Discovering Relations using Matrix Factorization Methods

Ervina Cergani*
Max-Planck-Institut für Informatik
Saarbrücken, Germany
ervina.cergani@mpi-inf.mpg.de

Pauli Miettinen
Max-Planck-Institut für Informatik
Saarbrücken, Germany
pauli.miettinen@mpi-inf.mpg.de

ABSTRACT

Traditional relation extraction methods work on manually defined relations and typically expect manually labelled extraction patterns for each relation. This strongly limits the scalability of these systems. In Open Relation Extraction (ORE), the relations are identified automatically based on co-occurrences of "surface relations" (contexts) and entity pairs. The recently-proposed methods for ORE use partition clustering to find the relations. In this work we propose the use of matrix factorization methods instead of clustering. Specifically, we study Non-Negative Matrix Factorization (NMF) and Boolean Matrix Factorization (BMF). These methods overcome many problems inherent in clustering and perform better than the k-means clustering in our evaluation.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*

Keywords

Relation extraction; Matrix factorizations; NMF; BMF

1. INTRODUCTION

The goal of Information Extraction (IE) from the Web is to extract facts from the free-form text. Traditional information extraction methods are based on manually prepared extraction rules or training examples, typically requiring the user to pre-specify the relations she is interested in. This severely restricts the scalability of these systems, especially on extracting new relations. The goal of Open (or shallow) Information Extraction (Open IE) [1] is to scale both relation and entity extraction to web scale using less sophisticated methods but leveraging the amount of information.

Typical Open IE tools address the whole workflow from plain-text documents to named entities and relations. In this short paper, however, we will only consider the problem of open relation extraction: given named and typed entities and the surface relations (context patterns) connecting entity pairs (for binary relations), find the true relations between the entities. Typically these relations are found by considering which contexts co-occur with similar sets of instance pairs, and considering a group of such contexts as a set of synonyms for some true relation. A common approach to find similar contexts (e.g. [4, 6, 11]) is to apply clustering techniques.

But clustering has some problems that makes it less optimal for this task. The typical clustering methods partition the object space (contexts or entity pairs), meaning that either a single context cannot refer to multiple relations or an entity pair cannot be linked with multiple relations. Further, all contexts or entity pairs must be associated with *some* true relation and the algorithms cannot ignore even the most obvious outliers. Both of these restrictions are unnatural to the task and removing them should improve the quality of the results. Another problem with clustering is the selection of the number of clusters. With the popular k-means algorithm (used in [11]), the number of clusters must be specified a priori. The Hierarchical Agglomerative Clustering (HAC) methods do not require setting the number of clusters, but with them, the user typically has to define a threshold dictating when the agglomeration process is stopped.

We propose to use the matrix factorization methods instead of clustering. In particular, we propose the use of the Non-Negative Matrix Factorization (NMF) and the Boolean Matrix Factorization (BMF). The NMF can be considered a relaxation of the k-means where the strict partition requirement is relaxed to only require non-negative weights. We can group similar objects together based on these weights, thus creating sort of clusters. These clusters, however, do not have to be disjoint or contain all of the objects. The BMF, on the other hand, returns the possibly overlapping clusters directly, but requires all involved matrices to be binary. With BMF we can also use the Minimum Description Length (MDL) principle to automatically decide the number of relations without any need for parameters. Hence, our algorithms overcome the issues associated with clustering, and in our empirical evaluation (Section 4), they typically outperform k-means.

2. RELATED WORK

The existing open relation extraction methods typically either consider the corpus one sentence (e.g. TextRunner [1])

*Current address: Department of Computer Science, Technische Universität Darmstadt, cergani@st.informatik.tu-darmstadt.de

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2507841.

or one category pair (or domain or type signature) at a time. In the latter case, including our methods, the entities are disambiguated first, after which the Noun Entity (NE) pairs appearing close to each other are grouped based on their types (e.g. all NE pairs of type *river–city* are grouped together). The relations are found separately for each category pair based on the NE pairs and the text between the NEs (context). In [4, 6], each NE pair is represented by a vector containing the term frequencies from the contexts. The NE pairs are clustered using HAC methods and and each cluster represents one relation. The relation names are based on the frequency of the terms in the contexts of a single cluster. Done this way, every entity pair can only appear in one relation and to compute the clustering, distance between every NE pair has to be computed, becoming easily prohibitive for the Web scale data. Further, the similarity of two NE pairs is based on the lexical similarity of the contexts, making it impossible to detect relations where the context patterns do not share terms.

To address these problems, `OntExt` [11] proposes to cluster the contexts based on how often they co-occur with other contexts. As there is usually much less contexts than NE pairs, this approach greatly reduces the object space. Also, the lexical similarity is not important anymore. For clustering, `OntExt` uses the k-means algorithm, which is typically faster than HAC methods as it does not have to compute the pair-wise distances. The label of the (candidate) relation is the context closest to the centroid of the cluster. To compute the seed instances (NE pairs) for a relation, `OntExt` ranks all NE pairs (based on how close they are to the cluster centroid) and selects the top-50. To remove the invalid relations, `OntExt` uses a classifier trained on labelled data to classify the clusters into valid and invalid relations.

The NMF is part of the standard toolkit for information retrieval and data mining alike. Probabilistic Latent Semantic Indexing (pLSI), for example, can be considered as a type of NMF with a specific optimization function and normalization. The BMF is somewhat less known, but it has been recently introduced to data mining as a generalization of frequent itemset mining and database tiling [8].

3. ALGORITHMS

Our basic framework follows that of `OntExt`, but instead of k-means clustering we use the matrix factorization methods. Replacing k-means with NMF is relatively straight forward; with BMF, more changes are needed.

3.1 The NMF Approach

The Approach. Similarly to `OntExt`, we build a context-by-context co-occurrence matrix $\boldsymbol{O}$ for each category pair (C_1, C_2). Initially, $\boldsymbol{O}(i,j)$ contains the number of NE pairs (from categories C_1 and C_2) that appear with both contexts i and j. Matrix $\boldsymbol{O}$ is then normalized so that each row sums to 1 after which the values in a row are divided by the number of non-zeros in that row (see [11] for details). Given this (non-negative) co-occurrence matrix $\boldsymbol{O}$ and an integer k, we apply NMF to it. That is, we try to find non-negative matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ of k columns and rows, respectively, such that $\|\boldsymbol{O} - \boldsymbol{W}\boldsymbol{H}\|_F^2$ is minimized.

We consider the columns of $\boldsymbol{W}$ as raw candidate relations. They are "raw" because instead of hard assignment of the contexts to candidate relations, $\boldsymbol{W}$ gives us nonnegative weights between each context and candidate relation. To obtain the final candidate clusters, we round the matrix $\boldsymbol{W}$ to be binary and interpret it so that if $\boldsymbol{W}(i,j) = 1$, then context i is assigned to candidate relation j. Unlike in clustering, a context can be assigned to multiple candidate relations—or to none.

Why NMF? To obtain intuition, let us consider two context patterns, called q and r. They correspond to row vectors $\boldsymbol{O}(q,:)$ and $\boldsymbol{O}(r,:)$ of the original matrix, and to row vectors $\boldsymbol{W}(q,:)$ and $\boldsymbol{W}(r,:)$ in the factorization. The factorization $\boldsymbol{O} \approx \boldsymbol{W}\boldsymbol{H}$ represents row i of $\boldsymbol{O}$ as a sum of rows of $\boldsymbol{H}$ scaled with the values of $\boldsymbol{W}(i,:)$, that is $\boldsymbol{O}(i,j) \approx \sum_k \boldsymbol{W}(i,k)\boldsymbol{H}(k,j)$. This means that if $\boldsymbol{O}(q,:)$ is similar to $\boldsymbol{O}(r,:)$, the corresponding vectors $\boldsymbol{W}(q,:)$ and $\boldsymbol{W}(r,:)$ are also similar and, consequently, q and r should be assigned to the same candidate relation(s) after the rounding.

The Algorithm. Computing the least-error rank-k NMF is NP-hard [14], and hence we must use heuristics. We use the Alternating Least-Squares (ALS) algorithm, where starting from random matrices, $\boldsymbol{W}$ is fixed and $\boldsymbol{H}$ is updated and then the roles are swapped until no more updates are possible (see [2] for more information). We compute the updated factor matrices using standard least-squares optimization and truncate all resulting negative values to 0.

To set k, the number of relations to extract, we followed [11] and used $k = 5$. We rounded values of $\boldsymbol{W}$ larger than $1/2$ to 1 and the rest to 0.

At this point we have the candidate relations. For the remaining steps, we follow [11]: we compute the geometric center of the set of contexts in each candidate relation to obtain the name of the relations and compute the seed instances similarly.

3.2 The BMF Approach

Using the NMF addresses one problem of the k-means (the partitioning), leaves one problem unaddressed (the number of candidate relations), and creates a problem of its own (the rounding of the factor matrix). Our second approach solves both of the problems of the k-means and avoids the rounding issue.

The Approach. Our second approach uses BMF instead of NMF. The BMF (approximately) represents the given binary matrix $\boldsymbol{C}$ as a Boolean product of binary factor matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, $\boldsymbol{C} \approx \boldsymbol{A} \circ \boldsymbol{B}$. The *Boolean matrix product* is defined as $(\boldsymbol{A} \circ \boldsymbol{B})(i,j) = \bigvee_k \boldsymbol{A}(i,k)\boldsymbol{B}(k,j)$; that is, it is the normal matrix product with the addition defined as $1+1 = 1$. The goal of the BMF is to minimize the Hamming distance between $\boldsymbol{C}$ and $\boldsymbol{A} \circ \boldsymbol{B}$, where the number of columns in $\boldsymbol{A}$ and rows in $\boldsymbol{B}$ is predefined (k).

As BMF requires all involved matrices to be binary, we cannot use the co-occurrence matrix $\boldsymbol{O}$. Instead, we use the context patterns-by-instance pairs matrix $\boldsymbol{C}$: we set $\boldsymbol{C}(i,j) = 1$ if context pattern i appears with entity pair j in $S(C_1, C_2)$. Assuming $\boldsymbol{A} \circ \boldsymbol{B}$ is an approximate BMF of such $\boldsymbol{C}$, we can interpret $\boldsymbol{A}$ and $\boldsymbol{B}$ as follows: matrix $\boldsymbol{A}$ will assign context patterns to candidate relations (similar to NMF but without need to round) and matrix $\boldsymbol{B}$ will assign instance pairs to candidate relations. Therefore, the kth candidate relation corresponds to the kth column of $\boldsymbol{A}$ (where $\boldsymbol{A}(i,k) = 1$ if context i is assigned to this relation) and to the kth row of $\boldsymbol{B}$ (where instance pair j is one of the seed instance pairs for this candidate relation if $\boldsymbol{B}(k,j) = 1$). For the name of the relation, we select the context corresponding to the row of $\boldsymbol{C}$ that is closest to the kth row of $\boldsymbol{B}$.

Method	Correct	Incorrect
k-means	53.5	46.5
NMF	66.0	34.0
BMF	60.1	39.9

Table 1: Average percentages of correct and incorrect relations based on manual evaluation of 500 relations discovered from the real-world data.

The Algorithm. Computing the BMF is computationally hard [8]. We used the heuristic `Asso` algorithm [8] for the task. When the matrix $\boldsymbol{C}$ is small enough, we can improve the results further. The idea is to use `Asso` to compute the initial factorization and then alternatively update one factor while keeping the other fixed resulting in ALS-type algorithm. Optimizing one factor matrix of BMF when the other is fixed, is, however, NP-hard even to approximate well [7, 8]. To solve the problem, we use Mixed Integer Programming (MIP) approach proposed by Lu et al. [5] (we used CPLEX as our MIP solver). The alternating update will converge in finite time as each step is guaranteed to reduce the error by at least 1. As solving the MIP is computationally expensive, we cannot use this approach with the largest matrices.

Selecting the Rank. With BMF we can get the assignment of contexts to candidate relations that avoids the problems with k-means (partitioning) and NMF (rounding). But this still leaves us with the problem of selecting the number of factors automatically. To find the correct rank for the BMF, Miettinen and Vreeken [9] proposed using the Minimum Description Length (MDL) principle. The idea is to encode the matrix $\boldsymbol{C}$ using the factors $\boldsymbol{A}$ and $\boldsymbol{B}$ and the error matrix $\boldsymbol{E} = (\boldsymbol{A} \circ \boldsymbol{B}) \oplus \boldsymbol{C}$, where $\oplus$ is the element-wise exclusive or. The more factors we have, the more bits it takes to encode the factor matrices, but the less bits it takes to encode the error matrix; we select the rank that lets us encode $\boldsymbol{C}$ with the least number of bits. To do the encoding, we used the *Typed-XOR Data-to-Model* encoding [10].

As [9, 10] also use `Asso` to compute the BMF, we can use their approach directly to find the correct rank. If the matrix is small enough to use the alternating update scheme with MIP, we will only do this after we have decided the rank, as otherwise we would have to re-compute the alternating updates for each potential rank k.

3.3 NMF with Contexts and Instance Pairs

For the sake of completeness, we also considered applying NMF to the contexts-by-instance pairs matrix $\boldsymbol{C}$. The results were not significantly different from NMF, and we omit them due to lack of space.

4. EXPERIMENTAL EVALUATION

To evaluate our algorithms we used semi-synthetic and real-world data. The purpose of the semi-synthetic data is to have data with known ground truth that still emulates real-world data well. The real-world data, which is the same as used in [11], was used to check that the results we obtained with the semi-synthetic data also hold in real applications.

Semi-Synthetic Data. The semi-synthetic data is based on the PATTY data set[1] [12]. PATTY contains a collection of disambiguated relations, the disambiguated noun entities the relations appear with, the frequencies of the relation-noun entity triples, and the context patterns together with the frequencies of how often they are used to refer to each relation. Further, we obtained the categories of the noun entities from the YAGO data set[2] [13]. We created the semi-synthetic data as follows: We considered every relation–instance pair as many times as it appears in PATTY's data. Every time we replace the relation with one of the contexts used to refer to it. The replacement is done independently of other replacements, and the probability of using particular context pattern is proportional to its frequency of referring to the relation in the PATTY data. This gives us the sets $S(C_1, C_2)$ for the category pairs (C_1, C_2) (defined using YAGO). The semi-synthetic data contains 358 223 noun instances that belong to 1 102 categories. PATTY contains 25 disambiguated relations, and our semi-synthetic data contains 23 645 context patterns.

In the experiment, we used the three algorithms, k-means, NMF, and BMF, to find the candidate relations. We compared the instance pairs associated to the found relation with the instance pairs associated to the true relations; the more similar the sets of instance pairs, the better the algorithm was able to recover the true relation. We say that a candidate relation corresponds to a real relation if the Jaccard coefficient of the instance pairs between the two is at least 0.95. For NMF, almost 85% of all found relations corresponded to a real relation, for k-means, almost 90%, and for BMF, more than 99%.

Scalability. To measure the scalability of our approach, we present the running times of NMF and k-means with the semi-synthetic data in Figure 1. The NMF was implemented in Matlab and for k-means we used Matlab's build-in implementation. The experiments were run on a server with 40 2.27 GHz Intel Xeon processors. We grouped the context-by-context matrices based on their size (x-axis) and measured the time in seconds a single run of NMF or k-means took (y-axis). As both methods require random re-starts, they should be run multiple times, but the effects of this simply scale the time linearly. For most of the matrices, k-means is slightly faster on average, but its running times vary much more, especially with the larger matrices. The higher variance is probably due to the k-means requiring more iterations to converge with few data sets, while converging faster with the others.

As BMF uses different type of data (contexts-by-instance pairs versus contexts-by-contexts), its running times are not directly comparable. Nevertheless, it is clearly slower than NMF or k-means (up to an order of magnitude with the implementation we used), and the CPLEX MIP solver should only be used with small matrices (at most 400 contexts in our experiments). We omit further details due to lack of space.

Real-World Data. We used the real-world data from [11] (with the pre-processing described there). As we did not have any ground truth, the results are analysed manually. We also use the classifier-based post-processing to measure the quality of the results at the end of the workflow.

We manually examined a subset of the results from the different algorithms. 500 randomly selected relations were manually labelled as correct or incorrect. In addition to ourselves, we used three external reviewers. The averages

[1] `http://www.mpi-inf.mpg.de/yago-naga/patty/`

[2] `http://www.mpi-inf.mpg.de/yago-naga/yago/`

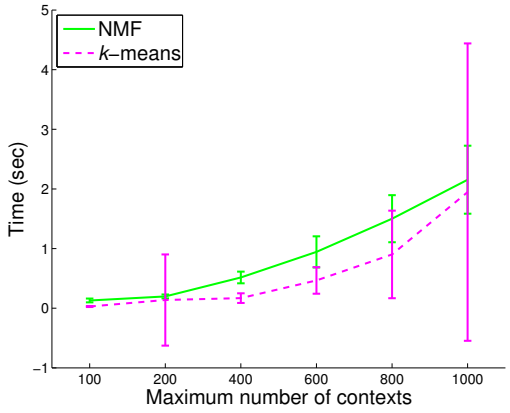

Figure 1: Average running times of NMF and k-means with different number of contexts. The width of the error bars is twice the standard deviation.

Method	Precision	Recall	ROC Area
k-means	0.541	0.540	0.540
NMF	0.431	0.640	0.485
BMF	0.650	0.650	0.650

Table 2: Classifier results.

of the results are reported in Table 1. NMF gives the highest percentage of correct relations, followed by BMF and k-means. There was some variation between individual reviewer's results, but the differences in the averages are big enough to absorb those variances. The success rate is not very high (66% for the best method), but is still better than what was reported with `OntExt` [11], where the authors state that more than half of the candidate relations were incorrect.

Classification Results. Using a classifier to post-process the candidate relations can greatly improve the results with a one-time cost of training the classifier. It is not clear that the results from different methods are equally easy to classify, however. To test this, we trained a classifier on the manually labelled candidate relations of the real-world data, and studied how well the classifier performs with unseen relations. To build the classifier, we build the features following [11]. As a classifier, we used a support vector machine classifier with an RBF kernel (the SMO classifier of the Weka package[3] [3]), as it gave the best results of the tried classifiers. The parameters were optimized using a grid search. To train the classifiers, we used 400 labelled relations and 10-fold cross-validation. For the evaluation, we used the remaining 100 labelled relations. The labels were obtained by taking the majority vote of the reviewers' labels. The results of the classification are presented in Table 2.

From the results we see that BMF is the best in precision, recall, and ROC. NMF is better than k-means in recall, but worse in precision.

5. CONCLUSIONS

We have proposed to use matrix factorization instead of clustering in relation discovery. The matrix factorizations are more robust than clustering as they do not require partitioning. Furthermore, using the MDL principle together with the BMF allows automatic selection of the correct number of clusters. In the experimental evaluation, BMF seems to be the best all-round performer, but this quality comes with a cost in running times. As each category pair is dealt independently, however, the work can be trivially distributed in the cloud, alleviating the problem. If BMF is deemed too slow, NMF also improves over k-means in many cases without significant hit on the running times.

More comprehensive study against the HAC-based methods is the next step in future work. Further, it is of interest to study whether we can use the specific knowledge of the domain to improve the BMF's running times (e.g. by the sparsity of the matrices, or using the Buckshot-type approach used by SONEX [6]).

[3] `http://www.cs.waikato.ac.nz/ml/weka/`

6. REFERENCES

[1] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open Information Extraction from the Web. In *IJCAI '07*, pages 2670–2676, 2007.

[2] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and Applications for approximate Nonnegative Matrix Factorization. *Comput. Stat. Data An.*, 52(1):155–173, 2007.

[3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[4] T. Hasegawa, S. Sekine, and R. Grishman. Discovering relations among named entities from large corpora. In *ACL '04*, 2004.

[5] H. Lu, J. Vaidya, and V. Atluri. Optimal Boolean Matrix Decomposition: Application to Role Engineering. In *ICDE '08*, pages 297–306, 2008.

[6] Y. Merhav, F. Mesquita, D. Barbosa, W. G. Yee, and O. Frieder. Extracting information networks from the blogosphere. *ACM Trans. Web*, 6(3), Sept. 2012.

[7] P. Miettinen. On the positive-negative partial set cover problem. *Inform. Process. Lett.*, 108(4):219–221, 2008.

[8] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The Discrete Basis Problem. *IEEE Trans. Knowl. Data En.*, 20(10):1348–1362, Oct. 2008.

[9] P. Miettinen and J. Vreeken. Model Order Selection for Boolean Matrix Factorization. In *KDD '11*, pages 51–59, 2011.

[10] P. Miettinen and J. Vreeken. MDL4BMF: Minimum Description Length for Boolean Matrix Factorization. Technical Report MPI–I–2012–5-001, Max-Planck-Institut für Informatik, June 2012.

[11] T. P. Mohamed, E. R. Hruschka Jr, and T. M. Mitchell. Discovering Relations between Noun Categories. In *EMNLP '11*, pages 1447–1455, 2011.

[12] N. Nakashole, G. Weikum, and F. Suchanek. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *EMNLP-CoNLL '12*, 2012.

[13] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge. In *WWW '07*, pages 697–706, 2007.

[14] S. A. Vavasis. On the Complexity of Nonnegative Matrix Factorization. *SIAM J. Optim.*, 20(3):1364–1377, 2009.