ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse34952020.pdf
https://doi.org/10.30534/ijatcse/2020/34952020

Application of the Swarm Intelligence for the Semantic Annotation of the Web of Things

Ismail NADIM[1], Yassine El Ghayam[2], Abdelalim Sadiq[3]
[1]Ibn Toufail University Faculty of science Kenitra, Morocco, ismail.nadim.gi@gmail.com
[2]SMARTiLab EMSI Rabat Honoris Universities, Morocco, yassine.elgh@gmail.com
[3]Ibn Toufail University Faculty of Science Kenitra, Morocco, a.sadiq@uit.ac.ma

ABSTRACT

Entity Linking (EL) is an important tool for many researches including the Internet of Things (IoT). Particularly, it helps to automate the annotation of the IoT devices and services with semantic vocabulary. This semantic annotation improves the interoperability between IoT devices during services discovery and composition. This paper describes an EL-based semantic annotation framework. The latter processes a web table to extract the mentions, generates candidate entities from the Knowledge base (KB) for each mention and then disambiguates the mentions to the correct entry in the KB. The main contribution of this paper is the use of both genetic algorithm (GA) and Artificial Bee Colony (ABC) to handle the combinatorial problem of the collective disambiguation. The proposed framework implements, in addition, some text preprocessing techniques such as the bag of words (BoW) model and provides a tool for IoT data collection. The framework is a first step to demonstrate how swarm intelligence (SI) can exceed the numerical approaches and even some meta-heuristic techniques like the probabilistic based ones.

Key words: Artificial Bee Colony, Entity Linking, Genetic Algorithm, Semantic Web of Things

1. INTRODUCTION

The Internet of Things (IoT) [1] is a system of connected objects leveraging the information and communication technologies (ICT). The Web of Things (WoT) [2] brings the IoT to the Web in order to improve the interoperability between the IoT devices. For this end, the WoT leverages well-known Web technologies such as HTTP and Web APIs to publish the IoT devices as Web resources. Furthermore, the semantic Web of Things (SWOT) is a combination between the WoT and the semantic web [3] that attempts to describe the WoT resources with semantic vocabulary in order to reach the semantic interoperability between the devices. The semantic interoperability [4] which means the ability for the IoT devices to exchange comprehensible data is crucial to minimize the human intervention in IoT systems. The task of describing WoT resources with meaningful vocabulary leveraging semantic Web technologies such as ontologies is known as the semantic annotation [5].

Practically, the semantic annotation should be automatic because of the huge number of the WoT data to be annotated. For this end, the famous Entity linking (EL) [6] task is used. EL task consists in linking a piece of data called mention from a source document to the entity it represents in a knowledge base (KB) through three steps. Given a source document, the first step is the detection of the mentions to be annotated. Once the mentions are defined, the second step consists in generating a set of candidate entities for each mention. Finally, to be able of selecting the correct entity of a mention a third step is necessary: the disambiguation. The disambiguation consists in ranking the discovered candidate sets using some features in order to map each mention to the entity it represents the best in the KB [7].
The remaining of this paper is organized as follows: in Section 2, a general overview of the problem is presented. Section 3 describes the data pre-processing task. Section 4 details the join inference of the disambiguation task based on the swarm intelligence techniques. Section 5 discusses the obtained results. And Section 6 provides conclusions and work perspectives.

2. POBLEM DESCRIPTION

Formally, given a text document D, a knowledge base KB and N mentions M=$\{m_1,m_2,...,m_N\}$, M in D. The EL task consists in identifying a set of entities $E=\{e_1,e_2,...,e_N\}$, E in KB such as : e_i represents the reference entity of the mention m_i, in [1,N].
The disambiguation process can be considered as the most difficult step of the EL task, because through it a decision should be made. Two disambiguation approaches are worth to site here; the first is the local disambiguation [8] which ranks the candidate entities using some features. After that, the best-ranked entities are mapped to their corresponding mentions. This approach does not consider the interdependencies between the candidate entities. The second is the global or collective disambiguation [9], which considers that the correct disambiguation entities are not only the most similar to their corresponding mentions but further are the most “coherent” concepts. This approach has many advantages, especially if the data to be annotated are stored in form of tables with rows and columns [10][11]. However, the most important drawback of the collective disambiguation remains the hard combinatorial problem that will be generated during the inference of the reference

entities. This problem can be tackled basing on numerical approaches by computing and comparing all the possible solutions. However, in most cases, a brute force approach to finding the optimal combinations is not feasible because of the complexity of the optimization problem to be resolved. For this reason, metaheuristic techniques like swarm intelligence [12] and evolutionary algorithms [13] could be preferred to face the complexity of this inference problem. Genetic algorithm (GA) is a heuristic search that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. Swarm intelligence is also similar to real nature, because large number of simple organisms like bees are performing some simple task in order to get a complex task accomplished with ease. Metaheuristic techniques are, in general, faster and more robust solutions to solve complex set of problems in different domains. Particularly, such techniques would give a boost to the emerging IoT ecosystems, improve the consumer experience of IoT applications and services [14].

This paper looks into the application of these techniques in the semantic annotation of the Web of Things (WoT) data. Particularly, this paper describes a global disambiguation approach (Figure 1). The latter processes a WoT table (a table containing WoT data) to extract the mentions leveraging the classical text preprocessing techniques (stemming, normalization, tokenization, stop words). It generates candidate entities from the Knowledge base for each mention based on numerous features (string similarity, popularity, entity type, etc.) and then disambiguates these extracted mentions to the correct entries in the given knowledge base leveraging the semantic relatedness and the context similarities. The main contribution of this paper is the use of both genetic algorithm (GA) and Artificial Bee Colony (ABC) to resolve the combinatorial inference problem of the disambiguation modelled as a Travelling Salesman Problem (TSP).

The details of the framework structure and implementation is presented in what follows:

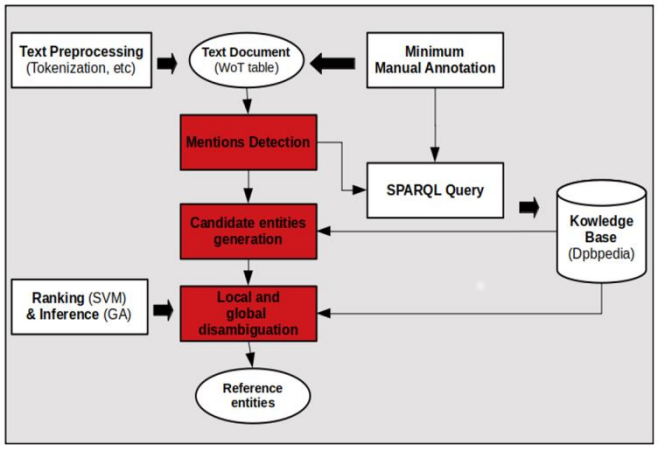

Figure 1: The Framework Overview

3. DATA PREPROCESSING

3.1 WoT Data

The framework is a Java web application intended for the domain of smart irrigation. However, it is designed to host all kind of IoT devices having web APIs such as smart parking or green energy devices. The web application is used to populate relational tables with IoT devices data and services. An example of the attributes of the central table are presented in Table 1. Each of these attributes is briefly described and manually annotated with Dbpedia vocabulary.

The manual annotation of the header is justified by two assumptions: (1) the manual annotation is not anymore a tedious task when the number of the data to be annotated is reasonable (2) the manual annotation is safer when the data to be annotated are used as an important context to annotate other data. For example: the location (place), the type (sensor, actuator), the property observed or acts on, the unit of measurement, the value of properties for a given WoT device are important contextual data. Furthermore, the complete manual annotation of these data may guarantee a well construction of the KB, and helps in search and mash ups [15].It is worth to note that different ontologies like the Semantic Sensor Network (SSN) ontology [16] can be used instead of Dbpedia. After annotating the WoT schema manually with the appropriate entities and properties from ontologies. The next step is the mentions detection from the content of the cells. As already mentioned, a text preprocessing is needed. This latter is described below:

3.2 Mentions Detection

After populating the WoT table with the data, the next step is the mentions detection. The goal of this process is to find the most relevant mentions for the annotation task by removing irrelevant, redundant and noisy data. Indeed, before extracting the mentions, it is required to preprocess the text of each row. The preprocessing task adopted by the framework involves two steps: the tokenization and the stop words removal. In addition, the bag of words model (BoW) is used to propose the most suitable words for the annotation.

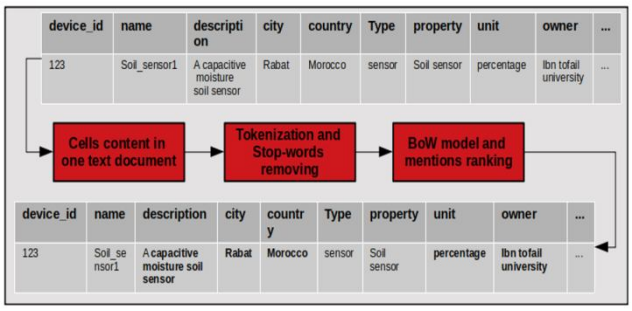

Figure 2: The mentions detection using the BoW model

- Tokenization

The tokenization consists of splitting the text into words (tokens) separated by whitespaces or punctuation characters.

The result of this operation is a set of words. Numerous NLP (Natural Language Processing) libraries in many programming languages can be used. The current framework uses The Apache OpenNLP library [17], which is a machine learning based toolkit for the processing of natural language text.

- Removing the Stop-words

Stop-words like pronouns, articles and prepositions and other words are used frequently in the text but they are few significant. That is why all stop words are removed before the annotation process.

- Bag of Words

A bag of words model is a representation of text that describes the occurrence of words within a document. The framework uses the BoW to choose a set of mentions that may deserve a semantic annotation. To be able of applying the BoW, the data of the processed row is gathered in one text document. Then, the BoW sorts the words of the row in the order of descending relevance (occurrence, length, etc.). In general, the number of mentions have to be limited to a maximum value (e.g. 20) for computing reasons. Therefore, if the number of the words of the BoW is lower than this value all the words will used as mentions otherwise the framework choose the most relevant ones. The Figure 2 illustrates the mentions detection task of one row.

3.3 Candidate Entities Generation

The search or the candidate entity generation is the process of generating a set of candidate entities for each mention from the KB. This process is very important for at least two reasons; the first is when a mention have no reference entity on the target KB (no-annotation), it will be removed from the mentions list which will reduce the computation time. The second and according to [18] consists in capturing the most probable entities to link the mention while maintaining a small set of candidates. Consequently, to take the most of this step, the framework leverages the following features.

Table 1: WoT table header annotation

	Attribute name	Description	Annotation
1	Device_id	The ID of the device	ttp://dbpedia.org/ontology/id
2	Name	The name of the device	http://dbpedia.org/ontology/Name
3	Description	The description of the device	http://dbpedia.org/ontology/description
4	Country	The country of the device	http://dbpedia.org/ontology/Country
5	Spot	The spot of the device	http://dbpedia.org/ontology/Location
6	Region	The city of the device	http://dbpedia.org/ontology/City
7	Latitude	The latitude of the device	http://dbpedia.org/ontology/Latitude
8	Longitude	The longitude of the device	http://dbpedia.org/ontology/Longitude
9	Elevation	The elevation of the device	http://dbpedia.org/ontology/Elevation
10	Height	The height from the ground	http://dbpedia.org/ontology/Height
11	Organization	The owner of the device	http://dbpedia.org/ontology/Organisation
12	Type	Sensor or actuator	http://dbpedia.org/ontology/type
13	unit	The unit of measurement of the property	http://dbpedia.org/page/Units_of_measurement
14	provider	The manufacturer of the device	http://dbpedia.org/ontology/manufacturer
15	Tags	Some tags for indexing purposes	http://dbpedia.org/ontology/tag
16	property	The property measured by the device	http://dbpedia.org/property
17	sensorType	The type of the sensor	http://dbpedia.org/ontology/type

Table 2: Example of Sparql query and its result

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX vrank:<http://purl.org/voc/vrank#>
SELECT ?p ?c substr( ?d, 0, 200 ) as ?d
FROM <http://dbpedia.org>
FROM <http://people.aifb.kit.edu/ath/#DBpedia_PageRank>
WHERE {
?p rdf:type dbo:Place.
?p vrank:hasRank/vrank:rankValue ?c.
?p dbo:abstract ?d .
?p rdfs:label ?x .
?x bif:contains "(Rabat)" .
Filter regex (str(?p),"resource").
}
ORDER BY DESC(?c) LIMIT 20
```

```
http://dbpedia.org/resource/Rabat41.0279 ……
http://dbpedia.org/resource/Rabat 41.0279 ……...
http://dbpedia.org/resource/Rabat 41.0279 ……...
http://dbpedia.org/resource/Rabat 41.0279 ……
http://dbpedia.org/resource/Rabat 41.0279 ……
http://dbpedia.org/resource/Rabat 41.0279 ……
http://dbpedia.org/resource/Rabat 41.0279 ……
http://dbpedia.org/resource/Rabat 41.0279 ……
http://dbpedia.org/resource/Victoria,_Gozo 4.6521 ……
http://dbpedia.org/resource/Victoria,_Gozo 4.6521 ……
http://dbpedia.org/resource/Victoria,_Gozo 4.6521 ……
http://dbpedia.org/resource/Rabat,_Malta 4.23087 ……
….
```

- String similarity*:* indicates how similar is the query mention to the title of the candidate entity.
- Prior Popularity: indicates how "famous" is a candidate entity in the KB.
- Entity Type: indicates the coherence between the mention and the candidate entity types (location, etc.)
- Context: indicates how similar is the contextual texts of the mention and the candidate entity.
- Semantic coherence: measures the semantic relatedness between the candidate's entities by using the table header, which is already annotated as context when querying the KB.

This approach allows to obtain a matrix of candidate entities which is formalized for later use. The Table 2 shows the result of a Sparql query against Dbpedia to generate 20 top candidate entities to the mention "Rabat". The example uses the surface form, the entity type "Place" and the entity popularity features.

The use of these three features is explained below:

- The surface form: the individuals of the knowledge base Dbpedia which contain the string "Rabat" will be chosen.
- The entity type: the individuals which have the entity type "http://dbpedia.org/ontology/Place" will be privileged.
- The entity popularity: the filtered individuals and which are the best ranked -in terms of presence in the knowledge base- will be chosen. For example, the candidate "http://dbpedia.org/resource/Rabat" having the rank 41.0279 has more chance to be the reference entity of the mention "Rabat" than the candidate "http://dbpedia.org/resource/Victoria,_Gozo" which has only the rank 4.6521.

The candidate entities of each mention are stored in a text document forming a disambiguation matrix. The important elements of this matrix are the mentions, the candidate entities URLs and a description of each candidate. This matrix may contain a score column representing a ranking score for each candidate. The role of this matrix is to prepare all the data helping to take the decision of the reference entities during the disambiguation task. The disambiguation matrix will be mainly used by two modules; the local and the global disambiguation processes. Figure 3 shows how such a matrix looks like.

3.4 Local Disambiguation

To re-rank the disambiguation matrix based on the features previously discussed, the framework leverages a SVM (Support vector machine) classifier. The used classifier is open-sourced [19] and it is trained on a set of string similarity and entity popularity metrics as its feature vectors similarly to [5]. The final result of the local disambiguation is a set of re-ranked candidate entities. The different steps of this process are detailed in a previous work [7].

MentionID	Mention	CandidateID	CandidateURL	Candidate description	...	score

Figure 3: Example of the disambiguation matrix

4. GLOBAL DISAMBIGUATION APPROACH

The method leverages genetic algorithm as well as artificial bee colony to jointly infer the target entities.

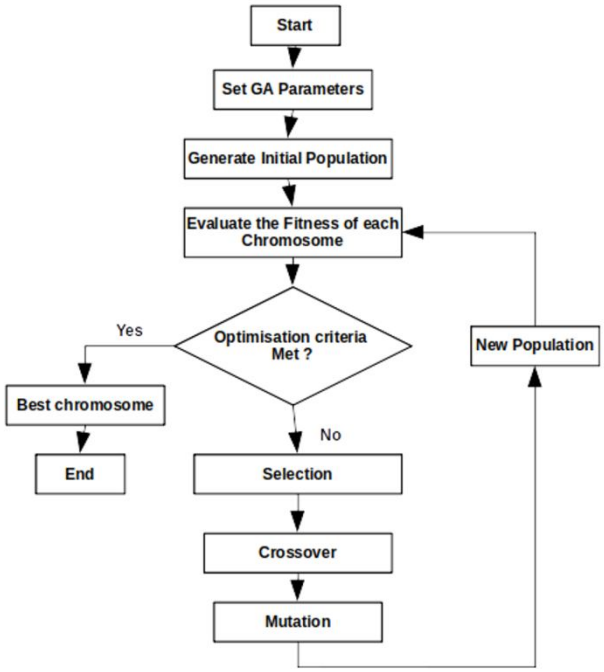

Figure 4: GA Steps Organigram

4.1 Genetic Algorithm (GA)

GA is an iterative process where each iteration is called a generation. Each generation concerns a population composed of individuals or chromosomes. The fitness of a population calculates the quality of the chromosomes and remains stable for a number of generations before a superior chromosome appears. A common practice is to terminate a GA after a specified number of generations and then examine the best chromosomes in the population. From a generation to another the chromosomes selection, crossover, mutation and fitness computing are applied. Finally, if no satisfactory

solution is found, the GA is restarted. The selection means the use of the solutions with high fitness to pass on to next generations. The crossover consists in swapping parts of the solution with another in chromosomes or solution representations. The main role is to provide mixing of the solutions and convergence in a subspace. Finally, the mutation is the change of parts of one solution randomly, which increases the diversity of the population and provides a mechanism for escaping from a local optimum. (Figure 4)

4.2 Join Inference Using GA

Given a WoT table, the approach attempts to execute the GA as follows:

- **GA initial parameters**

The idea is to create chromosomes whose length of each is equal to the number of mentions to be treated. Consequently, the number of genes depends on the processed row. For example, if for a given row only 5 mentions were detected, each chromosome will be composed of 5 genes. However, this number should be lower or equal to the maximum length Q for performance purposes. Q could be simply the number of the web table attributes (Ideally Q=20). As far as the population size K is concerned, it will be the number of candidate entities (for example K=100). In the case where the set of the generated candidates for a given mention is too small, then the last generated candidate for this mention is duplicated many times until reaching K. The crossover probability P_c is preferred to be high (e.g. P_c= 0.75) and the mutation to be very low or null (P_m= 0).

- **First Generation**

Instead ofrandomly generating the initial population of chromosomes. We choose the first generation of size K: X_1,X_2,...,X_K. such as $X_1 = (G_{11},G_{12},...,G_{1Q})$; $X_2 = (G_{21},G_{22},...,G_{2Q})$; …; $X_K = (G_{K1},G_{K2},...,G_{KQ})$ where G_{ij} is the candidate entity of mention of index i and of rank j. In the example mentioned in Figure 5, the first population is composed of 4 chromosomes with 5 genes each.

Figure 5: First GA generation example (K=4, Q=5)

- **Fitness Function**

The fitness function to measure the performance of each combination (chromosome) is defined as follows:

$$f(X_w) = \underset{(i,j)}{argmin} \sum_{j=1, j\neq i}^{Q} S(G_{l,i},G_{l,j}) \text{ , } w \in [1,K]$$

where S is a similarity function between two given candidate entities.

- **Fitness Computing**

Calculating the fitness of each individual chromosome: $f(X_1)$,$f(X_2)$,...,$f(X_K)$ is equivalent of finding the optimal order of the genes which minimize f. This can be viewed as a travelling salesman problem (TSP). The TSP is a problem in which a sales person has to visit certain cities following some path, such that each city is visited only once and then reach back to the place he started from. The travelling person should travel in such a way that his travelling cost (distance) is minimum. In the context of this paper, finding the shortest path of the TSPis equivalent of finding the set of genes (candidate entities) which are the most semantically related to each other. For instance the TSP for the chromosome $X_1 = (G_{11},G_{12},...,G_{15})$ is represented in the graph in figure 3. And a possible solution of this example is $(G_{11},G_{13},G_{15},G_{12},G_{14})$.

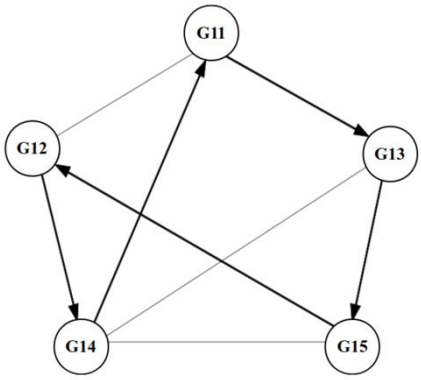

Figure 6: TSP illustration (Q=5)

However, the TSP (Figure 6) is a complex combinatorial optimization problem that is difficult to solve with numerical approach. To illustrate this, with only 20 cities and even if we could evaluate 1 million paths per second, examining all 20! Possible paths would require more than 77,000 years. Therefore, the paper addresses the TSP leveraging the Artificial Bee Colony (ABC) algorithm. The method is explained as follows:

4.3 Artificial Bee Colony (ABC)

- **ABC Overview**

The ABC is a widely used technique in many research fields namely to address combinatorial problem such as the travelling salesman problem [20]. Bees by nature are organized into groups of active, inactive and scout bees. Active bees travel to a food source, examine neighbour food sources, gather food and return to the hive. Scout bees investigate the area surrounding the hive looking for attractive new food sources. Once food source is found active and scout bees inform the others by waggle dance.

This dance conveys information to the inactive bees about the location and quality of the food source. Inactive foragers receive this food source information from the waggle dance and may become active foragers. In general, an active foraging bee continues gathering food from a particular food source until that food source is exhausted, at which time the bee becomes an inactive forager.

- **ABC Application**

The key concept in ABC algorithm is the idea that each virtual food source that represents a solution has some sort of neighbour solution. In the case of the TSP, where a solution can be represented as an array of cities representing a path from city to city, a natural neighbour solution relative to a current solution is a permutation of the current solution where two adjacent cities have been exchanged.
The Figure 7 shows an illustration of how the framework apply the ABC algorithm for the chromosomes fitness calculation.

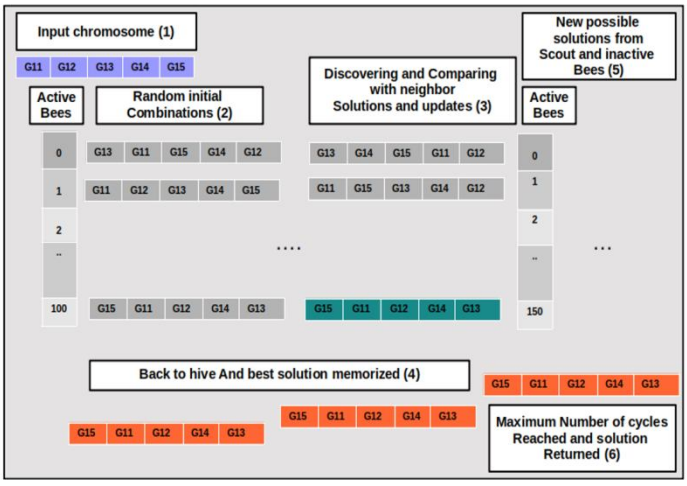

Figure 7: ABC application example

1. Given a chromosome X as input, a number of active bees (e.g. 100) is generated with random solutions (combinations of the genes of X).
2. Initially, the number of the generated solutions is equal to the number of the active bees.
   Recruited active bees leave the hive to search for neighbour food sources according to the solution they have in their memories.
3. Each bee examines a set of neighbour solutions (as defined previously) and updates its solution.
   The framework leverages the context similarity between two genes (candidates) by comparing their contextual description, captured in the disambiguation matrix, leveraging the BoW model and the Jaccard similarity similarly to [21].
   When a bee found a better solution than its, the new solution is memorized, the bee returns to the hive and inform inactive bees with the new solution.
4. After that, the hive global solution is updated as well as the persuaded inactive bees' solutions.
   The active bees continue its search in neighbour food sources, evaluate their fitness, update its solution and inform inactive bees. In case where a counter of the number of times a particular virtual food source has been visited without finding a better neighbour food source.
5. The bee returns to the hive and become inactive and a randomly chosen inactive bee become an active one.
6. A maximum number of cycles is used to define the number of iterations of the ABC. One cycle represents processing of each bee in the hive. When this number is reached the ABC is ended and the hive solution is returned.

5. RESULTS DISCUSSION

The present framework which is a Spring boot based web application implements the different methods and algorithms as explained in this paper using Java libraries like OpenNLP and opensource code like for the SVM. The BoW model has been used twice by the framework. The first time as a mentions detection helper and the second to measure the context similarity between the candidates leveraging the Jaccard distance. The ABC implementation is inspired from the simulated bee colony implemented by [22] in C#. The evaluation of the framework is done at two levels: the applicability and the efficiency. These two criteria will be compared against a baseline join inference method using a probabilistic graphical model (PGM). In what follows, the authors give a brief overview of the PGM based approach and an applicability comparison with the swarm intelligence approach. After that the framework efficiency is discussed.

5.1 Baseline

During the last years, alternative approaches for semantic annotation using the Entity Linking task and the collective disambiguation were proposed. However, the join disambiguation was handled leveraging, in most cases, numerical computing. Recently, an approximate approach based on probabilistic graphical models was used by different researches. For instance, [9] have used three features in their graphical model: the local similarities, the semantic relatedness and the prior popularity of a candidate entity. [23] have conducted a probabilistic approach which consists in learning a conditional probability model from data and employing approximate probabilistic inference in order to find the maximum a posteriori (MAP) assignment [24]. Particularly to annotate web tables, [10] have used a probabilistic method to annotate columns and cells values with entities. Recently, [5] have provided a unified WoT Knowledge Base construction framework. The EL framework they have proposed, annotates entities, types and relations using features from [20] . To infer the best disambiguation entities, they have adopted an iterative message passing (IMP) algorithm from [11]. The frequent idea of PGM, which was also First the

variable nodes of the graph are initialized with the candidate entities. After that, the variable nodes in the graph send their current assignment to the factor nodes they are connected to.

Once the factors receive the values of their neighboring variable nodes, they calculate the agreement between the received values using the function S. To decide if two values agree or not the result of the function S is compared to a threshold T. If the value of S satisfactory, the variable nodes values are accepted else, they are rejected and the variables receive a change message to update their assignments. The algorithm converges when no variable node receives a change message.

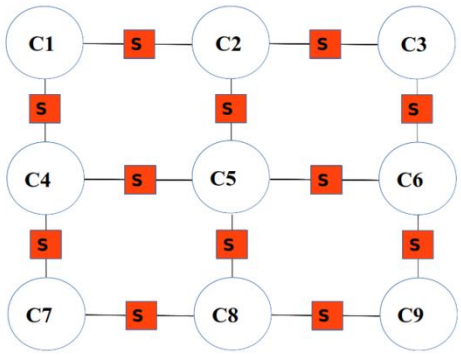

Figure 4: AWoT table represented as a Factor Graph

Accordingly, the authors shed light on the superiority of the proposed swarm intelligence method compared with the probabilistic based approach in terms of applicability. The Table 3 below gives a summary of this comparison.

Table 3: Applicability comparison between GA and PGM based approaches

	Probabilistic approach	GA based approach
Convergence	Not sure	Generally convergent
Annotation	All cells at the time	Row by row
Applicability	Hundreds of rows	Illimited
Use of semantic relatedness	Between rows and columns cells	Only between one row cells
Dependence on initial values	High dependence	Medium dependence

5.2 Performance and Accuracy

As can be noted, the efficiency and the accuracy of the framework depends on several criteria namely the initial parameters of the GA and the ABC. For instance, annotating a row containing only 5 extracted mentions is, in general, faster that annotating a 12 mentions row. To evaluate the relationship between the initial parameters and the performance and accuracy a row containing exactly 20 mentions have been suggested to the framework. 10 candidates were generated for each mention. After that, the annotation process is run with different ABC initial parameters. An augmentation of the response time was noted by approximately three times in function of the number of bees. However, when the annotation process is re-executed for the same data (row), this approximation changes. The authors conclude that since the GA and ABC are approximate techniques, the static relationship between number of bees and response time cannot be given. However, intuitively augmenting the number of active bees means that more solutions have to be evaluated and the annotation process may be slower. The Table 4 below shows the response time (RT1) in number of GA generations for the first round as well as RT2 for the second round.

Table 4: Number of Bees and Response Time Relationship

	NB = 100	NB = 200	NB = 300	NB = 400
RT 1	3	9	24	37
RT 2	3	12	2	4

After the semantic annotation process was ended, the accuracy of the results was checked and only 54% of the mentions were correctly matched. Two reasons were suggested, the first is the small number of candidate entities and the second is the similarity measure. Therefore, a second round was performed using the Normalize Google Distance (NGD) similarity [25]. The authors noted that the accuracy was improved to approximatively 60%. However, the performance was decreased and that was explained by the definition of the NGD measure itself. The Table 5 shows the explained results.

Table 5: Accuracy and Time Response Approximations for Context and NGD Similarities

	Context similarity	NGD
Accuracy	54%	60%
Time response (generations)	3	9

Finally, the GA approach seems to be superior than [5] which found the average time consumed in each annotation on the WoT table equal to 4.1 s using a similar PGM approach. The best result found for the proposed GA succeeded to annotate a 20 mentions row in only three generations (nearly 5 seconds) which gives an average of 0.25 seconds (5/20) per mention.

6. CONCLUSION

The paper describes a semantic annotation framework of WoT tables. The current framework implements various techniques and efficient algorithms for the EL task such as the well-known text preprocessing tools for the mentions detection task as well as swarm and evolutionary algorithms for the disambiguation task. A relational database was populated with WoT data and services. The annotation performance and accuracy of the framework were captured. Furthermore, a comparative study met the GA based inference technique with the probabilistic graphical one, was elaborated. In the future, a large scale evaluation of the present framework will be performed and other high computing techniques will be integrated [26][27]..

REFERENCES

1. L. Atzori, A. Iera, and G. Morabito, **The Internet of Things: A survey**, *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
2. D. Guinard, V. Trifa, F. Mattern, and E. Wilde, **From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices**, *Architecting the Internet of Things*, pp. 97–129, 2011.
3. T. Berners-Lee, J. Hendler, and O. Lassila. **The semantic web**. *Scientific american*, *284*(5), 28-37. 2011.
4. M. Noura, M. Atiquzzaman, and M. Gaedke, I**nteroperability in Internet of Things: Taxonomies and Open Challenges**, Mobile Networks and Applications, vol. 24, no. 3, pp. 796–809, Jul. 2018.
5. Z. Wu, Y. Xu, Y. Yang, C. Zhang, X. Zhu, and Y. Ji, **Towards a Semantic Web of Things: A Hybrid Semantic Annotation, Extraction, and Reasoning Framework for Cyber-Physical System**, *Sensors*, vol. 17, no. 2, p. 403, Feb. 2017.
6. W. Shen, J. Wang, and J. Han, **Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions**, *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 443–460, Feb. 2015.
7. I. Nadim, Y. El Ghayam, and A. Sadiq. **Towards the semantic annotation of Web of Things: A collective disambiguation approach.***In Proceedings of the 2nd international Conference on Big Data, Cloud and Applications* (pp. 1-6). March 2017.
8. L. Ratinov, D. Roth, D. Downey, and M. Anderson. **Local and global algorithms for disambiguation to wikipedia**. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (pp. 1375-1384). Association for Computational Linguistics.2011.
9. S. Rong, and M. Iwaihara. **A collective approach to ranking entities for mentions.***2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS). 2016.*
10. G. Limaye, S. Sarawagi, and S. Chakrabarti. **Annotating and searching web tables using entities, types and relationships.***Proceedings of the VLDB Endowment*, 3(1-2), 1338–1347. 2010.
11. V. Mulwad, T. Finin, and A. Joshi. **Semantic message passing for generating linked data from tables**. *In International Semantic Web Conference* (pp. 363-378). Springer, Berlin, Heidelberg. October, 2013.
12. J. Kennedy. (2006). ***Swarm intelligence***. In *Handbook of nature-inspired and innovative computing* (pp. 187-219). Springer, Boston, MA.
13. Mitchell, Melanie. ***An introduction to genetic algorithms***. MIT press, 1998.
14. T. Chakraborty, and S. K. Datta. **Application of swarm intelligence in Internet of Things**. In *2017 IEEE International Symposium on Consumer Electronics (ISCE)*(pp. 67-68). IEEE. November, 2017.
15. V. Mulwad, T. Finin, Z. Syed, and A. Joshi. **Using linked data to interpret tables**. In *Proceedings of the First International Conference on Consuming Linked Data-Volume 665* (pp. 109-120). November, 2010.
16. M. Compton, P. Barnaghi, L. Bermudez, R. Garcca-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. Le-Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor, **The SSN Ontology of the W3C Semantic Sensor Network Incubator Group**, SSRN Electronic Journal, 2012
17. **OpenNLP**, A. (2011). *Apache software foundation. URL http://opennlp. apache. Org.*
18. B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J.R. Curran. **Evaluating entity linking with Wikipedia.***Artificial intelligence*, *194*, 130-150. 2013.
19. T. Joachims. **Training linear SVMs in linear time**. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 217-226). ACM. August, 2006.
20. D. Karaboga, and B. Gorkemli. **A combinatorial artificial bee colony algorithm for traveling salesman problem.***In 2011 International Symposium on Innovations in Intelligent Systems and Applications*(pp. 50-53). IEEE. June, 2011.
21. S. Niwattanakul, J. Singthongchai , E. Naenudorn, and S. Wanapu. **Using of Jaccard coefficient for keywords similarity**. In *Proceedings of the international multiconference of engineers and computer scientists* (Vol. 1, No. 6, pp. 380-384). March, 2013.
22. J. McCaffrey. **Natural Algorithms: Use Bee Colony Algorithms to Solve Impossible Problems.** *MSDN Magazines, erişim linki: http://msdn. microsoft. com/enus/magazine/gg983491. Aspx*, *24*. 2011.
23. O.-E. M. Ganea, A. Lucchi, C. Eickhoff, and T. Hofmann. **Probabilistic Bag-Of-Hyperlinks Model for Entity Linking.***Proceedings of the 25th International Conference on World Wide Web - WWW '16*. 2016.
24. D. Koller, and N. Friedman. ***Probabilistic graphical models: principles and techniques***. MIT press. 2009.
25. R. L. Cilibrasi, and P. M. Vitanyi. ***The google similarity distance***. *IEEE Transactions on knowledge and data engineering*, *19*(3), 370-383. 2007.
26. S. B. J, **Enhancing Performance of IoT Networks through High Performance Computing**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 3, pp. 432–442, Jun. 2019.
27. R. N. S., **Optimal Reactive Power Control Using Compensating Capacitor Based on Artificial Immune System**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 1.3, pp. 381–386, Aug. 2019.