Instance Classification using Co-Occurrences on the Web

Gijs Geleijnse[1], Jan Korst[1], and Viktor de Boer[2]

[1] Philips Research, High Tech Campus 34, 5656 AE Eindhoven, the Netherlands
{gijs.geleijnse,jan.korst}@philips.com
[2] Human-Computer Studies Laboratory, University of Amsterdam, the Netherlands
vdeboer@science.uva.nl

Abstract. We present a novel unsupervised approach to mapping art-related instances (such as music artists and painters) to subjective categories like genre and style. We base our approach on co-occurrences of the two on the web, found with Google. The co-occurrences are found using three methods: by identifying the search engine counts, by analyzing Google excerpts found by querying patterns and by scanning full documents. Per instance, we use the same co-occurrence-based approach to find its nearest neighbors, i.e. the most related instances. These results can be combined in order to create a more reliable classification. We tested and compared the three methods on two different domains: mapping music artists to genres, and painters to art-styles. The results show that the use of related instances indeed improves the precision of the classification. Moreover, the methods with the lowest Google Complexity perform best.

1 Introduction

We call *Paul Gauguin* a post-impressionist, *Madonna* a pop artist and *Annie Hall* a romantic comedy. Although rarely a complete consensus can be formed about such categories in art, the use of categories is a convenient mechanism to order and browse a collection. Moreover, it can be interesting to find closely related instances, for example to identify the relatedness between *Britney Spears* and *Christina Aguilera*. Such information is valuable for television and music recommenders, since it can be used to identify similarity between items.

We are interested in whether we can automatically find such meta-data using multiple web pages. In this paper we focus on two tasks. On the one hand the classification of instances (e.g. *pop artists*) into categories (Section 4) and on the other hand identifying a distance matrix of related instances (Section 5). We use multiple web pages to compute both a classification of instances into categories and a distance matrix between the instances. In Section 6 we utilize the distance matrix to create a more reliable instance classification. The information found can be used to create an automated folksonomy: a knowledge base where items are tagged using input from multiple users.

2 Problem Description

We want to map a set of instances – artists or items such as movies or paintings – to a set of categories. Given are two sets of instances I_a of size N and I_s of size M. Here, I_a is a set with instances such as paintings, or artists. The set I_s contains categories like movements in art or genres. Also given is a mapping $m : I_a \rightarrow I_s$.
Definition. We call a category $m(b)$ *most appropriate* for b if a domain expert would select $m(b)$ from the set I_s as the category best applicable to $b \in I_a$.
Problem. Find for each $b \in I_a$ the most appropriate $m(b) \in I_s$.

We use co-occurrences of instances in I_a and categories in I_s (e.g. *Johnny Cash* and *country*) on the web to extract such information to compute a preliminary mapping. Additionally, we assume that related instances in I_a often share a category in I_s. We compute distances between instances using web co-occurrences in order to identify the nearest neighbors for each instance. The preliminary mapping of each artist and its nearest neighbors are combined in a final mapping m.

3 Related Work

Early work on relation identification from the web can be found in [3]. Brin describes a website-dependent system to identify hypertext patterns that express a given relation. For each web site, such patterns are learned and explored to identify instances that are similarly related. SnowBall [1] is a successor of Brin's system with an embedded named-entity recognizer. The idea of extracting relations using patterns is similar to one of the methods presented here. However, in Snowball the relations gathered are not evaluated.

Cimiano and Staab [7] describe a method to use a search engine to verify a hypothesis relation. For example, if we are interested in the 'is a' or hyponym relation and we have the instance *Nile*, we can use a search engine to query phrases expressing this relation (e.g. *"rivers such as the Nile"* and *"cities such as the Nile"*). The number of hits to such queries is used to determine the validity of the hypothesis. Per instance, the number of queries is linear in the number of classes (e.g. *city* and *river*) considered.

The number of Google *hits* for pairs of terms can be used to compute a semantic distance between terms [6]. The nature of the relation is not identified, but the technique can for example be used to cluster painters. In [19] a similar method is used to cluster artists using search engine counts. In [17], the number of Google hits of combinations of artists is used in clustering artists. In one of our methods, we use the same techniques to obtain these figures.

KnowItAll is a hybrid named-entity extraction system [10] that finds lists of instances of a given class from the web using a search engine. It combines hyponym patterns [12] and learned patterns for instances of the class to identify and extract named-entities. Moreover, it uses adaptive wrapper algorithms [8] to

extract information from html markup such as tables. Contrary to our method, it does not use instances to formulate queries. In [9] the information extracted by KnowItAll is evaluated using a combinatorial model based on the redundancy of information on the web.

In [2] a number of documents on art styles are collected. Names of painters are identified within these documents. The documents are evaluated by counting the number of painters in a training set (of e.g. *expressionists*) that appear in the document. Painters appearing on the best ranked documents are then mapped to the style. De Boer et al. use a training set and page evaluation, where we simply observe co-occurrences.

A document based technique in artist clustering is described in [13]. For all music artists in a given set, a number of documents is collected using a search engine. For sets of related artists a number of discriminative terms is learned. These terms are used to cluster the artists using support vector machines. The documents are obtained in a similar way in our document-based method. However, we restrict ourselves to identifying names of artists and categories in the documents.

4 Three Classification Methods

In this section, we present three methods to classify instances in I_a using web data. The first method is based on analyzing the total numbers of co-occurrences of instances in I_a and categories in I_s on the web. To retrieve this data we use Google [4]. An important drawback of this page count method is that it has a high *Google Complexity*, i.e. it requires many queries to a search engine, $\mathcal{O}(N \cdot M)$. For large sets this can be problematic, since search engines currently allow only a limited amount of automated queries per day [5]. Moreover, the number of hits can fluctuate over time [18], which hampers the reuse of old hit counts. In Sections 4.2 and 4.3 we present two alternative methods that do not suffer from these drawbacks.

We are interested in a mapping m', based on co-occurrences of elements in I_a and I_s. In Section 6 we combine this mapping m' with a distance matrix between related instances in I_a to find a definite mapping m.

4.1 Page-count-based mapping (PCM)

To obtain the mapping m' we perform a Google query `"`b`","`g`"` for each pair $(b, g) \in I_a \times I_s$. Per query, we extract the estimated number of *hits* $\mathrm{co}(b, g)$.

$$\mathrm{co}(b, g) = \text{‘the number of hits for query ”}b\text{”, ”}g\text{” ’}$$

We assume that the order of the terms b and g in the query does not effect the number of hits, thus we assume $\mathrm{co}(b, g) = \mathrm{co}(g, b)$.

This Page-Count-based Mapping (PCM) is simple and intuitive. If we are for example interested in categorizing music artists into genres, we analyze the number of hits to queries for combinations of the names of the artist and each genre.

Assuming Johnny Cash to be a country artist, we expect that more documents contain both the terms *Country* and *Johnny Cash* than *Reggae* and *Johnny Cash*.

Per $b \in I_a$ we could map the $g \in I_s$ with the most hits. However, we observe that frequently occurring categories in I_s have a larger probability to be mapped to any instance in I_a. For example, the query '*Pop*' results in 8 times more hits than the query '*Disco*'. Although we consider *Boney M* as a disco-act, the query *Boney M, pop* gives twice the amount of hits as *Boney M, disco*. This observation leads to a normalized approach, inspired by the theory of pointwise mutual information [14,9].

For $b \in I_a$ and $g \in I_s$, we define a scoring function $S(b, g)$ as follows.

$$S(b, g) = \frac{\text{co}(b, g)}{1 + \sum_{c \in I_a} \text{co}(c, g)} \tag{1}$$

In the denominator we add 1 to the sum of all co-occurrences with g to avoid dividing by 0. Having computed the scores for all pairs, we select a preliminary mapping m' for each $b \in I_a$. Per instance we select the category $g \in I_s$ with the highest score S.

$$m'(b) = \text{argmax}_{h \in I_s} S(b, h) \tag{2}$$

Using PCM we thus need to perform $N \cdot M$ queries.

4.2 Pattern-based mapping (PM)

The Pattern-based Mapping (PM) is based on occurrences of terms in phrases on the web. We observe combinations of terms in phrases that express the relation we are interested in. For example, if we are interested in the relation between music artists (in I_a) and their genres (in I_s), an appropriate phrase that links terms of the two could be *'[artist] is one of the biggest [genre] artists'*.

We can identify these patterns automatically by using a training set of related instances and categories [11,16]. Learning patterns can be done with $\mathcal{O}(N)$ queries.

We use combinations of a pattern and an instance or a category as a query to the search engine [11]. For example, if we have the pattern *"[genre] artist such as [artist]"*, we use *"artist such as"* in queries in combinations with all names of genres and artists. We use this pattern e.g. both for the query *"Country artists such as"* and for the query *"artists such as Prince"*. In the excerpts found with the first query, we identify instances in I_a, while in the results for the second query we search for categories in I_s related to *Prince*.

These queries provide access to relevant data. From the excerpts returned by the search engine, we thus identify the elements of either I_a or I_s to measure the number of co-occurrences of the pairs. Hence, using PM $\text{co}(b, g)$ is defined as follows.

$$\text{co}(b, g) = \text{'number of occurrences of } b \text{ by querying patterns with } g\text{'} + \text{'number of occurrences of } g \text{ by querying patterns with } b\text{'}$$

Using PM we only need $\mathcal{O}(N + M)$ queries. We use the same scoring function $S(b, g)$ as given in (1) to obtain a preliminary mapping as given in (2).

4.3 Document-based mapping (DM)

In the Document-based Mapping (DM) approach we collect the first k URLs of the documents returned by the search engine for a given query. These k URLs are the most relevant for the query submitted based on the ranking used by the search engine [4].

In the first phase of the algorithm, we query all instances in both I_a and I_s and collect the top k documents for each of the queries. For instances in I_a, we retrieve each document using the URLs found by the search engine. We count the occurrences of the categories in I_s (thus the names of the categories) in the retrieved documents for the intermediate mapping m'. From the documents retrieved with a category $g \in I_s$, we similarly extract the occurrences of instances in I_a.

The documents obtained using DM are the most relevant for each element $b \in I_a$. For the artists queried we expect biographies, fan pages, pages of museums, entries in database sites and so on. The categories in I_s (e.g. the genres or styles) mentioned in these pages will most probably reflect the genre of the artist queried.

The co-occurrences function is here thus defined as follows.

$$\text{co}(b, g) = \text{'number of occurrences of } b \text{ in documents found with ``}g\text{''} \text{'} + \text{'number of occurrences of } g \text{ in documents found with ``}b\text{''} \text{'}$$

The co-occurrences of elements in I_a and I_s again are used for an intermediate mapping using the same scoring function.

This method also requires only $\mathcal{O}(N + M)$ queries. However, additional data communication is required since for each query up to k documents have to be downloaded instead of using only the data provided by the search engine.

5 Finding Related Instances

We use the same three co-occurrence-based methods to compute the relatedness between elements in I_a. We consider instances in I_a to be related, when they frequently co-occur in the same context. In Section 6, we use this information in a final mapping m between elements in I_a and I_s.

Per pair $(b, c) \in I_a \times I_a$ we compute the score T, similar to the score S in (1).

$$T(b, c) = \frac{\text{co}(b, c)}{1 + \sum_{y, y \neq b} \text{co}(b, y) \cdot \sum_{x, x \neq c} \text{co}(x, c)} \tag{3}$$

Again, we do not use a majority voting to prevent frequently occurring instances to be strongly related to many other instances.

In PCM we combine the names of two artists into a query and extract the number of *hits*. Using this method this phase requires N^2 queries.

If we use PM to obtain the numbers of co-occurrences of instances in I_a, we can specify the nature of the relatedness. For example, for instances of the class *pop artist*, we can be solely interested in artists who have played together. A pattern such as *"[pop artist] recorded a duet with [pop artist]"* could be suitable for this purpose. This phase of the method consists of $k \cdot N$ queries (with k the number of patterns).

In the documents obtained with the DM method we only expect occurrences of other terms in I_a that are strongly connected with the $b \in I_a$ queried. For DM no additional queries have to be performed in this phase, since we can reuse the documents obtained in the first phase.

6 Combine results in final mapping

We use the assumption that related instances in I_a often share the same category. We investigate whether the use of relatedness between instances in I_a helps to improve to the precision of the mapping m'.

We combine the scores T with the preliminary mapping m' as follows. Per $b \in I_a$, we inspect m' to determine the category that is assigned most often to b and its n closest related instances. We thus expect that the most appropriate category g for b is most often mapped by m' among b and its nearest neighbors.

Per instance $b \in I_a$, we construct an ordered list with b and its n nearest neighbors, $\mathcal{B} = (b_0, b_1, ..., b_n)$ with $b = b_0$ as its first element and $T(b, b_i) \geq T(b, b_{i+1})$, for $i > 0$.

For a final mapping m of instances $b \in I_a$ to a category in I_s, we inspect the most occurring category mapped by m' to b and its n nearest neighbors.

$$m(b) = \text{argmax}_{h \in I_s} (\sum_{c \in \mathcal{B}} I(c, h))$$

with

$$I(b_i, h) = \begin{cases} 1 & \text{if } m'(b_i) = h \\ 0 & \text{otherwise.} \end{cases}$$

If two categories have an equal score, we select the first occurring one. That is, the category that is mapped by m' to b or to the artist most related to b.

7 Experiments

We present two experiments. The first experiment is on the music domain where we map a list of 224 artists to 16 genres [13]. In the second, we construct a list of painters and a list of movements in art using Wikipedia and map the two.

In these experiments, we only evaluate precision. If for an $b \in I_a$ (thus either an artist or a painter) no mapping could be found, we consider it incorrectly classified.

In the first experiment, I_s is the set of all artist names in the list composed by Knees et al. [13]. This list consists of 14 genres, each with 16 artists. The genres mentioned in the list are not all suitable for finding co-occurrences. For example, the term *classical* is ambiguous and *Alternative Rock/Indie* is an infrequent term. We therefore manually rewrote the names of the genres into unambiguous ones (such as *classical music*) and added some synonyms. After collecting the numbers of co-occurrences of artists and genres, we summed up the scores of the co-occurrences for synonyms. Thus, for each artist b the number of co-occurrences with the terms *Indie* and *Alternative Rock* are added to the co-occurrences of b with the genre *Alternative Rock/Indie*.

For the second experiment, we extracted from Wikipedia a list of 1,280 well-known painters from the article *List of painters* and a list of 77 movements in art from *List of art movements*[3]. We tested the performance of the algorithm on the subset of 160 painters who could be extracted from pages describing movements (e.g. from the page on *Abstract Expressionism*). The other 1,120 painters are either not mentioned on the pages describing styles or are mentioned on more than one page. However, when computing similarities between the painters, we take all 1,280 painters into account.For the elements of I_s in this test no synonyms were added. For fairness, we excluded pages from the domain `wikipedia.org` in the Google search queries.

We performed the experiments using each of the methods described to obtain the co-occurrences.

Motivated by the results in [17], for PCM we used the `allintitle` option in the artist classification experiment. We also added the extra term *music* for finding co-occurrences of the artists. Due to the rareness of some of the painters and names of movements, we did not use these restrictions in the second experiment.

For PM, we selected learned patterns for the mapping between the elements in I_a and I_s. For learning, we used instance-pairs outside the test set. For the relation between the instances in I_a, these patterns found were mostly enumeration patterns, e.g. *"including b and"*. The complete details of both experiments and the patterns used in PM can be found on the web page[4].

In Table 1 the performance of the preliminary mapping m' can be found for the three methods ($n = 0$) for both experiments. The experiments show that in general the use of related instances improves the classification (see Table 1 and Figure 1). It is remarkable that the methods with the lowest Google Complexity thus PM and DM perform better than PCM.

We were able to map all artists to a genre. Co-occurrences of genres and artists thus could be found using PCM, PM as well as DM. The latter performs best. With respect to the preliminary mapping, the method with the smallest amount of Google queries performs best. Moreover, DM shows that the first k

[3] www.wikipedia.org Both pages visited in April 2006.
[4] http://gijsg.dse.nl/webconmine/)

	ARTIST-GENRE			PAINTER-MOVEMENT		
method	$n = 0$	best	(corresp. n)	$n = 0$	best	(corresp. n)
PCM	71.4	81.3	(13)	35.0	35.0	(0)
PM	72.2	88.8	(8)	53.8	63.8	(18)
DM	83.9	87.1	(5)	65.0	81.2	(20)
PM-STEMMING				53.2	61.9	(28)

Table 1. Precision (%) without related instances and best precision per method.

pages ranked by Google indeed contain relevant and reliable information for our purposes.

Using DM, only few related artists can be found on the documents visited. This leads to a stable performance for the final mapping when expanding the list of related artists (Figure 1). That is, we only consider artists that co-occur at least once. Contrary to especially PCM, large numbers of n do not deteriorate the precision.

With the supervised music artist clustering method discussed in [13] a precision of 87% was obtained using complex machine learning techniques and a relatively large training set. In [17] a precision of up to 85% precision was obtained using $\mathcal{O}(N^2)$ Google queries. We can conclude that our simple and unsupervised method produces similar results. Moreover, we compute a classification of artists into genres instead of clusters of artists.

Although in the painter-movement experiment the number of categories identified (77) is much larger than in the first one (16), the performance of PM and especially DM is still good. The results of PCM indicate that when the precision of the intermediate mapping is low (35%), the use of related instances does not improve the results. In this experiment we even observe a deterioration of the performance. Here DM clearly outperforms PM. This can be explained by the fact that using PM considerably less painter-movement pairs could be extracted. We expected the recall of PM to increase when applying stemming on the names of movements and the texts extracted [15]. Although the number of pairs extracted slightly increases, the precision does not improve (Table 1).

8 Conclusions and Future Work

We have discussed three alternative methods PCM, PM and DM to obtain co-occurrences of terms using a search engine. These methods are applied to gain a preliminary mapping between instances such as artists or painters and categories such as genres or art-movements. The distance between related instances is used to obtain a more reliable mapping.

The three alternatives used have a different complexity with respect to the number of queries to a search engine. The method using patterns and the one using complete documents are linear in the number of items in the sets of instances

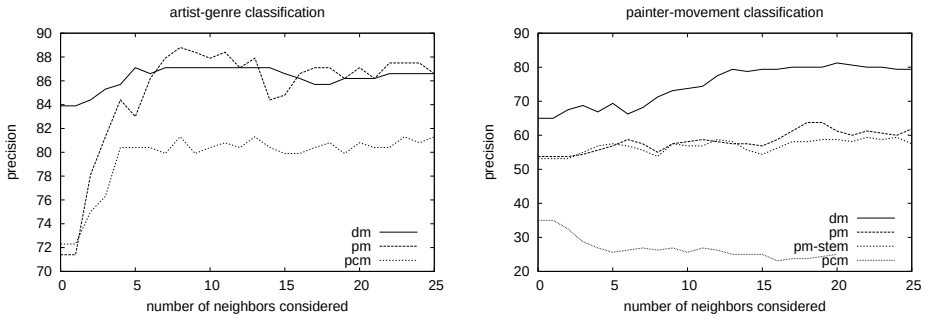

Fig. 1. Precision (%) for classification.

and categories, where the page-count-based mapping is quadratic. This distinction is important for classifying large sets of instances, since search engines allow only a limited amount of automated queries per day.

We can precisely classify artists to genres, where the most efficient methods with respect to the Google complexity perform best. A second experiment consisted of the mapping of painters to their movements. This experiment underlines that the document-based and pattern-based method outperform the query-expensive method based on Google page counts.

We showed that simple and unsupervised methods can be used for a reliable classification. Using related instances indeed helps to improve the classification of instances. The experiments show an increase of the performance in both experiments. However, the Google count based method in painter classification shows that this additional step deteriorates the precision, if the classification is very unreliable.

In future work, we want to further exploit the field of mapping categories to art-related items.

In this work, we assumed that the mapping was functional, i.e. only one category could be assigned to an instance. However, in some tasks multiple categories can apply to an instance. We therefore want to adapt the system such that multiple categories accompanied with a confidence mark can be assigned to an instance. Moreover, methods can be exploited to learn other words related to some category, e.g. with the *tf·idf*-approach [13,14].

In folksonomies users are asked to add tags to an artist or album. These uncontrolled lists of terms characterize the artist. We want to investigate whether it is possible to map artists to such tags as well.

References

1. E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, 2000.

2. V. d. Boer, M. v. Someren, and B. J. Wielinga. Extracting instances of relations from web documents using redundancy. In *Proceedings of the Third European Semantic Web Conference (ESWC'06)*, Budva, Montenegro, June 2006.
3. S. Brin. Extracting patterns and relations from the world wide web. In *WebDB Workshop at sixth International Conference on Extending Database Technology (EDBT'98)*, 1998.
4. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
5. M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in NLP*, pages 563–570, Vancouver, Canada, 2005.
6. R. Cilibrasi and P. Vitanyi. Automatic meaning discovery using Google. http://www.cwi.nl/˜paulv/papers/amdug.pdf, 2004.
7. P. Cimiano and S. Staab. Learning by Googling. *SIGKDD Explorations Newsletter*, 6(2):24–33, 2004.
8. V. Crescenzi and G. Mecca. Automatic information extraction from large websites. *Journal of the ACM*, 51(5):731–779, 2004.
9. D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 1034–1041, 2005.
10. O. Etzioni, M. J. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.
11. G. Geleijnse and J. Korst. Learning effective surface text patterns for information extraction. In *Proceedings of the EACL 2006 workshop on Adaptive Text Extraction and Mining (ATEM 2006)*, pages 1–8, April 2006.
12. M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA, 1992.
13. P. Knees, E. Pampalk, and G. Widmer. Artist classification with web-based data. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, pages 517–524, Barcelona, Spain, October 2004.
14. C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.
15. M. F. Porter. An algorithm for suffix stripping. In *Readings in information retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1997.
16. D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 41–47, Philadelphia, PA, 2002.
17. M. Schedl, P. Knees, and G. Widmer. A Web-Based Approach to Assessing Artist Similarity using Co-Occurrences. In *Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing (CBMI'05)*, Riga, Latvia, June 2005.
18. J. Véronis. Weblog, 2006. http://aixtal.blogspot.com.
19. M. Zadel and I. Fujinaga. Web services for music information retrieval. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, October 2004.