ↑A(T)∩C(T) f_e: ↓C(T)\A(T)	Optimistic f_c: ↑ A(T) f_e: ↓A(T)\C(T)	Stochastic f_c: ↑ A(T) f_e: ↓ E[A(T)\C(T)]	Delta f_c: ↑ ±A(T)-C(T) f_e: ↓C(T)\A(T)
*confidence	0.20	0.14	0.029	0.29	0.62	0.020
*support	20	48	5	1	1	355
*contextualScore	0.0	0.132	0.0508	0.0	0.0	0.115
*percentageOf SecondRank	0.0	0.0290	0.596	0.0	0.0	0.459
*finalScore	0.0	0.491	0.128	0.0	0.228	0.004
C(T)	4.031	4.031	4.031	4.031	4.031	4.031
A(T)	**14.89**	0.072	4.227	5.042	1.496	3.153
A(T)∩C(T)	**2.085**	0.055	1.477	1.8	0.878	1.022
C(T)\A(T)	**1.946**	3.976	2.554	2.231	3.153	3.009
A(T)\C(T)	12.8	**0.017**	2.75	3.242	0.618	2.131
PRECISION	0.190	0.0373	0.403	**0.406**	0.403	0.334
RECALL	**0.522**	0.0157	0.372	0.451	0.218	0.263
F-SCORE	0.251	0.0211	0.345	**0.384**	0.259	0.262
EMPTY%	**12.4%**	96%	22.6%	17.6%	48.5%	36%

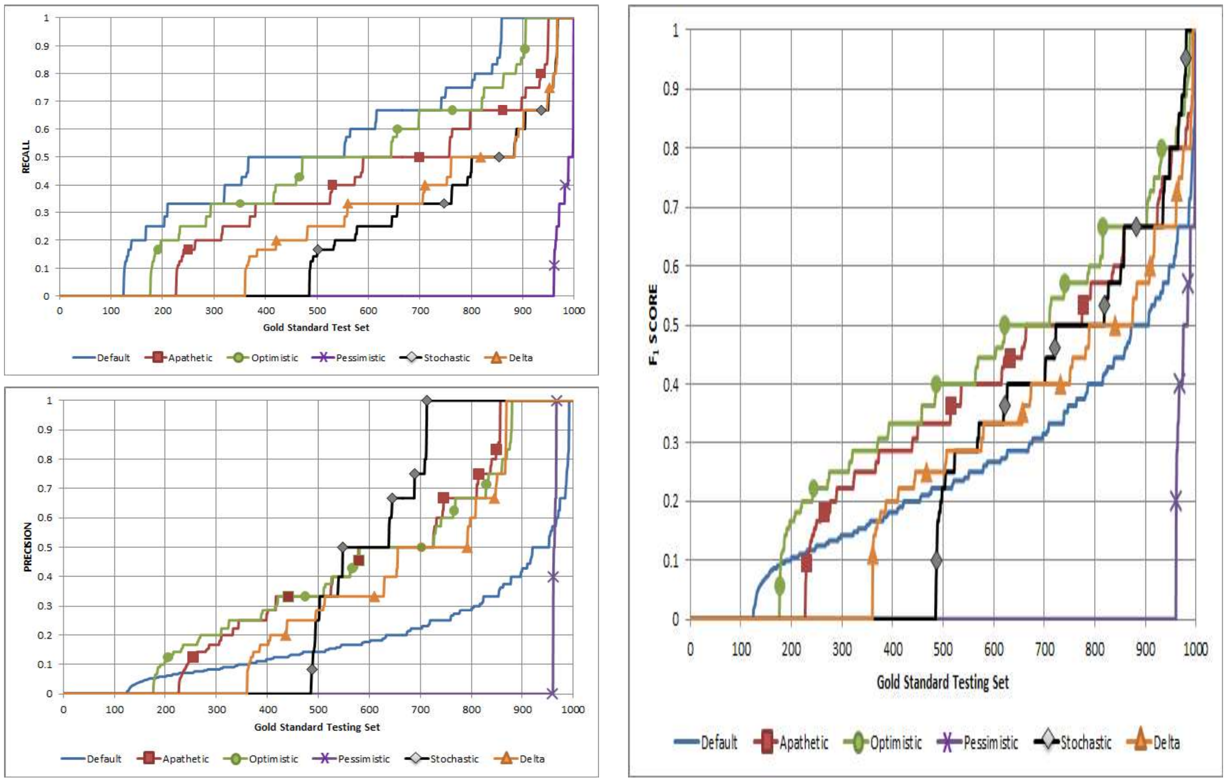


Figure 4. Graphs of recall (top left), precision (bottom left), and F_1-score (right) for each variant of PTA including default configuration used by DBpedia Spotlight on 1000 text fragments.

Table 8: Default and recommended values for YCA's tunable parameters (first 2 rows), followed by summary statistics (mean values) for comparison metrics computed on 1000 text fragments gold-standard using tunable parameters' default values (1st column) and values recommended by different forms of PTA.

	Default	Pessimistic f_c: ↑ A(T)∩C(T) f_e: ↓ A(T)\C(T)	Apathetic f_c: ↑A(T)∩C(T) f_e: ↓C(T)\A(T)	Optimistic f_c: ↑ A(T) f_e: ↓A(T)\C(T)	Stochastic f_c: ↑ A(T) f_e: ↓ E[A(T)\C(T)]	Delta f_c: ↑ ±A(T)-C(T) f_e: ↓C(T)\A(T)
*max	100	9	18	6	99	98
*score	0.0	0.917	0.455	0.594	0.192	0.007
C(T)	4.031	4.031	4.031	4.031	4.031	4.031
A(T)	1.439	0.243	**1.457**	1.233	1.444	1.443
A(T)∩C(T)	1.071	0.199	**1.082**	0.938	1.075	1.070
C(T)\A(T)	2.960	3.832	**2.949**	3.093	2.956	2.961
A(T)\C(T)	0.368	**0.044**	0.375	0.295	0.369	0.373
PRECISION	0.520	0.160	**0.529**	0.511	0.525	0.521
RECALL	0.268	0.0491	**0.271**	0.238	0.268	0.266
F-SCORE	0.331	0.0719	**0.335**	0.306	0.332	0.329
EMPTY%	39.9%	83.6%	**38.7%**	42.4%	39.2%	39.6%

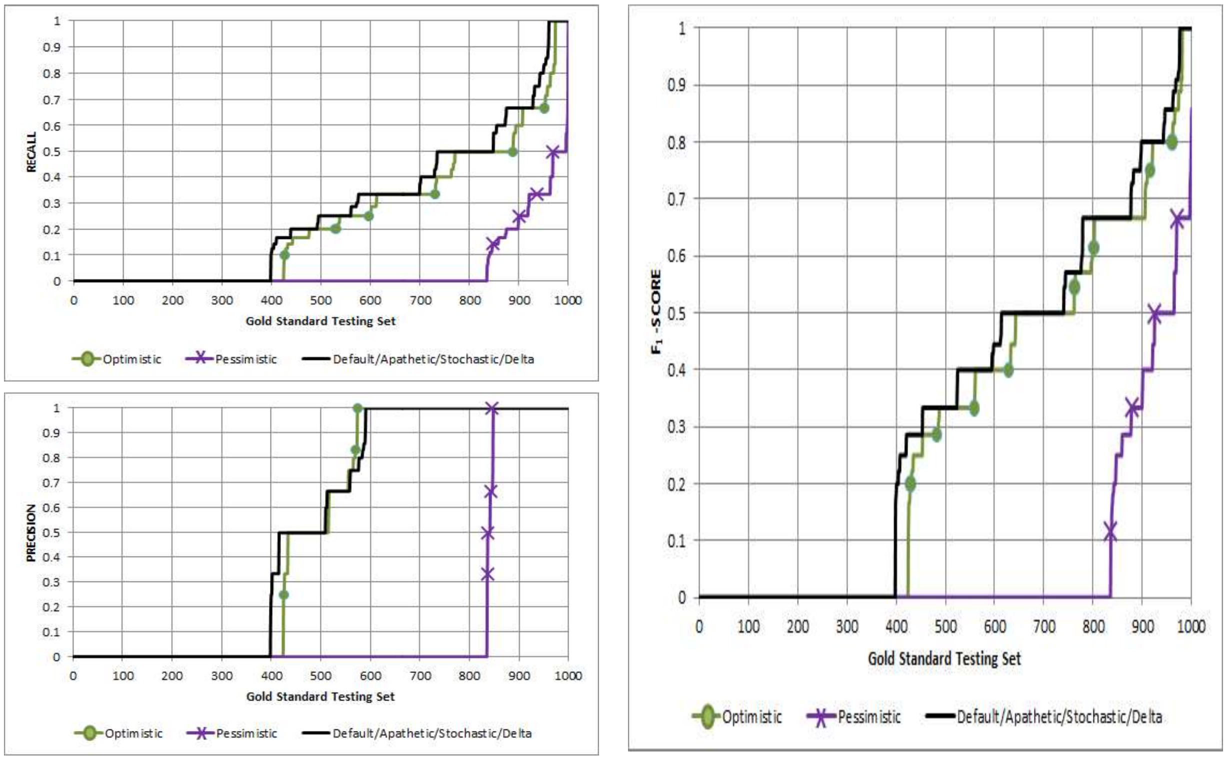


Figure 5: Graphs of recall (top left), precision (bottom left), and F_1-score (right) for each variant of PTA including default configuration used by YCA on 1000 text fragments.

VI. Result Summary

Table 9 provides a summary of the experimental results. For each examined annotator, the table presents the forms of PTA that evaluated as the best(+) and the worst(-) performing for mean measures of: most annotations, most annotations matching the gold standard, least unrecognized annotations, least empty answers and best precision/recall. Also included is the overall (recommended) PTA variant based on AUC graphs. Noteworthy is that the best PTA variant per individual dimensions does not necessarily indicate best performing overall solution. As an example, consider WikipediaMiner. Apathetic PTA scored highest for most annotations, most matched, least empty, and best recall; however, the recommended solution is stochastic which was not the top (nor bottom) of any single measure. The table also shows that no single variant of PTA is best as this is a function of annotator behavior, testing set, and user assumptions (unknown annotations are mostly right, mostly wrong, or partially correct). Nonetheless, it would appear pessimistic PTA is excessively conservative and is often surpassed by its probabilistic counterpart: stochastic. Consequently, the pessimistic PTA variant should be avoided. From Table 9, the following summary observations can be made:

- Apathetic PTA emerged as a good choice for most annotations, most matched and best recall.
- Optimistic PTA performed well when precision is the primary concern.
- Stochastic PTA was the prevailing general-purpose strategy with an overall best solution for three of the four annotators tested.

Table 9: Summary table indicating the best(+) and the worst(-) solutions for individual measures by averages plus overall recommended solution based on AUC graphs.

	most Annotations A(T)	**most matched A(T)∩C(T)**	**least unknown A(T)\C(T)**	**least empty answers EMPTY %**	**best recall**	**best precision**	**Overall best AUC-F$_1$**
TagME	+default -pessimistic	+default +apathetic -pessimistic	+pessimistic -default	+default +apathetic -pessimistic	+default +apathetic -pessimistic	+optimistic -default	+stochastic -pessimistic
WikipediaMiner	+apathetic -default -pessimistic	+apathetic -default -pessimistic	+default +pessimistic -apathetic	+apathetic -pessimistic	+apathetic -default -pessimistic	+default +pessimistic -apathetic	+stochastic -apathetic
DBpedia Spotlight	+default -pessimistic	+default -pessimistic	+pessimistic -default	+default -pessimistic	+default -pessimistic	+optimistic -pessimistic	+optimistic -pessimistic
Yahoo (YCA)	+apathetic -pessimistic	+apathetic -pessimistic	+pessimistic -apathetic	+apathetic -pessimistic	+apathetic -pessimistic	+apathetic -pessimistic	+default +apathetic +stochastic +delta -pessimistic

Our experimental results demonstrate that PTA is effective. In all annotators tested, PTA either: 1) suggested an improved configuration, or 2) validated the default configuration as a local maximum solution. PTA successfully found alternative configurations that better fitted the testing set than the default parameters of TagME, WikipediaMiner, and DBpedia Spotlight.

VII. Related Literature

Semantic annotation tools offer the possibility of deeper analysis of textual content by disambiguating terms and phrases present in the text and linking them to appropriate concepts from a knowledge base, hence enabling more efficient and accurate classification, organization, search and retrieval of textual content. The research community has already developed a critical mass of both research prototypes and usable software products in this area. Automated semantic annotation tools examined in this paper: TagMe (Ferragina & Scaiella, 2012), Denote (Cuzzola et al., 2013), DBPedia Spotlight (Mendes, Jakob, García-Silva & Bizer, 2011), Wikipedia Miner (Milne & Witten, 2013), among others, provide means to automatically semantically process textual content and identify relevant semantic concepts.

Recently, these tools have been increasingly referred to as entity linking tools since they link entity mentions in the text with the corresponding entry/entries in a knowledge base. Shen, Wang & Han (2015) have provided a very comprehensive and detailed survey of the state-of-the-art automated entity linking (i.e., semantic annotation) tools. They have identified three modules that such tools consist of, namely candidate entity generation, candidate entity ranking and unlinkable menton prediction modules, and for each module presented and analyzed different methods and techniques that were proposed in the literature and applied in existing annotation tools. They also reported that the examined tools "differ along multiple dimensions and are evaluated over different data sets".

Large majority of today's semantic annotators rely on a general purpose knowledge base (KB) such as Wikipedia or more structured and semantically rich KBs like DBpedia, YAGO, and Wikidata. However, there are also domain-specific semantic annotators, many of them in the biomedical domain, e.g., MetaMap (Aronson & Lang, 2010) and NCBO annotator (Whetzel et al., 2013), that rely on domain-specific KBs such as Unified Medical Language System (UMLS) (Bodenreider, 2004), DrugBank (Law et al., 2014) or medical ontologies available at NCBO BioPortal (Whetzel et al., 2011). In order to assure good domain coverage, both in terms of breadth and depth of the entities covered, some annotation tools rely on more than one KB. For instance, the semantic annotation method proposed by Berlanga, Nebot & Pérez (2015) relies on the use of several KBs of arbitrary size and domain specificity, and is also independent of any specific characteristic of a KB (e.g., disambiguation pages, internal links and other Wikipedia specific features). Some semantic annotators, such as TagMe and the one developed by WalmartLabs (Gattani et al., 2013), are specifically designed and developed for semantic annotation of social media content that is typically short, fragmented, poorly spelled and grammatically incorrect. Since such an annotator requires a global and almost real-time KB, Gattani et al. (2013) expanded Wikipedia with data from various structured sources (e.g., Adam (health), MusicBrainz (albums), City DB, and Yahoo Stocks), as well as new interesting events extracted from Twitter data stream.

The algorithms developed as a part of semantic annotation systems, e.g. spot identification and disambiguation, often rely on user-dependent fine-tuning of input parameters. These parameters allow the user to customize the algorithms for the specific topic or textual content type (Jovanovic et al., 2014). While annotation tools do provide a suggested default value on these parameters, they are not by any means optimal for all scenarios. To our knowledge, no existing related work or implemented software has attempted to address the problem of automated parameter fine-tuning for semantic annotation tools; therefore, our proposed PTA framework serves as a first foundational step in this direction.

Having said that, it is important to point out that while researchers have not addressed the problem of parameter fine-tuning for semantic annotators, there have been a few fruitful work on the systematic evaluation of semantic annotation systems. For instance, Cornolti, Ferragina & Ciaramita (2013) have developed a framework consisting of metrics and standard datasets for measuring the efficiency and effectiveness of existing semantic annotation tools. The major contribution of their work is its novel approach to the systematic classification of different tasks of a semantic annotator and the development of suitable metrics for each of these specific tasks. The work by Steinmetz, Knuth & Sack (2013) is also focused on the evaluation of semantic annotation tools; however, their attention is centered more on the statistical analysis of different benchmark and dictionary datasets that can be used in the evaluation process. Heuss, Humm, Henninger & Rippl (2014) compared the performance of several state-of-the-art semantic annotation tools on domain specific texts (namely texts about museum collections). The study found that, on average, each tool achieved roughly just a third of its F1 score on texts covering general/common topics. The results also showed very high standard deviations for all performance measures (recall, precision and F1), indicating not only lower performance than in a common case, but wider distribution of the results. This is consistent with the findings of Shen, Wang, & Han (2015) who reported that the tools examined in their survey tend to perform very differently for different data sets and domains. It is worth pointing out that existing comparative studies of semantic annotation tools have all relied on the suggested default parameter settings for the compared systems; therefore, they do not necessarily reflect the best case performance of the annotator tool on the objects of the experiment. It could very well be the case that if the optimal parameter values were chosen (as opposed to the default values) that the obtained results could be significantly different.

It should be mentioned that besides automated semantic annotation tools, there are also semi-automated semantic annotators that allow for user's intervention during the annotation process. This intervention often takes the form of choosing the best option from a list of candidate annotations, or removing some of the proposed annotations that the user considers incorrect or irrelevant. While considerable research and development efforts were put into the design and development of semi-automated annotation tools (Uren et al., 2005), (Oliveira & Rocha, 2013), their reliance on human active participation impacts their efficiency, and thus they have been largely superseded by fully automated tools. Still, there are some usage scenarios, e.g., scholarly reading, where, due to their mixed-initiative annotation approach, semi-automated annotators are preferred. For instance, based on a study of scholarly annotation practices, Müller-Birn, Klüwer, Breitenfeld, Schlegel, & Benedix (2015) have designed and developed Neonion, a lightweight annotation tool for creating, sharing and reusing annotation data. Neonion users can accept, reject or modify annotations recommended by the tool; annotations take the form of references to appropriate Wikidata (Vrandečić & Krötzsch, 2014) entities. This feedback that users provide is leveraged for improving subsequent recommendations. Annotation of medical texts is another domain where human involvement is often needed to assure the accuracy of automatically produced annotations. For example, RapTAT is a semi-automated semantic annotation tool based on an interactive and iterative machine learning approach, and aimed at assisting end users with annotation of various kinds of medical texts (Gobbel et al., 2014). In each iteration, the tool annotates potentially relevant phrases within a document, presents the annotations to a reviewer for correction, and then uses the obtained feedback (i.e., corrected annotations) to re-train its machine learning model before annotating subsequent document.

Although work on finding optimal parameter values for semantic annotation tools is novel, it is important to point out that the use of evolutionary algorithms as Genetic Algorithms for optimal parameter estimation in control systems has been a commonplace (Chang, 2006). For instance, Seng et al (1999) used Genetic Algorithms to simultaneously tune the parameters of a self-tuning fuzzy logic control system. Similarly, Yao & Sethares (1994) used Genetic Algorithms for optimizing the structure and parameters of feedforward and recurrent neural networks and shown to be able to reduce estimation error in probability to zero. Genetic Algorithms have also been widely used for parameter tuning in Proportional-Integral-Derivative (PID) controllers (Panda, 2011), power system optimization (Kothari, 2012) and HVAC systems (Kusiak, Tang & Xu, 2011). They were also used to deal with challenges of image annotation (Bahrami & Abadeh, 2014). In the Semantic Web domain, Genetic Algorithm-based approaches can be observed in a variety of applications (Chen, Wu & Cudré-Mauroux, 2012) such as finding optimal ontology alignments between multiple ontologies (Martinez-Gil, Alba & Aldana-Montes, 2008); identification and alignment of datasets of the Linked Open Data cloud (Gunaratna, Lalithsena & Sheth, 2014); RDF query answering (Oren, Guéret & Schlobach, 2008); and semantic Web service discovery (Sangers, Frasincar, Hogenboom & Chepegin, 2013) and composition (Fanjiang & Syu, 2014), among others, and have shown to be effective for such optimization problems.

VIII. Limitations and Future Work

We have proposed a method, called Parameter Tuning Architecture (PTA), for finding suitable values for tunable parameters of semantic annotators (Section IV) and demonstrated this method on four semantic annotators (Section V). Note that the annotators themselves were not trained by PTA; instead, alternative parameter values were proposed that better suited the targeted evaluation/testing set (*wiki-annot30*). PTA could be extended for semantic annotator training, namely, as a tool to decide on the fallback parameter defaults when no alternative values are supplied. This would simply require the use of a training set rather than a testing set with PTA. However, since we were unable to acquire the exact training set used for each of the four annotators in our experiments, we were not able to compare how general purpose default values suggested by PTA would perform relative to the defaults currently chosen by the annotators' designers. Even if the training sets were available, there may be *internal* tunable parameters that are only accessible to the designers, and not exposed through the annotator's public interface. Moreover, since each of the tested annotators were trained with different gold standards on (most-likely) different versions of Wikipedia, subsequent work should include experimentation with other datasets such as Microsoft's "*Entity Recognition and Disambiguation Challenge*"[6] and/or *ClueWeb*[7] for further validation of our PTA framework.

Future work may also include validation with other knowledge based annotation systems (non-Wikipedia) that rely on life sciences, biomedical, or other ontologies. Additionally, we propose an extension to our fitness function (equation 1) with the introduction of weights (δ_c, δ_e) :

$$FIT(T) = arg\ max_{v \varepsilon V} \delta_c \left[f_c(T)\right]^2 - \delta_e \left[f_e(T)\right]^2 \ where\ \delta_c + \delta_e = 1$$

This would allow for an emphasis toward either finding more correct annotations f_c or less annotations in error f_e. The weights themselves (δ_c, δ_e) could be part of the tunable parameter vector v in which

[6] http://web-ngram.research.microsoft.com/ERD2014/

[7] http://lemurproject.org/clueweb09/FACC1/

PTA would be tasked to not only find suitable parameter values for the annotator in question but also suggesting parameter values for itself. Finally, our future work will investigate other evolutionary algorithms such as swarm optimization and compare its performance with the currently used genetic algorithm (Figure 1).

IX. Conclusion

In this paper we present Parameter Tuning Architecture (PTA), a general method to determine "best-fit" values of configuration parameters for semantic annotators. We explain the caveats of supervised testing specific to semantic annotators and devised PTA variants to tackle the uncertainty of unlabeled annotations. We tested these variants on four well-known semantic annotators and provided a method for selecting the best solution using a genetic algorithm and area-under-the-curve metric. Experimental results indicate that PTA is capable of suggesting configurable parameters that improve upon specific individual areas of most annotations, most matched gold standard answers, and least uncertainty. Finally, our tests demonstrate that PTA can consistently find a configuration that provides an overall best solution, i.e., solution with the best precision versus recall trade-off for many semantic annotators. We balance our findings by acknowledging the limitations of our work and propose five future research directions for further study (Section VIII).

Since the PTA fitness function, the core component of the proposed method, does not rely on any annotator-specific feature, our PTA method is applicable to any semantic annotator, and can be used to enhance the annotator's performance on any specific annotation task. Besides being directly beneficial for semantic annotation tools, the proposed method might also be indirectly useful to any intelligent system that relies on semantic-rich annotation of textual content, such as text search and retrieval systems, various content-based recommender systems, systems that rely on semantics of textual content to support personal or business decision making and the like.

References

A. R. Aronson and F.-M. Lang. (2010). *An overview of MetaMap: historical perspective and recent advances*, 17(3), pp.229-236, JAMIA. http://metamap.nlm.nih.gov/

S. Atdağ, and V. Labatut. (2013). *A Comparison of Named Entity Recognition Tools Applied to Biographical Texts*, (pp. 228-233). 2nd International Conference on Systems and Computer Science.

S. Bahrami and M.S. Abadeh, (2014). *Automatic image annotation using an evolutionary algorithm (IAGA)*, pp.320 - 325, 7th International Symposium on Telecommunications (IST 2014).

R. Berlanga, V. Nebot, M. Pérez. (2015). *Tailored semantic annotation for semantic search*, 30, pp. 69-81, Web Semantics: Science, Services and Agents on the World Wide Web.

O. Bodenreider. (2004). *The Unified Medical Language System (UMLS): integrating biomedical terminology*, 32 (Database-Issue), pp. 267-270, Nucleic Acids Research. http://www.nlm.nih.gov/research/umls

W.D. Chang, (2006). *An improved real-coded genetic algorithm for parameters estimation of nonlinear systems,* 20(1), pp. 236-246. Mechanical Systems and Signal Processing.

H. Chen, Z. Wu & P. Cudré-Mauroux, (2012). *Semantic Web Meets Computational Intelligence: State of the Art and Perspectives,* 7(2), pp. 67-74, IEEE Computational Intelligence Magazine.

R. Chiong, and O.K. Beng, (2007). *A Comparison between Genetic Algorithms and Evolutionary Programming based on Cutting Stock Problem.* vol. 14, pp. 72-77. Engineering Letters.

M. Cornolti, P. Ferragina, M. Ciaramita, (2013), *A Framework for Benchmarking Entity-Annotation Systems,* pp. 249-260. 22nd International World Wide Web Conference.

J. Cuzzola, D. Gasevic, E. Bagheri, Z. Jeremic, J. Jovanovic, R. Bashash, (2013). *Semantic Tagging with Linked Open Data,* vol. 1054, pp 52-53. 4th Canadian Semantic Web Symposium (CSWS 2013).

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, et al. (2012), *Knowledge vault: a web-scale approach to probabilistic knowledge fusion,* pp. 601-610, 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '14).

Y. Fanjiang and Y. Syu, (2014). *Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach*, 56(3), pp. 352-373, Information and Software Technology.

P. Ferragina, U. Scaiella, (2012). *Fast and Accurate Annotation of Short Texts with Wikipedia Pages.* 29(1), pp. 70-75, IEEE Software.

A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, et al. (2013). *Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach*, pp. 1126-1137, Proc. VLDB Endow. 6, 11 (August 2013).

G. Gobbel, J. Garvin, R. Reeves, R.M. Cronin, J. Heavirland et al. (2014). *Assisted annotation of medical free text using RapTAT*, 21(5), pp.833-841, J Am Med Inform Assoc.

J. Grefenstette, (1992). *Genetic Algorithms for Changing Environments,* pp. 139-146, Parallel Problem Solving from Nature 2.

K. Gunaratna, S. Lalithsena, and A.P. Sheth, (2014). *Alignment and Dataset Identification of Linked Data in Semantic Web*, 4 (2), pp. 139-151, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J.R. Curran, (2013). *Evaluating Entity Linking with Wikipedia*, 194, pp. 130-150, Journal of Artificial Intelligence.

T. Heuss, B. Humm, C. Henninger, and T. Rippl. (2014). *A comparison of NER tools w.r.t. a domain-specific vocabulary*, pp.100-107, In Proceedings of the 10th International Conference on Semantic Systems (SEM '14), H. Sack, A. Filipowska, J. Lehmann, and S. Hellmann (Eds.). ACM, New York, NY, USA.

E. Hovy, R. Navigli, S.P. Ponzetto, (2013). *Collaboratively built semi-structured content and Artificial Intelligence: The story so far*, 194, pp. 2-27. Journal of Artificial Intelligence.

J. Jovanovic, E. Bagheri, J. Cuzzola, D. Gasevic, Z. Jeremic, R. Bashash, (2014). *Automated Semantic Annotation of Textual Content*, 16(6), pp. 38-46. IEEE IT Professional.

D.P. Kothari, (2012). *Power system optimization,* pp. 18-21, 2nd National Conference on Computational Intelligence and Signal Processing (CISP).

A. Kusiak, F. Tang, & G. Xu, (2011). *Multi-objective optimization of HVAC system with an evolutionary computation algorithm, 36(*5), pp. 2440-2449, Energy.

V. Law, C. Knox, Y. Djoumbou, T. Jewison, A.C. Guo, et al. (2014). *DrugBank 4.0: shedding new light on drug metabolism*, 42(1), pp.D1091-7, Nucleic Acids Res.

B. Liu, (2012). *Sentiment Analysis and Opinion Mining (Synthesis Lectures on Human Language Technologies)*. Morgan & Claypool Publishers.

J. Martinez-Gil, E. Alba, & J.F. Aldana-Montes, (2008). *Optimizing ontology alignments by using genetic algorithms,* Workshop on Nature based Reasoning for the Semantic Web.

D. Maynard, (2008). *Benchmarking Textual Annotation Tools for the Semantic Web.* 6th International Conference on Language Resources and Evaluation.

P.N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, (2011). *DBpedia spotlight: shedding light on the web of documents.,* pp. 1-8, 7th International Conference on Semantic Systems. ACM.

D. Milne, & I.H. Witten, (2013). *An open-source toolkit for mining Wikipedia.* vol. 194, pp. 222-239, Journal of Artificial Intelligence.

D. Moriarty, A. Schultz, J. Grefenstette, (1999), *Evolutionary Algorithms for Reinforcement Learning,* 11, pp. 241-276, Journal Artificial Intelligence Research (JAIR).

C. Müller-Birn, T. Klüwer, A. Breitenfeld, A. Schlegel, and L. Benedix. (2015). *Neonion: Combining Human and Machine Intelligence*, pp. 223-226, Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing (CSCW'15 Companion).

P. Oliveira, and J. Rocha. (2013). *Semantic annotation tools survey*, pp.301-307, IEEE Symposium on Computational Intelligence and Data Mining (CIDM).

E. Oren, C. Guéret, & S. Schlobach, (2008). *Anytime query answering in RDF through evolutionary algorithms,* pp. 98-113, 7th International Semantic Web Conference (ISWC 08).

S. Panda, (2011). *"Multi-objective PID controller tuning for a FACTS-based damping stabilizer using Non-dominated Sorting Genetic Algorithm-II,*, 33(7), pp. 1296-1308, International Journal of Electrical Power & Energy Systems.

L. Ratinov and D. Roth, (2009). *Design challenges and misconceptions in named entity recognition*, (pp. 147-155). Thirteenth Conference on Computational Natural Language Learning (CoNLL '09).

J. Sangers, F. Frasincar, F. Hogenboom, and V. Chepegin, (2013). *Semantic Web service discovery using natural language processing techniques*, 40(11), pp. 4660-4671, Expert Systems with Applications.

T.L. Seng, M. Bin Khalid, & R. Yusof, (1999). *Tuning of a neuro-fuzzy controller by genetic algorithm,* 29(2), pp. 226-236, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Trans.,.

W. Shen, J. Wang, and J. Han. (2015). *Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions,* 27(2), pp.443-460, IEEE Transactions on Knowledge & Data Engineering.

N. Steinmetz, M. Knuth & H. Sack, (2013). *Statistical Analyses of Named Entity Disambiguation Benchmarks,* pp. 21-25, 1st International Workshop on NLP and DBpedia.

H. Szczerbicka, M. Becker, M. Syrjakow, (1998). *Genetic Algorithms: a Tool for Modelling, Simulation, and Optimization of Complex Systems,* 29(7), pp. 639-659, Cybernetics and Systems.

V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, S. Ciravegna, (2005). *Semantic annotation for knowledge management: Requirements and a survey of the state of the art.* 4(1). pp. 14-28. Journal of Web Semantics.

D. Vrandečić, M. Krötzsch. (2014). *Wikidata: A Free Collaborative Knowledge Base*. 57(10). pp. 78–85. Communications of the ACM.

J. Weston, A. Bordes, O. Yakhnenko, N. Usunier, (2013). *Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction,* (pp.1366-1371), Conference on Empirical Methods in Natural Language Processing.

P.L. Whetzel, N.F. Noy, N.H. Shah, P.R. Alexander, C. Nyulas, et al. (2011). *BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications*, 39, pp.W541-5. Nucleic Acids Res. http://bioportal.bioontology.org/

P.L. Whetzel and NCBO Team. (2013). *NCBO Technology: powering semantically aware applications*, 4 (Suppl 1), S8, J. Biomed. Semantics.

Y. Yan, N. Okazaki, Y. Matsuo, Z. Yang, and M. Ishizuka, (2009). *Unsupervised relation extraction by mining Wikipedia texts using information from the web*. vol. 2, pp. 1021-1029. Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP.

L. Yao, & W. A. Sethares, (1994). *Nonlinear parameter estimation via the genetic algorithm,*, 42(4), pp. 927-935, Signal Processing, IEEE Transactions on.