Lints, T. (2008). Let AI learn from Web 2.0. Tag co-occurrence based text categorization as an example. In *Artificial Intelligence and Soft Computing (ASC 2008)*. ACTA Press. 5 pages.

T. Lints, "Let AI learn from Web 2.0. Tag co-occurrence based text categorization as an example," in *Artificial Intelligence and Soft Computing (ASC 2008)*, ACTA Press, 2008. 5 pages.

```
@inproceedings{Lints08_Tags,
  author = {Taivo Lints},
  title = {Let {AI} Learn from {W}eb 2.0. {T}ag Co-Occurrence
based Text Categorization as an Example},
  year = {2008},
  booktitle = {Artificial Intelligence and Soft Computing (ASC
2008)},
  publisher = {ACTA Press},
  note = {5 pages}
}
```

LET AI LEARN FROM WEB 2.0. TAG CO-OCCURRENCE BASED TEXT CATEGORIZATION AS AN EXAMPLE

Taivo Lints
Research Laboratory on Proactive Technologies
Tallinn University of Technology
Ehitajate tee 5
19086 Tallinn, Estonia
email: taivo@taivo.net

ABSTRACT
The paper draws attention to the potentially high value of user-generated web content from the viewpoint of Artificial Intelligence. Web 2.0 sites can be used as a quickly growing base of knowledge to learn from automatically. To motivate researchers to make use of that data source, a demonstration is provided about the ease of obtaining and using tag co-occurrence data from a social bookmarking site. The described demo uses this derived info to categorize texts into predefined categories and, according to subjective evaluation, does so sensibly well. Though, being just a simple demo case for raising AI researchers' interest in Web 2.0 data usage, it is definitely not meant to be comparable to industry grade text categorizers in its current state.

KEY WORDS
Artificial Intelligence, Knowledge Acquisition, Text Categorization.

1 Introduction

The last decade has seen remarkable advances in World Wide Web technologies, and especially in the ways the web is used. The current stage is occasionally called Web 2.0, which can roughly be defined as "a trend in the use of World Wide Web technology and web design that aims to facilitate creativity, information sharing, and, most notably, collaboration among users" [1]. Examples of Web 2.0 applications include wikis (especially Wikipedia), various social bookmarking sites (e.g. del.icio.us), blogs, and content aggregators that automatically merge data drawn from other sites through RSS feeds or otherwise. The term Web 2.0 has already become a "buzzword" as it is used frequently with the intention to impress the audience rather than to convey clear ideas. This obviously creates negative emotions towards the concept in more serious researchers. However, this negative attitude should not be allowed to carry over to the underlying principles and technologies of Web 2.0.

The main motivation of this paper is to draw more attention to the huge potential of easily accessible collaboratively created content for Artificial Intelligence. That content provides for near future AI systems a very valuable source of knowledge to learn from. The challenge is how to exploit that source efficiently, and the first steps towards overcoming that challenge are to make enough AI researchers aware of the problem and possibilities, and to motivate them to work on it.

One way for emphasizing the potential of Web 2.0 to AI would be creating a thorough overview of possible existing data sources and providing a lot of ideas for extracting useful knowledge from them. A notable step in this direction is a paper by Damme, Hepp, and Siorpaes [2] which lists many web resources and methods that could be used together for automatically building large ontologies. But Web 2.0 resources have many other potential benefits for AI in addition to the help in ontology building, and hopefully more papers will appear soon that both extensively review the known possibilities and propose new ones.

Current paper takes a different approach and presents a simple prototype of text categorization system that uses tag co-occurrence data from del.icio.us. The goal was not to create as good as possible text classifier, but instead to show how easily the Web 2.0 data can be used and that even the crudest programs based on that data are already quite functional. Although the parts of the classification system described in this paper are more or less widely known, their specific combination that makes up this method is, at least to my knowledge, novel.

2 Text Categorization

Text categorization is the task of assigning predefined categories to given text documents. In the case of single-label categorization a document can belong to one category only (e.g., 'this text is about "technology"'), while multilabeling allows for several categories per text (e.g., 'this text is about "technology", "computers" and "society"').

As described in [3], the field of automated text categorization was until late 1980s mostly about knowledge engineering – expert knowledge was manually encoded into classification rules. Then machine learning quickly took over and has continued to dominate ever since. In machine learning approaches the text classifier is automatically built by learning from a set of preclassified documents.

The method proposed in this paper is somewhere in between the knowledge engineering and machine learning. What is being learnt automatically is not some features of preclassified texts, but relations between words, i.e., a very simplistic ontology is automatically created. The word sets themselves, being actually tags extracted from a social bookmarking site, are on the other hand manually constructed by the collective of human bookmarkers. The texts of web pages that bookmarkers have tagged are not taken into account in this simple prototype (otherwise the system would mostly learn features of preclassified texts just as the rest of machine learning methods do). Then a very simple and crude classification algorithm is constructed that uses this automatically extracted tag co-occurrence data.

3 Social Tagging Systems and AI

The websites that support collective tagging of resources (bookmarks, photos, etc.) have only appeared in recent years, and as it took some time for their content to accumulate to a notable size, it is no wonder the scientific use of that data is only starting to become a topic of interest.

Tag data from social bookmarking sites is up to now mostly viewed as useful for Semantic Web, in the sense that while authors of web pages are typically not interested in adding metadata to their pages, social bookmarking sites provide enough incentive for ordinary web users to add that metadata themselves. Although the result is neither very rigorous nor detailed (tags are only applied to pages, not elements of pages), it is a lot better than having no metadata at all and might help considerably in information retrieval from the web. Many papers indeed deal with using tag data from social bookmarking sites to develop better web search methods, e.g. [4] and [5]. Although very useful, their general idea is quite straightforward in the sense that tags are used in the same context they were initially intended to – as manually added informational markers of specific web pages. What is being improved is mostly just the speed with which users can find the web pages they are looking for.

A somewhat more innovative (and more relevant to this paper) use of tags is analyzing their co-occurrence and using this information to build ontologies. In general this is what my example system also does. However, most ontology-building papers tend to be concerned with the low quality of ontologies built only on co-occurrence data (e.g., [6] and [2]) and thus might frighten the readers with the potentially huge amount of work necessary to get decent result. My example, on the other hand, tries to demonstrate that even an extremely simple approach can already be quite useful in practice. This should encourage researchers to get started with building simple functional prototypes. These could of course later be iteratively developed into considerably better systems at will. Inspiration for further developments can be found, for example, in [2].

4 Analyzing Tag Co-Occurrence

As a source of tag data I chose del.icio.us, because it is a very popular bookmarking site (thus containing a lot of data), it's range of topics is very wide (as opposed to sites built for tagging more specialized resources), and it provides relatively easy access to the data (at least to some part of it).

I selected a small subset of del.icio.us users (3573 in total), and automatically downloaded the newest postings (bookmarked URLs with added tags) of each of them using RSS feeds provided by del.icio.us. Automatization scripts, as well as all the following analysis programs, were written in Python programming language. As it would be impolite to put a load on del.icio.us server by downloading everything as fast as possible, I left on average 10 second pauses between each request. Also, I made sure that if the server would give an error message for some reason, my program would stop sending the requests. However, no such error was raised during the download.

Next, the received postings were processed. All URLs were collected into a list of unique URLs (the number of which turned out to be 236 572), and all tags attached to a certain URL by different users were associated with that URL in this list. An example of list entry: 'http://brandsoftheworld.com/': ['Design', 'Graphic', 'Logo', 'brands', 'logos', 'design', 'resources', 'logos', 'resources', 'vector', 'logos', 'design', 'Resource', 'vectors', 'logos'].

As can be seen, the multiple occurrence of a tag (i.e., the URL is tagged with the same word by more than one user) is preserved for later analysis. E.g., there are 4 instances of 'logos' in the given example. To keep the system simple, URLs are not analyzed for similarity, and thus the same page bookmarked by some users with the address `http://www.brandsoftheworld.com/` is considered to be a different URL. Also, the information about individual user behavior is left behind at this point – only aggregate data is used.

To calculate the relatedness of the tags to each other (in the sense of co-occurrence, not necessarily synonymity), I used cosine similarity as it has already been shown to give useful results, e.g. in [7] and [8]. In principle, cosine similarity just finds the angle between two vectors that are derived from tag data:

$$\theta = arccos \frac{A \cdot B}{\|A\|\|B\|}$$

There are several ways to create these vectors, but I chose one of the simplest, described in [7]: "Tags are aggregated into *tag vectors*, for which the index $v_{t_l}\,[O_m]$ is equal to the number of times that the tag t_l annotates the object O_m". As the number of unique URLs in my case is relatively big (236 572) and that determines the length of vectors, then using full vectors would have been computationally prohibitive. Luckily, the vectors were sparse and allowed for fast computational manipulations. Only nonzero val-

ues were saved, e.g. "towerdefense", {31648: 4, 142178: 1}, which says that the word 'towerdefence' is used four times for tagging URL number 31648 and one time for URL number 142178. The numbering of URLs (i.e., the correspondence between vector index and URLs) is arbitrary, except that an URL is guaranteed to have the same index in every vector.

For simplicity, no semantic analysis was done in the process of tag vector creation. This means that for example tags 'dogs', 'dog' and 'Dog' were all treated as being different. Even though in this case it is likely all these tree words were used in the same meaning by site users and uniting them into one tag by the analyzer would have made some sense, it is not always so easy to be sure if the 's' in the end of the word signifies plurality and if capital letters have any importance to the meaning of the word under analysis. This is not to say semantic analysis should not be used, just that it was not feasible to apply it in this intentionally simple prototype system. The number of literally (as opposed to semantically) unique tags in the dataset turned out to be 69 530.

Strictly speaking, cosine similarity finds the angle between given vectors. Thus cosine similarity value 0 would actually mean the vectors overlap (fully, for typical encodings). To make the value more intuitively understandable, I converted it with the formula

$$intuitive\ similarity = -(angle - \pi/2)$$

where angle is measured in radians. Value 0 of ("intuitive") similarity now means the two tag vectors are NOT similar (in the context of this approach), and $\pi/2$ would be maximal similarity. An example of resulting similarity scores would be: "inkpen", {'inkpen': 1.5708, 'paper': 0.0492, 'gadgets': 0.0072, 'pen': 0.1724, 'scroll': 0.3063}.

5 Using Co-Occurrence Data for Text Categorization

Analyzing tag co-occurrence in itself is not a novel thing to do. However, we will go further and use the found similarity scores for text categorization. As the goal of current prototype is simplicity, not accuracy, we will again use a somewhat primitive, but still functional, approach.

We assume that the words used to denote categories also exist in the tag data analyzed previously. To estimate how related some text is to a given category, we will take the previously found similarity score vector and use the following formula:

$$relatedness = S1 * C1 + S2 * C2 + \dots$$

where S1 is the similarity score of the first word in similarity score vector, C1 is the count of that word in the text under analysis, etc. For example, if the category is 'inkpen' and text is "all the gadgets were wrapped in paper", the relatedness score would be $1.5708*0+0.0492*1+0.0072*1+0.1724*0+0.3063*0 = 0.0564$ (see the end of previous section for the similarity score vector of 'inkpen'). Here, again, word counting uses literal comparison for simplicity, e.g. when counting the occurrences of the word 'paper' no attention would have been paid to instances of 'Paper' in the text.

This method is strongly dependent on the popularity of the category label in the bookmarking community. For example the word 'car' is very widely used as a tag, which also means it co-occurs with very many other tags, and the similarity score vector for it is very long (i.e., we know it's similarity to very many other words). Thus the relatedness formula would potentially have very many $S * C$ terms and the sum could be large. On the other hand, if the category label is not widely used by taggers, we would know its similarity score to only a few other words, and the sum that gives relatedness score would inevitably be small, no matter how related the text is to that category. Probably the simplest way to reduce that problem is limiting the number of terms in the sum. E.g., we can take the 70 most similar words to given category and use only them in the relatedness formula. If the similarity score vector is shorter than 70, we would just have to skip calculating the relatedness of text to that category due to lack of information. What would be appropriate limit depends on our data set and goals. Obviously, this approach is very crude. Even the dependence on tag popularity is not fully removed – the top 70 (or whatever other limit) of a very long vector contains only high similarity score words, while top 70 of a length 75 vector has similarity scores down to almost zero. Also, the length of text itself strongly affects word counts and thus the relatedness score. Still, this approach is good enough for our purposes here as we are not going to compare relatedness scores of different texts to each other.

Now, to categorize a text into one of given categories, we calculate the relatedness of the text to each of those categories and just pick the category with the highest relatedness score. Here the result is only one category per text, which is good when the goal is strict single-label categorization. But as we are using continuous relatedness scores, the method could potentially be extended to multi-labeling by, for example, accepting all categories whose relatedness score is above some special value (however, this value should then depend on text length, or the relatedness measure itself would have to be changed to become independent of the size of the text).

6 Evaluation

Evaluating text categorization is not a straightforward procedure. As noted in [3], human experts often tend to disagree with each other about what category a text belongs to. The evaluation of machine learning based text categorizers is often done on certain subsets of the so-called Reuters collection, which contains human labeled news stories. But, as [3] explains, comparing different classifiers even on the same text collection requires strict consistency in experimentation procedures, including using exactly the

same subset of the collection, exactly the same split between training and test set, and exactly the same evaluation measures with same parameters. The lack of such consistency renders many published results not fully comparable with each other. As of evaluating the simple categorizer described in current paper, the aforementioned approach is not even applicable, in the sense that this categorizer cannot be made to learn from a given training set of texts, but instead uses external knowledge derived from social bookmarking sites. Thus I decided to be satisfied with a subjective evaluation only, especially given the goal here is not to propose the most accurate categorizer, but to demonstrate the usability of Web 2.0 data by AI related applications.

The set of texts for this subjective evaluation was acquired by looking at the list of most popular (at some point in December 2007) postings made in November 2007 at digg.com (another Web 2.0 site where people can submit links to news stories and other web resources, and then these links are ranked by other users). All the popular pages (those that were still online) were then downloaded and processed to extract human-readable text. This resulted in 2534 text files.

Categorizer was then run on the text collection. An example of the set of categories used for testing is 'nature', 'politics', 'religion' and 'technology'. Trying different limits on how many most related words of each category to use (as described in the section "Using Co-occurrence Data for Text Categorization"), the limit of 50 gave acceptable results. While obviously far from perfect, they were most certainly also far from random, and actually surprisingly sensible given the simplicity of the categorizer and the small tag data set that was used (only a tiny subset of what is available on the web). I also showed the results to an unbiased semantic web researcher whose opinion on the quality of the results was similar to mine (he was actually the one who suggested publishing the results, as for me it was just a toy project that I was playing with).

Due to space restrictions of the paper it is not possible to present here the full experimentation results, which consist of listings of page titles (as provided by digg users to describe their posts) divided to categories that were provided to the system by me. Interested readers can obtain them by request. A tiny example of descriptions of the top 12 stories under category 'nature' is given in the following list. However, this is far too short to be of any use in assessing the quality of the categorizer.

1. Surfer Dude Stuns Physicists With Theory of Everything
2. Unlocking the Benefits of Garlic
3. Hwy Patrolman arresting and tasing man for breaking law which doesn't exist
4. Voters Speak: No More State Income Tax and Criminal Marijuana Penalties
5. 11 phenomenal images of earth
6. Helping Coral Reefs That Help People
7. Unveiling the winners of the Oxygen contest for wallpapers
8. Google Earth Heading Towards Extinction?
9. Hey Digg: Your "Upcoming" Model Blows, Here's How to Fix It
10. This is the greenest building ever built!
11. Better Gmail Firefox Extension for New Gmail
12. You Silly Boys: Blondes Make Men Act Dumb

The number of texts in each category when using limit 50 is as follows:

nature (100)
politics (485)
religion (182)
technology (1552)
uncategorized (67)

where 'uncategorized' contains texts with zero relevance to any other category. To make sure it really was worth to use the tag co-occurrence data in this categorizer, I also let it sort the texts without that data, i.e., by only counting the occurrences of category labels ('nature', 'politics', 'religion', 'technology') in the texts. This resulted in notable degradation of quality – very many texts remained uncategorized even though they would have fit to some of the categories:

nature (258)
politics (157)
religion (64)
technology (468)
uncategorized (1439)

Taking into account all the results of the experimenting I did, it is possible to say, at least subjectively, that the described simple categorizer gave quite sensible results and that the use of tag co-occurrence data was an important factor in this.

7 Conclusion

As demonstrated, it requires relatively little effort to develop a system that uses data from Web 2.0 sites and provides useful functionality. Uncountable possibilities exist to enhance the described simple text categorizer, many of them very obvious. But more importantly, an AI system that learns from the data available from Web 2.0 sites will have a steady input of new information even without any

further development efforts, because the large user communities of those sites are continuously producing additional data without any cost to the AI system developer. Thus, making better use of these valuable sources of information should be considered an important direction of near-future research in AI.

8 Acknowledgements

Thanks to Leo Mõtus for providing me the possibility to do highly unconstrained research. Financial supporters of my research include Research Laboratory on Proactive Technologies and the Department of Computer Control in Tallinn University of Technology, Estonian Information Technology Foundation, Estonian Doctoral School in ICT, Estonian Ministry of Education and Research, and Estonian Science Foundation.

9 References

[1] Web 2.0, Wikipedia, `http://en.wikipedia.org/wiki/Web_2.0`, accessed on April 11, 2008.

[2] C. Van Damme, M. Hepp, and K. Siorpaes, FolksOntology: An Integrated Approach for Turning Folksonomies into Ontologies, *Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)*.

[3] F. Sebastiani, Machine Learning in Automated Text Categorization, *ACM Computing Surveys*, 34(1), 2002, 1–47.

[4] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su, Optimizing Web Search Using Social Annotations, *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 2007.

[5] G. Begelman, P. Keller, and F. Smadja, Automated Tag Clustering: Improving search and exploration in the tag space, *Proc. of WWW2006, Collaborative Web Tagging Workshop*, Edinburgh, UK, 2006.

[6] M. Chen and J. Qin, Deriving Ontology from Folksonomy and Controlled Vocabulary (poster presentation), *iConference 2008, iFutures: Systems, Selves, Society*, Los Angeles, US, 2008.

[7] P. Heymann and H. Garcia-Molina, Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems, *Stanford InfoLab Technical Report*, 2006.

[8] R. Li, S. Bao, Y. Yu, B. Fei, and Z. Su, Towards effective browsing of large scale social annotations, *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 2007, 943–952.