Of mice and terms: clustering algorithms on ambiguous terms in folksonomies

Nicola Raffaele Di Matteo
Department of Computer Science
University of Bologna
Bologna (Italy)
dimatteo@cs.unibo.it

Silvio Peroni
Department of Computer Science
University of Bologna
Bologna (Italy)
speroni@cs.unibo.it

Fabio Tamburini
Department of Linguistics and Oriental Studies
University of Bologna
Bologna (Italy)
fabio.tamburini@unibo.it

Fabio Vitali
Department of Computer Science
University of Bologna
Bologna (Italy)
fabio@cs.unibo.it

ABSTRACT

Developed using the principles of the Model-View-Controller architectural pattern, FolksEngine is a parametric search engine for folksonomies that allows us to test arbitrary search improvement algorithms by specifying them in three phases: *expansion*, where the original query is converted in multiple ones according to semantic rules associated to the query terms, *search*, executing the queries on a standard folksonomy search engine such as Delicious, and r*anking*, sorting the results according to rules. In this paper we extend our previous studies using FolksEngine and offer a new query expansion algorithms based on Natural Language Processing techniques, and a new view for the results based on Semantic Web technologies. We also describe some tests of the algorithms developed, in order to obtain a clear and effective evaluation of them.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Text analysis*.

H.3.1 [**Information Storage And Retrieval**]: Information Search and Retrieval – *Clustering*.

General Terms

Algorithms.

Keywords

Delicious, FolksEngine, Folksonomy, INFOMAP-NLP, RDF, Web Search Engine, Wordnet.

1. INTRODUCTION

In the last years, much work on document search has been exploring new approaches for search based on the exploitation of semantic data. Usually, this kind of search is performed through content analysis, that is obviously imprecise, or by retrieving and analysing annotations added to documents.

SAC'10, March 22-26, 2010, Sierre, Switzerland.

These annotations could be added by classification professionals, that categorize and describe the documents by using rich and well-organized thesauri, and yet hard to master, or by the collective action of a non-professional audience that describes documents in a free-from-constraint approach, in particular without the need to impose a controlled and unambiguous vocabularies, i.e. by defining *folksonomies*.

Obviously, folksonomies do not generate a consistent structure of concepts associated to the document content: in fact, they introduce well-known problems such as terms' ambiguity or synonymity, that make it hard to offer a fully shareable and synthetic view of the content of the documents.

Literature exists that offers *a posteriori* semantic structuring of the terms that are actually used in folksonomies, and contextualize and explain the terms that are used there in. But it is clear that is not possible to impose a precise semantic to them exactly because of their main advantage: they allow the collective making of complex vocabularies exactly by ignoring the issues of ambiguity and synonymity.

An alternative path to imposing firm and definite semantics to terms used in folksonomies is to associate them to well-defined terms in a predefined ontology, i.e to cluster folksonomic tags around concepts placed in a clear and well-defined semantic structure. Providing such association in an explicit way is a task as wide as ultimately futile, given the number of terms being used and the variety of meanings associated to them.

On the other hand, the identification of clusters of folksonomic tags, that refer to a particular concept, implies the creation of a distance value between folksonomic terms and the specification of a cluster as the set of terms that fall within a certain distance from the query. This can be performed with a number of different algorithms, many of which are well studied in literature.

Starting from the basis introduced in [5], in this paper we present the architecture of an application prototype, *FolksEngine*, that we use to develop and test in an integrated manner all sorts of different clustering algorithms in order to verify their different effectiveness. To do that we have created a flexible architecture, that allows to separate the commonalities and the differences of each algorithm we are testing. In particular, we introduce a three-step process, expansion, search and ranking, to which all algorithms must comply with, placing all differences in the individual expansion and ranking steps.

The architecture also made to test our own contribution to such algorithms. Here we also report on the implementation of algorithms bringing to folksonomies a few Natural Language Processing (NLP) techniques, thereby applying well-studied

algorithms originally created for ample document collections to the restricted domain defined by folksonomies.

We finally introduce a particular view for visualizing results, completely based on Semantic Web technologies, such as RDF [8] and OWL [12]: each result returned by the engine is described using entities concerning a particular ontology for folksonomies, expressly developed to address this issue using some Content Ontology Design Patterns [9].

2. EXISTING APPROACHES FOR TERM CLUSTERING

Various approaches can be pursued in order to improve the user's query by enriching it with sophisticated meanings or associations to get better results from search engines.

The first approach uses lexical resources (such as WordNet) and its ontological organization in order to extract concepts related to the query terms. WordNet semantic relations – such as synonymy, implicitly expressed in the synsets definition, hyponymy (type-subtype relation) and meronimy (part-whole relation) – can be successfully used to add to the original query further concepts, directly related to those used by the user. The list of concepts connected to the original query can be expanded also considering measures of semantic relatedness between the lexical elements in WordNet [3].

Another query expansion approach starts from a radically different view: instead of using an existing lexical resource, all the query terms are expanded by using statistical and machine learning techniques on huge quantities of text resources. Words and multi-word expressions can be compared by means of distributional similarity measures and then clustered using various techniques. Examples and explanations of some of these techniques – based on latent semantic analysis and n-gram extraction from large text corpora – can be found in [7] [10] [11] [15] [14].

Though they do not strictly concern query expansion algorithms, some other tools frameworks and techniques have been developed to try to enrich in some way a particular tag set given as input. For example, the *FoLksonomy Ontology enRichment*tool (*FLOR*) [2]performs an automatic semantic enrichment of folksonomies, without any user contribution, using a three steps procedure that guarantees a lexical analysis, a thesaurus-based semantic expansion and a final semantic enrichment through ontologies.

Our goal in this work, for what concerns the query expansion issue, is to explore in particular the theoretical proposal made in [14] using its implementation, the package INFOMAP-NLP[1], as an NLP method to expand the user's queries.

3. FOLKSENGINE

In this section we introduce FolksEngine[2] the search engine framework we developed to exploit clustering methods for folksonomic tags. The basic principles of our framework depend on both design and theoretical aspects.

First of all, the flexibility of the infrastructure represents the main feature we want to guarantee. We did not want to focus on a specific clustering algorithm but we wanted to develop a framework that allowed us to test a number of clustering algorithms for folksonomies in very different ways, in order to understand their efficacy in each particular context considered.

FolksEngine allows adding easily new algorithms simply by extending the appropriate internal framework structures, without modifying the application core. Moreover, it is be able to expand the current knowledge base with new folksonomic repositories and ontological thesaurus, in order to have new and up-to-date data. Clearly, this knowledge base enrichment guarantees, every time we extend the knowledge base, a much more accurate analysis and evaluation of each tested algorithm.

Each algorithm need to be organized around the following three steps:

- the user's query is expanded into multiple keywords;
- the new keyword set is given as input to a general folksonomic search engine, generating multiple result sets;
- the result sets are collated, filtered and sorted according to a particular ranking function.

3.1 Clustering algorithms

The first clustering algorithm we developed is based on the semantic expansion of the user's query, according to one of the semantic relations available in the WorldNet thesaurus, trying to avoid ambiguity among terms in some way. A possible simplest solution to solve the term ambiguity could be to specify the query as a "keytag:keyword" pairs, in which the first term represents a sort of category for the latter, avoiding completely the common space-separated keywords form. Through this mechanism, we expand the query *t:w* as *t:w*, *t w*, *t' w*, *t" w*, … , $t^n w$, where *t'* … t^n are terms semantically related – through synonymy, hyponymy, hyperonymy, semantic distance, etc. – to *t*.

Another mechanism we developed in our framework concerns the use of lexical resources based on a formal ontology: not just WordNet, but also FrameNet[3], PropBank[4], and so on. Using these kind of resources, we can navigate the lexical relations among terms coded into the ontology, finding concepts somehow related to the user's query.

Some of the basic relations cited, such as synonymy and hyponymy, have already been experimented in [5]. Clearly, more complex metrics – for instance, based on semantic relatedness between words and concepts (e.g., the lowest super-ordinate in taxonomic relation or overlapping measures between word definitions) – can be profitably used to identify concepts that are related to the terms of the original query.

Hand-made lexical resources can be very useful for these tasks, but, unfortunately, the availability of such resources is very limited: they are often restricted to the most common languages, mainly English, and often their coverage can not be adequate to guarantee a complete and useful term expansion of the user's query. A lot of terms occurring inside user's queries can contain ambiguities between words and proper names (e.g. bush/Bush, Paris Hilton/Hilton hotels, etc.). The problem here is that methods heavily based on such resources cannot handle this kinds of ambiguities.

Yet, finding words that are related together by some linguistic relation or that exhibit similar distributional behaviour is a field of research in NLP that has been actively explored in the last few years.

[1] http://infomap-nlp.sourceforge.net/.
[2] http://folksengine.web.cs.unibo.it/.
[3] http://framenet.icsi.berkeley.edu/.
[4] http://verbs.colorado.edu/~mpalmer/projects/ace.html.

A viable solution is the use of large amount of real text as statistical evidence of linguistic phenomena: starting form a large amount of text corpora, such as the British National Corpus for English[5] or the CORIS/CODIS for Italian[6], it is possible to apply statistical and machine learning techniques in order to identify the words that exhibit a similar distributional behaviour of the user's terms.

One of these approaches, largely used by the NLP and information retrieval communities, is grounded on the Latent Semantic Analysis, firstly introduced by Deerwester, et al. [4]. The occurrence of each word is represented as a vector of words co-occurring with it in the same context. All vectors are thus collected in a [word x word in context] matrix A. This is a huge matrix, because each original vector spans through the entire lexicon of the target language. Applying a singular value decomposition technique to A we can map each vector into a subspace, called the WORD-SPACE, with reduced dimensionality *k*, keeping most of the original distributional information.

$$\mathrm{sim}(w_1, w_2) = \frac{w_1 * w_2}{\|w_1\| * \|w_2\|} \quad (1)$$

Each word is then mapped into a *k*-dimensional vector in order to efficiently compare it to other word-vectors to find similarities using, for example, the cosine distance refSim – where *w1*w2* is the euclidean scalar product and ||*w*|| is the norm of the vector *w*. The closer the vectors in the *k*-dimensional space are, the more the distributional behaviours of the two words are similar. So, appropriate clustering methods can identify sets of similar words.

Starting from this standard technique, Widdows [13] studied the interesting properties of this WORD-SPACE, adding the possibility of using logical connectives inside user queries. The notion of distributional similarity between words has been extended with the *negation*, in terms of vector orthogonality, and *disjunction*, as linear sum of subspaces. Since the same logical operators on vectors were used by Birkhoff and von Neumann in the 1930s to describe the logic of a quantum mechanical system, these logical operators are called *quantum connectives* and the system as a whole is called a *quantum logic*.

This quantum connectives are implemented inside the INFOMAP-NLP package, allowing us to write very specific queries. For example the query "*rock NOT band*" allows for the retrieval of the word most similar to *rock* when it refers to "a lump or mass of hard consolidated mineral matter" and not to "a genre of popular music". We have extensively used these methods inside our framework for what concerns the query expansion phase. The results will be presented in Section 4.

Moreover, elaborating these methods as proposed, for example, by Widdows and Dorow [15], we can build small graphs for each target word, showing the most similar words in a structured way and showing also the different senses the target word can have in real texts.

3.2 The implementation of FolksEngine

The core of FolksEngine is composed of a set of cooperating classes that take care of different aspects of the search and retrieval in folksonomies.

Each extension, implementing a specific clustering algorithm, is obtained by extending the framework core classes. Such user-defined classes contain only the specific sub-algorithm, while the framework deals with the relationships among classes and with their interactions. The core classes of the framework are organized following the Model - View – Controller pattern.

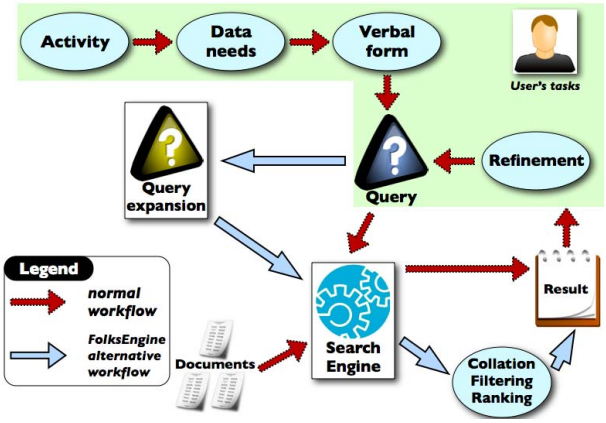

Figure 1. An alternative workflow for retrieving documents through folknsonomic queries.

As shown in Figure 1, our framework describes a classical information retrieval flow with a particular variant: the introduction of the expansion of the user's query. Algorithms to expand queries – e.g., using semantic relations, clustered similarity, etc. – are defined by the programmer by extending a core class. Both the application flow and the semantic distance algorithm (used for ranking the search results) are controlled by particular classes, easily extendible depending on the particular needs.

Our framework also provides facilities for using WordNet in any part of the algorithms. The entire Wordnet thesaurus database is integrated using WordNet Sql Builder2, and it is accessible directly from the framework. This is especially useful for the query expansion.

Obviously, it is possible to improve the framework, in order to access other dictionaries, by extending the class concerning the database handling of classes and term relations.

We have created a search engine accessing directly Delicious. The HTML pages returned by Delicious are cleaned and transformed into a well-formed XML document, and it is then possible to use XPath to retrieve the data sought.

The framework controller listens for requests from the user and dispatches them to the framework model. Once retrieved, the results are returned to the controller and, finally, they are shown by the viewer. Thus the typical process execution is "read the query, expand it, execute the search, rank the result".

3.3 RDF view for query results

The results returned by the first FolksEngine version were written in a simple HTML document. Though they are very comprehensible by humans, who watch them through a browser and give easily a meaning to each string they read, the real semantics for those data are completely hidden behind the document structure. For this reason, the particular format used here is not the best way to return meaningful data for machines, that are not able to rightly interpret the implicit semantics of the result document. Trying to describe more precise and machine-processable data, in the past years the World Wide Web

[5] http://www.natcorp.ox.ac.uk/.
[6] http://sciling.dslo.unibo.it/accesso_coris_eng.html.

Consortium developed a set of specifications with the intent to allow machines to easily understand data and the relations among them in a Web context.

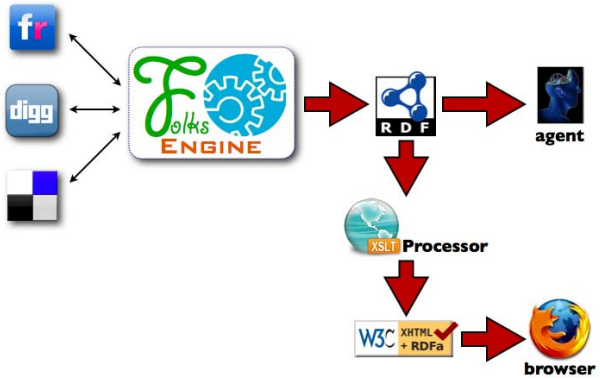

Figure 2. The general process of returning results performed by the FolksEngine. Depending on the kind of client that queried it, two different output are returned: an RDF document, easily readable for machines, or an XHTML +RDFa one, readable for human being too.

Within the set of these specifications, RDF [8] represents the standard models for data interchange on the Web:it is based on *statements*, each of them identified by the triple *subject*, *property* and object. Taking into consideration that, in order to return more machine-processable results, we developed a new particular view for our engine based on RDF. All the RDF triples involved in the answer refer to a particular OWL ontology[7] developed to address this issue.

This ontology takes inspiration from an interesting article by Thomas Gruber [6] in which are addressed some design issues for the development of ontologies describing folksonomies. Taking into consideration that work, the Ontology Design Patterns [9] for the design, we are able to describe semantically all the results returned by the FolksEngine.

Moreover, using the same view, we can offer two different kind of outputs, depending on the client that queries the engine, as shown in Figure 2. After retrieving results from (possibly) different folksonomic search engines and after re-ordering them, an RDF document is produced as output. Because of its machine-processability, it can be interpreted easily by general agents and it can also be shown by browser as a standard XHTML document.

Obviously, the default output of this view is an RDF document, made using the ARC2 library[8], that is easily readable by machines. To obtain a human-readable output too, we need to apply a specific XSLT document, specified through a process instruction, to the RDF output document. It is important to note the latter transformation, that is performed automatically by browsers, results in a well formed XHTML document in which we embed all the semantic assertions by using RDFa [1].

[7] http://www.essepuntato.it/2009/07/folksonomy.
[8] http://arc.semsol.org/.

4. EVALUATION

In order to understand the quality of the evaluated outcomes of the algorithms developed, we need to describe how Delicious performs a search.

Delicious uses "AND" as the default operator. This means that if we search documents using "fruit apple" as query, we will receive all the documents that have both these tags or that contain both the words in the text content. Note that, as default, Delicious looks for documents not only via tags but it also tries to find the query terms in the document text content.

Since looking for documents analysing their text content in respect to the query could be misleading, Delicious gives us the possibility to restrict the search to tags only, just putting the flag "tag:" before the terms you specify in the query. Querying "tag:apple", we ask Delicious to look for all the documents that have "apple" as tag, and not in the content.

Even if the use of such flag could help users to make more precise queries, in many cases we can encounter some problems concerning the term ambiguity. For example, through "tag:apple" we can find both documents concerning the fruit and the computer company. A possible way to solve this situation is to build the query by adding a tag, for example "tag:fruit". Of course, we will not retrieve documents with similar tags such as "tree" or "food".

Obviously, to address this last issue an an automatic way, we developed four different query expansion algorithms, following the principle "keytag:keyword" introduced in Section 3:

- using Wordnet, we expand the *keytag*with the hypernyms, hyponyms and synonyms – the query "fruit:apple", using the hypernyms for "fruit", is expanded into "fruit apple", "reproductive structure apple", "product apple", "production apple", "aftermath apple" and "consequence apple";
- through INFOMAP-NLP, we get the words that are in relation with the *keytag* according to the distances defined within the algorithm – the query "fruit:apple" is expanded in "fruit apple", "vegetable apple", "herbs apple", "peas apple", "vegetables apple", "beans apple", "bread apple", "potatoes apple", "meat apple", "butter apple", "rice apple", "cheese apple", "seeds apple", "cooked apple", "chicken apple".

The rank calculation for these algorithms is, at the moment, rather simple: taking into consideration the set *T*of all the terms obtained after the expansion, we sum 10 to each result every time a term matches with a tag in *T*.

Starting from these assumptions, we made seven queries, always using ambiguous terms. We performed them using six different algorithms: the four ones previously explained and the one developed by Delicious, trying to query it using "tag:*keyword*" and "tag:*keytag*tag:*keyword*" respectively. All the results are summarized in Table 1.

As shown, all the Wordnet-based algorithms are not as good as the simpler "tag:*keytag*tag:*keyword*" returned by Delicious. On the other hand, the INFOMAP algorithms increase the average precision of results. Even if these are preliminary tests only, they tend to show the effective power of the INFOMAP-NLP package. Moreover, note that, even if it is not shown in Table 1, the recall of the INFOMAP algorithm is better than the Delicious (with keytag) one, because the former returns much more results for the query.

Table 1. The results for all the tests performed. Taking the first 20 results returned by the FolksEngine for each algorithm, the precision for each test is calculated dividing the relevant results by 20. The "Precision" in the last column refers to the average precision measure of each query (a = "fruit:apple", b = "animal:mouse", c = "animal:computer", d = "bible:genesis", e = "band:genesis", f = "organization:police", g = "band:police").

Algorithm	a	b	c	d	e	f	g	Precision
Delicious (without *keytag*)	0	0	20	4	6	19	0	0.278
Delicious (using *keytag*)	18	16	20	20	20	15	7	0.828
Wordnet hypernyms	0	15	17	19	8	19	0	0.557
Wordnet hyponyms	6	17	20	20	10	15	3	0.65
Wordnet synonyms	20	19	18	20	18	11	0	0.757
INFOMAP	20	17	18	20	20	16	7	0.842

5. CONCLUSIONS AND FUTURE WORKS

In this paper we have introduced a flexible PHP framework for folksonomy-based document search.

Our ongoing studies concern two different issues. First of all, we are carrying on the development of a systematic approach for the evaluation of different clustering algorithms, in order to prove in a concrete way the quality of the search results returned by each algorithm. Moreover, we are implementing and testing other (more sophisticated) algorithms for query expansions and result ranking.

An alternative methodology for driving the terms' expansion that we plan to implement in our system relies on a dependency-parsed corpus [7]. Triples (word, syntactic relation, word) are extracted and various statistical and stochastic measures are computed on them in order to define a similarity measure between two words. Again, a clustering algorithm relying on such similarity measures can group words exhibiting similar distributional and syntactic behaviour.

All the experiments presented in this paper that require the use of large corpora were based on the *BNC*, a 100-million-word written and spoken corpus of British English developed at the beginning of the 90s. The use of real text, instead of hand-made lexical resources can capture also information from real life events or situations, often sources of ambiguous queries, allowing a certain kind of efficient disambiguation possibilities. In order to obtain reliable results from these operations we need a huge amount of recent real texts. That is why the next experiments we aim to perform will be based on the *ukWaC*, a 2 billion-word corpus of British English built recently – and available at http://wacky.sslmit.unibo.it/ – by collecting documents from the Web.

As outlined before, the research in this field of NLP community is very active, and a lot of alternative methods and techniques can be devised and successfully applied to the query expansion task.

Apart from these NLP aspects, we are improving the ontology model for describing folksonomies and we are exploring the possibilities to associate the results returned by the engine with other similar data, for example by linking them to the Linked Data graph.

REFERENCES

[1] Adida, B., Birbeck, M., McCarron, S., Pemberton, S. (2008). RDFa in XHTML: Syntax and processing. W3C Recommendation. World Wide Web Consortium. http://www.w3.org/TR/rdfa-syntax/.

[2] Angeletou, S. (2008). Semantic Enrichment of Folksonomy Tagspaces. International Semantic Web Conference ISWC'08, Doctoral Consortium. Karlsruhe, Germany.

[3] Budanitsky, A., Hirst, G. (2006). Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Computational Linguistics, 32 (1), 13-47.

[4] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41, 391-407.

[5] Di Matteo, N. R., Peroni, S., Tamburini, F., Vitali, F. (2009). A parametric architecture for tags clustering in folksonomic search engines. It will be presented during the 9th International Conference on Intelligent System Design and Applications (ISDA'09). Pisa, Italy.

[6] Gruber, T. (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.html.

[7] Lin, D. (1998). Automatic retrieval and clustering of similar words. In Proceedings of the 17th international conference on Computational linguistics, 768 – 774. Montreal, Canada.

[8] Manola, F., Miller, E. (2004). RDF Primer. W3C Recommendation. World Wide Web Consortium. http://www.w3.org/TR/rdf-primer/.

[9] Presutti, V., Gangemi, A. (2008). Content Ontology Design Patterns as practical building blocks for web ontologies. In Proceedings of ER2008. Barcelona, Spain.

[10] Purandare, A., Pedersen, T. (2004). SenseClusters - Finding Clusters that Represent Word Senses. In Proceedings of the HLT-NAACL 2004: Demonstration Papers, 26-29. Boston, MA, USA.

[11] Sinha, R., Mihalcea, R. (2007). Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity. In Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007), 363-369. Irvine, CA, USA.

[12] W3C OWL Working Group (2009). OWL 2 Web Ontology Language Document Overview. W3C Working Draft. http://www.w3.org/TR/owl2-overview/.

[13] Widdows, D. (2003). Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, 136 – 143. Sapporo, Japan.

[14] Widdows, D. (2004). Geometry and Meaning. CSLI Publication.

[15] Widdows, D., Dorow, B. (2002). A Graph Model for Unsupervised Lexical Acquisition. In Proceedings of the 19th International Conference on Computational Linguistics, 1093-1099. Taipei, Taiwan.