



ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERÍA INFORMÁTICA

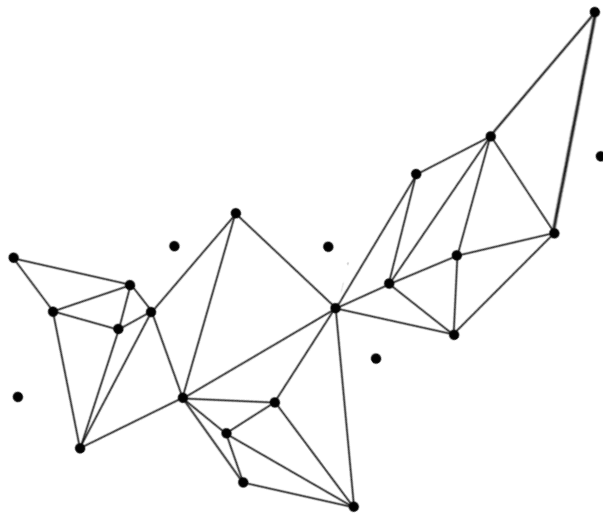
Título del proyecto:

**S-SAM: A SEMANTIC SELF-ADAPTED METHOD FOR
CATEGORIZING ANNOTATED RESOURCES**

Olivier Moriones Oyón

Alberto Córdoba Izaguirre

Pamplona, 15 de junio de 2015



*“Cuando realmente quieres que algo suceda,
el universo entero conspira para que realices tu deseo”*

El Alquimista, Paulo Coelho

ACKNOWLEDGEMENTS



To my mother and my father, because they gave me everything.

To my sisters, because despite the distance we all keep together.

ABSTRACT



The present Final Degree Project propose a new method for automatic classification of resources labelled with tags coming from a folksonomy of social tagging systems. It's the result of a variation of SAM, a Self-Adapted Method, that is, a method of automatic classification of annotated resources, which have been done by some researchers of the Public University of Navarra.

The method, called S-SAM (or Semantic SAM) have as their goal to improve the classification of annotated resources by means of this automatic method, without use human force, in order to make more accurate the knowledge representation and information recovery.

To do so, it's been chosen the final degree project of [\(Ciordia, 2011\)](#) as a pattern to follow in the implementation of SAM and S-SAM, which is a Java program that needs some data allocated in MySQL format databases.

The research is divided into two parts. The first part studies the way a subset of resources is classified using the number of occurrences versus using the fitness of the annotation (that is, a consensus evaluation from experts); and the second part also studies this but using the whole set of resources (all the annotations).

Once obtained the results, they will be compared finding out which way classifies the best.

TABLE OF CONTENTS

Chapter I: Antecedents	25
Chapter II: Introduction	29
2.1. Background	29
2.1.1. A little of history	29
2.1.2. Web 2.0	31
2.1.3. Folksonomies.....	32
2.1.4. Semantic Web.....	36
2.1.5. SAM	38
2.2. Motives	39
2.3. Goals and hypothesis	40
2.4. Proposed solution	41
2.5. Document Organization	41
Chapter III: Analysis and design	45
3.1. SAM prototype.....	45
3.1.1. SAMsModel definitions.....	46
3.1.2. SAM classifier definition.....	48
3.1.3. SAM actions and components	49
3.1.3.1. SAMsModel_creation.....	49
3.1.3.2. SAMsModel_evolution.....	51
3.1.4. Descriptive example.....	51
3.2. S-SAM prototype.....	54
3.2.1. S-SAMsModel definitions	55
3.2.2. S-SAM classifier definition	56
3.2.3. S-SAM actions and components	56
3.2.3.1. S-SAMsModel_creation	56
3.2.3.2. S-SAMsModel_evolution	56

3.2.4.	Descriptive example.....	57
3.3.	S-SAM FULL prototype.....	60
3.3.1.	S-SAMFULLsModel definitions.....	60
3.3.2.	S-SAM FULL classifier definition.....	61
3.3.3.	S-SAM FULL actions and components.....	61
3.3.3.1.	S-SAMFULLsModel_creation.....	62
3.3.3.2.	S-SAMFULLsModel_evolution.....	62
3.3.4.	Descriptive example.....	62
3.4.	SAM, S-SAM and S-SAM FULL for R47.....	65
3.4.1.	Databases.....	66
3.4.2.	Configuration file.....	67
3.4.3.	Input parameters.....	68
3.4.4.	Output results.....	70
3.5.	SAM y S-SAM for Del.icio.us.....	70
3.5.1.	Databases.....	71
3.5.2.	Configuration file.....	72
3.5.3.	Input parameters.....	72
3.5.4.	Output results.....	73
3.6.	Other related support programs.....	74
Chapter IV: Implementation.....		79
4.1.	Software tools.....	79
4.1.1.	Java.....	80
4.1.2.	NetBeans.....	80
4.1.3.	WampServer.....	81
4.1.4.	Couchbase.....	81
4.2.	Installation and configuration guide.....	82
4.2.1.	Installation of WampServer.....	82
4.2.2.	Installation of NetBeans.....	82

4.2.3.	Importing databases	83
4.2.4.	Importing NetBeans projects	84
4.2.5.	Configuring phpMyAdmin	84
4.2.6.	Configuring NetBeans	85
4.3.	How to run SAM and S-SAM.....	85
Chapter V: Experimental results		89
5.1.	SAM, S-SAM and S-SAM FULL for R47.....	89
5.1.1.	Dictionary SAM results.....	90
5.1.2.	Dictionary S-SAM results	96
5.1.3.	Dictionary S-SAM FULL results	103
5.1.4.	Results comparison among Dictionary SAM, S-SAM and S-SAM FULL 109	
5.2.	SAM and S-SAM for delicious	111
5.2.1.	SAM results	111
5.2.2.	S-SAM results.....	115
5.2.3.	Results comparison between SAM and S-SAM	122
Chapter VI: Conclusions.....		129
Chapter VII: Future lines		133
Chapter VIII: Bibliography.....		139

TABLE OF FIGURES AND TABLES

Figure 1. Dictionary SAM - Dictionary tags.....	91
Figure 2. Dictionary SAM - Dictionary tags zoomed in.	92
Figure 3. Dictionary SAM - Classification with SIM.	93
Figure 4. Dictionary SAM - Classification with DELTA.....	93
Figure 5. Dictionary SAM - Classification comparison.....	94
Figure 6. Dictionary SAM - Classification comparison (correct).....	95
Figure 7. Dictionary S-SAM - Dictionary tags with MAX.	97
Figure 8. Dictionary S-SAM - Dictionary tags with MEAN.	97
Figure 9. Dictionary S-SAM - Dictionary tags MAX/MEAN comparison.....	98
Figure 10. Dictionary S-SAM - Classification with SIM.	99
Figure 11. Dictionary S-SAM - Classification with DELTA.....	100
Figure 12. Dictionary S-SAM - Classification comparison.	101
Figure 13. Dictionary S-SAM - Classification comparison (correct).	102
Figure 14. Dictionary S-SAM FULL - Dictionary tags with MAX.....	104
Figure 15. Dictionary S-SAM FULL - Dictionary tags with MEAN.	104
Figure 16. Dictionary S-SAM FULL - Dictionary tags MAX/MEAN comparison. ...	105
Figure 17. Dictionary S-SAM FULL - Classification with SIM.....	106
Figure 18. Dictionary S-SAM FULL - Classification with DELTA.....	107
Figure 19. Dictionary S-SAM FULL - Classification comparison.	108
Figure 20. Dictionary S-SAM FULL - Classification comparison (correct).	109
Figure 21. DSAM, DSSAM and DSSAMFULL comparison DELTA-MAX (correct).	110
Figure 22. SAM - Dictionary tags.	112
Figure 23. SAM - Dictionary tags zoomed in.	113
Figure 24. SAM - Classification with SIM.	114
Figure 25. SAM - Classification with DELTA.	114
Figure 26. SAM - Classification SIM/DELTA comparison.	115
Figure 27. S-SAM - Dictionary tags with MAX.	116
Figure 28. S-SAM - Dictionary tags with MAX zoomed in.	117
Figure 29. S-SAM - Dictionary tags with MEAN.	117
Figure 30. S-SAM - Dictionary tags with MEAN.	118
Figure 31. S-SAM - Dictionary tags MAX/MEAN comparison.	119
Figure 32. S-SAM - Dictionary tags MAX/MEAN comparison zoomed in.	119

Figure 33. S-SAM - Classification with SIM.....	120
Figure 34. S-SAM - Classification with DELTA.....	121
Figure 35. S-SAM - Classification comparison.....	122
Figure 36. SAM and S-SAM comparison (DELTA-MAX).....	123
Figure 37. SAM comparison with SIM-DELTA (correct).....	124
Figure 38. S-SAM comparison with SIM (correct).	125
Figure 39. S-SAM comparison with DELTA (correct).	125
Figure 40. S-SAM comparison (correct).	126

Table 1. Initial phase: SAMsModel_creation.....	51
Table 2. Example of social tagging system for SAM.	52
Table 3. Example of V and SCR sets for SAM.	53
Table 4. Example of social tagging system for S-SAM.	57
Table 5. Example of V and SCR sets for S-SAM.	59
Table 6. Example of social tagging system for S-SAM FULL.....	63
Table 7. Example of V and S_CR sets for S-SAM FULL.....	64
Table 8. General categorization algorithm for S-SAM.....	135

Chapter I:

ANTECEDENTS



Internet is one of the most powerful tools nowadays. At the beginning was made up for static pages, but quickly dynamic pages arose. Web as it was knew evolved into a new concept where non-expert users could generate their own content: Web 2.0. Social networks, blogs and wikis appeared establishing new types of relationships, business and communications.

As information grew up, new representation and classification methods were needed in order to manage that huge amount of data. That's why two paradigms turned up: folksonomies and ontologies.

The first one is very common in collaborative systems. It allows users to take a several amount of text labels and tag them to resources of the systems. This method is very simple, non-hierarchical, easy to update and can be done by a big set of users.

In contrast to folksonomies, exists ontologies, a formal framework for representing knowledge composed of concepts and relationships within a domain. They are complex systems of hierarchical structure with a high level creation and update cost that only can be performed by a limited group of persons: experts.

Both paradigms have their own supporters and detractors. Many experts believe that folksonomies are very easy systems to maintain and develop but incurs into several weaknesses, such as non-controlled vocabulary which arises ambiguity, polysemy and synonymy problems. However, others think that these problems have nothing to do with which ontologies. Not only the scalability problems issues, but also the need of experts to create content (not taking into account the creation and maintenance costs). Despite of all this, it ensures a very good classifications of the resources within a domain.

A lot of authors have contemplated to combine both (or, at least, reduce the space between them) in order to take the better of each one: the ontologies power and the creation and update flexibility of folksonomies. In this aspect, some proposals had been presented trying to improve folksonomies navigability. However, others authors believe that both paradigms can't be reconciled and its paths have to be separated and improve individually.

But one thing is clear, the optimal solution for this problem have not been reached yet.

Chapter II:

INTRODUCTION

■ ■ ■

In this chapter, it will be set a brief introduction about the present project. First of all, it's going to place a background in order to understand from where all of this is coming, then why to research about resources classification and knowledge representation and information recovery (also other related motivations), the objectives and goals desired to achieve and, finally, what is the proposed solution.

2.1. Background

2.1.1. A little of history

From the beginning of Web to the present, the user's roll and the need of classify information contained in have considerably evolved. Together with the appearance of the called Web 2.0, blogs and social networks, users have turned from being simple information consumers into generators of content, which have caused that the large amount of existing information within Internet exponentially increases.

This new user's roll has required the appearance of new simpler ways of information representation and classification than the traditional ones, based on taxonomies, thesaurus or even ontologies.

One of the most frequently used ways today for carrying out this classification are the collaborative tagging systems, in which information is classified by means of free assignment of text labels by users.

In these systems, normally called folksonomies ([Vander Wal, 2007](#)), users use a non-controlled vocabulary for classify existing resources, that's why they do not require have previous technical knowledge. When user assign a certain amount of text labels to the resource (photos, videos... depend on the social network), it's said that user labels the resource. Fruit of this labelling, it's obtained a set of annotations that relates user with his resource and labels.

Labels can, therefore, be used to users of social networks can classify information and access to it. Thus, folksonomies have a very interesting social aspect, which is based on that labels and resources are shared by any user. This means, in many cases, that any user can access to resources published by other users, using the labels with which it have been labelled.

Definitely, folksonomies offer users a simple way of organizing information and, at the same time, strengthening the social dimension of web, making that labelled information by users is available for the rest. Their huge ease of use and social aspect that provides have made folksonomies turn into a real standard for representing knowledge in Web 2.0.

Nevertheless, folksonomies suffer from important problems for being based on non-controlled vocabularies. These problems become evident according to information volume of folksonomy arise (search and location).

Several authors have made proposals that search to solve some of the folksonomy problems from different perspectives, in many cases focused on providing structure to that space of labels, in a way similar to a bottom-up ontology construction.

Quite the contrary of folksonomies, Semantic Web is based on information representation in an explicit, formal and agreed way. This representation is made by using ontologies that allow represent with precision top-down knowledge models, explicitly characterizing their several elements and relationships.

The characteristics offered by Semantic Web will allow to resolve the knowledge representation and information recovery problems of folksonomies, but it generates other important problems related to complexity of modelling domains, of maintaining it updated and, fundamentally, with the biggest complexity associated with the resource classification in comparison with the assignment of a set of text labels.

Even though both approaches can be considered as opposite, different authors ([Gruber, 2007](#)) believe that both techniques can help each other. This way, on one hand, Web 2.0 and folksonomies can be a useful information source to help to reduce some of the ontology problems, such as its problems with creation, maintenance and insertion of information (population). On the other hand, ontologies can provide folksonomies with a structure which allows navigation and search information problems go down and help this way to users can access to that information in a more efficient way.

2.1.2. Web 2.0

When Internet was born, web pages were merely static code normally in HTML language offering information to users, basically text, which were constantly updated.

As time goes by, web sites were turning dynamic thanks to the use of other languages, such as PHP or similar, providing multimedia content (like images, sounds, video, more user-friendly interfaces) and allowing self-content creation without being an expert. This change was denoted as Web 2.0.

Although Web 2.0 suggests a new version of the World Wide Web, it does not refer to an update to any technical specification, but rather to cumulative changes in the way Web pages are made and used.

A Web 2.0 site can allow users to interact and collaborate with each other in a social media dialogue as creators of user-generated content in a virtual community, in contrast to Web sites where people are limited to the passive viewing of content. That's why this "old" version of World Wide Web was called Web 1.0. Thanks to this evolution, different management content tools were born such as social network sites, blogs, wikis, folksonomies, video sharing sites, hosted services, Web applications...

2.1.3. Folksonomies

One of the meaningful concepts of Web 2.0 is the social tagging systems. Besides create web content in a simple way, users now can also organize or classify it in categories by labels assignment (tags). A tag is a key word that is added to a digital object (website, photography, video) with the purpose of describe it, not as a part of a classification formal system, but as a way to easy any localization of information and contents.

Tagging systems has quickly became in a usual tool to classify information in many social networks. Two of the most meaningful and popular apps that have raised are:

- **Del.icio.us**¹: a social bookmarking application in which users store their favourite webpages and freely navigate freely the set of sent pages by all users.
- **Flickr**²: a social network in which users can upload their photographs and navigate the sent ones by other users.

Folksonomies offer a way of knowledge representation and information recovery, based on the semantic information of resources and tags that it's obtained from user annotations. Result of this user collective labelling emerge a semantic representation of tags and resources. This way, if some users frequently use "art" and "museum" tags simultaneously, can be considered that there is a certain semantic relationship between

¹ Del.icio.us, <http://www.delicious.com>

² Flickr, <http://flickr.com>

them. Also can be considered that these resources which are classified by them are related to each other and that its topic are related to the semantic of these tags.

Thus, folksonomies are based on the interjection of numerous users that jointly create a “collective intelligence” which define the semantic of information. This semantic is not obtained by the knowledge of one expert user, but the sum of the vision of several users’ information, hence it’s called “collective intelligence”.

It’s usual to split folksonomies in two big categories ([Vander Wal, 2005](#)):

- **Broad folksonomies:** is the one in which multiple user tag certain content with a variety of terms from a variety of vocabularies, thus creating a greater amount of metadata of that content.
- **Narrow folksonomies:** is the one in which a few users, primarily the content creator, tag an object with a limit number of terms.

Some other authors also make differences between narrow folksonomies and extended narrow folksonomies, being the first one those in which the creator of resource uniquely can label it, and the second one those in which it’s extended to other social network users. This difference between who are authorized to tag resources and who are not makes that the way in which emerge semantic information of resources and tags was different.

In broad folksonomies it’s satisfied that, as users tag labels to resources, there is a few number of tags that are the most used ones ([Golder and Huberman, 2006](#)). These tags and its frequency with which have been labelled represent the semantic information of that resource. The semantic information of tags is obtained from resources to which are assigned and from the rest of tags which are used to classify these resources.

In narrow folksonomies, the semantic of a resource depends only on assigned tags by creator or by a few number of users (extended narrow).

Usually, a tag is labelled only one time by each resource, so the contribution of each tag to semantic representation of resource is homogeneous. As far as the semantic information of tags, in this kind of folksonomies is more complicated to get, since exist a few number of annotations of tags to resources.

Though folksonomies have become a current standard in knowledge representation of web, they have important issues if they are compared to formal systems of knowledge

representation. These problems mainly exist because folksonomies are based on non-controlled vocabularies, which imply the use of syntactic variations ([Echarte et al., 2009](#)) of the same tag, of tags that can mean different things (polysemy), or of tags with different granularity. For example, two users can use the tags “eiffel” and “eifel” to label a photography of Eiffel Tower in Paris (syntactic error); can use the same tag “apple” to classify two different resources, such as two pics of a fruit and a computer (polysemy); can label the tags “photography” and “photo” to the same resource (synonymy) or classify the pic of a cat with “animal” or “cat” (granularity).

Other problem of folksonomies are related to the fact that, when a user makes an annotation (label a tag to a resource), this tag can be different kinds according to user intention ([Golder and Huberman, 2006](#)). A user can assign descriptive tags to a resource and collaborate this way to represent its semantic, but also can assign personal use tags, such as “toRead” or “myJob”. These tags do not represent the semantic of the resource and it’s not possible to obtain its semantic information from the resources that are related to. Folksonomies are based on, therefore, a combination of tags that can be used to classify information in a collective way and other terms that are only useful to authors.

The use of non-controlled vocabulary limits, thus, the ability to represent knowledge of folksonomies, however it also affects to ability to recover existing information, mainly as information volume within folksonomy enlarges ([Chi and Mytkowicz, 2008](#)).

The habitual way to assume folksonomies information is navigate through them using tags to explore and find out the information. The structure of folksonomies and its application in social networks also allows to make exploratory searches. In this kind of searches, user do not leave with a specific aim, but from an element (normally a user, resource or tag) use the available hyperlinks in order to navigate through folksonomy and to find interesting information. In that way, for example, user can explore information surfing among the more popular tags, profiles from other users, dates, etc. This kind of navigation is much related to the concept of serendipity, that belongs to situations in which users find some information that are not actually looking for but it’s interesting to them.

Nevertheless, issues derived from use of non-controlled vocabularies also complicate navigation and exploration of information. These problems can be summed up in two:

- Limited searches
- Limited exploration

The abilities of search are mainly limited by linguistic and semantic limitations of tags as described in previous points. When a user searches contents by a tag or a set of tags, results will arrive determined by tags which have been used in the classification of existing contents. Thus, for example, when searching photographs of apples using those resources labelled with “apple”, it will be obtained photographs of apples and computers, and when searching photographs of Eiffel tower using “eiffel” tag, it will not be obtained photographs labelled with “eifel”.

In respect to exploration, the two main methods for navigating through tags are search/refine and using some kind of tag space visualization such as a tag cloud, but these methods are not much satisfactory ([Begelman, 2006](#)).

Search/refine method is based on selection and search of a tag and later refinement of obtained results. In a certain way, if a user search in Del.icio.us the “book” tag, he will get a set of websites labelled with this tag and a set of related tags with which the search can be extended. These related tags offer a basic way to widen or refine the user’s search. For example, it can be checked the appearance of personal-type tags, such as “toRead” or “of”, that possibly can’t help to user at all, that’s why several times it’s more convenient to refine the search with the obtained results.

Folksonomies also have another navigation mechanism denoted as tag cloud. In it, most frequent tags within folksonomy are shown, pointing out with bigger size those that more times are used and with other colour those sometime user have already used.

However, the usefulness of this tag clouds for navigating is also many times limited, due to different reasons:

- Tags appear in alphabetic order without taking account possible existing relations amongst them.
- Showing uniquely the most common tags, just as the annotation number grows, the access to less common tags, but can contribute important points of view about information, is lost.
- It’s shown redundant tags that can be grouped as, for example, “blog” and “blogs”, or “tool” and “tools”.

- It's shown personal type tags such as "toRead", that haven't got a semantic related to resources.
- This visualization system is uniquely used in the first level of navigation, but it usually don't have similar mechanisms as navigation goes deep and navigate amongst contents or tags.

As a result of all of this, users have a very simple way to classify information, but its recover is more complex or, sometimes, not possible to obtain it in a direct way. Ontologies would allow to reduce these problems thanks to being based on a structured model of knowledge and to capacities of inference that they have. As shown in ([Shadbolt et al., 2006](#)), information of Web 2.0 can be more useful if it's semantically structured and Semantic Web can supply tools that make possible this task.

2.1.4. Semantic Web

In front of folksonomies, it exists the formal classification systems. These systems offer better mechanisms for representing knowledge and access information, but at the expense of a bigger difficulty when classifying information.

In the early of this century, Tim Berners Lee suggested his vision of Web Semantic, based on a web in which all information has an associated semantic meta-information, that allows it to be extended and managed not only by humans, as it have been doing, but also by computers. Since then it have been made continuous progresses, but, still today, its application keep being very complex ([Shadbolt et al., 2006](#)) and it's constrained mainly to the theoretical and academic sphere, and to certain business environments.

Semantic Web use ontologies as representation tools of knowledge. They allow describe each resource of web according to a controlled vocabulary and that, ideally, must be agreed amongst all users with the purpose of having a shared knowledge.

An ontology is defined as "a formal, explicit specification of a shared conceptualization" ([Gruber, 1993](#)). In this definition, *explicit* is referred to the need of enumerate all concepts and elements that set up part of a domain; *formal* to the need of use a formalized language in its representation; and *shared* to a need of including in the

representation the points of view of all users that are going to use it. Ontologies build up structured knowledge basis, in which concepts, instances, their attributes and relationships are modelled. Thus, they are formal specifications that can be used by agents to talk in a common language, since they model a domain in a strict way.

In the Web Semantic field, it's used the OWL language (Web Ontology Language) (McGuinness and van Harmelen, 2004) as representation language of knowledge with ontologies. OWL language is a standard for knowledge representation promoted by World Wide Web Consortium³ (W3C) in order to create a standardization that allows the Web Semantic vision proposed by Tim Berners-Lee. To that end, the development committee was based on three essential pillars: descriptive logic formality, representation abilities of frame-based systems and RD-F/XML as method to information exchange.

The main problems of formal classification systems, like ontologies, come derived from the power of expression and formality that they have, that make them, in many cases, very hard to create and use, besides presenting important scalability problems.

It's very complicated, for example, that users that generate content in a social network, normally not experts in knowledge representation, are capable of classifying each content that they produce according to a large ontology full of options of classification. In addition it's difficult to represent a priori within an ontology all the spaciousness of knowledge domain susceptible to be used and, therefore, to do it having reached an agreement with other related social networks. Do not forget that the knowledge structure that is wanted to represent in a system is rarely static and that needs change throughout time, making required a continuous guidance of classification criteria and its adaptation of needs in each moment.

Some authors think that two of the main reasons that make hard to create and use ontologies are, on one hand, the dynamicity of concepts they represent and, on the other hand, the difference between benefit and cost to create or use and ontology in contrast to not use.

Other people point out that folksonomies can reduce distance between Social Web and Semantic Web, thanks to that they can generate steady structures of knowledge using

³ W3C, <http://w3c.org>

the categorization provided by folksonomies as base on which build ontologies. Thus, ontologies derived from folksonomies will represent a collective knowledge instead of the perception of a limited number of experts. They also considers that another folksonomies important contribution is the help in the ontologies evolution. The knowledge represented in ontologies is not static, but evolving over time, in consequence it's required appropriated mechanisms that allow to capture these changes in the modelled domains and apply them to corresponding ontologies.

The monitoring changes in folksonomies can help to obtain this variations in domains, in order to the person responsible of manage ontologies can carry out the changes. Lastly, the dynamicity of folksonomies can be also used to help populating ontologies with information supplied by users. This way, folksonomies can help with the creation and evolution of ontologies, as well as its population with information coming from tagging of users.

2.1.5. SAM

In the thesis defended by [\(Francisco Echarte, 2011\)](#), it's proposed a method for the Automatic Classification of Annotated Resources, ACoAR (which is another nomenclature of SAM), in collaborative tagging systems. It has as main objective to improve the knowledge representation and information recovery in collaborative tagging systems, making an automatic classification of resources of a set of classification concepts.

This automatic method classify resources of a folksonomy into a set of classification concepts, using the semantic information obtained from the number of users' annotations.

SAM can be used in existing systems creating automatically classification concepts and classifying the available resources in that moment. Therefore, it's applied to the evolution of these systems, processing the new generated information and adapting concepts and classification to that evolution.

This method is based on an open architecture of components that allows its application to tagging systems of different characteristics. Thanks to this modularity, methods and

algorithms used in SAM can be changed in an easy way without affecting other parts of the program.

One of the keys of SAM success is that it's only used a subset of all existing tags for representing the semantic of resources and classification concepts. This technic allows to choose those tags which are the most representative ones to represent the whole set of tags. This makes that the method get a meaningful fewer computational cost than using the full set of tags without losing relevant information.

This subset of tags is made by only keeping in mind the number of times a tag is labelled to a resources. In other words, that one tag which have more number of occurrences in a resource.

At first sight, this could be a very nice approach. Indeed, this method gets a quite good classification of resources. It can be seen at ([Córdoba, Astrain, Villadangos and Echarte, 2013](#)), the explanatory paper of SAM, how resources are classified depending on the number of occurrences of tags. It shows not only the classification but the evolution too.

2.2. Motives

The motives to do this research can be divided in 3 types: scientific, personal, and related with other projects in which this work is, somehow, connected. Evidently, scientific motives are the most important ones and which, as a last resort, joined together with the two others, being the same as a whole.

The main scientific incentive to carry out this project is to find out a better classification of resources in social tagging system in order to improve the knowledge representation and information recovery. In this sense, since a couple of years, several researchers from the Public University of Navarra created a method for the automatic classification of annotated resources called SAM (or ACoAR), as shown previously, which was also the thesis of Francisco Echarte. However, the manner SAM classifies content could not be the best one (as it will be explained on the following section) so a new methods were proposed to improve it.

Regarding the personal motivations, I can consider myself as a restless person always looking for a challenge. I have previously cooperated with my university thanks to some collaboration grants and those experiences reinforced my passion of learning new things and increased my knowledge about computer science, touching very dissimilar fields. So I faced with this new challenge where I could do more than learn but contribute.

As for related projects, [\(Ciordia, 2011\)](#) took the advantages of SAM to implement a classification of resources from a medical image folksonomy and to obtain new ways of recovery information without modifying how users tag content.

It should be pointed out that there are many research lines related to SAM that have not been investigated yet, which are very interesting and can give us brighter lights of classifying resources in social tagging systems.

2.3. Goals and hypothesis

However, and despite the good results of SAM classification, it suffers from a weakness and it lies on its basis. The fact that to choose the number of occurrences of a tag as a good characteristic to mark it as “representative” could not be the best election. This means that the tags which appears the most do not have to be the most characteristic.

For example, and as shown at [\(Golder and Huberman, 2006\)](#), there are users that labels resources with non-descriptive tags but as personal use tags, such as “toRead”, “myJob”, “interesting” etc., which prove that not each tag of a resource could be representative. This situation makes the number of tags occurrences not be the best choice for selecting the representative subset of tags.

The solution of this problem would be to find out the degree of fitness of each tag for each resource, so that make the subset of tags with those tags with the most qualification, that is, the most representative ones.

2.4. Proposed solution

This last hint became the first step of this project. As shown at the previous section, choosing the number of occurrences of a tag could not be the most suitable characteristic to decide if a tag is representative or not.

So that's why from here it's proposed a variation of SAM classification method. Instead of taking into account the occurrences of a tag, it's decided to choose by a semantic evaluations. This will ensure that the selected tags will be the most representative ones.

It's desired to check how resources are classified using semantic evaluations and compare the resulting classification with which that uses the syntactic information (number of occurrences).

To that end, this project will be made up of two parts. For the first part, a subset of annotations will be chosen so that a couple of 10 experts can evaluate it; concretely, the annotations of 47 random resources. This subset will be called "R47". The whole set of annotations belongs to that one used in the [\(Córdoba, Astrain, Villadangos and Echarte, 2013\)](#) research, corresponding a dump of a database of Del.icio.us. So that's why it will be called "delicious set". It will be study how the classification is done in this little subset, comparing the use of syntactic (occurrences) or semantic (fitness) information.

For the second part, it will be used the semantic information of that subset to establish which tags are the most representative ones and to make with them the classification of the whole set of annotations, varying a threshold which define the strength of the fitness chosen. Also, the results will be comparing with the classification obtained using the syntactic method.

The aim of this research is to verify that this new semantic method gets a better and accurate classification than the syntactic one due to the expert evaluation of tags.

2.5. Document Organization

This project is structured as follows:

- Chapter I – Antecedents:

A brief preamble of what is going to be about.

- Chapter II – Introduction:

An introduction to how information evolve with time, how change the roll of users, what folksonomies are, why knowledge representation and information recovery were needed and what current methods were used to deal with it, concretely SAM.

- Chapter III – Analysis and design:

Both SAM, S-SAM and S-SAM FULL prototypes will be defined, analysed in depth and explained with descriptive examples. Then, it's going to detail what inputs they need and what they output for R47 and for delicious databases.

- Chapter IV – Implementation:

It will establish which tools are necessary to run the programs and how they have to be configured in order to execute properly. Some tips are mentioned too.

- Chapter V – Experimental Results:

It will be showed the output results and summarized into graphics and statistics and extracted some conclusions.

- Chapter VI – Conclusions:

It will be gathered all results from the previous chapter and analysed in depth in order to choose the best method for categorizing resources.

- Chapter VII – Future Lines:

Some new research lines will be established to continue going deep with this study of SAM.

- Chapter VIII – Bibliography:

All the used resources and online references to develop this project is gathered here.

Chapter III:

ANALYSIS AND DESIGN



In this chapter it will be examined the different project needs and the implementation of all goals giving application an easy, clear and useful design. This is desired to be done in the most efficient way, due to this program manage huge amounts of data and have to be as fast as possible. That's why ACoAR is going to be analysed and totally redesigned in order to satisfy the new requirements.

3.1. SAM prototype

The SAM method classifies resources annotated by users by means of a classifier. The method needs the extension of the tagging system definition ([Hotho, Jäschke, Schmitz & Stumme, 2006a, 2006b](#)) in order to provide the automatic classification of resources.

This extension is called *SAMsModel*. The classification of resources is performed by means of the classifier *SAM_classifier* and two actions:

- *SAM_classifier*: classifies the resources.
- *SAMsModel_creation*: generates the *SAMsModel* from a previously existing tagging system providing classified resources according to their similarities (initial phase).
- *SAMsModel_evolution* (self-adapting phase): provides resources classified as the tagging system evolves.

3.1.1. SAMsModel definitions

According to [\(Hotho, Jäschke, Schmitz & Stumme, 2006b\)](#), a social tagging system can be defined as a tuple TS as depicted in Definition 1.

Definition 1. A social tagging system can be defined as a tuple $TS := \langle U, R, T, A \rangle$ where U is the set of users, R is the set of resources, T represents the set of tags and $A: U \times R \times T$ is a ternary relation representing the set of annotations:

$$A \subseteq \{\langle u, r, t \rangle : u \in U, r \in R, t \in T\}$$

Definition 2. An occurrence of a given tag t_i for a resource r_j (a given resource r_j for a tag t_i) is given by the number of triplets (u_k, r_j, t_i) where u_k is any user that belongs to U .

$$occur_{i,j}(r_j, t_i) = card(u_k, r_j, t_i) \in A | u_k \in U$$

Definition 3. A given tag t_i is representative when the total amount of its occurrences is greater or equal than a certain threshold value. SAM uses this set to represent the similarity of the resources and to reduce the computational and memory cost.

Definition 4. Let $SAMsModel := \langle U, R, T, A, S_{rt}, RC, Z, V, S_{CR} \rangle$ be SAM to represent the information needed by the classification method where: R is the set of resources where each resource has three different states (pending, converged and classified); S_{rt} is the set of representative tags of the social tagging system; RC is the set of resource categories or categories; Z is the set which links the categories with the resources they classify; V is the set of vectors associated to categories, resources and tags of the set S_{rt} ; and finally, S_{CR} is a set of similarity measures between a resource and a category.

Definition 5. The set of resources of the tagging system R consists of three pairwise disjoint subsets: $R_{pending}$, $R_{converged}$, $R_{classified}$; such that:

$$\begin{aligned} R &= R_{pending} \amalg R_{converged} \amalg R_{classified} = \\ &= R_{not\ classified} \cup R_{classified} | R_{not\ classified} = R_{pending} \cup R_{converged} \end{aligned}$$

The resources classified under a given category belong to $R_{classified}$. The resources not classified belong initially to $R_{pending}$, and the move to $R_{converged}$ as they converge. This convergence depends on the kind of social tagging system. For example, in a broad folksonomy typically depends on the number of occurrences by resources. A resource converges on Del.icio.us when it reaches 100 occurrences ([Golder and Huberman, 2006](#)).

Definition 6. Let RC be the set of categories on which resources of the social tagging system are classified.

Definition 7. Let Z be the set of pairs (r, c) where $r \in R_{classified}$ and $c \in RC$, representing that resource r is classified under the category c .

Definition 8. The set V consists of three subsets V_R , V_C and $V_{S_{rt}}$, which are the set of vectors which represent the resources, the categories and the tags of S_{rt} , respectively.

$$V = V_R \cup V_C \cup V_{S_{rt}} | V_R = \{V_r | r \in R\}, V_C = \{V_c | c \in RC\}, V_{S_{rt}} = \{V_s | s \in S_{rt}\}$$

The corpus consisting of a set of resources and a set of representative tags (S_{rt}) is represented by a matrix $M = (m_{ij}) \in \mathfrak{R}^{R \times S_{rt}}$. Each row vector m_i corresponds to a resource $r_i \in R$ and each column vector corresponds to a tag t_i of S_{rt} . Each m_{ij} represents the number of occurrences that relates t_i to r_i .

Definition 9. Let $V_r \in V_R$ be the vector defined as follows:

$$V_r = (v_i)_{1 \leq i \leq |S_{rt}|} | v_i = |(u_k, r, s_i) \in A| \quad \forall u_k \in U, \quad \forall s_i \in S_{rt}$$

Definition 10. Let $V_c \in V_C$ be the vector defined as the summation of the vectors of the resources classified by it. Each position of the vectors represents the total amount of occurrences assigned to the corresponding tag t ($t \in S_{rt}$).

$$V_c = \sum_{k=1}^{k \leq |R|} V_{r_k} | (r_k, c) \in Z$$

Definition 11. Let $V_s \in V_{S_{rt}}$ be the vector defines as follows:

$$V_s = (v_i)_{1 \leq i \leq |S_{rt}|} | v_i = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if rest} \end{cases}$$

3.1.2. SAM classifier definition

The SAM classifier (*SAM_classifier*) provides the classification of the converged resources of a given social tagging system using a set of resource categories. *SAM_classifier* may use two different classification modes: *classif_sim* and *classif_delta*.

Definition 12. Given a certain resource $r_i \in R_{converged}$, a category $c_{cand} \in RC$ is its candidate category if, and only if, $\forall c_j \in RC, c_j \neq c_{cand}, similarity(c_j, r_i) < similarity(c_{cand}, r_i)$.

Definition 13. A resource $r \in R_{converged}$ is classified under a candidate category $c_{cand} \in RC$ by `classif_sim` if and only if the similarity degree between r and c_{cand} is greater than a given th_{sim} . That is, $classif_sim_{c_{cand}}(r) = true \leftrightarrow S_{CR}(c_{cand}, r) \geq th_{sim}$.

Definition 14. A resource $r \in R_{converged}$ is classified under a candidate category $c_{cand} \in RC$ by `classif_delta` if and only if $\forall c_p, c_k \in RC | c_p \neq c_{cand} \wedge c_k \neq c_{cand}, S_{CR}(c_{cand}, r) - S_{CR}(c_k, r) \geq S_{CR}(c_{cand}, r) - S_{CR}(c_p, r) \geq th_{\delta}$. That is, c_p is the category whose similarity degree is the closest to $similarity(c_{cand}, r)$.

The classifier mode `classif_delta` is used to increase the precision of the classification when there exists similar categories or resources that can be classified under different candidate categories. A high value of $S_{CR}(c_{cand}, r) - S_{CR}(c_p, r)$ implies a correct classification of the resource under the category.

3.1.3. SAM actions and components

This subsection describes the actions used by SAM to classify resources from an existing social tagging system (`SAMsModel_creation`) and how the social tagging system evolves with new occurrences (`SAMsModel_evolution`), creating and evolving the `SAMsModel` respectively.

3.1.3.1. SAMsModel_creation

Given a social tagging system, SAM initially creates a set of categories where resources are classified, and it assigns a name to each category according to the information

provided by the resources classified in each category and their occurrences. Once created the categories, each new annotation of the social tagging system is processed and updating the similarity information and adapting the categories when necessary.

SAM starts with the creation of the vectorial representation of the resources and the set of representative tags (S_{rt}) of the social tagging system. Component *Representations* is in charge of these tasks.

Each resource is assigned to subsets $R_{converged}$ or $R_{pending}$ in terms of whether they have converged or not. The component *Convergence* uses the number of occurrences associated to the S_{rt} set to assign the resources to $R_{converged}$.

The component *Clustering* clusters the resources of the social tagging system belonging to $R_{converged}$ in a set of categories, generating the set of similar categories on which resources of the social tagging system are classified and the set of pairs (r, c) where $r \in R$ and $c \in RC$, representing that resource r is classified into the category c (Z).

The component *MergingSplitting* analyses the categories provided in order to evaluate the convenience of merging or splitting any of them, updating RC and Z sets.

The component *Classifier* is in charge of grouping the resources under those categories with which they have high similarity by comparing all the resources belonging to the $R_{converged}$ set with the categories in which they are grouped in Z . The component assigns the resource to the $R_{classified}$ set according to the similarity measures between each resource and its category or keeps it on $R_{converged}$ and removes it from Z .

Finally, the component *Representations* creates the vector representations for the categories using the Z set, and the component *Naming* assigns meaningful names to these categories according to the tags with higher weights.

The following table shows all this just commented actions of *SAMsModel_creation*.

Table 1. Initial phase: *SAMsModel_creation*.

1. Representations:: *createS_{rt}*()
2. Representations:: *createVectors*()
3. **forall** $r \in R$ **do**
4. **if** Convergence:: *hasConverged*(r) **then**
5. assign r to $R_{converged}$
6. **else** assign r to $R_{pending}$
7. **endif**
8. **endforall**
9. Clustering:: *create*($R_{converged}$)
10. MergingSplitting:: *process*(RC, Z)
11. **forall** $r \in R_{converged}$ **do**
12. **if** Classifier:: *isCorrectlyClassified*(Z, r) **then**
13. assign r to $R_{classified}$
14. **else** drop r from Z
15. **endif**
16. **endforall**
17. Representations:: *createConceptVectors*(RC, Z)
18. Naming:: *process*(RC, Z)

3.1.3.2. *SAMsModel_evolution*

The scope of this project does not involve the evolution of *SAMsModel*. Hence this part of the project has not been developed.

3.1.4. Descriptive example

This subsection illustrates the behaviour of the actions *SAMsModel_creation* using the social tagging system described in Table 2. Example of social tagging system for SAM..

Table 2. Example of social tagging system for SAM.

Tag		Resource							
		r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
t_1	blog	47	3	3	4	6	1	21	0
t_2	blogs	11	4	0	0	0	3	0	0
t_3	twitter	26	0	5	2	9	4	0	0
t_4	socialweb	31	1	0	3	0	5	7	0
t_5	blogging	7	2	3	0	0	0	5	0
t_6	art	0	98	37	1	0	0	0	0
t_7	museum	4	41	75	0	0	0	0	4
t_8	warhol	0	23	11	4	0	0	1	4
t_9	picasso	1	81	15	0	0	4	0	0
t_{10}	history	0	1	9	0	0	2	0	1
t_{11}	architecture	4	33	6	0	7	0	21	2
t_{12}	java	0	1	1	18	24	3	0	11
t_{13}	programming	0	0	3	75	21	29	18	27
t_{14}	python	1	0	0	0	13	0	13	36
t_{15}	mysql	0	0	0	6	0	2	1	4
t_{16}	database	0	0	0	7	0	7	0	10
t_{17}	ruby	1	4	4	0	0	41	0	1
t_{18}	ajax	1	1	3	2	12	4	1	6
t_{19}	ror	1	4	0	0	0	2	5	1
t_{20}	opensource	4	5	0	15	3	3	19	21

The rows correspond to the tags of the social tagging system (20), and the columns to the resources (8). The value of each cell is the number of occurrences. For example, $occur_{2,4}(r_4, blogs) = 0$ and $occur_{12,5}(r_5, java) = 24$. The information concerning users is not displayed since it is not necessary. In this example, we use the following components:

- (1) *Representations*, which includes all the tags with a minimum number of 20 occurrences.
- (2) *Converge*, with a total minimum of 100 occurrences associated to tags of S_{rt} .
- (3) The *classif_sim* mode with a $th_{sim} = 0.7$ as *Classifier*.
- (4) *Clustering*, based in k-means with $k = 3$.
- (5) *MergingSplitting*, which merges two categories when their similarity is greater than 0.8.
- (6) *Naming*, which assigns the name of categories concatenating the two more representative tags in each category.

(7) *RecalculationCondition*, which performs the re-calculus of S_{rt} and RC each time a resource moves from $R_{converged}$ to $R_{classified}$.

Vectors representing R , RC and S_{rt} are compared using the cosine similarity measure. Table 3. Example of V and S_{CR} sets shows V and S_{CR} sets.

First of all, the action *SAMsModel_creation* is applied to the social tagging system. Representation builds $S_{rt} = \{t_1, t_3, t_4, t_6, t_7, t_8, t_9, t_{11}, t_{12}, t_{13}, t_{14}, t_{16}, t_{17}, t_{18}, t_{20}\}$. After the execution of the component *Convergence*, all the resources move to the set $R_{converged}$ except the resource r_5 which remains in the set $R_{pending}$ since it does not reach the total minimum number of occurrences (100). After applying the k-means algorithm, three categories are obtained. The resources classified under those categories are $c_1 = \{r_1\}$, $c_2 = \{r_2, r_3\}$ and $c_3 = \{r_4, r_6, r_7, r_8\}$.

The component *MergingSplitting* does not update the sets RC and Z because the values of the similarity measures among categories are lower than 0.8. As the values of the similarity measures between r_7 and all the categories are lower than 0.7 (th_{sim}), the resource r_7 does not belong to $R_{classified}$. Then,

$$R_{pending} = \{r_5\}, R_{converged} = \{r_7\} \text{ and } R_{classified} = \{r_1, r_2, r_3, r_4, r_6, r_8\}.$$

In this example, the categories are renamed as $c_1 = \text{"blog \& socialweb"}$, $c_2 = \text{"art \& museum"}$ and $c_3 = \text{"programming \& ruby"}$. Finally, the component *Classifier* performs the calculus of the similarity measures between categories and resources.

Table 3. Example of V and S_{CR} sets for SAM.

Vectors	
V_{r_1}	(47, 26, 31, 0, 4, 0, 1, 4, 0, 0, 1, 0, 1, 1, 4)
V_{r_2}	(3, 0, 1, 98, 41, 23, 81, 33, 1, 0, 0, 0, 4, 1, 5)
V_{r_3}	(3, 5, 0, 37, 75, 11, 15, 6, 1, 3, 0, 0, 4, 3, 0)
V_{r_4}	(4, 2, 3, 1, 0, 4, 0, 0, 18, 75, 0, 7, 0, 2, 15)
V_{r_5}	(6, 9, 0, 0, 0, 0, 0, 7, 24, 21, 13, 0, 0, 12, 3)
V_{r_6}	(1, 4, 5, 0, 0, 0, 4, 0, 3, 29, 0, 7, 41, 4, 3)
V_{r_7}	(21, 0, 7, 0, 0, 1, 0, 21, 0, 18, 13, 0, 0, 1, 19)
V_{r_8}	(0, 0, 0, 0, 4, 4, 0, 2, 11, 27, 36, 10, 1, 6, 21)

S_{CR}	c_1	c_2	c_3
r_1	1.0000	0.0912	0.0950
r_2	0.0658	0.9552	0.0588
r_3	0.1145	0.8777	0.0881
r_4	0.0799	0.0394	0.9238
r_5	0.2399	0.0651	0.7275
r_6	0.1139	0.0803	0.7595
r_7	0.5221	0.1288	0.5822
r_8	0.0460	0.0835	0.7924

Vectors

V_{d_1}	(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_2}	(0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_3}	(0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_4}	(0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_5}	(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_6}	(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_7}	(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_8}	(0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)
V_{d_9}	(0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
$V_{d_{10}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)
$V_{d_{11}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
$V_{d_{12}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0)
$V_{d_{13}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)
$V_{d_{14}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
$V_{d_{15}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)
V_{c_1}	(47, 26, 31, 0, 4, 0, 1, 4, 0, 0, 1, 0, 1, 1, 4)
V_{c_2}	(6, 5, 1, 135, 116, 34, 96, 39, 2, 3, 0, 0, 8, 4, 5)
V_{c_3}	(5, 6, 8, 1, 4, 8, 4, 2, 32, 131, 36, 24, 42, 12, 39)

3.2. S-SAM prototype

Our proposal, i.e. Semantic SAM or S-SAM, consists in a lightly modification in dictionary creation. Instead of taking into account the occurrences of a tag, it's decided to choose tags by semantic evaluations. Hence it will ensure that the selected tags will be the most representative ones. That's why this new method was called Semantic SAM, because it takes the meaning (fitness) of a tag regarding its resource.

3.2.1. S-SAMs Model definitions

So it's necessary some modifications of the definition of SAM given above. Next, it will be rewritten those one that need changes.

Definition 2.1. An occurrence of a given tag t_i for a resource r_j (a given resource r_j for a tag t_i) is given by the number of triplets (u_k, r_j, t_i) where u_k is any user that belongs to U .

$$occur_{i,j}(r_j, t_i) = \text{card}(u_k, r_j, t_i) \in A | u_k \in U$$

Definition 2.2. The fitness of a given tag t_i for a resource r_j (a given resource r_j for a tag t_i) is an evaluation between 0 and 1 that measures the level of adequacy of that tag over that resource.

$$fitn_{i,j}(r_j, t_i) = n \in \mathbb{R} | 0 \leq n \leq 1$$

Definition 3. A given tag t_i is representative when its fitness value is greater or equal than a certain threshold value. S-SAM uses this set to represent the similarity of the resources and to reduce the computational and memory cost.

Definition 3.1. There are two ways to choose the fitness of a tag: the MAX method, which selects the higher fitness value of all annotations of the tag, and the MEAN method, which makes a mean of all fitness value of the annotations of the tag.

The rest of the definitions that do not appear here remains as they are.

3.2.2. S-SAM classifier definition

The definitions related to classifier also remain in the same way as SAM.

3.2.3. S-SAM actions and components

This subsection describes the actions used by S-SAM to classify resources from an existing social tagging system (*S-SAMsModel_creation*) and how the social tagging system evolves with new occurrences (*S-SAMsModel_evolution*), creating and evolving the *S-SAMsModel* respectively.

3.2.3.1. S-SAMsModel_creation

Due to S-SAM its quite similar to SAM actions, it's going to focus only in the changed parts and leave the rest as it is.

S-SAM also starts with the creation of the vectorial representation of the resources and the set of representative tags (S_{rt}) of the social tagging system. However, those tags are not going to be chosen by the number of occurrences, but the consensus semantic evaluation from experts that have been previously rated a subset of the annotations. This evaluation is going to be called *fitness*.

Despite of these changes, *S-SAMsModel* follows exactly the same actions that *SAMsModel* does, so the algorithm pseudocode it will be the same.

3.2.3.2. S-SAMsModel_evolution

The scope of this project does not involve the evolution of *S-SAMsModel* too. Hence this part of the project has not been developed.

3.2.4. Descriptive example

As done with SAM, S-SAM will also be explained with an example. Using the following social tagging system as described in Table 4. Example of social tagging system for S-SAM.. The rows correspond to the tags of the social tagging system (20), and the columns to the resources (8). The value of each cell is a pair where the first number is the fitness value and the second one, the number of occurrences.

Table 4. Example of social tagging system for S-SAM.

Tag	Resource	Resource								MAX
		r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	
t_1	blog	(0.9 , 47)	(0.4, 3)	(0.0, 3)	(0.2, 4)	(0.7, 6)	(0.5, 1)	(0.7, 21)	(0.0, 0)	0.9
t_2	blogs	(0.5, 11)	(0.8 , 4)	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.6, 3)	(0.0, 0)	(0.0, 0)	0.8
t_3	twitter	(0.4 , 26)	(0.0, 0)	(0.4, 5)	(0.3, 2)	(0.1, 9)	(0.4, 4)	(0.0, 0)	(0.0, 0)	0.4
t_4	socialweb	(0.8 , 31)	(0.2, 1)	(0.0, 0)	(0.5, 3)	(0.0, 0)	(0.4, 5)	(0.3, 7)	(0.0, 0)	0.8
t_5	blogging	(0.8, 7)	(1.0 , 2)	(0.5, 3)	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.6, 5)	(0.0, 0)	1.0
t_6	art	(0.0, 0)	(0.7 , 98)	(0.5, 37)	(0.6, 1)	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.0, 0)	0.7
t_7	museum	(0.2, 4)	(0.5 , 41)	(0.5, 75)	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.0, 4)	0.5
t_8	warhol	(0.0, 0)	(0.6, 23)	(0.7 , 11)	(0.4, 4)	(0.0, 0)	(0.0, 0)	(0.5, 1)	(0.6, 4)	0.7
t_9	picasso	(0.5, 1)	(0.6 , 81)	(0.6, 15)	(0.0, 0)	(0.0, 0)	(0.1, 4)	(0.0, 0)	(0.0, 0)	0.6
t_{10}	history	(0.0, 0)	(0.7, 1)	(0.6, 9)	(0.0, 0)	(0.0, 0)	(0.8 , 2)	(0.0, 0)	(0.5, 1)	0.8
t_{11}	architecture	(0.5, 4)	(0.9 , 33)	(0.6, 6)	(0.0, 0)	(0.7, 7)	(0.0, 0)	(0.6, 21)	(0.5, 2)	0.9
t_{12}	java	(0.0, 0)	(0.7 , 1)	(0.3, 1)	(0.7, 18)	(0.5, 24)	(0.5, 3)	(0.0, 0)	(0.7, 11)	0.7
t_{13}	programming	(0.0, 0)	(0.0, 0)	(0.5 , 3)	(0.5, 75)	(0.5, 21)	(0.4, 29)	(0.5, 18)	(0.5, 27)	0.5
t_{14}	python	(0.0, 1)	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.4, 13)	(0.0, 0)	(0.6, 13)	(1.0 , 36)	1.0
t_{15}	mysql	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.5, 6)	(0.0, 0)	(0.2, 2)	(1.0 , 1)	(1.0, 4)	1.0
t_{16}	database	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.4, 7)	(0.0, 0)	(0.8, 7)	(0.0, 0)	(0.9 , 10)	0.9
t_{17}	ruby	(0.0, 1)	(0.2, 4)	(0.2, 4)	(0.0, 0)	(0.0, 0)	(0.3, 41)	(0.0, 0)	(0.3 , 1)	0.3
t_{18}	ajax	(0.8, 1)	(0.6, 1)	(1.0 , 3)	(0.5, 2)	(0.5, 12)	(0.7, 4)	(0.7, 1)	(0.6, 6)	1.0
t_{19}	ror	(0.0, 1)	(0.1, 4)	(0.0, 0)	(0.0, 0)	(0.0, 0)	(0.8, 2)	(0.9 , 5)	(0.8, 1)	0.9
t_{20}	opensource	(0.4, 4)	(0.5 , 5)	(0.0, 0)	(0.5, 15)	(0.2, 3)	(0.1, 3)	(0.5, 19)	(0.2, 21)	0.5

For example, $occur_{2,4}(r_4, blogs) = 0$ and $occur_{12,5}(r_5, java) = 24$ and $fitn_{2,4}(r_4, blogs) = 0.0$ and $fitn_{12,5}(r_5, java) = 0.5$. The information concerning users is not displayed since it is not necessary. In this example, we use the following components:

- (1) *Representations*, which includes all the tags with a minimum fitness value of 0.6 choosing the MAX method.
- (2) *Convergence*, with a total minimum of 100 occurrences associated to tags of S_{rt} .
- (3) The *classif_sim* mode with a $th_{sim} = 0.7$ as *Classifier*.
- (4) *Clustering*, based in k-means with $k = 3$.
- (5) *MergingSplitting*, which merges two categories when their similarity is greater than 0.8.
- (6) *Naming*, which assigns the name of categories concatenating the two more representative tags in each category.
- (7) *RecalculationCondition*, which performs the re-calculus of S_{rt} and RC each time a resource moves from $R_{converged}$ to $R_{classified}$.

Vectors representing R , RC and S_{rt} are compared using the cosine similarity measure. Table 5 shows V and S_{CR} sets.

First of all, the action *S-SAMsModel_creation* is applied to the social tagging system. Representation builds $S_{rt} = \{t_1, t_2, t_4, t_5, t_6, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{14}, t_{15}, t_{16}, t_{18}, t_{19}\}$. After the execution of the component *Convergence*, all the resources move to the set $R_{converged}$ except the resource r_5 which remains in the set $R_{pending}$ since it does not reach the total minimum number of occurrences (100). After applying the k-means algorithm, three categories are obtained. The resources classified under those categories are $c_1 = \{r_1\}$, $c_2 = \{r_2, r_3\}$ and $c_3 = \{r_4, r_6, r_7, r_8\}$.

The component *MergingSplitting* does not update the sets RC and Z because the values of the similarity measures among categories are lower than 0.8. As the values of the similarity measures between r_4, r_6 and r_8 and all the categories are lower than 0.7 (th_{sim}), these resources do not belong to $R_{classified}$. Then,

$$R_{pending} = \{r_5\}, R_{converged} = \{r_4, r_6, r_8\} \text{ and } R_{classified} = \{r_1, r_2, r_3, r_7\}.$$

In this example, the categories are renamed as $c_1 = \text{"blog \& socialweb"}$, $c_2 = \text{"art \& museum"}$ and $c_3 = \text{"blog \& architecture"}$. Finally, the component *Classifier* performs the calculus of the similarity measures between categories and resources.

Table 5. Example of V and S_{CR} sets for S -SAM.

Vectors	
V_{r_1}	(47, 11, 31, 7, 0, 0, 1, 0, 4, 0, 1, 0, 0, 1, 1)
V_{r_2}	(3, 4, 1, 2, 98, 23, 81, 1, 33, 1, 0, 0, 0, 1, 4)
V_{r_3}	(3, 0, 0, 3, 37, 11, 15, 9, 6, 1, 0, 0, 0, 3, 0)
V_{r_4}	(4, 0, 3, 0, 1, 4, 0, 0, 0, 18, 0, 6, 7, 2, 0)
V_{r_5}	(6, 0, 0, 0, 0, 0, 0, 0, 7, 24, 13, 0, 0, 12, 0)
V_{r_6}	(1, 3, 5, 0, 0, 0, 4, 2, 0, 3, 0, 2, 7, 4, 2)
V_{r_7}	(21, 0, 7, 5, 0, 1, 0, 0, 21, 0, 13, 1, 0, 1, 5)
V_{r_8}	(0, 0, 0, 0, 0, 4, 0, 1, 2, 11, 36, 4, 10, 6, 1)

S_{CR}	c_1	c_2	c_3
r_1	1.0000	0.0640	0.6341
r_2	0.0579	0.9950	0.1745
r_3	0.0816	0.9572	0.1374
r_4	0.2289	0.0918	0.1491
r_5	0.1852	0.0739	0.3955
r_6	0.3611	0.2222	0.2137
r_7	0.6341	0.1663	1.0000
r_8	0.0221	0.0390	0.3674

Vectors	
V_{d_1}	(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_2}	(0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_3}	(0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_4}	(0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_5}	(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_6}	(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_7}	(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_8}	(0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)
V_{d_9}	(0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
$V_{d_{10}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)
$V_{d_{11}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
$V_{d_{12}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0)
$V_{d_{13}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)
$V_{d_{14}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
$V_{d_{15}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)
V_{c_1}	(47, 11, 31, 7, 0, 0, 1, 0, 4, 0, 1, 0, 0, 1, 1)
V_{c_2}	(6, 4, 1, 5, 137, 34, 96, 10, 39, 2, 0, 0, 0, 4, 4)
V_{c_3}	(21, 0, 7, 5, 0, 1, 0, 0, 21, 0, 13, 1, 0, 1, 5)

3.3. S-SAM FULL prototype

Another idea that arose from this project was the use of fitness evaluation not only to decide the representative tags of a social tagging system but to make the following calculus with them. This way it will ensure that the selected tags will be the most representative ones and, with this information, can act accordingly. This variation of S-SAM method has been called S-SAM FULL, because it pretends to perform only with the meaningful information (fitness).

3.3.1. S-SAMFULLsModel definitions

Since it's going to change the information type (and its origin), it's necessary some modifications of the definition of SAM given above. That's the reason why it will be rewritten all of them, except Definition 1, which remains, and Definition 2, which disappears.

Definition 2. The fitness of a given tag t_i for a resource r_j (a given resource r_j for a tag t_i) is an evaluation between 0 and 1 that measures the level of adequacy of that tag over that resource.

$$fitn_{i,j}(r_j, t_i) = n \in \mathbb{R} | 0 \leq n \leq 1$$

Definition 3. A given tag t_i is representative when its fitness value is greater or equal than a certain threshold value. S-SAM FULL uses this set to represent the similarity of the resources and to reduce the computational and memory cost.

Definition 3.1. There are two ways to choose the fitness of a tag: the MAX method, which selects the higher fitness value of all annotations of the tag, and the MEAN method, which makes a mean of all fitness value of the annotations of the tag.

Definition 8. The set V consists of three subsets V_R , V_C and $V_{S_{rt}}$, which are the set of vectors which represent the resources, the categories and the tags of S_{rt} , respectively.

$$V = V_R \cup V_C \cup V_{S_{rt}} | V_R = \{V_r | r \in R\}, V_C = \{V_c | c \in RC\}, V_{S_{rt}} = \{V_s | s \in S_{rt}\}$$

The corpus consisting of a set of resources and a set of representative tags (S_{rt}) is represented by a matrix $M = (m_{ij}) \in \mathfrak{R}^{R \times S_{rt}}$. Each row vector m_i corresponds to a resource $r_i \in R$ and each column vector corresponds to a tag t_i of S_{rt} . Each m_{ij} represents the fitness value that relates t_i to r_i .

Definition 9. Let $V_r \in V_R$ be the vector defined as follows:

$$V_r = (v_i)_{1 \leq i \leq |S_{rt}|} | v_i = |(u_k, r, s_i) \in A| \forall u_k \in U, \quad \forall s_i \in S_{rt}$$

Definition 10. Let $V_c \in V_C$ be the vector defined as the summation of the vectors of the resources classified by it. Each position of the vectors represents the total average fitness assigned to the corresponding tag t ($t \in S_{rt}$).

$$V_c = \sum_{k=1}^{k \leq |R|} V_{r_k} | (r_k, c) \in Z$$

3.3.2. S-SAM FULL classifier definition

The definitions related to classifier also remain in the same way as SAM.

3.3.3. S-SAM FULL actions and components

This subsection describes the actions used by S-SAM to classify resources from an existing social tagging system ($S-SAMFULLsModel_creation$) and how the social tagging system evolves with new occurrences ($S-SAMFULLsModel_evolution$), creating and evolving the $S-SAMFULLsModel$ respectively.

3.3.3.1. S-SAMFULLsModel_creation

Due to S-SAM FULL differs in some parts from SAM actions and we are going to focus only in the changed parts and leave the rest as it is.

S-SAM FULL also starts with the creation of the vectorial representation of the resources and the set of representative tags (S_{rt}) of the social tagging system. As S-SAM does, the tags are chosen not by the number of occurrences but by their fitness.

Then, each resource is assigned to subsets $R_{converged}$ or $R_{pending}$ in terms of whether they have converged or not. The component *Convergence* uses the average fitness value associated to the S_{rt} set to assign the resources to $R_{converged}$.

From this point forward, *S-SAMFULLsModel* follows exactly the same actions that *SAMsModel* does, so the algorithm pseudocode it will be the same.

3.3.3.2. S-SAMFULLsModel_evolution

The scope of this project does not involve the evolution of *S-SAMsModel* too. Hence this part of the project has not been developed.

3.3.4. Descriptive example

As done with SAM, S-SAM FULL will also be explained with an example. Using the following social tagging system as described in Table 6. The rows correspond to the tags of the social tagging system (20), and the columns to the resources (8). The value of each cell is the value of fitness.

Table 6. Example of social tagging system for S-SAM FULL.

Tag	Resource	Resource								MAX	CVG
		r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8		
t_1	blog	0.9	0.4	0.0	0.2	0.7	0.5	0.7	0.0	0.9	0.43
t_2	blogs	0.5	0.8	0.0	0.0	0.0	0.6	0.0	0.0	0.8	0.24
t_3	twitter	0.4	0.0	0.4	0.3	0.1	0.4	0.0	0.0	0.4	0.20
t_4	socialweb	0.8	0.2	0.0	0.5	0.0	0.4	0.3	0.0	0.8	0.28
t_5	blogging	0.8	1.0	0.5	0.0	0.0	0.0	0.6	0.0	1.0	0.36
t_6	art	0.0	0.7	0.5	0.6	0.0	0.0	0.0	0.0	0.7	0.23
t_7	museum	0.2	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.5	0.15
t_8	warhol	0.0	0.6	0.7	0.4	0.0	0.0	0.5	0.6	0.7	0.35
t_9	picasso	0.5	0.6	0.6	0.0	0.0	0.1	0.0	0.0	0.6	0.23
t_{10}	history	0.0	0.7	0.6	0.0	0.0	0.8	0.0	0.5	0.8	0.33
t_{11}	architecture	0.5	0.9	0.6	0.0	0.7	0.0	0.6	0.5	0.9	0.48
t_{12}	java	0.0	0.7	0.3	0.7	0.5	0.5	0.0	0.7	0.7	0.43
t_{13}	programming	0.0	0.0	0.5	0.5	0.5	0.4	0.5	0.5	0.5	0.36
t_{14}	python	0.0	0.0	0.0	0.0	0.4	0.0	0.6	1.0	1.0	0.25
t_{15}	mysql	0.0	0.0	0.0	0.5	0.0	0.2	1.0	1.0	1.0	0.34
t_{16}	database	0.0	0.0	0.0	0.4	0.0	0.8	0.0	0.9	0.9	0.26
t_{17}	ruby	0.0	0.2	0.2	0.0	0.0	0.3	0.0	0.3	0.3	0.13
t_{18}	ajax	0.8	0.6	1.0	0.5	0.5	0.7	0.7	0.6	1.0	0.68
t_{19}	ror	0.0	0.1	0.0	0.0	0.0	0.8	0.9	0.8	0.9	0.33
t_{20}	opensource	0.4	0.5	0.0	0.5	0.2	0.1	0.5	0.2	0.5	0.3

For example, $fitn_{2,4}(r_4, blogs) = 0.0$ and $fitn_{12,5}(r_5, java) = 0.5$. The information concerning users is not displayed since it is not necessary. In this example, we use the following components:

- (1) *Representations*, which includes all the tags with a minimum fitness value of 0.6 choosing the MAX method.
- (2) *Convergence* (CVG), with an average value of 0.01 associated to tags of S_{rt} .
- (3) The *classif_sim* mode with a $th_{sim} = 0.7$ as *Classifier*.
- (4) *Clustering*, based in k-means with $k = 3$.
- (5) *MergingSplitting*, which merges two categories when their similarity is greater than 0.8.
- (6) *Naming*, which assigns the name of categories concatenating the two more representative tags in each category.
- (7) *RecalculationCondition*, which performs the re-calculus of S_{rt} and RC each time a resource moves from $R_{converged}$ to $R_{classified}$.

Vectors representing R , RC and S_{rt} are compared using the cosine similarity measure. Table 7 shows V and S_{CR} sets.

First of all, *S-SAMFULLsModel_creation* is applied to the social tagging system. *Representation* builds $S_{rt} = \{t_1, t_2, t_4, t_5, t_6, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{14}, t_{15}, t_{16}, t_{18}, t_{19}\}$. After the execution of the component *Convergence*, all the resources move to the set $R_{converged}$, without exception, since all of them reach the minimum average fitness (0.01). After applying the k-means algorithm, three categories are obtained. The resources classified under those categories are $c_1 = \{r_1\}$, $c_2 = \{r_2, r_3\}$ and $c_3 = \{r_4, r_5, r_6, r_7, r_8\}$.

The component *MergingSplitting* does not update the sets RC and Z because the values of the similarity measures among categories are lower than 0.8. As the values of the similarity measures between r_5 and all the categories are lower than 0.7 (th_{sim}), the resource r_7 does not belong to $R_{classified}$. Then,

$$R_{pending} = \{\emptyset\}, R_{converged} = \{r_5\} \text{ and } R_{classified} = \{r_1, r_2, r_3, r_4, r_6, r_7, r_8\}.$$

In this example, the categories are renamed as $c_1 = \text{"blog \& socialweb \& blogging \& ajax"}$, $c_2 = \text{"blogs \& blogging \& architecture \& ajax"}$ and $c_3 = \text{"mysql \& java"}$. Finally, the component *Classifier* performs the calculus of the similarity measures between categories and resources.

Table 7. Example of V and S_{CR} sets for *S-SAM FULL*.

Vectors	
V_{r_1}	(0.9, 0.5, 0.8, 0.8, 0.0, 0.0, 0.5, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.8, 0.0)
V_{r_2}	(0.4, 0.8, 0.2, 1.0, 0.7, 0.6, 0.6, 0.7, 0.9, 0.7, 0.0, 0.0, 0.0, 0.6, 0.1)
V_{r_3}	(0.0, 0.0, 0.0, 0.5, 0.5, 0.7, 0.6, 0.6, 0.6, 0.3, 0.0, 0.0, 0.0, 1.0, 0.0)
V_{r_4}	(0.2, 0.0, 0.5, 0.0, 0.6, 0.4, 0.0, 0.0, 0.0, 0.7, 0.0, 0.5, 0.4, 0.5, 0.0)
V_{r_5}	(0.7, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.7, 0.5, 0.4, 0.0, 0.0, 0.5, 0.0)
V_{r_6}	(0.5, 0.6, 0.4, 0.0, 0.0, 0.0, 0.1, 0.8, 0.0, 0.5, 0.0, 0.2, 0.8, 0.7, 0.8)
V_{r_7}	(0.7, 0.0, 0.3, 0.6, 0.0, 0.5, 0.0, 0.0, 0.6, 0.0, 0.6, 1.0, 0.0, 0.7, 0.9)
V_{r_8}	(0.0, 0.0, 0.0, 0.0, 0.0, 0.6, 0.0, 0.5, 0.5, 0.7, 1.0, 1.0, 0.9, 0.6, 0.8)

S_{CR}	c_1	c_2	c_3
r_1	1.0000	0.7227	0.5248
r_2	0.6928	0.9844	0.6022
r_3	0.5428	0.8774	0.5752
r_4	0.3752	0.5265	0.7364
r_5	0.5777	0.5403	0.6824
r_6	0.4827	0.5582	0.7865
r_7	0.5774	0.5029	0.8388
r_8	0.1723	0.4103	0.8966

Vectors	
V_{d_1}	(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_2}	(0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_3}	(0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_4}	(0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_5}	(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_6}	(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_7}	(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
V_{d_8}	(0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)
V_{d_9}	(0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
$V_{d_{10}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)
$V_{d_{11}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
$V_{d_{12}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0)
$V_{d_{13}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)
$V_{d_{14}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
$V_{d_{15}}$	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)
V_{c_1}	(0.9, 0.5, 0.8, 0.8, 0.0, 0.0, 0.5, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.8, 0.0)
V_{c_2}	(0.4, 0.8, 0.2, 0.75, 0.6, 0.65, 0.6, 0.65, 0.75, 0.5, 0.0, 0.0, 0.0, 0.8, 0.05)
V_{c_3}	(0.35, 0.15, 0.3, 0.15, 0.15, 0.38, 0.25, 0.33, 0.28, 0.48, 0.4, 0.68, 0.53, 0.63, 0.63)

3.4. SAM, S-SAM and S-SAM FULL for R47

Once explained how SAM, S-SAM and S-SAM FULL work, it's necessary to go deeper in the way they get data and they output the results. This subsection, and the next one, is going to be about what is needed to run the programs, what are the meaning of the input parameters and what the outputs are.

But before this, it's necessary to point out the origin of data that is going to manage. This project is focused in the treatment of two sources of information. The first one we will called "delicious" due to they came from a dump of a database of Del.icio.us. This

data correspond to a table with a big amount of annotations, that is, when a tag is labelled to a certain resource. And the second one it's a subset of these annotations, which will be called "R47" because they correspond to 47 resources randomly chosen.

The corresponding programs for R47 have been named as "Dictionary SAM", "Dictionary S-SAM" and "Dictionary S-SAM FULL" respectively in order to differentiate from SAM and S-SAM for Delicious.

3.4.1. Databases

Both Dictionary SAM, Dictionary S-SAM and Dictionary S-SAM FULL take data from a database called "dictionary". In this database there is just one table, "annotation", with the following attributes and their meanings:

- resourceID: the unique identifier of the resource. It's primary key of the table.
- resourceName: the name of the resource.
- tagID: the unique identifier of the tag. It's primary key of the table.
- tagName: the name of the tag.
- count: the number of annotations have that tag in that resource.
- fitness: the consensus expert evaluation of that tag in that resource.

Then programs do their calculus and they are poured in two different databases. Due to there are 3 program, they are called "d-sam" and "d-samtest" for Dictionary SAM, "d-ssam" and "d-ssamtest" for Dictionary S-SAM and "d-ssamfull" and "d-ssamfulltest" for Dictionary S-SAM FULL. This is because they use different criteria to select representative tags, which are different types of data, and hence, different way to store it. Dictionary SAM uses "count" attribute to decide whether or not a tag is representative, while Dictionary S-SAM and Dictionary S-SAM FULL use "fitness" instead.

The first of these databases have 11 tables:

- concept: store the classification concepts (ID and name).
- resource: store the resources (ID and state).

- scc: store the similarity concept–concept.
- scd: store the similarity concept–dictionaryTag.
- scr: store the similarity concept–resource.
- srr: store the similarity resource –resource.
- tag: store the tags from folksonomy (ID, name and if it’s dictionary tag or not).
- vc: store the vectorial representation of concepts (concept ID, tag ID and value).
- vd: store the vectorial representation of dictionary tags (tag1 ID, tag2 ID and value).
- vr: store the vectorial representation of resources (resource ID, tag ID and value).
- z: store the classification of resources in concepts (resource ID and concept ID).

And the second one have only 2 tables:

- test: where input parameters are stored.
- time: store the timings of each phase.

The Dictionary S–SAM and Dictionary S–SAM FULL have the same database with the same tables than Dictionary SAM does, but some data types are different (fitness is double while count is integer).

3.4.2. Configuration file

There is a unique configuration file where all database information is gathered. In this file is written down all the database names and the user and password needed to access them. It exists because in several java files from the program is necessary to specify the databases names.

It’s very important the order they are distributed. For example, the first line refers to URL dictionary database, the second and third line refers to the database where calculus are going to be store (“d–sam” and “d–samtest” for Dictionary SAM, “d–ssam” and “d–ssamtest” for Dictionary S–SAM and “d–ssamfull” and “d–ssamfulltest” for Dictionary

S-SAM FULL) and the fourth and fifth line refers to the database name and password respectively (in order to access phpMyAdmin databases).

So if these lines are switched in some way, program will not work properly. Therefore it's very important to maintain the line order, unless you want to modify the program code and reorder the line as desired.

3.4.3. Input parameters

In Dictionary SAM there are 8 input parameters to make program run:

- dicMinAnnotation: is the minimum annotation value to decide if a tag belongs to dictionary or not.
- convergMinAnnotations: is the minimum annotation value that a resource has to have to converge.
- classifThreshold: a minimum value to correctly classify resources. Interval value between 0 and 1.
- mergingThreshold: a minimum similarity value between 2 concepts to merge them. Interval value between 0 and 1.
- namingThreshold: a minimum value to classify name tags as representative to rename concepts with respect the highest one. Interval value between 0 and 1.
- simMinValue: a minimum value to store similarities of S set. Interval value between 0 and 1.
- simSRRThres: a minimum value of two concepts to compare its resources. Interval value between 0 and 1.
- classifierT: type of classifier. Possible values: SIM and DELTA.

In Dictionary S-SAM there are 9 input parameters to make program run:

- dicMinFitness: is the minimum fitness value to decide if a tag belongs to dictionary or not. Interval value between 0 and 1.
- methodDictionary: the method of create dictionary. Possible values: MAX and MEAN.

- convergMinAnnotations: is the minimum annotation value that a resource has to have to converge.
- classifyThreshold: a minimum value to correctly classify resources. Interval value between 0 and 1.
- mergingThreshold: a minimum similarity value between 2 concepts to merge them. Interval value between 0 and 1.
- namingThreshold: a minimum value to classify name tags as representative to rename concepts with respect the highest one. Interval value between 0 and 1.
- simMinValue: a minimum value to store similarities of S set. Interval value between 0 and 1.
- simSRRThres: a minimum value of two concepts to compare its resources. Interval value between 0 and 1.
- classifierT: type of classifier. Possible values: SIM and DELTA.

And finally in Dictionary S-SAM FULL there are also 9 input parameters:

- dicMinFitness: is the minimum fitness value to decide if a tag belongs to dictionary or not. Interval value between 0 and 1.
- methodDictionary: the method of create dictionary. Possible values: MAX and MEAN.
- convergMinFitness: is the minimum average value that a resource has to have to converge.
- classifyThreshold: a minimum value to correctly classify resources. Interval value between 0 and 1.
- mergingThreshold: a minimum similarity value between 2 concepts to merge them. Interval value between 0 and 1.
- namingThreshold: a minimum value to classify name tags as representative to rename concepts with respect the highest one. Interval value between 0 and 1.
- simMinValue: a minimum value to store similarities of S set. Interval value between 0 and 1.
- simSRRThres: a minimum value of two concepts to compare its resources. Interval value between 0 and 1.
- classifierT: type of classifier. Possible values: SIM and DELTA.

3.4.4. Output results

Once finished the programs, the databases *d-sam* and *d-samtest* (or their respective databases for Dictionary S-SAM and Dictionary S-SAM FULL) are filled with data and also a file with the statistics of the execution are made. This Excel file gathers all the information about the performance of algorithm, the entry parameters and the output results.

The resultant file consists of several tabs:

- Parameters: contains the entry parameters of the execution.
- Folksonomy: summarizes the number of annotations, resources, tags and dictionary tags.
- Resources: displays how classification have been carried out and shows how many resources have been classified, converged and pending.
- Concepts: shows the concepts created and the number of resources and tags contains each one.
- Classification: display in detail what resources have been classified under which concepts.
- Timings: contains the timings of the execution.

3.5. SAM y S-SAM for Del.icio.us

As done with Dictionary SAM, S-SAM and S-SAM FULL, the corresponding programs for the whole database annotations (delicious) have been simply named as “SAM” and “S-SAM” respectively. Make sense that S-SAM FULL have not been considered because of the lack of fitness value in the rest of database. Despite of this, the corresponding program is done but not taken into account.

3.5.1. Databases

The same as Dictionary triad, SAM and S-SAM also take data from the database “dictionary”, with which most representative tags are selected.

After this, programs do their calculus and they are poured in two different databases too, called “sam” and “samtest” for SAM and “ssam” and “ssamtest” for S-SAM. This is because they use different criteria to select representative tags, which are different types of data, and hence, different way to store it. SAM uses “count” attribute to decide whether or not a tag is representative, while S-SAM uses “fitness” instead.

The first of these databases have 11 tables:

- concept: store the classification concepts (ID and name).
- resource: store the resources (ID and state).
- scc: store the similarity concept-concept.
- scd: store the similarity concept-dictionaryTag.
- scr: store the similarity concept-resource.
- srr: store the similarity resource -resource.
- tag: store the tags from folksonomy (ID, name and if it's dictionary tag or not).
- vc: store the vectorial representation of concepts (concept ID, tag ID and value).
- vd: store the vectorial representation of dictionary tags (tag1 ID, tag2 ID and value).
- vr: store the vectorial representation of resources (resource ID, tag ID and value).
- z: store the classification of resources in concepts (resource ID and concept ID).

And the second one have only 2 tables:

- test: where input parameters are stored.
- time: store the timings of each phase.

3.5.2. Configuration file

There is a unique configuration file where all database information is gathered. In this file is written down all the database names and the user and password needed to access them. It exists, as Dictionary triad does, because in several java files from the program is necessary to specify the databases names.

It's also very important the order they are distributed. The first line refers to URL dictionary database, the second one refers to delicious database, the third and fourth line refers to the database where calculus are going to be store ("sam" and "samtest" for SAM and "ssam" and "ssamtest" for S-SAM) and the fifth and sixth line refers to the database name and password respectively (in order to access phpMyAdmin databases).

3.5.3. Input parameters

In SAM there are 8 input parameters to make program run:

- dicMinAnnotation: is the minimum annotation value to decide if a tag belongs to dictionary or not.
- convergMinAnnotations: is the minimum annotation value that a resource has to have to converge.
- classifThreshold: a minimum value to correctly classify resources. Interval value between 0 and 1.
- mergingThreshold: a minimum similarity value between 2 concepts to merge them. Interval value between 0 and 1.
- namingThreshold: a minimum value to classify name tags as representative to rename concepts with respect the highest one. Interval value between 0 and 1.
- simMinValue: a minimum value to store similarities of S set. Interval value between 0 and 1.
- simSRRThres: a minimum value of two concepts to compare its resources. Interval value between 0 and 1.
- classifierT: type of classifier. Possible values: SIM and DELTA.

And in S-SAM there are 9 input parameters to make program run:

- dicMinFitness: is the minimum fitness value to decide if a tag belongs to dictionary or not. Interval value between 0 and 1.
- methodDictionary: the method of create dictionary. Possible values: MAX and MEAN.
- convergMinAnnotations: is the minimum annotation value that a resource has to have to converge.
- classifThreshold: a minimum value to correctly classify resources. Interval value between 0 and 1.
- mergingThreshold: a minimum similarity value between 2 concepts to merge them. Interval value between 0 and 1.
- namingThreshold: a minimum value to classify name tags as representative to rename concepts with respect the highest one. Interval value between 0 and 1.
- simMinValue: a minimum value to store similarities of S set. Interval value between 0 and 1.
- simSRRThres: a minimum value of two concepts to compare its resources. Interval value between 0 and 1.
- classifierT: type of classifier. Possible values: SIM and DELTA.

3.5.4. Output results

As with Dictionary SAM, S-SAM and S-SAM FULL, at the end of the programs execution, the databases *sam* and *samtest* (and *ssam* and *ssamtest*) are filled with data and also a file with the statistics of the execution are made. This Excel file gathers all the information about the performance of algorithm, the entry parameters and the output results.

The resultant file contains the same tabs that Dictionary triad does:

- Parameters: contains the entry parameters of the execution.
- Folksonomy: summarizes the number of annotations, resources, tags and dictionary tags.

- Resources: displays how classification have been carried out and shows how many resources have been classified, converged and pending.
- Concepts: shows the concepts created and the number of resources and tags contains each one.
- Classification: display in detail what resources have been classified under which concepts.
- Timings: contains the timings of the execution.

***NOTE:** In Dictionary SAM, S-SAM and S-SAM FULL, similarities are calculated, but in delicious version are not. This is because the huge amount of data of delicious which implies a super high computational and memory cost for a current computer. Apart from this project is just focused on how classification is carried out. But the implementation is done and it works fine, but it's commented. If it's desired to carry out the similarity phase, you just have to remove the forward slashes of the code.*

3.6. Other related support programs

Despite the project have been implemented with the programs from above, it have been needed some other programs to carry out this tasks. The list below contains all of them:

- TRANSFORM: allows to convert the original format of delicious database in a more suitable form, that is, join 3 tables in just one.
- DCREATOR: this program convert the dictionary data from excel file to database.
- RENAMETAGIDS: as name points out, this program rename the tags from the original excel files from where dictionary database have been made in such a way they match with those from delicious database. S-SAM FULL.
- INCLUDEDICTIONARY: its function is to join the dictionary annotations with the delicious annotations.

- S-SAM FULL: it's the S-SAM FULL version for delicious.
- ACOAR ADRIANC/FlickrBackup: this program download images from flicker using the input tags and store all the information in a database called "medicalimagesdb".
- ACOAR ADRIANC/ACoAR: classify the resources obtained by FlickrBackup using the ACoAR method (SAM).

Chapter IV:

IMPLEMENTATION

■ ■ ■

So as to carry out the implementation of S-SAM, it's been needed to install and reconfigure some software programs so that desired results can be obtained. Some of this used tools are listened below. In closing this chapter, there is a short guide of installation and configuration to try both SAM and S-SAM at your home with some little steps.

4.1. Software tools

In the following list it will be shown all the programs that have been needed to develop this research. Note that all the tools used in this project are for Windows (concretely Windows 8) and no other operative system. This doesn't mean that all the programs

created in this final degree project can be run in others OS, but it probably will be need other software tools. Moreover, only one computer will be necessary to execute the programs.

4.1.1. Java

In order to implement SAM and S-SAM it was necessary to find a programming language which had high-level performance and specification, could handle lot of memory resources and threads pool and preferably have a huge amount of documentation available.

So that's why Java was chose. This language fulfils all the requirements needed. In addition, Java is cross-platform, that is, any compiled Java program runs on all platforms for which there exists a JVM (Java Virtual Machine) and this holds for all major operating systems, including Windows, Mac OS and Linux.

Furthermore, it's object-oriented, so it's possible to define abstract data types in a clear modular structure and easy to maintain and modify existing code. It's the perfect programming language for this project.

4.1.2. NetBeans

NetBeans⁴ is a software development platform written in Java that allows applications to be developed from a set of modular software components, called modules. It is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++, and HTML5. NetBeans is cross-platform and runs on Microsoft Windows, MAC OS X, Linux, Solaris and other platforms supporting a compatible JVM.

⁴ <https://netbeans.org/>

This tool was chosen because of its versatility, flexibility, high-level performance and supports Java, which is one of the best programming languages nowadays regarding memory management.

4.1.3. WampServer

WampServer⁵ is a software stack for Microsoft Windows operative system consisting of the Apache web server, OpenSSL for SSL support, MySQL database and PHP programming language. It's a very complete application server platform, which it can be performed a lot of different tasks, like web designing, database managing, etc.

I decided to use this tool instead of other ones because I have already installed in my computer and I use it for a long time.

***NOTE:** If you are not up to use WampServer to manage databases for this project because it contains a lot of unuseful stuff, you can use another tools that use MySQL technology, such as phpMyAdmin⁶ (which is already included in WampServer), MySQL Workbench⁷, Webmin MySQL module⁸ or Emma⁹, among others.*

4.1.4. Couchbase

Couchbase¹⁰, originally known as Membase, is an open-source, distributed (shared-nothing structure) NoSQL document-oriented database that is optimized for interactive applications. It is designed to provide easy-to-scale key-value or document

⁵ <http://www.wampserver.com/>

⁶ http://www.phpmyadmin.net/home_page/index.php

⁷ <http://www.mysql.com/products/workbench/>

⁸ <http://www.webmin.com/standard.html>

⁹ <http://freecode.com/projects/emma>

¹⁰ <http://www.couchbase.com/>

access with low latency and high sustained throughput. It is designed to be clustered from a single machine to very large-scale deployments spanning many machines.

Besides, Couchbase Server provides on-the-wire client protocol compatibility but is designed to add disk persistence, data replication, live cluster reconfiguration, rebalancing and multitenancy with data partitioning.

There were multiple reasons to add this tool to the implementation of SAM and S-SAM. First of all, the application generated so much information that overflowed the memory allocated to Java process and it was needed a cache server. And hence Couchbase were chose because it's well prepared to run in Windows.

4.2. Installation and configuration guide

***IMPORTANT:** Before starting this guide, be advised that it's been done for Windows 8 and is valid for Windows 7 too.*

4.2.1. Installation of WampServer

In order to run SAM and S-SAM, the computer will have to have previously installed Microsoft Visual C++ 2012. This program can be downloaded in the link below: <http://www.microsoft.com/en-us/download/details.aspx?id=30679>

Then, go to the official page of the product WampServer and download the version x64 and x86 according to the computer architecture in which is going to be installed. Regarding the configuration parameters, in this case it will be set "Chrome" as default navigator (route C:\Program Files(x86)\Google\Chrome\Application\chrome.exe) and "localhost" as SMTP.

4.2.2. Installation of NetBeans

Before installing this tool, it will be needed to have Java SE Development Kit 8¹¹ in our computer, which can be directly downloaded from the Oracle official page.

For NetBeans installation, go to the product webpage and to choose the version that includes all the available functionalities.

4.2.3. Importing databases

Once we have all programs installed, it will be needed to import all the databases in order to store the information which SAM and S-SAM generates. To do this, go to phpMyAdmin in localhost (or the SMTP name you put it before in the WampServer installation) typing in your browser <http://localhost/phpmyadmin/> and import the following databases:

- dictionary.sql: contains the evaluated annotations.
- delicious.sql: contains a dump of delicious database.
- d-sam: store the calculus of Dictionary SAM.
- d-samtest: store the input parameters and the timings of execution of Dictionary SAM.
- d-ssam: store the calculus of Dictionary S-SAM.
- d-ssamtest: store the input parameters and the timings of execution of Dictionary S-SAM.
- d-ssamfull: store the calculus of Dictionary S-SAM FULL.
- d-ssamfulltest: store the input parameters and the timings of execution of Dictionary S-SAM FULL.
- sam: store the calculus of SAM.
- samtest: store the input parameters and the timings of execution of SAM.
- ssam: store the calculus of S-SAM.
- ssamtest: store the input parameters and the timings of execution of S-SAM.

¹¹ Java SE Development Kit 8u31, <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

For each one of the databases, click on the “Import” menu button, then click the “Select archive” and then “Go” button. Repeat the process with all the files from above.

Note that depending the weight of the file and the power of your computer, the process will take several minutes, such as with “delicious.sql”, which contains a dump of delicious with more than 521.000 annotations.

4.2.4. Importing NetBeans projects

In order to import the NetBeans projects, open the program and follow the route: File > Import Project and choose one of them. Repeat this with each project.

***NOTE:** In the case of “DCreator”, a message will pop up because is necessary to add some external libraries to be able to run it. To do this, in the LIBRARIES folder, it will be all the required ones and you only have to select one by one those which project demands us until conflicts are solved.*

4.2.5. Configuring phpMyAdmin

Since there are some heavy database files, it’s required to increase some phpMyAdmin parameters to upload and allow these archives. Open the *php.ini* file from WampServer (C:\wamp\bin\php\php5.5.12\php.ini) and modify these variables with the following values:

- max_execution_time = 120
- post_max_size = 30M
- upload_max_filesize = 600M
- memory_limit = 600M

4.2.6. Configuring NetBeans

These programs need from big amounts of computer resources, therefore it's necessary to supply more RAM memory to NetBeans. To do this, do a right-click on the project and select "Properties". Then, chose the "Run" tab and in the VM Options field, type the VM arguments:

- "-Xmx1500M" for Dictionary triad.
- "-Xmx15000M" for SAM and S-SAM.

This way you allocate the enough power to the projects and will can run them properly.

4.3. How to run SAM and S-SAM

Once followed all these steps, everything is ready to be run. Now you only have to select the project you want to run and click on "Run Project" button or pressing F6 in NetBeans.

The results obtained depend on the input parameters that have been chosen. In order to modify them, go to the "Main.java" file from whatever project and, among the first lines, you will find the parameters. Change them accordingly with the constraints previously mentioned.

Depend on which project you run, the execution will take more or less time. Dictionary triad programs take few minutes while SAM and S-SAM for delicious take several hours, even days (it may vary among computers).

Chapter V:

EXPERIMENTAL RESULTS



The purpose of his project is to compare the results of SAM and S-SAM and to see which one classify the best. In this chapter, the programs will be executed with certain input parameters and then to analyse the results and make some statistics. To do this, it will be done for the Dictionary programs and then for the SAM and S-SAM delicious version.

5.1. SAM, S-SAM and S-SAM FULL for R47

For this experimental results, it's going to work with a specific set called "R47". This set is, in fact, a subset of annotations of delicious database. This subset is called this way

because it corresponds to the annotations of 47 resources, in particular, 11122 annotations with 7711 tags.

***NOTE:** We are going to work first with this little subset in order to manually analyse the results and extract some conclusions, and then perform the same programs with the whole database (delicious).*

These annotations have been evaluated one by one in order to give them a grade between 0 and 1 which reflects the level of adequacy of the annotation, called fitness. These assessments have been done by the consensus of 10 experts which evaluate each tag of each resource with a mark between 0 to 1, being 0 the lower representative value and 1 the higher one. The fitness of each tag is, hence, the mean of the 10 grades, being each one belonging to each expert. So a total of 111220 evaluations have been carried out.

Therefore, each annotation has two values: the number of occurrences and the fitness value. With these two numbers, it have been carried out some experiments.

5.1.1. Dictionary SAM results

Dictionary SAM uses the number of occurrences to establish which tags are representative and to do the corresponding calculus.

The input parameters that have been used are the following ones:

- `dicMinAnnotation` = 5 / 10 / 15 / 20 / 25 / 30 / 35 / 40 / 50 / 100 / 150 / 350 / 500 / 750 / 850 / 1000 / 1500
- `convergMinAnnotations` = 1
- `classifThreshold` = 0.1
- `mergingThreshold` = 0.7
- `namingThreshold` = 0.5
- `simMinValue` = 0.5
- `simSRRThres` = 0.4
- `classifierT` = SIM / DELTA

The following graphic reflects the behaviour of tags from dictionary when modifying the *dicMinAnnotation* parameter, which represents the minimum number of annotations that a representative tag have to have.

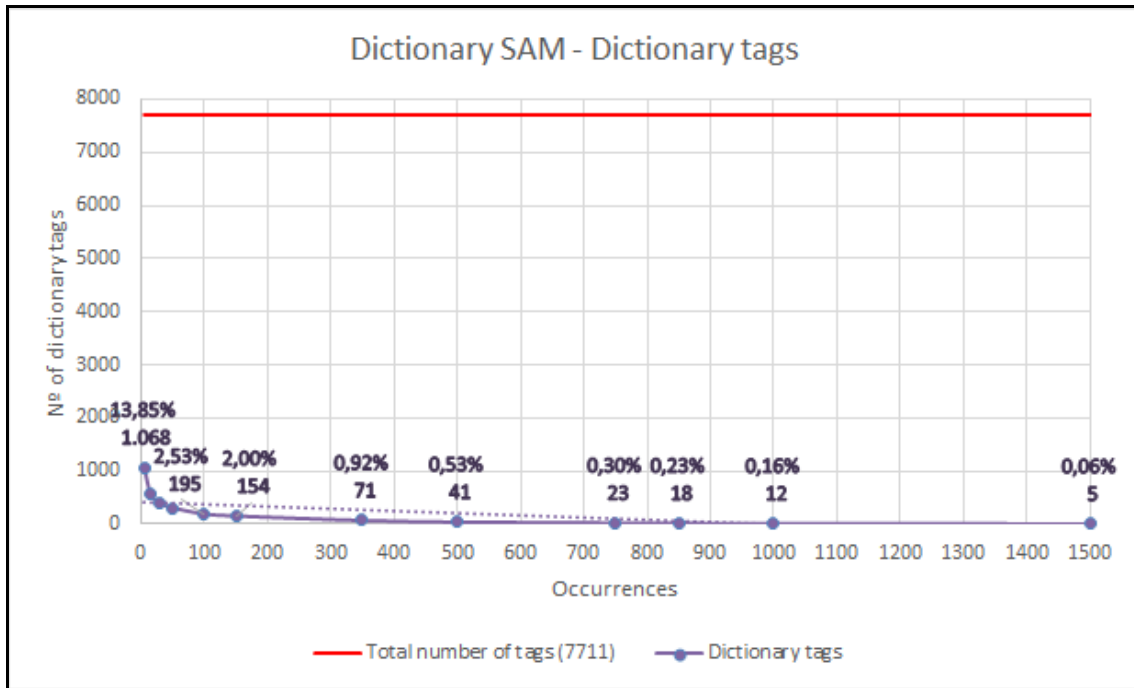


Figure 1. Dictionary SAM - Dictionary tags.

Let's zoom in to see in detail.

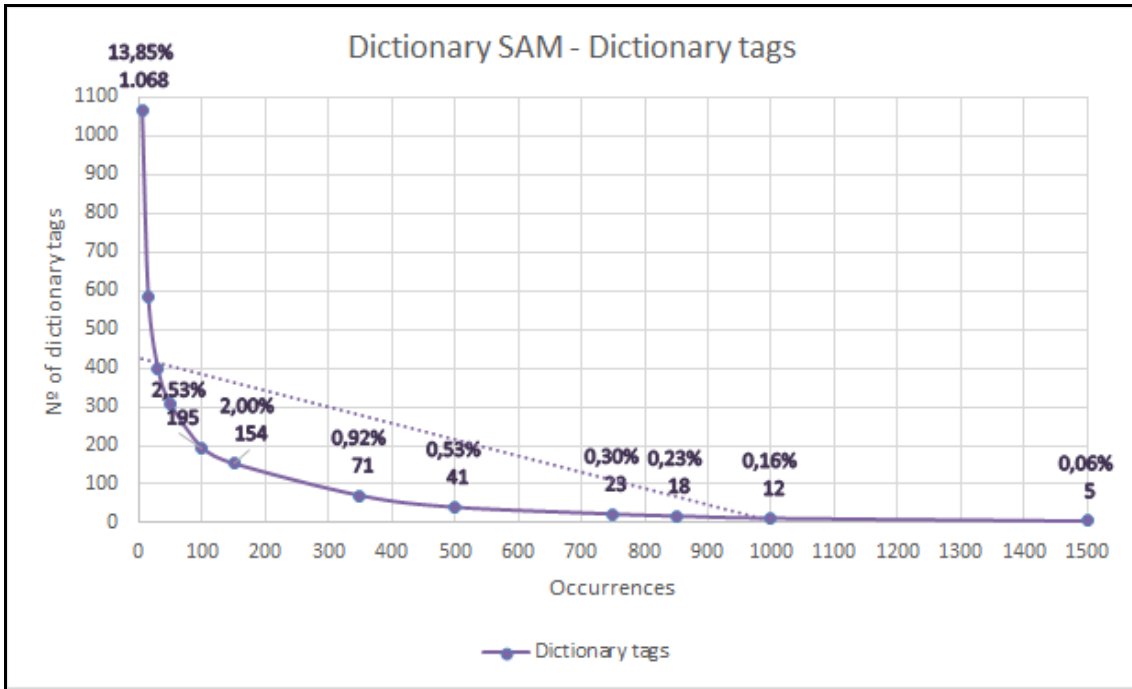


Figure 2. Dictionary SAM - Dictionary tags zoomed in.

Here, the lower is the number of occurrences of tags (less restrictive), the greater is the dictionary set. This is because it allows to enter more tags in the set as the restriction of the parameter is weaker.

The next graphics capture the behaviour of resources classification varying the *dicMinAnnotation* parameter and, also, the *classifierT*, which is the type of classifier that perform the similarities:

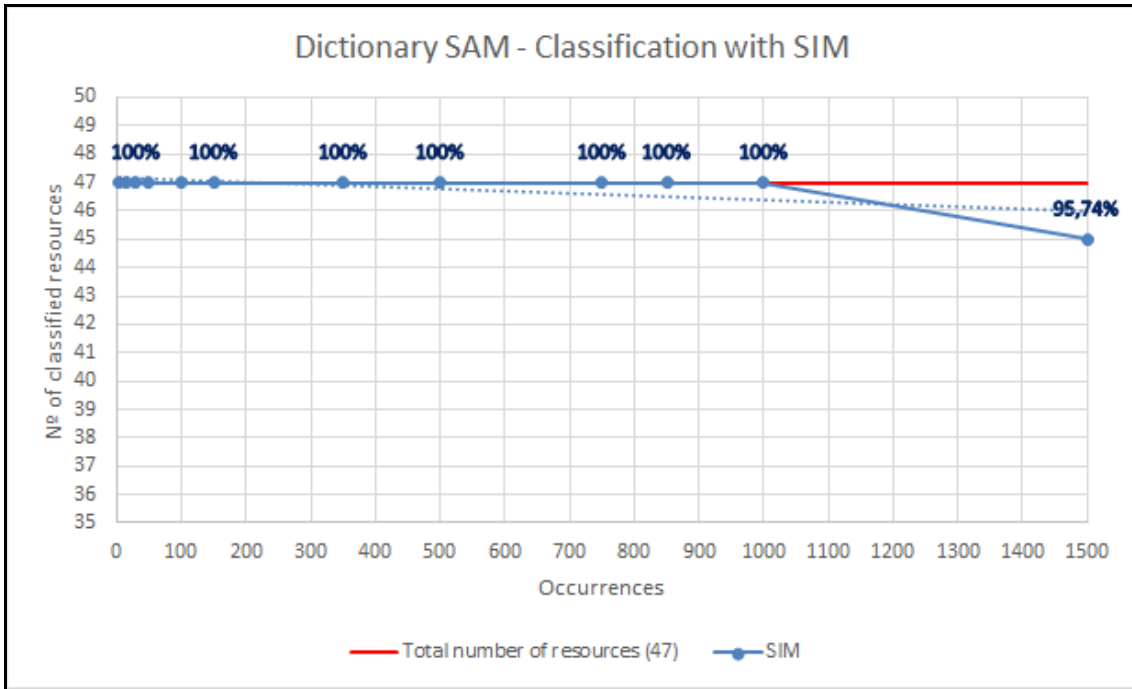


Figure 3. Dictionary SAM - Classification with SIM.

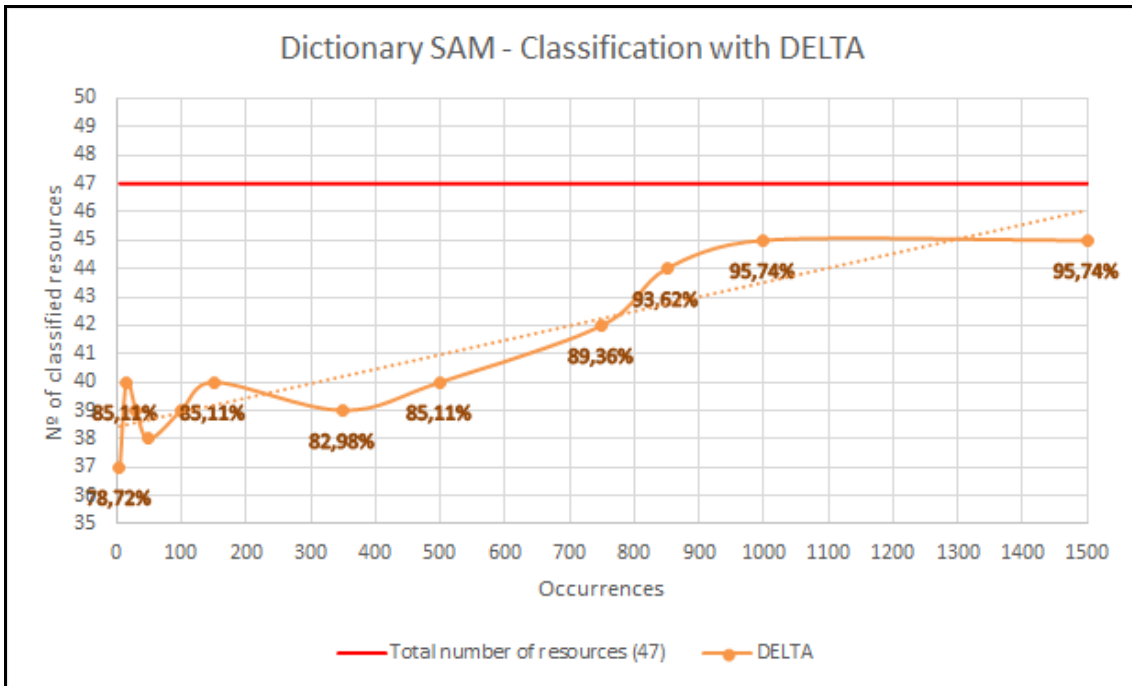


Figure 4. Dictionary SAM - Classification with DELTA.

In these charts is gathered how DSAM classifies resources depending on the classifier. We can see that in classifier SIM practical every resources is classified even if with a

dictionary set of 5 tags (Occurrences = 15000), while in DELTA classifier, classifies less resources. Here, we can see the comparison between them.

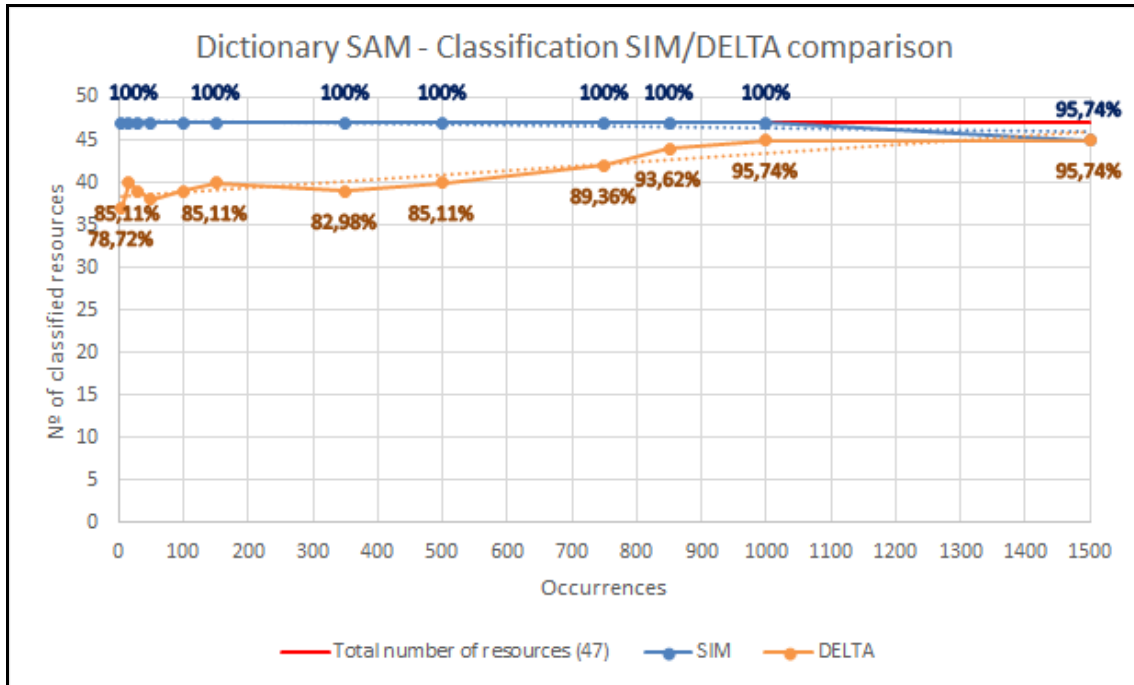


Figure 5. Dictionary SAM - Classification comparison.

It's clear that SIM classifier classifies more resources than DELTA, but, are they correctly classified? In order to know this, each resource of each classification concept of each classifier has been evaluated. The next graphics show those resources that are correctly classified under each classifier.

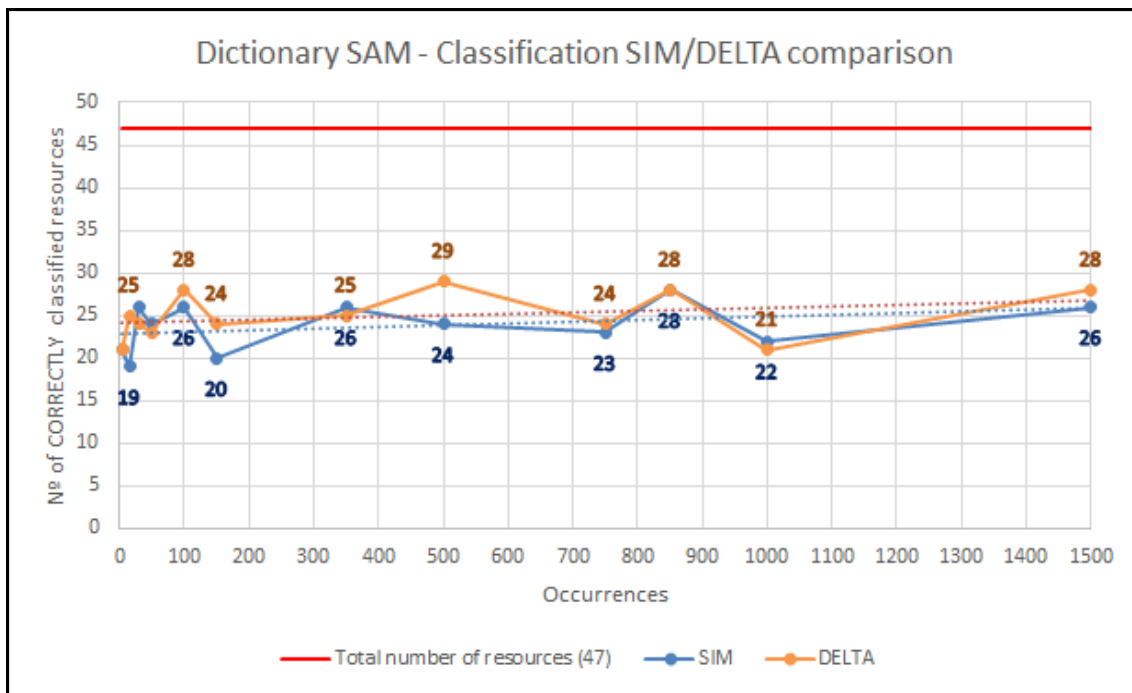


Figure 6. Dictionary SAM - Classification comparison (correct).

As it can be seen that both of them practically classify the same number of resources, DELTA slightly better, but we can say that they are equal. However, the classification groups under they form, and under which they classify the resources, have nothing to do with.

In SIM classifier these groups are very heterogeneous while in DELTA classifier, they are highly united and homogeneous. This makes that, in SIM, resources under the same group are very different and, in DELTA, they are very alike. For example, it have randomly chosen 2 different names of classification concepts from SIM and DELTA groups where it can be seen this:

SIM = “wii & blog & powerpoint & crafts & science & education & resources & collaboration”

DELTA = “design & web2.0 & webdesign & web”

This is a behaviour that constantly can be seen throughout this chapter when comparing these two classifiers.

5.1.2. Dictionary S-SAM results

Dictionary S-SAM uses the fitness value to establish which tags are representative and the number of occurrences to do the corresponding calculus.

The input parameters that have been used are the following ones:

- `dicMinFitness` = 0.1 / 0.2 / 0.3 / 0.4 / 0.5 / 0.6 / 0.7 / 0.8 / 0.9 / 1
- `methodDictionary` = MAX / MEAN
- `convergMinAnnotations` = 1
- `classifThreshold` = 0.1
- `mergingThreshold` = 0.7
- `namingThreshold` = 0.5
- `simMinValue` = 0.5
- `simSRRThres` = 0.4
- `classifierT` = SIM / DELTA

The following graphic reflects the behaviour of tags from dictionary when modifying the *dicMinAnnotation* parameter, which represents the minimum number of annotations that a representative tag have to have, and the *methodDictionary* parameter, which is the followed method to select the fitness value of the tag (MAX or MEAN).

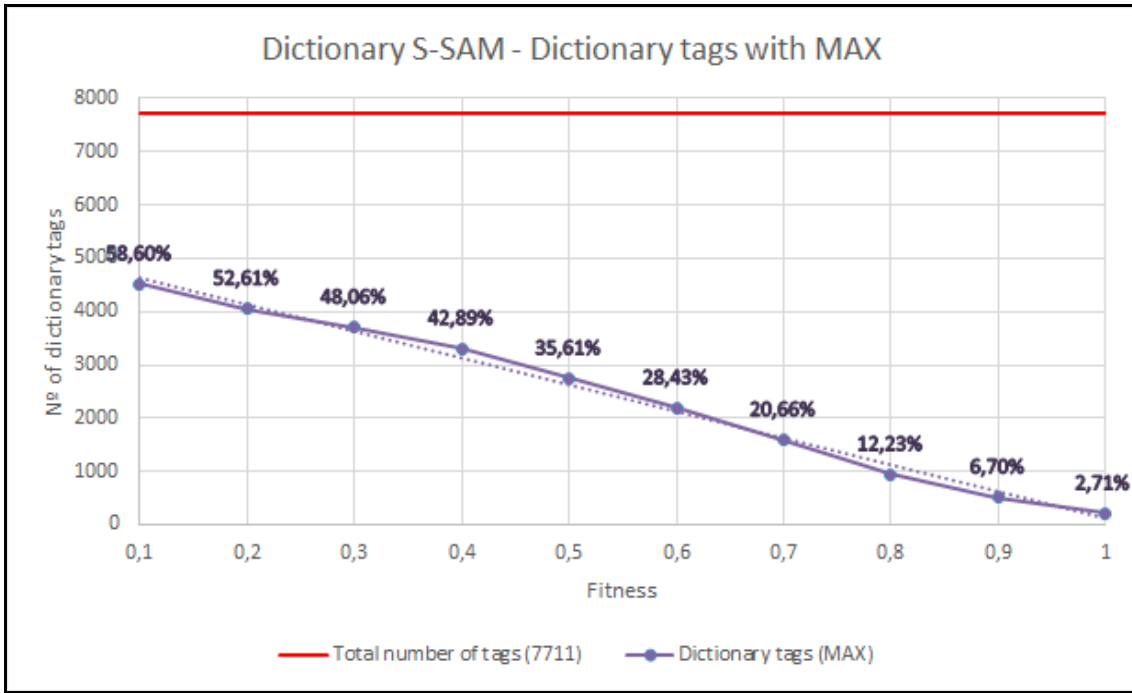


Figure 7. Dictionary S-SAM - Dictionary tags with MAX.

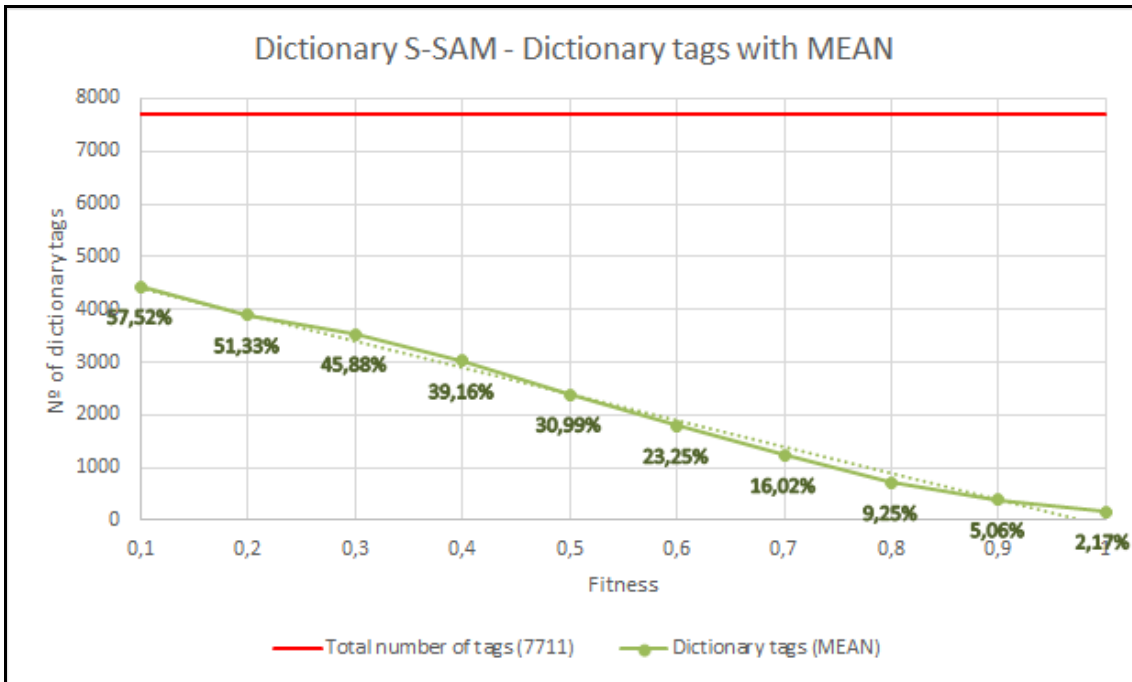


Figure 8. Dictionary S-SAM - Dictionary tags with MEAN.

The next chart represent the comparison between them.

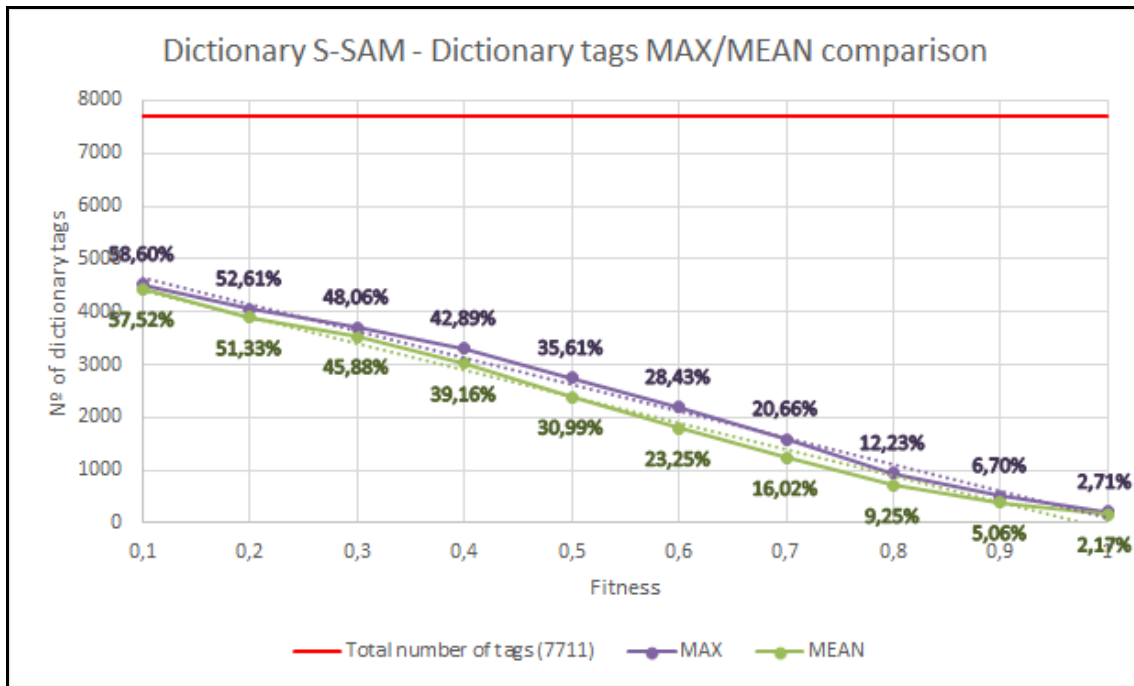


Figure 9. Dictionary S-SAM - Dictionary tags MAX/MEAN comparison.

As in Dictionary SAM, the lower is the value of the tag fitness (less restrictive), the greater is the dictionary set. This is because it allows to enter more tags in the set as the restriction of the parameter is weaker. Both are quite similar.

The next graphics capture the behaviour of resources classification varying the *dicMinFitness* and *methodDictionary* parameters, also, the *classifierT*, which is the type of classifier that perform the similarities:

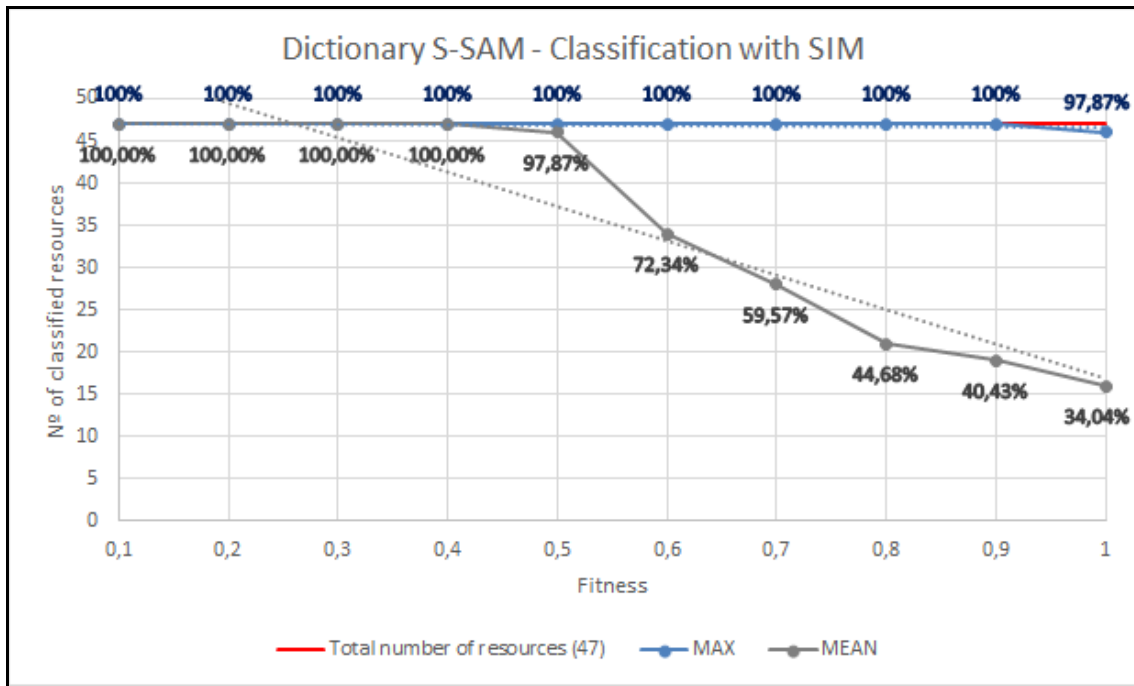


Figure 10. Dictionary S-SAM - Classification with SIM.

As we can see, using MEAN method classifies better in lower fitness values and, using MAX, it constantly classifies fine. This is because MEAN method uses an average of the tags from dictionary to do the classification. This implies that there are more meaningful tags in low fitness values than in the higher ones. And as MAX method chooses the best one, the classification is good in all cases.

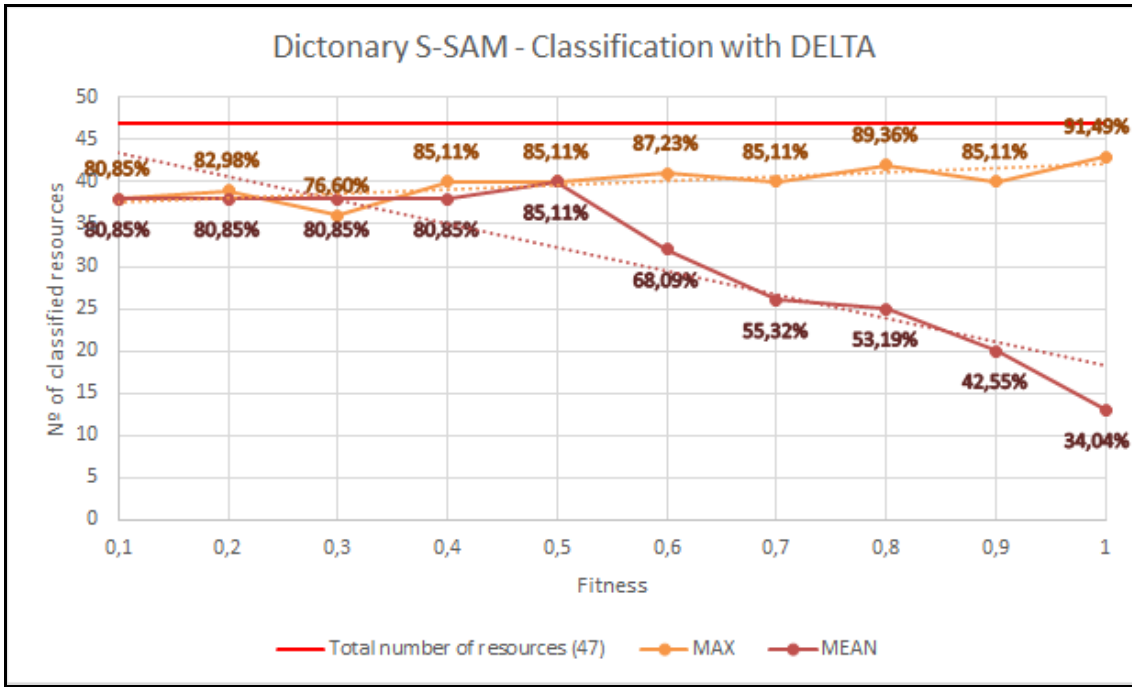


Figure 11. Dictionary S-SAM - Classification with DELTA.

With DELTA classifier it's experimented the same thing with the *methodDictionary* input parameter. MEAN method gives better results in lower fitness values and MAX method is always better.

In the next chart all these classifications are gathered in just one to have a clear viewpoint:

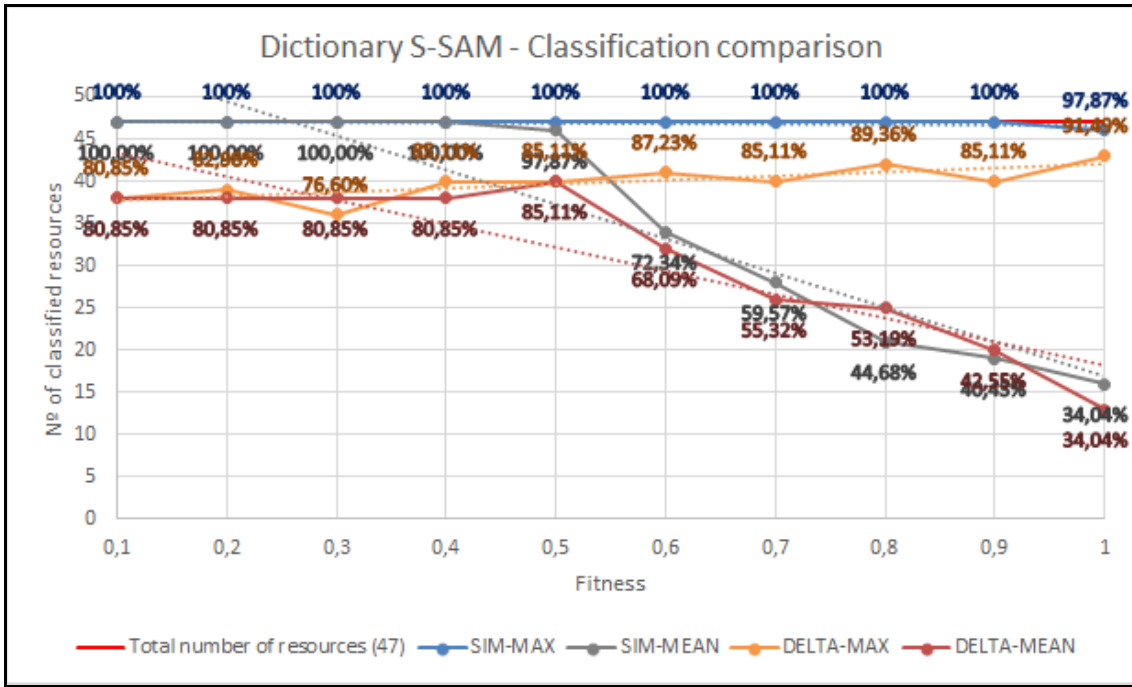


Figure 12. Dictionary S-SAM - Classification comparison.

It's obvious that SIM-MAX classifies the best but, are these resources correctly classified? In order to know this, each resource of each classification concept of each classifier have been evaluated. The next graphics shows those resources that are correctly classified under each classifier:

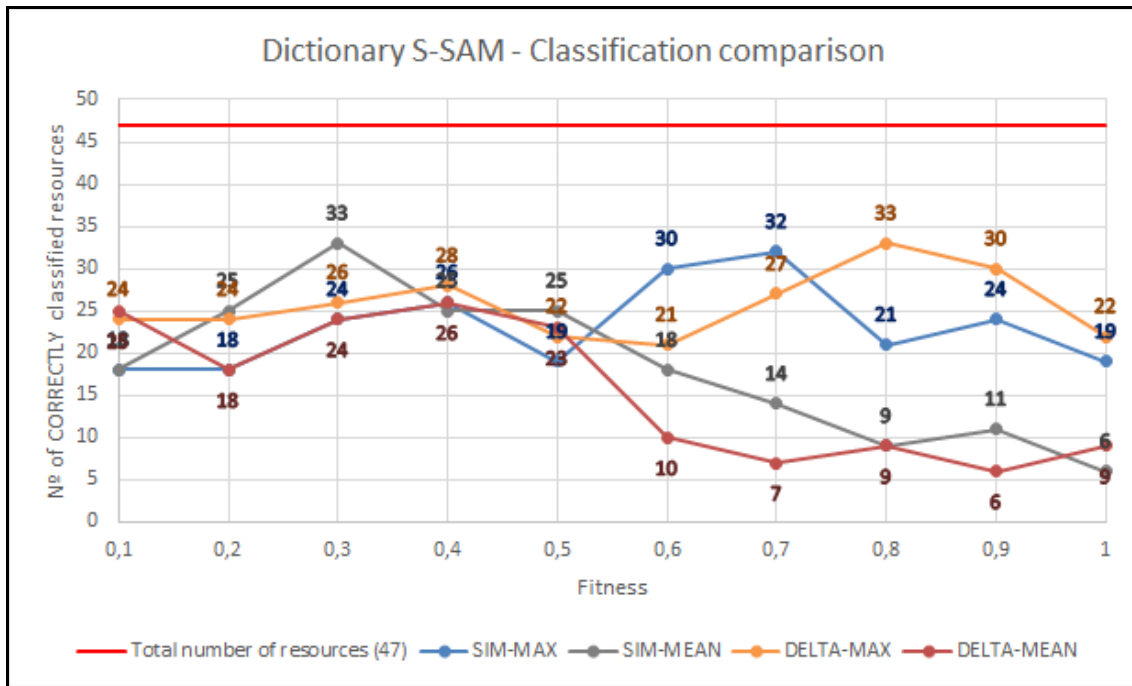


Figure 13. Dictionary S-SAM - Classification comparison (correct).

There are some curious things that happens here. The first one is that SIM-MAX is the method that more resources classify (practically everyone in every situation) but is not an accurate classification. The second one is that both SIM-MEAN and DELTA-MEAN classify fewer resources than the others. And the third one is that DELTA-MAX classify better when the value of the fitness is higher.

So which one is the best? First of all, we can directly dismiss DELTA-MEAN because it gets the worst results; the rest of the classification are quite better. With these three, we can also reject SIM-MEAN due to what it's looked for is a dictionary which contains few tags that represent the whole tags set. If the MEAN method is chosen, it means that we will get better results as fitness value is lower, and this is not desired. And SIM-MEAN belongs to this unwanted kind of methods. Therefore, it remains SIM and DELTA of MAX method. But as previously commented, SIM makes heterogeneous resources group up and lower connected among them.

So the conclusion that we can extract from this reflection is that the best one is by far DELTA-MAX, which classifies the best when the fitness value is high and build up good resources groups, concretely when the value of *dicMinFitness* is 0.8.

5.1.3. Dictionary S-SAM FULL results

Dictionary S-SAM FULL uses the fitness value to establish which tags are representative and to do the corresponding calculus.

The input parameters that have been used are the following ones:

- `dicMinFitness` = 0.1 / 0.2 / 0.3 / 0.4 / 0.5 / 0.6 / 0.7 / 0.8 / 0.9 / 1
- `methodDictionary` = MAX / MEAN
- `convergMinFitness` = 0.01
- `classifThreshold` = 0.1
- `mergingThreshold` = 0.7
- `namingThreshold` = 0.5
- `simMinValue` = 0.5
- `simSRRThres` = 0.4
- `classifierT` = SIM / DELTA

The following graphic reflects the behaviour of tags from dictionary when modifying the *dicMinAnnotation* parameter, which represents the minimum number of annotations that a representative tag have to have, and the *methodDictionary* parameter, which is the followed method to select the fitness value of the tag (MAX or MEAN).

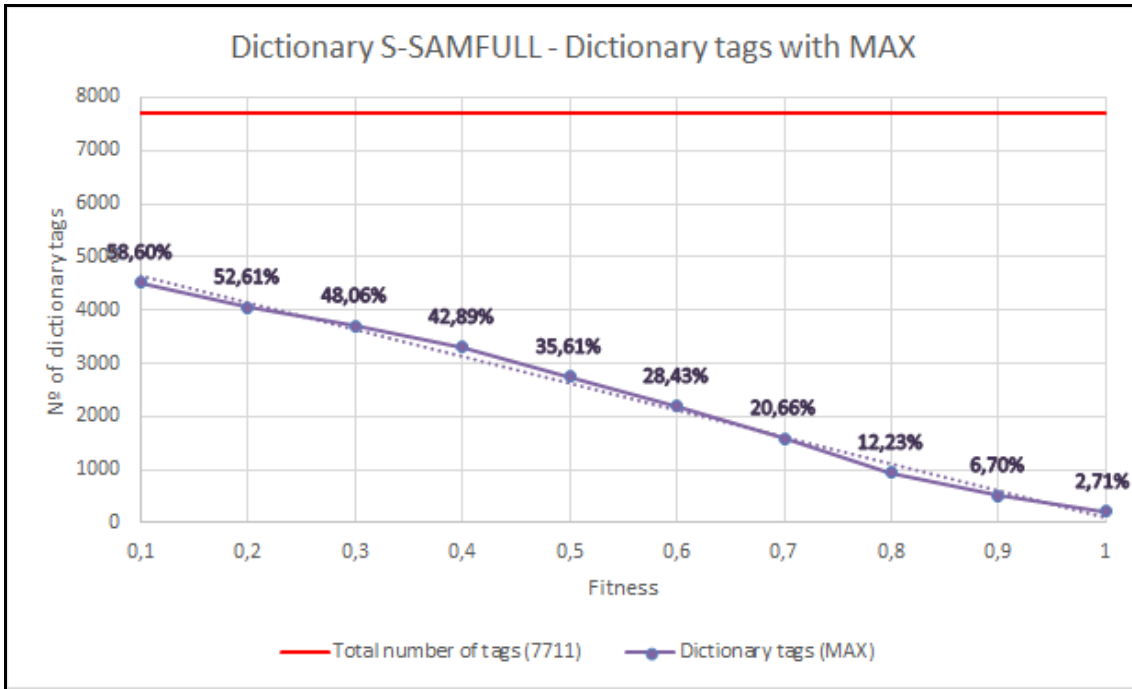


Figure 14. Dictionary S-SAM FULL - Dictionary tags with MAX.

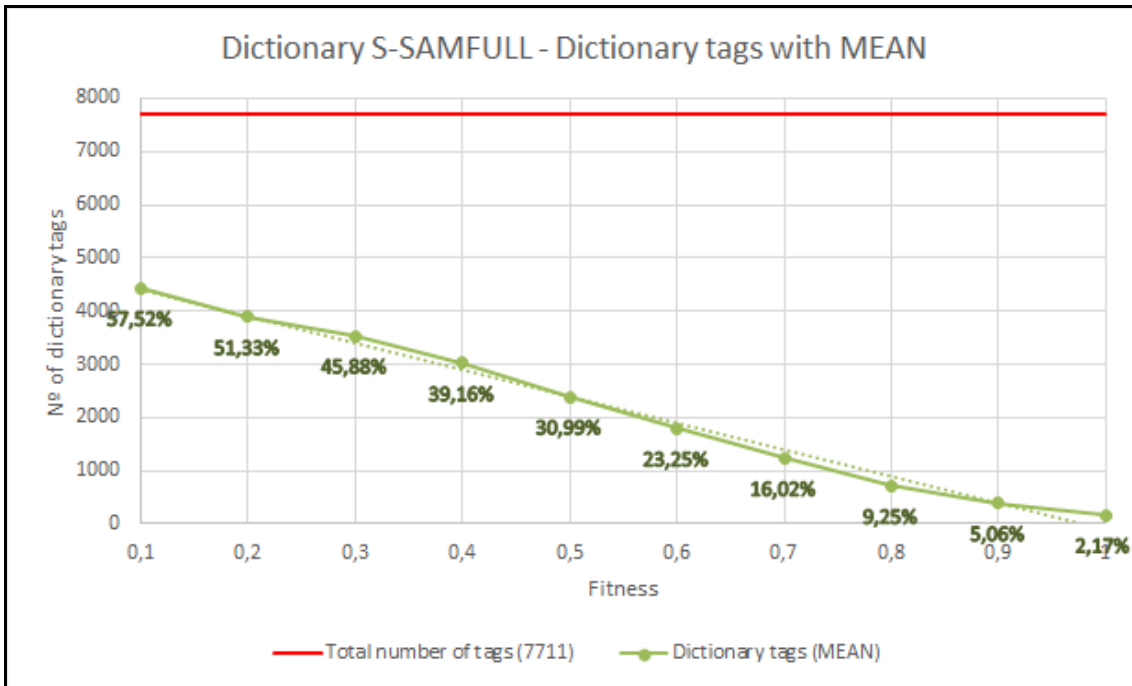


Figure 15. Dictionary S-SAM FULL - Dictionary tags with MEAN.

The next chart represent the comparison between them.

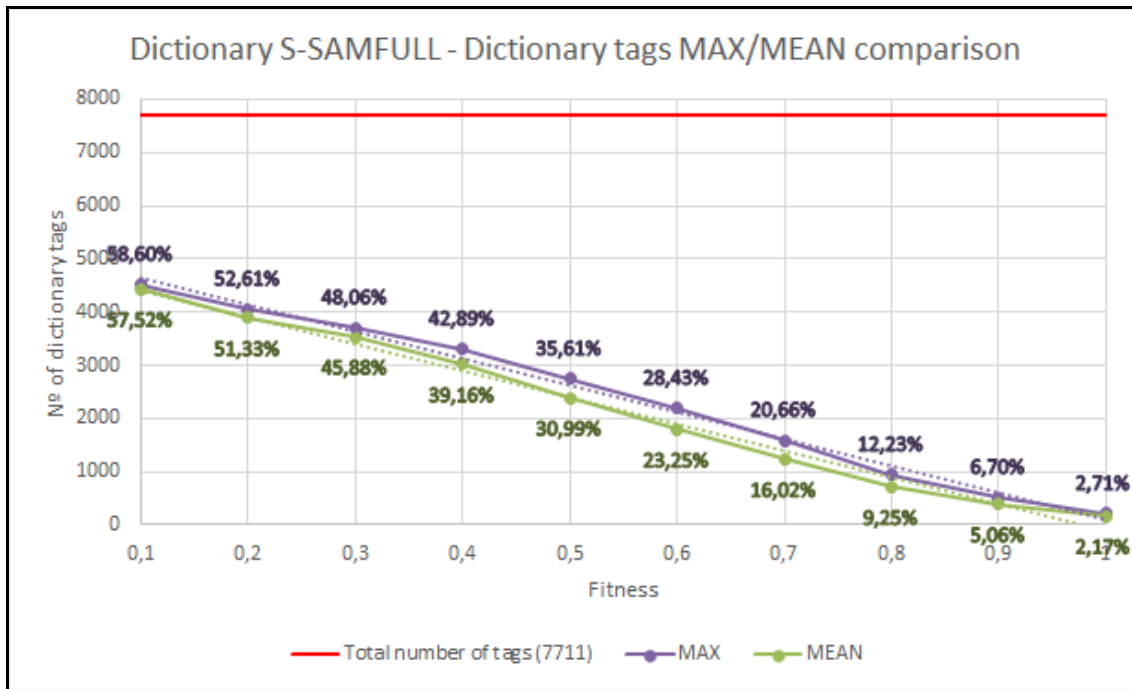


Figure 16. Dictionary S-SAM FULL - Dictionary tags MAX/MEAN comparison.

As in Dictionary SAM, the lower is the value of the tag fitness (less restrictive), the greater is the dictionary set. This is because it allows to enter more tags in the set as the restriction of the parameter is weaker. Both are quite similar.

The next graphics capture the behaviour of resources classification varying the *dicMinFitness* and *methodDictionary* parameters, also, the *classifierT*, which is the type of classifier that perform the similarities:

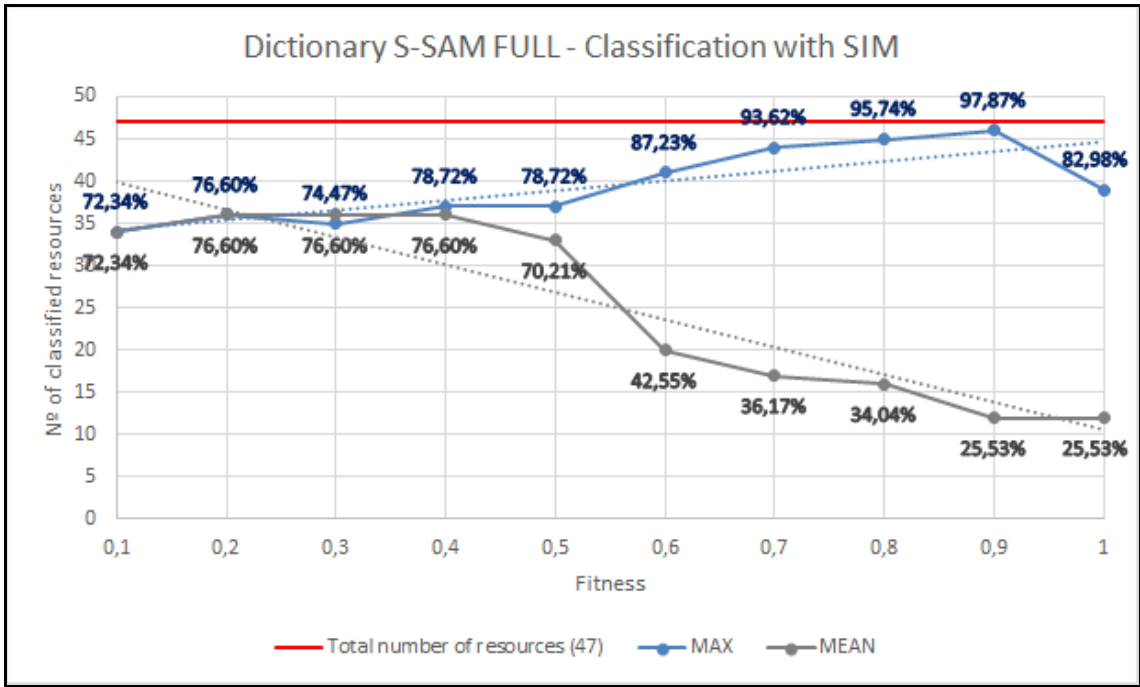


Figure 17. Dictionary S-SAM FULL - Classification with SIM.

As we can see, both methods classify in the same way in lower fitness values and, in the higher ones, it's MAX method which considerably grows while MEAN exponentially decreases. This is because MEAN method uses an average of the tags from dictionary to do the classification. This implies that there are more meaningful tags in low fitness values than in the higher ones. And as MAX method choses the best one, the classification is good in all cases.

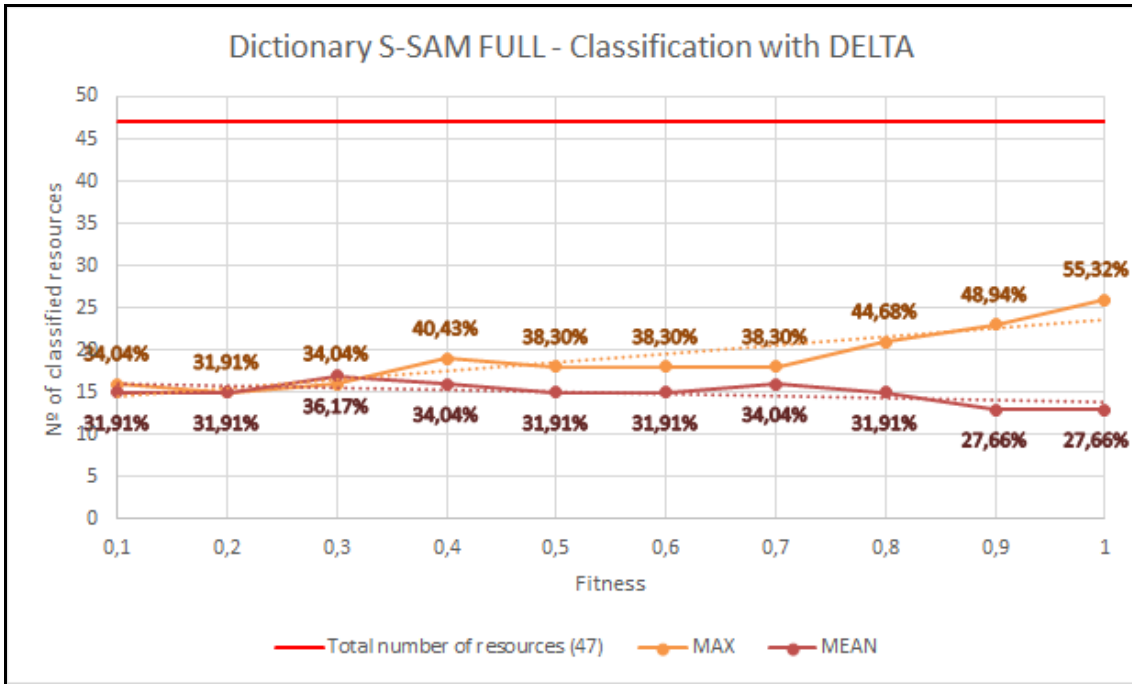


Figure 18. Dictionary S-SAM FULL - Classification with DELTA.

With DELTA classifier it's experimented the same thing that with SIM classifier. MEAN method gives slightly better results in lower fitness values and MAX method is always better.

In the next chart all these classifications are gathered in just one to have a clear viewpoint:

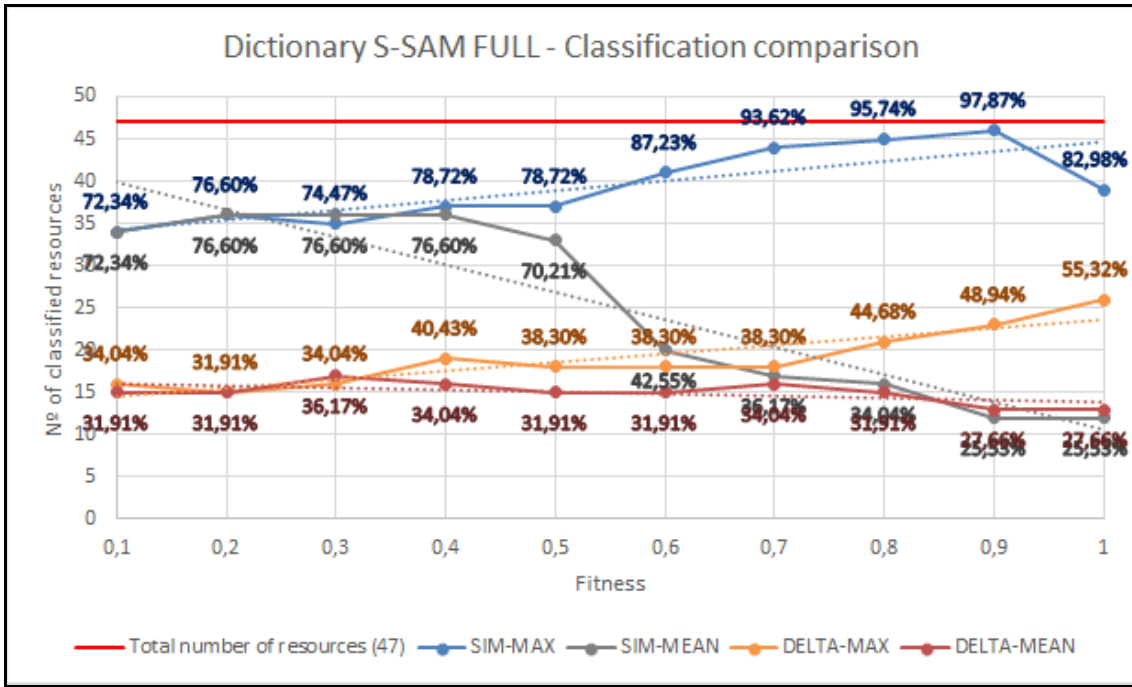


Figure 19. Dictionary S-SAM FULL - Classification comparison.

It's obvious that SIM-MAX gives the best results but this does not mean that the resources are correctly classified. In order to find it out, each resource of each classification concept of each classifier have been evaluated. The next graphics shows those resources that are correctly classified under each classifier:

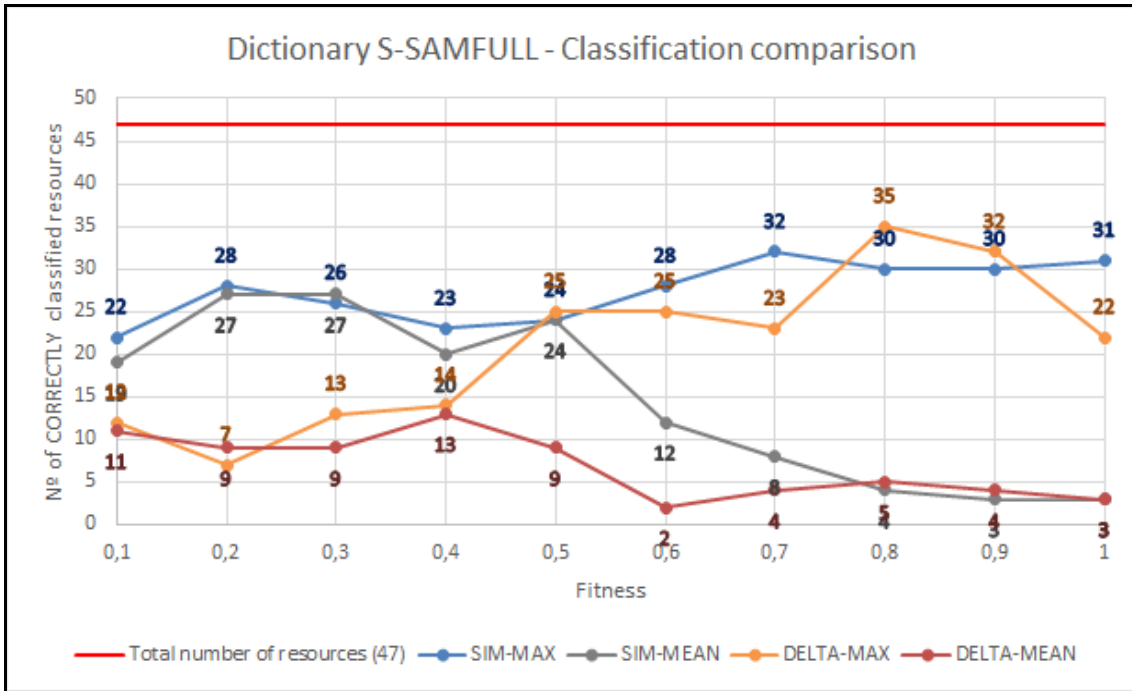


Figure 20. Dictionary S-SAM FULL - Classification comparison (correct).

It can be observed that SIM-MAX is, on average, the method that more resources classify, except in some specific cases. Both SIM-MEAN and DELTA-MEAN classify fewer resources than the others. And DELTA-MAX classify better when the value of the fitness is higher. Concretely, DELTA-MAX with *dicMinFitness* = 0.8 gives the best classification value of all the methods.

5.1.4. Results comparison among Dictionary SAM, S-SAM and S-SAM FULL

In order to find out what version of SAM is which classifies the best, it have taken the graphics of the correctly classified resources and compare between them.

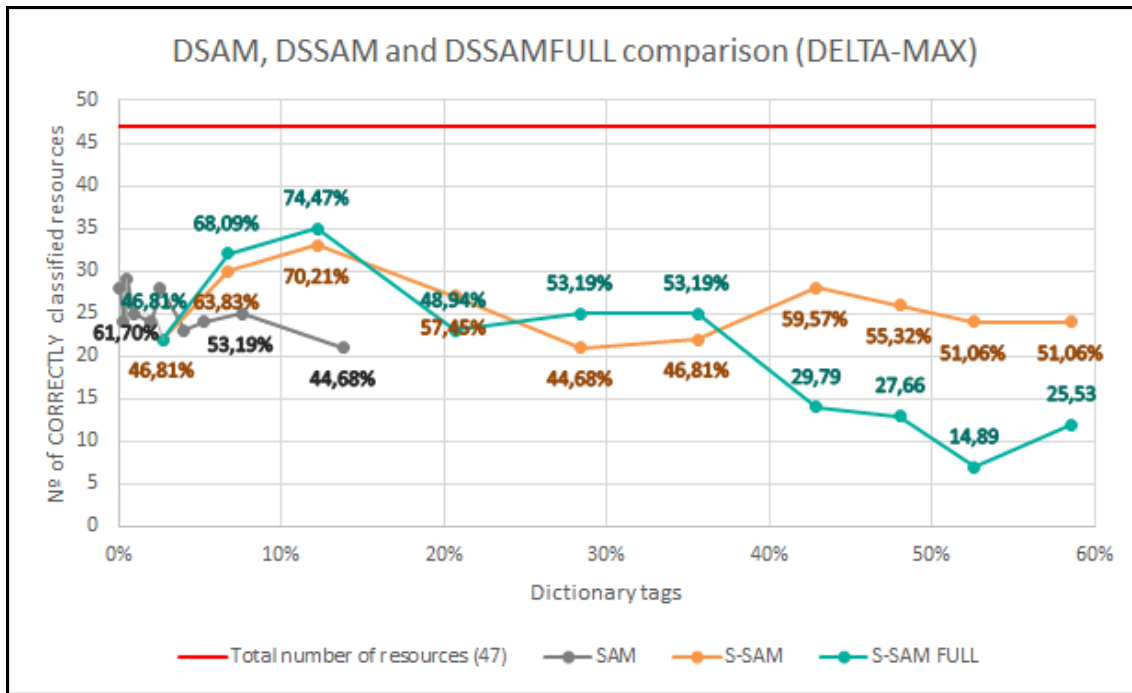


Figure 21. DSAM, DSSAM and DSSAMFULL comparison DELTA-MAX (correct).

This chart represent how SAM, S-SAM and S-SAM FULL DELTA-MAX for R47 behaves as dictionary tags set changes. In particular, if this set of tags represents the 5%-15%, which is the ideal case, the best results are obtained using S-SAM FULL.

It has to be said that using S-SAM FULL with DELTA-MAX and dicMinFitness = 0.8 (Dictionary tags = 12.23%) gives the best aggrupation of resources. First because the name of the classification concepts are very united and second because the resources classified under each resource are strongly related. The meaning among them is firm and with the name of the group too. It's by far the best classification among Dictionary triad.

However, this is a utopian approach. Because using a social tagging system with S-SAM FULL means that each tag labelled to each resource have to be evaluated by experts in consensus. And this is quite impossible. Or just reasonably unachievable due to the huge amount and heavy human work.

So the best, and rational, option is S-SAM, which is pretty similar to S-SAM FULL but with an exponentially less human work, since in this case only a reduced set of tags (the dictionary) have to be evaluated. Concretely, the 10% of tags of social tagging system approximately to obtain the best results.

5.2. SAM and S-SAM for delicious

Once performed the experiments with R47, it's time to do the same thing but with the whole delicious database. This bit set of annotations also contains R47. In particular, the delicious set contains 521611 annotations, with 6852 resources and 122786 tags. It's important to point out that the dictionary set will be made by means of R47 set.

Therefore, SAM will use the number of occurrences for making the dictionary and S-SAM will use the fitness value. With these two methods some experiments have been carried out.

5.2.1. SAM results

SAM uses the number of occurrences to establish which tags are representative and to do the corresponding calculus.

The input parameters that have been used are the following ones:

- `dicMinAnnotation` = 1.000 / 3.000 / 5.000 / 10.000 / 20.000 / 50.000 / 100.000
- `convergMinAnnotations` = 100
- `classifThreshold` = 0.1
- `mergingThreshold` = 0.75
- `namingThreshold` = 0.5
- `simMinValue` = 0.5
- `simSRRThres` = 0.4
- `classifierT` = SIM / DELTA

The following graphic reflects the behaviour of tags from dictionary when modifying the *dicMinAnnotation* parameter, which represents the minimum number of annotations that a representative tag have to have.

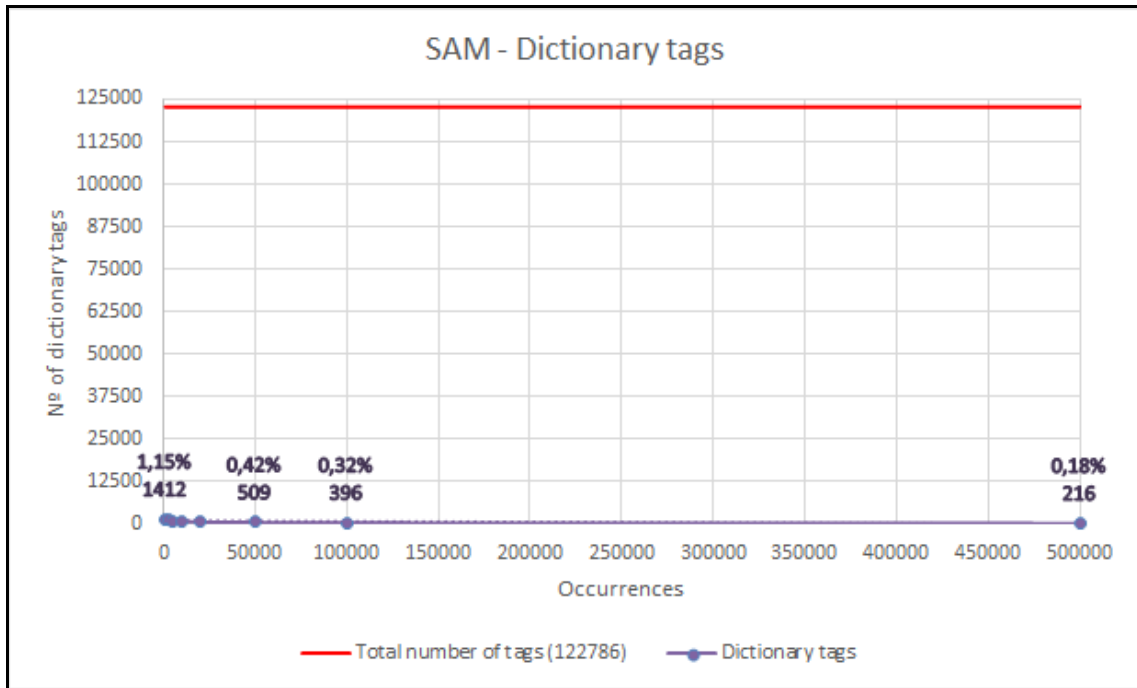


Figure 22. SAM - Dictionary tags.

Let's zoom in to see in detail.

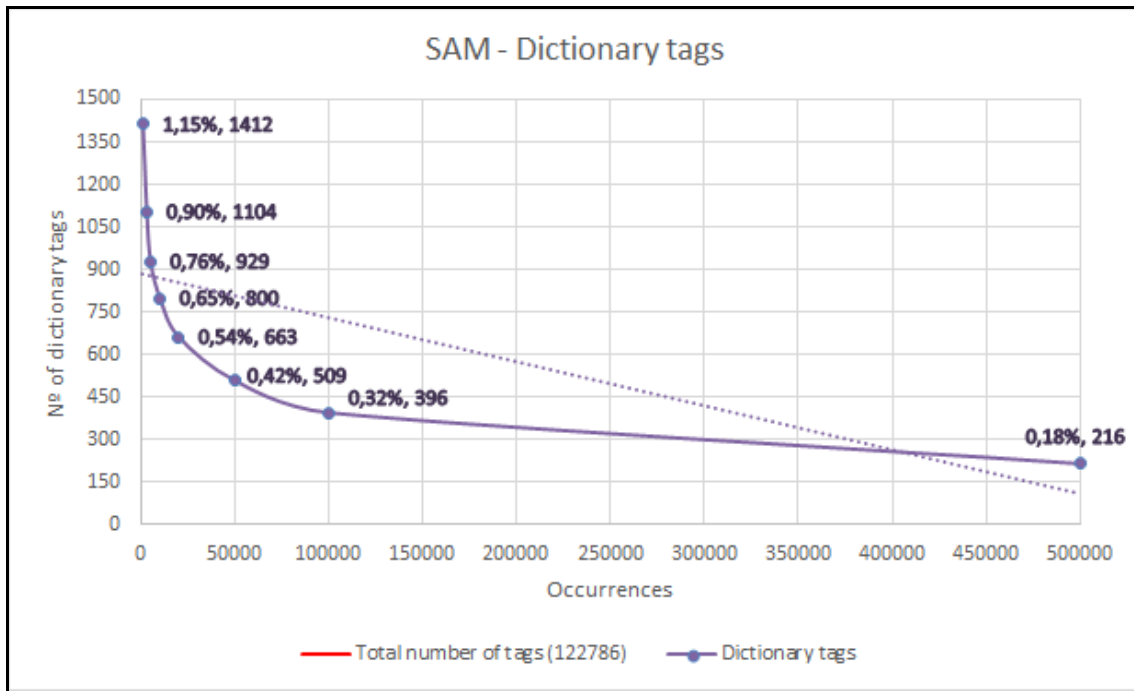


Figure 23. SAM - Dictionary tags zoomed in.

Here, the lower is the number of occurrences of tags (less restrictive), the greater is the dictionary set. This is because it allows to enter more tags in the set as the restriction of the parameter is weaker.

The next graphics capture the behaviour of resources classification varying the *dicMinAnnotation* parameter and, also, the *classifierT*, which is the type of classifier that perform the similarities:

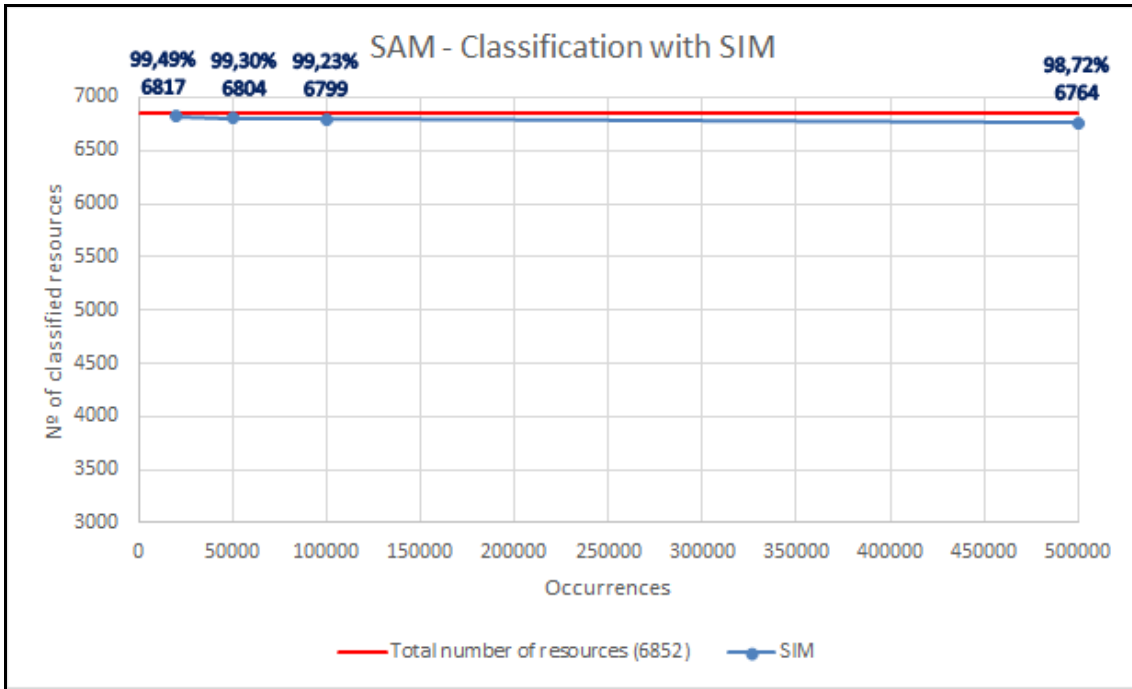


Figure 24. SAM - Classification with SIM.

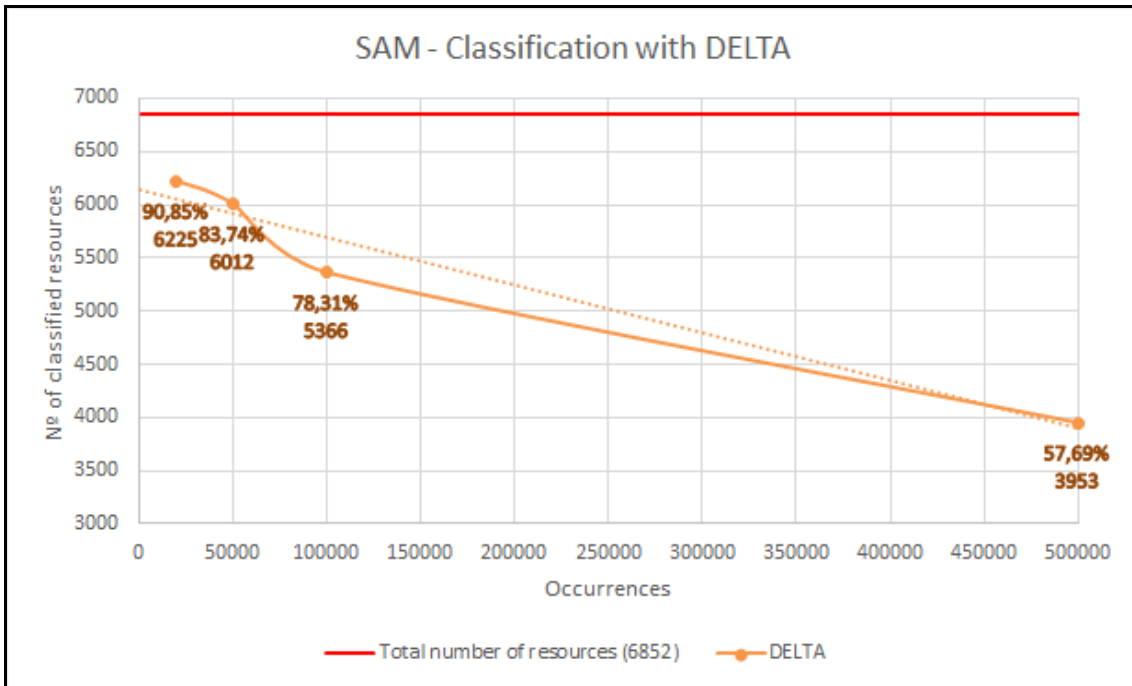


Figure 25. SAM - Classification with DELTA.

In these charts is gathered how SAM classifies resources depending on the classifier. We can see that in classifier SIM practically every resources is classified even if when the

dictionary set is very little, while in DELTA classifier, classifies less resources. Here, we can see the comparison between them.

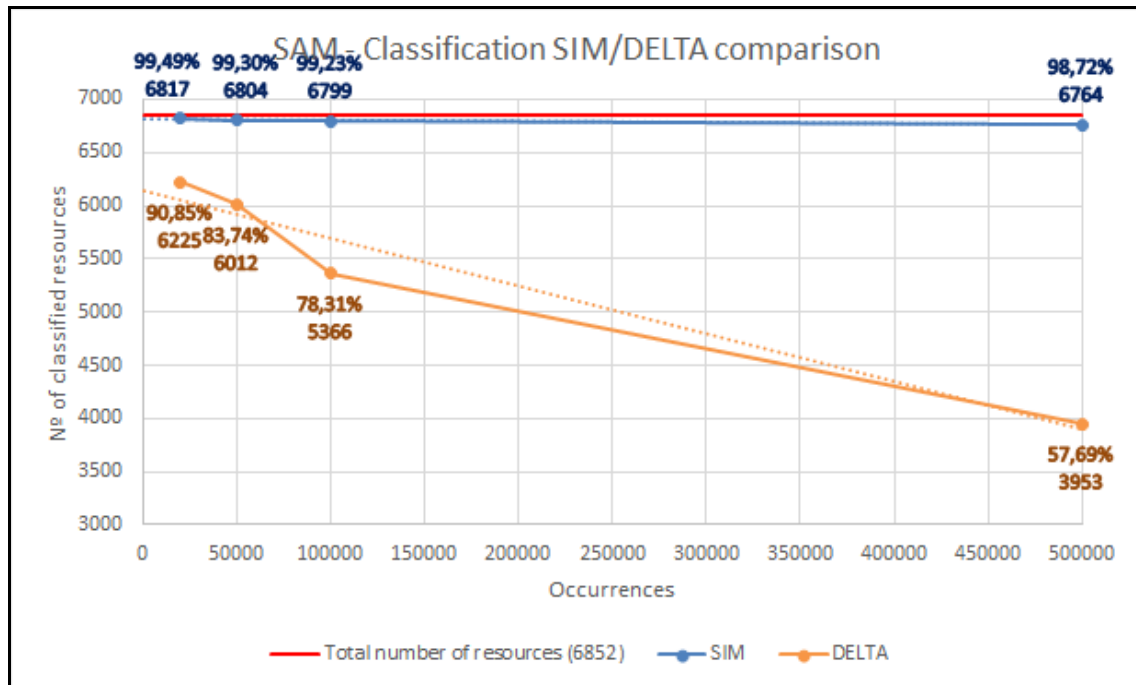


Figure 26. SAM - Classification SIM/DELTA comparison.

It's clear that SIM classifier classifies more resources (practically every one) than DELTA, but it's not sure that those resources are properly classified. So it have been checked how the 47 resources from R47 were been classified. This is reflected in the next graphic:

5.2.2. S-SAM results

S-SAM uses the fitness value to establish which tags are representative and the number of occurrences to do the corresponding calculus.

The input parameters that have been used are the following ones:

- $\text{dicMinFitness} = 0.1 / 0.2 / 0.3 / 0.4 / 0.5 / 0.6 / 0.7 / 0.8 / 0.9 / 1$
- $\text{methodDictionary} = \text{MAX} / \text{MEAN}$
- $\text{convergMinAnnotations} = 100$
- $\text{classifThreshold} = 0.1$
- $\text{mergingThreshold} = 0.75$
- $\text{namingThreshold} = 0.5$
- $\text{simMinValue} = 0.5$
- $\text{simSRRThres} = 0.4$
- $\text{classifierT} = \text{SIM} / \text{DELTA}$

The following graphic reflects the behaviour of tags from dictionary when modifying the *dicMinAnnotation* parameter, which represents the minimum number of annotations that a representative tag have to have, and the *methodDictionary* parameter, which is the followed method to select the fitness value of the tag (MAX or MEAN).

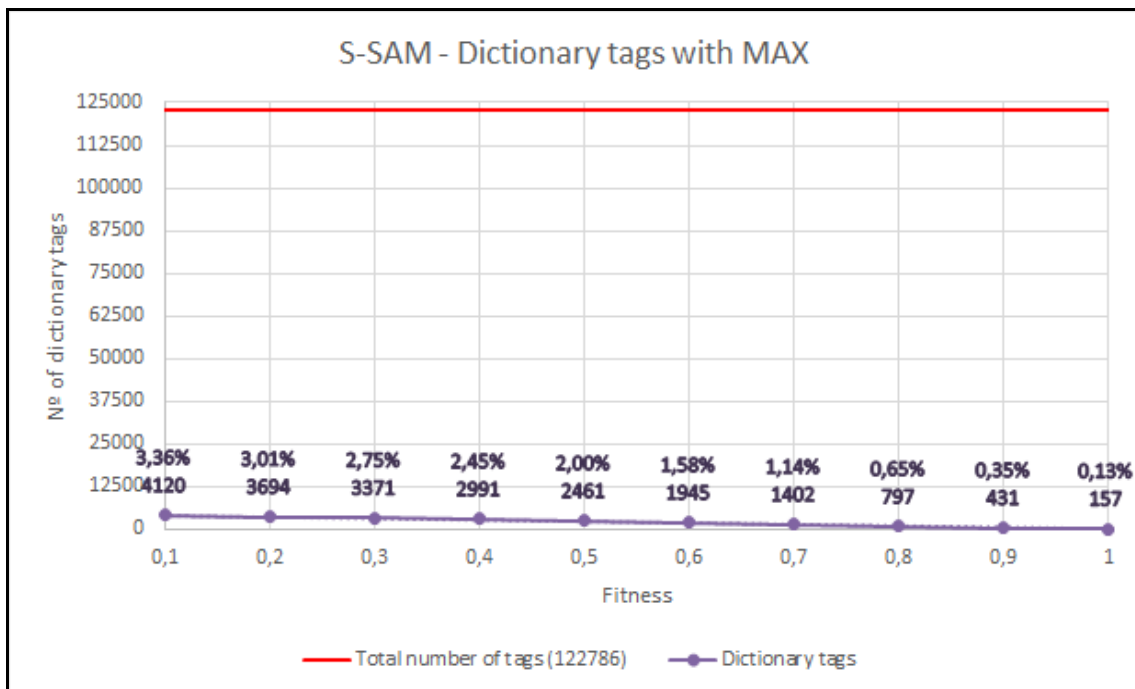


Figure 27. S-SAM - Dictionary tags with MAX.

Let's zoom in to see in detail.

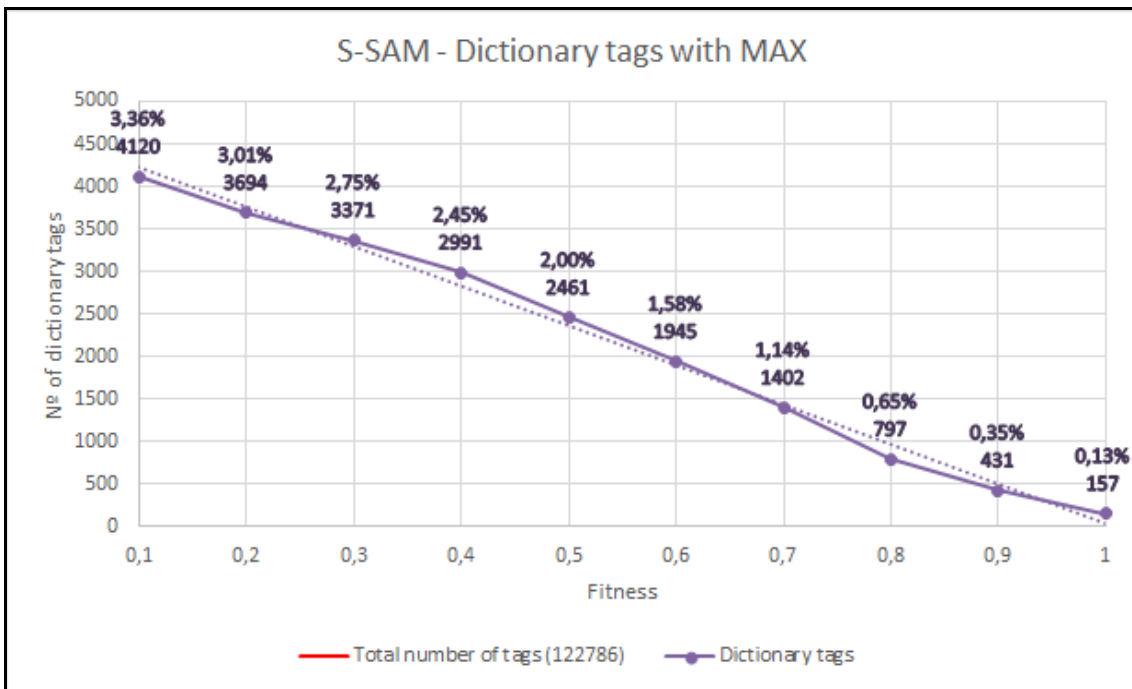


Figure 28. S-SAM - Dictionary tags with MAX zoomed in.

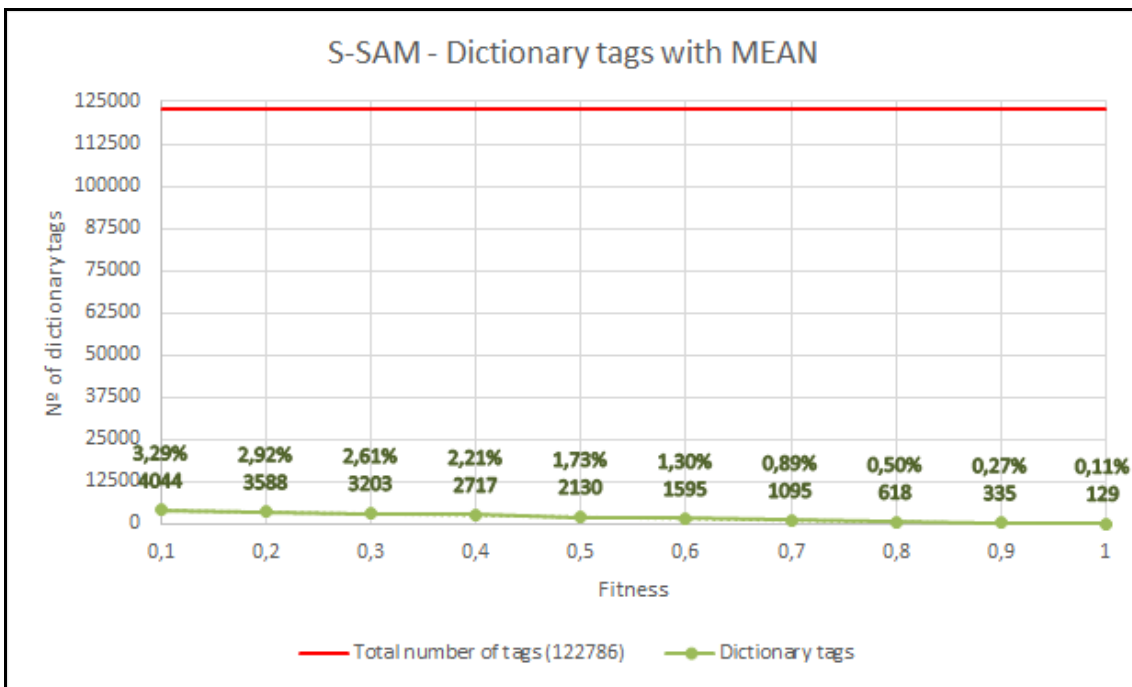


Figure 29. S-SAM - Dictionary tags with MEAN.

Let's zoom in to see in detail.

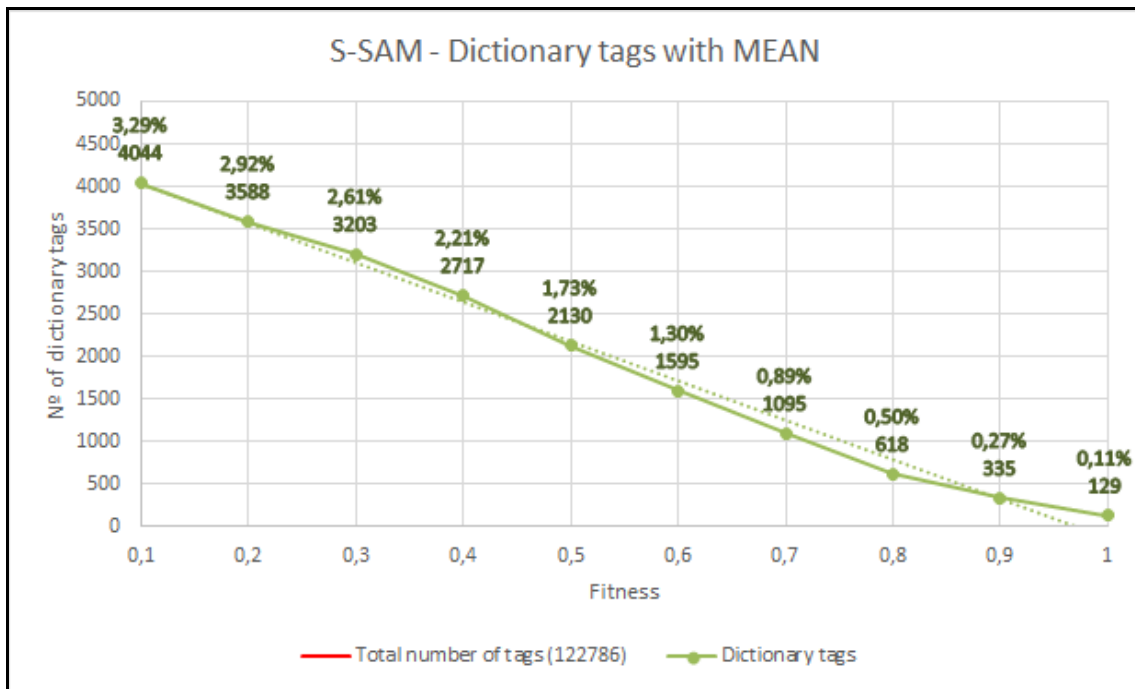


Figure 30. S-SAM - Dictionary tags with MEAN.

The next chart represent the comparison between them.

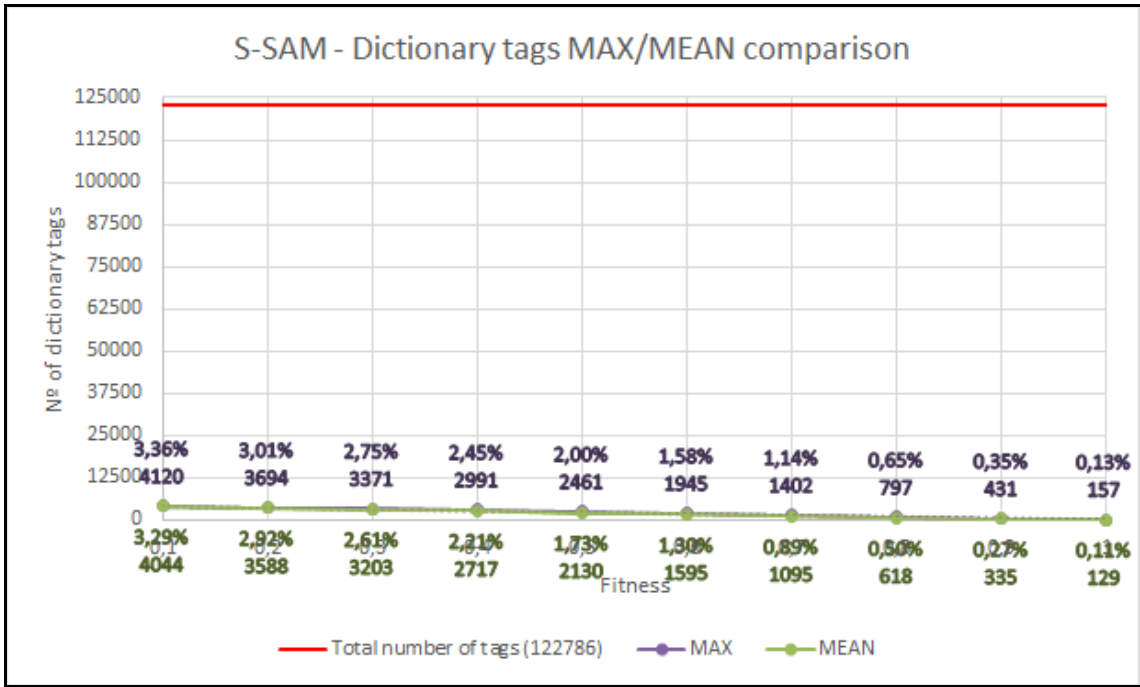


Figure 31. S-SAM - Dictionary tags MAX/MEAN comparison.

Let's zoom in to see in detail.

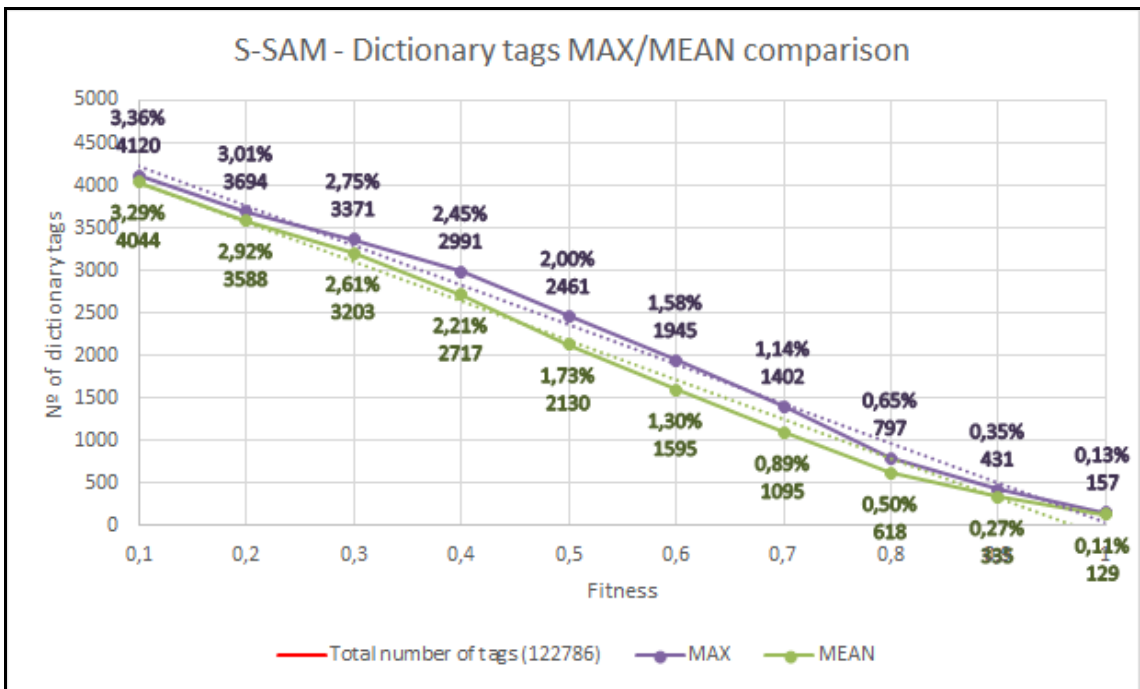


Figure 32. S-SAM - Dictionary tags MAX/MEAN comparison zoomed in.

As in SAM, the lower is the value of the tag fitness (less restrictive), the greater is the dictionary set. This is because it allows to enter more tags in the set as the restriction of the parameter is weaker. Both are quite similar.

The next graphics capture the behaviour of resources classification varying the *dicMinFitness* and *methodDictionary* parameters, also, the *classifierT*, which is the type of classifier that perform the similarities. Due to the interesting interval of fitness value is between 0.7 and 1, S-SAM only have been performed for these values.

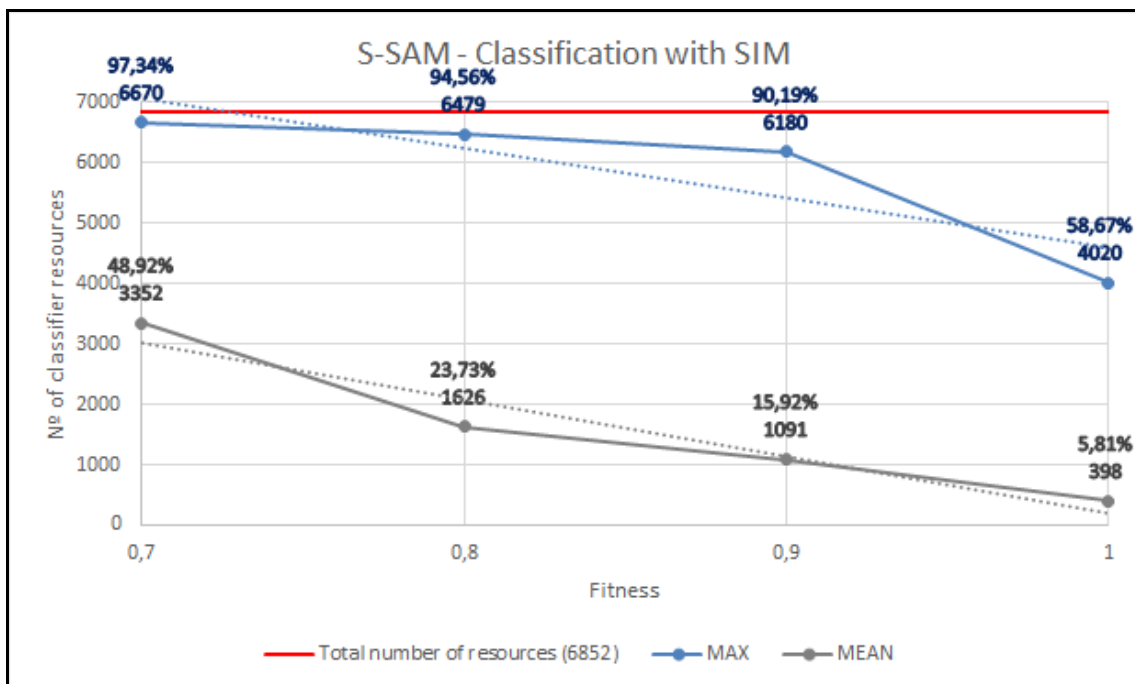


Figure 33. S-SAM - Classification with SIM.

As we can see, using MEAN method classifies less resources than MAX within the fitness values of interest. This is because MEAN method uses an average of the tags from dictionary to do the classification and this implies that there are more meaningful tags in low fitness values than in the higher ones. MAX method gives good results within this interval, especially when *dicMinFitness* = 0.7.

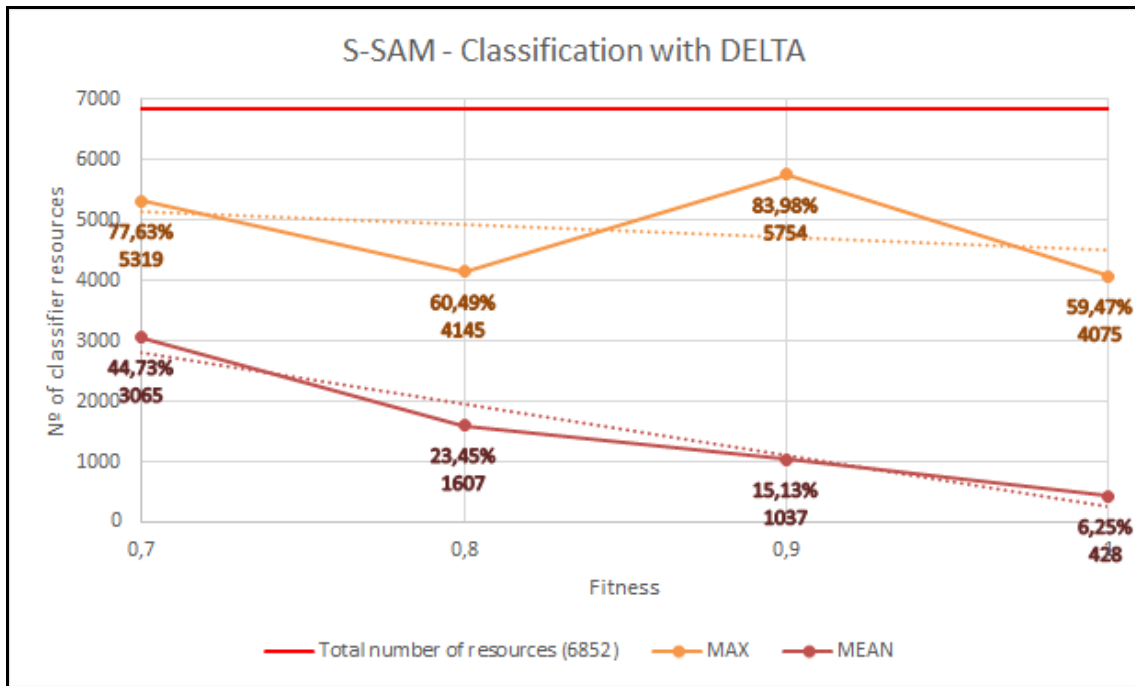


Figure 34. S-SAM - Classification with DELTA.

With DELTA classifier it's experimented the same thing that with SIM classifier. MEAN method gives worse results than MAX method, which is much better due to classify more resources.

In the next chart all these classifications are gathered in just one to have a clear viewpoint:

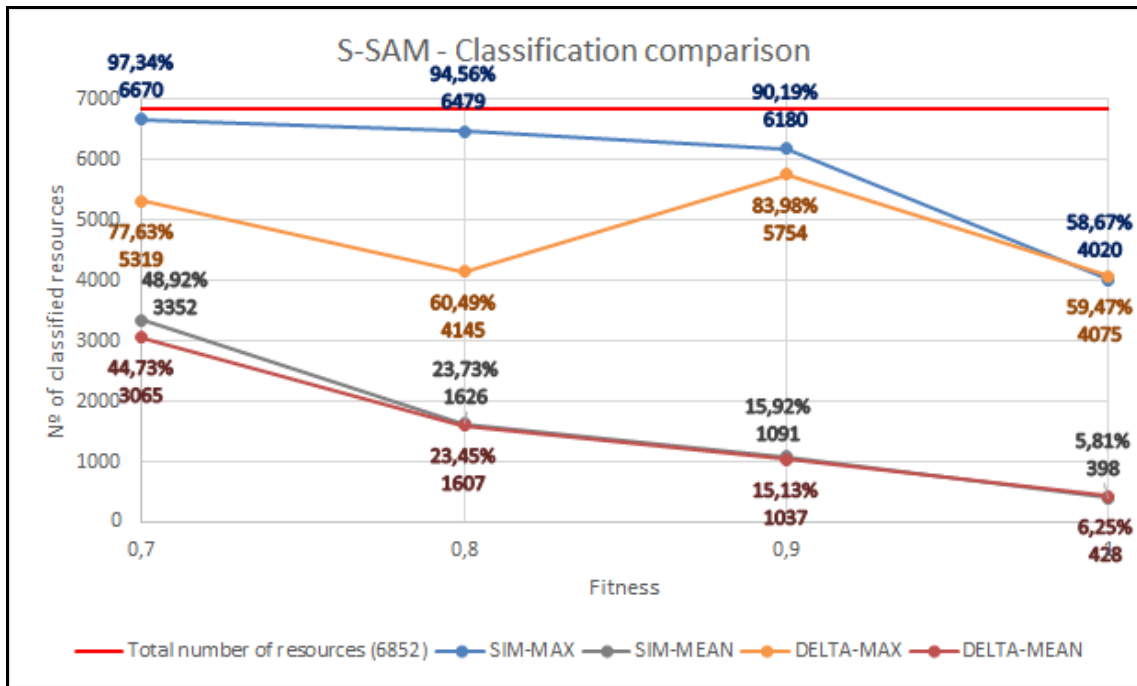


Figure 35. S-SAM - Classification comparison.

It's obvious that SIM-MAX gives the best results but this does not mean that the resources are correctly classified, as proved before. Both SIM-MEAN and DELTA-MEAN classify fewer resources than the others. And DELTA-MAX classify very similar to SIM-MAX when *dicMinFitness* is 0.9 and 1.

However, and despite the good results of SIM-MAX, we can assume after analysing the results in R47 that the number of correctly classified resources is lower than in DELTA-MAX, and also forms worse groups. So the best way to appropriately classify resources is DELTA-MAX.

5.2.3. Results comparison between SAM and S-SAM

Next, SAM and S-SAM results will be compared in order to see how these methods perform the classification.

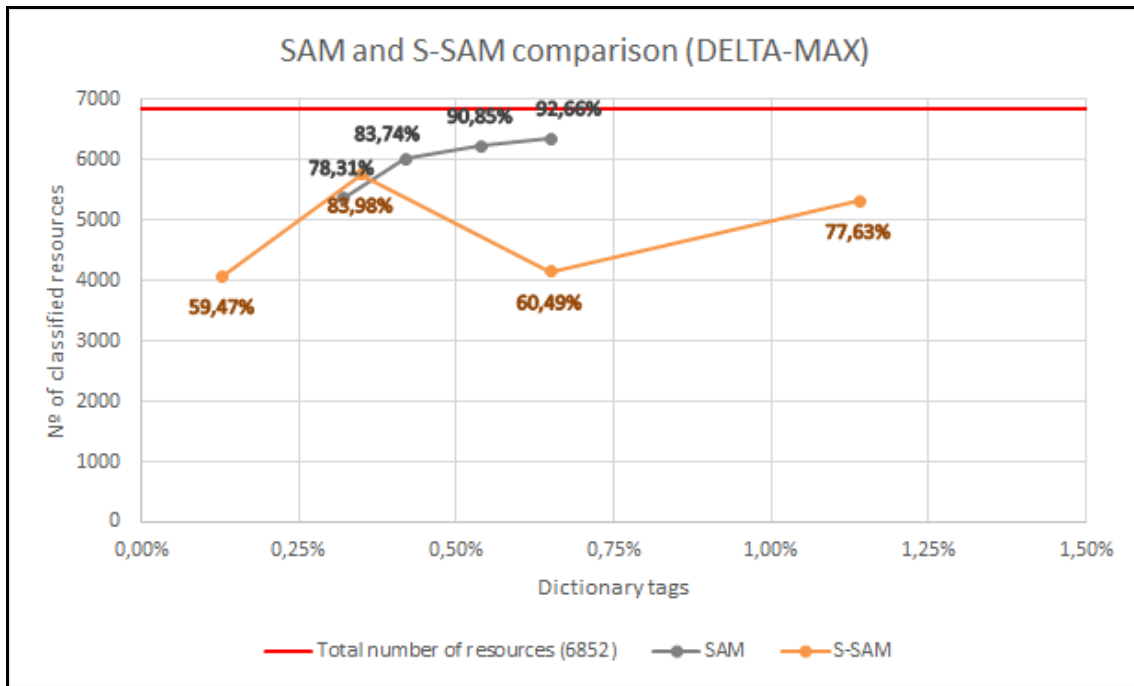


Figure 36. SAM and S-SAM comparison (DELTA-MAX).

This chart represents how SAM and S-SAM for delicious behaves as dictionary tags set changes. It can be seen that SAM classifies a big percentage of the resources of the social tagging system, while S-SAM classifies less resources. Experience with R47 tells us that S-SAM classifies more resources correctly than SAM.

Besides, it's important to point out that these results come from very little dictionary sets. The classification corresponds to, more or less, a 0.5% of the social tagging system tags; this is 600 tags out of 112786. The results from R47 suggest that if the dictionary grows enough to cover a 5%-10% of the tags, the classification would be much better than these results.

Let's see what happens with the 47 resources of R47 and how they were classified in delicious.

In the following graphic we can see the performance of SAM:

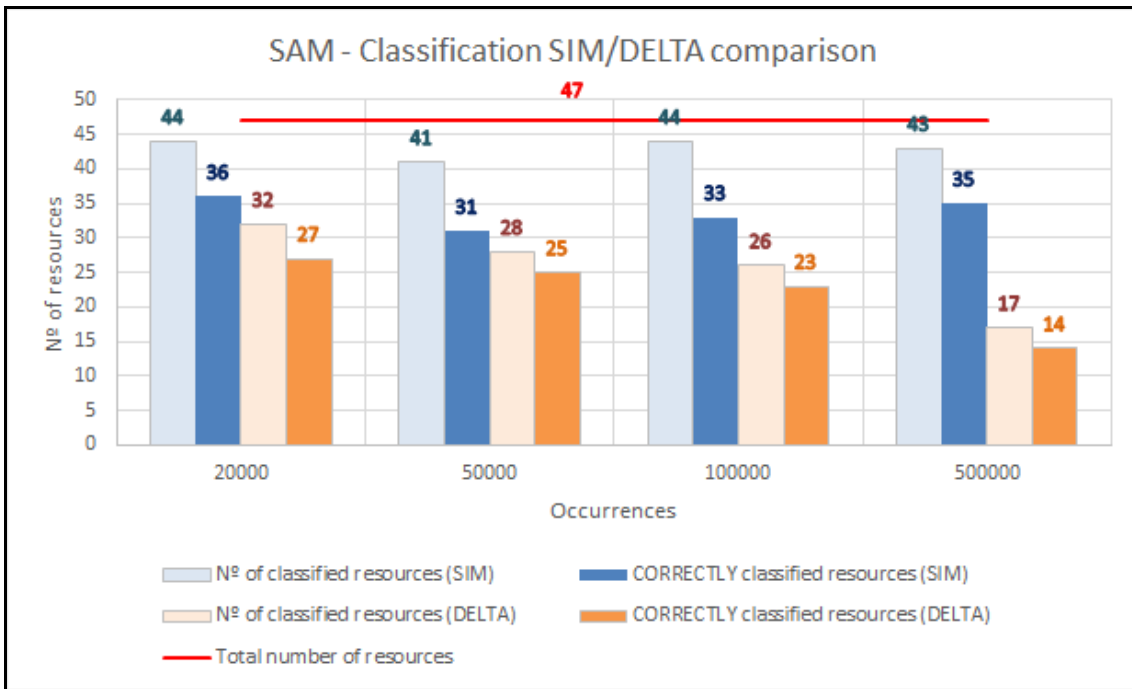


Figure 37. SAM comparison with SIM-DELTA (correct).

As it can be seen, SIM classifier classifies more resources with a lower percentage of correct resources over the classifieds while DELTA classifies less resources with a higher percentage of correct resources over the classifieds. This means that SIM has more probabilities to fail that DELTA does but classifies the 90% or more of resources of the social tagging system.

Regarding S-SAM, we have two graphics, one with the classification made up by SIM and another with DELTA:

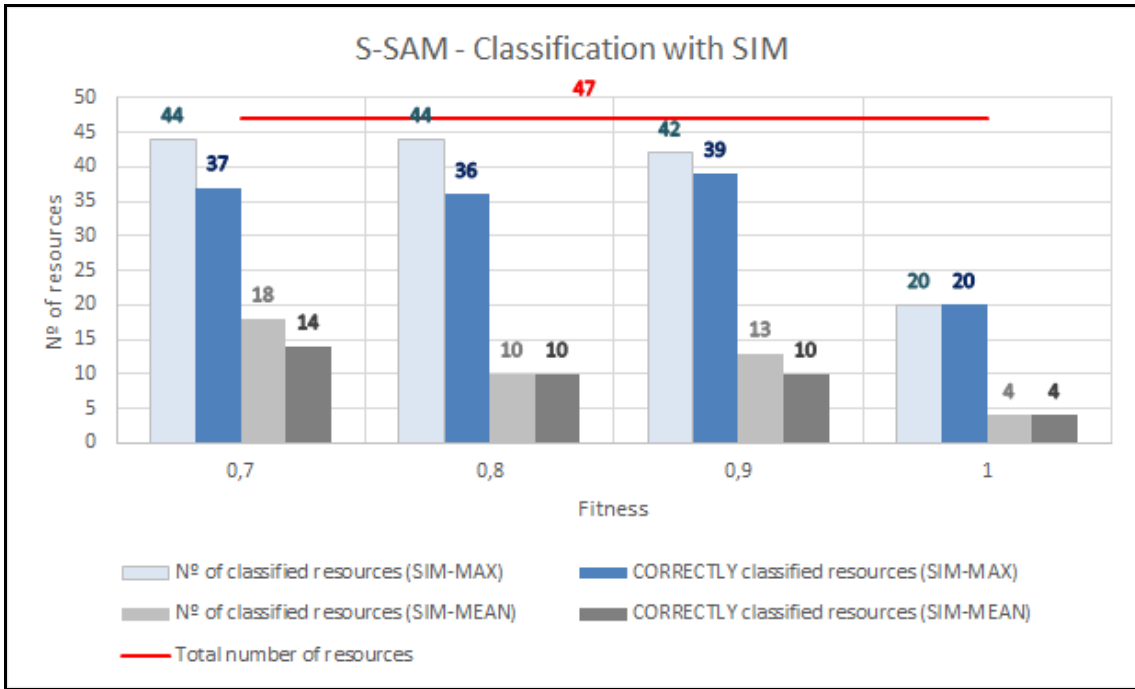


Figure 38. S-SAM comparison with SIM (correct).

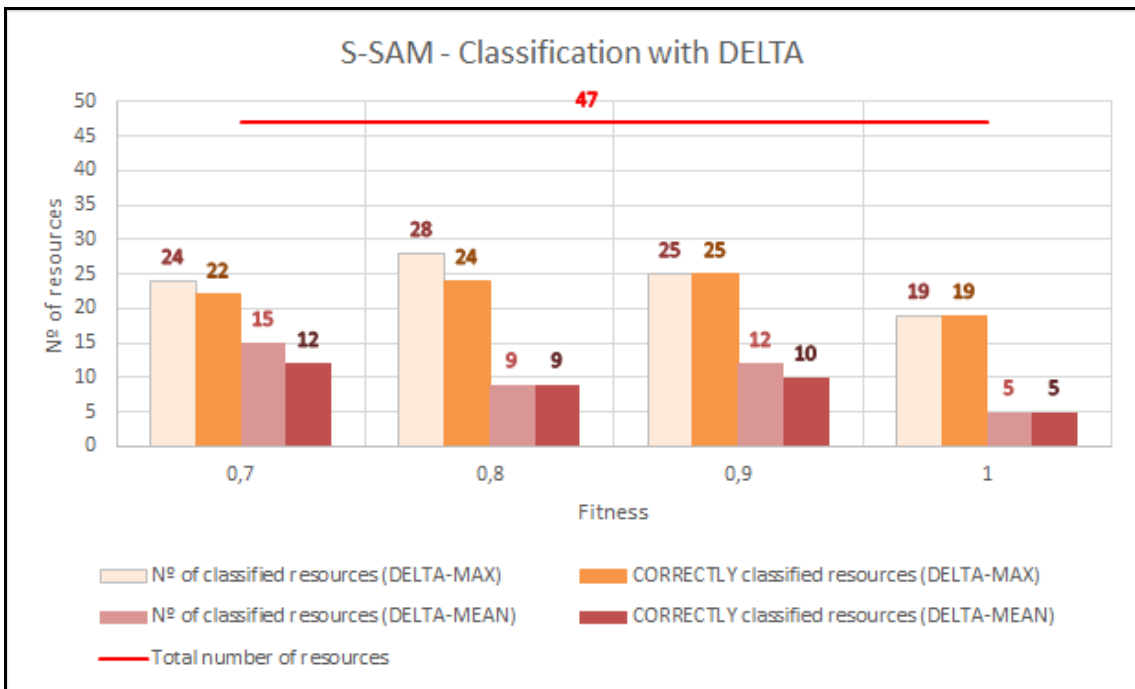


Figure 39. S-SAM comparison with DELTA (correct).

But as MEAN dictionary method gives bad results, we get rid of it and show the results without it as a summarized performance of S-SAM:

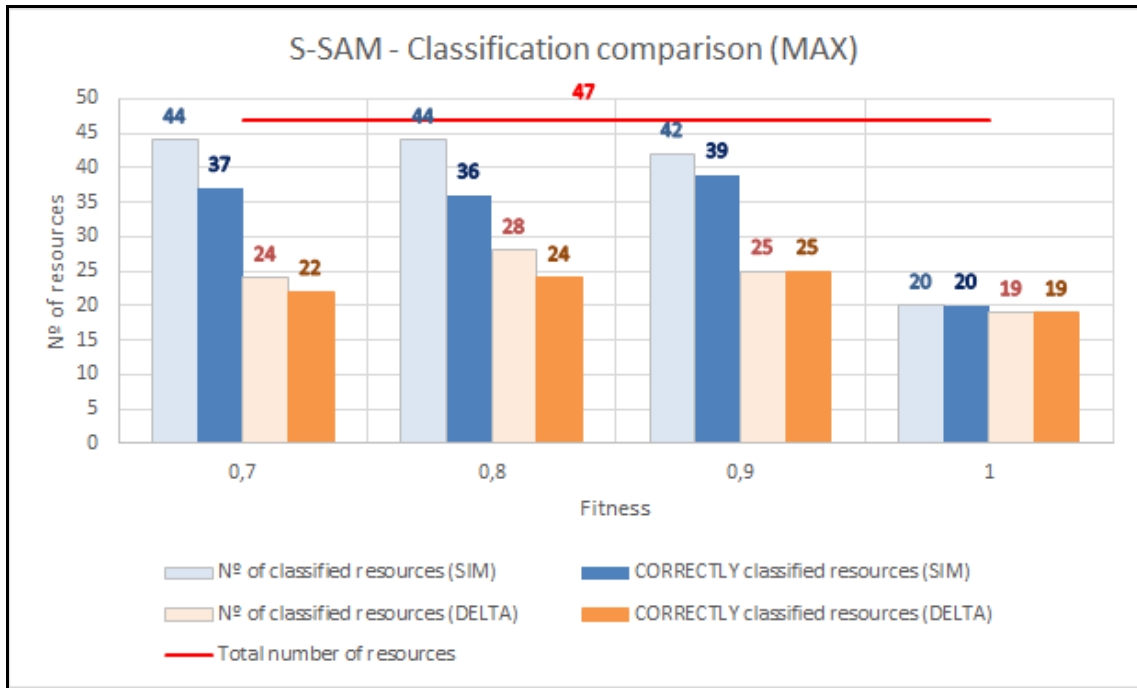


Figure 40. S-SAM comparison (correct).

It's obvious that SIM classifier, as every time, classifies more results than DELTA but DELTA what classifies is almost always correctly classified. Nonetheless, when fitness=0.9, the results got by using SIM classifier gives the best performance ever. First because the name of the formed groups are very accurate in relation to the resources it gathers and second because the classified resources within each group are strongly linked among them.

So S-SAM gives better results than SAM, and it does when SIM classifier is chosen, particularly when fitness=0.9.

Chapter VI:

CONCLUSIONS



In this chapter it will be gathered all the conclusions extracted from the experimental results from the previous chapter that have been carried out. Also, it will establish the background for the possible future research lines.

The first conclusion that have been obtained from this study is that DELTA classifies, in general, much better than SIM. DELTA forms the groups (classification concepts) with highly linked tags and also the resources are very similar among them. However, SIM classifies much more resources.

About S-SAM and S-SAM FULL, when choosing the tags of the dictionary, the best method is MAX not only because of it classifies more resources but it's interesting to form small dictionary tags set and not big ones as MEAN needs to output good results.

Regarding SAM and S-SAM, the most important conclusion from this research is that S-SAM is better than SAM, because it classifies more correctly resources from the social

tagging system. This is because the dictionary is not made by taking account the number of occurrences but the level of adequacy, that is, the fitness value. The possible reasons for this could be several, such as using non- descriptive tags to a resource and collaborate this way not to represent its semantic or assigning personal use tags ([Golder and Huberman, 2006](#)), such as “toRead” or “myJob”.

It has to be pointed out that when the social tagging system is little, in this case R47, it work better DELTA, but when using a big one, in this case delicious, the winner is SIM.

And finally, S-SAM FULL is the best alternative to get more correctly classified resources. However, this method is practically impossible to achieve because using it in a real social tagging system supposes that each annotation have to be evaluated by a group of experts.

Chapter VII:

FUTURE LINES



The results of this research we lead us to conclude that the S-SAM is better than SAM method. This have been verified using the results taken from R47 and delicious databases. Also S-SAM FULL is even better than S-SAM but reasonably unachievable.

Despite of the good results of this last one, it seems that it could classify more resources if naming threshold is recalibrated. For the S-SAM FULL the results have been done using *namingThreshold* = 0.9 and not 0.5 as in SAM and S-SAM. This was because S-SAM FULL uses the fitness value of tags to put the name to the groups while the others use the number of occurrences. As this last one do not have a fixed range but can be any positive natural number, using those tags which number of occurrences is higher than the 50% value of the highest (*namingThreshold* = 0.5) is enough for giving a good name. But with SAM, as the fitness value is limited to an interval from 0 to 1, if *namingThreshold* = 0.5, it allows too much tags names to form the name.

But it have been checked that the value used in S-SAM FULL (0.9) is too restrictive and the validation of each resource could be wrongly made up. So it would be a good idea to research about it: maybe reducing the value to 0.8 or 0.7 or finding out other new ways to give the names of the groups.

Regarding DELTA classifier, from the results it can be checked that it obtains the best percentages of correctly classified resources. That is, what classifies is almost always correct. The problem is that a lot of resources are not classified, the 50% of the resources in average. And its origin, I think, is in the beginning, when choosing some random annotations. Using a random method for evaluation tags gives random results, that is, the chosen tags set is created by using relevant and irrelevant tags. This outputs a mixed dictionary and hence the classification is not the best and could be improved. So if from the beginning a potential tags set is chosen, the experience with folksonomies tells me that the classification using DELTA, and SIM too, would became better. And how know which tags are potentially representative? Using those with more number of annotations.

So, in summary, I propose to select the annotations of those tags with the highest number of occurrences, evaluate them with a group of experts and perform the classification with S-SAM.

And eventually, respecting a way to improve S-SAM, using the following algorithm could classify more resources. It has to do with the remained set of unclassified resources. By each loop the fitness value decrease to insert the non-classified resources into the groups done by $\text{dicMinFitness} = 1$.

Table 8. General categorization algorithm for S-SAM.

```
1. var
2.   v_sem_max: integer    /*maximum semantic value (1)*/
3.   v_sem_increm: integer /*value of increment (0.1)*/
4. begin
5.   v_sem_max := 1;
6.   v_sem_increm := 0.1
7.   while v_sem_max <> 0.0 do
8.     S_SAMscreation(R)v_sem_max
9.     if |R|=|Rclassified| then
10.      /*all resources are classified*/
11.      v_sem_max := 0.0
12.     else
13.      R ← R − Rclassified;
14.      Rpending ← ε;
15.      Rconverged ← ε;
16.      v_sem_max := v_sem_max − v_sem_increm;
17.     endif;
18.   enddo;
19. end;
```


Chapter VIII:

BIBLIOGRAPHY



Bibliographic reference

A. Córdoba, J.J. Astrain *, J. Villadangos, F. Echarte. (2013). “A self-adapted method for the categorization of social resources.” *Elsevier*.

Begelman, G. (2006). Automated tag clustering: Improving search and exploration in the tag space.

- <http://www.ambuehler.ethz.ch/CDstore/www2006/www.rawsugar.com/www2006/20.pdf>

Chi, E. H. y Mytkowicz, T. (2008). Understanding the efficiency of social tagging systems using information theory.

- <http://www.aaai.org/Papers/ICWSM/2008/ICWSM08-032.pdf>

Ciordia, A. (2011). Sistema colaborativo de clasificación de imágenes médicas basado en folsonomías.

- <http://academica-e.unavarra.es/bitstream/handle/2454/4714/577693.pdf>

Echarte, F. (2011). ACoAR: a method for the Automatic Classification of Annotated Resources.

Golder, S. A. y Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *Journal of Information Science*.

- <http://arxiv.org/ftp/cs/papers/0508/0508082.pdf>

Gruber, T. R. (1993). Knowledge acquisition.

- <http://tomgruber.org/writing/ontolingua-kaj-1993.pdf>

Gruber, T. R. (2007). Ontology of folksonomy: A mash-up of apples and oranges. *International Journal on Semantic Web & Information Systems*.

- <http://tomgruber.org/writing/ontology-of-folksonomy.htm>

Hotho, A., Jäschke, R., Schmitz, C., & Stumme, G. (2006a). Folkrank: A ranking algorithm for folksonomies. In *Proceedings of the workshop der fachgruppe information retrieval*.

- <http://www.kde.cs.uni-kassel.de/stumme/papers/2006/hotho2006folkkrank.pdf>

Hotho, A., Jäschke, R., Schmitz, C., & Stumme, G. (2006b). Information retrieval in folksonomies: Search and ranking. *Proceedings of the 3rd European semantic web conference*. Budva, Montenegro: Springer.

- <http://www.kde.cs.uni-kassel.de/stumme/papers/2006/hotho2006information.pdf>

McGuinness, D. L. y van Harmelen, F. (2004). OWLWeb Ontology Language - Overview. *W3C Recommendation*.

- <http://www.w3.org/TR/2004/REC-owl-features-20040210/>

Shadbolt, N., Berners-Lee, T., y Hall, W. (2006). The semantic web revisited.

- http://eprints.soton.ac.uk/262614/1/Semantic_Web_Revisited.pdf

Vander Wal, T. (2005). Explaining and showing broad and narrow folksonomies.

- <http://www.vanderwal.net/random/entrysel.php?blog=1635>

Vander Wal, T. (2007). Folksonomy.

- <http://vanderwal.net/folksonomy.html>

Online reference

- Couchbase
<http://www.couchbase.com/>
- Java
<https://java.com/es/>
- NetBeans
<https://netbeans.org/>
- WampServer
<http://www.wampserver.com/en/>