DIMACS Technical Report 2005-42
February 2006

Simulated Entity Resolution by Diverse Means: DIMACS Work on the KDD Challenge of 2005

by

Andrei Anghelescu, Aynur Dayanik, Dmitriy Fradkin,
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08854

Alex Genkin,
DIMACS
Rutgers University
New Brunswick, New Jersey 08854

Paul Kantor,
SCILS
Rutgers University
New Brunswick, New Jersey 08854

David Lewis,
David D. Lewis Consulting

David Madigan,
Department of Statistics
Rutgers University
New Brunswick, New Jersey 08854

Ilya Muchnik, Fred Roberts,
DIMACS
Rutgers University
New Brunswick, New Jersey 08854

ABSTRACT

This report describes DIMACS work on two of the groups of entity resolution problems, ER1 and ER2 for the KDD Challenge in 2005. We presume that the situation is intended to mimic, using abstracts and author information from the life sciences, some real world problem, in which it is important to recognize the identity of an individual, even though he may share that name with other individuals (ER1), or may actively seek to hide his identity by removing his own name from a work, or replacing it with an alias (ER2a, and ER2b,c). Thus specific problems investigated include author resolution, finding a missing author of a paper, and detecting a false author of a paper. The methods used to attack these problems include combinatorial cluster analysis, fusion of methods, penalized logistic regression / maximum entropy approaches, and dependency modeling.

1 Introduction

This report describes DIMACS work on two of the groups of entity resolution problems, ER1 and ER2. Before presenting the details of our approach, we offer an overview here. We presume that the situation is intended to mimic, using abstracts and author information from the life sciences, some real world problem, in which it is important to recognize the identity of an individual, even though he may share that name with other individuals (ER1), or may actively seek to hide his identity by removing his own name from a work, or replacing it with an alias (ER2a, and ER2b,c). We can describe our methods in terms of a cross tabulation relating three different levels at which the problem can be addressed, and three categories of tools.

The notion of identifying an individual can be thought of at a first level by asking about the matrix of that individual's interrelations with other individuals. This is the model used, for example, in social network analysis. We have therefore tried a number of baseline computations for which the only information that we use is the frequency of co-occurrence of individuals as named authors of a paper, compared in some way to the frequency of co-occurrence that might be expected naively. We think of this as a population-based measure of author 'affinity'.

The second level at which individuals might be identified and disambiguated is by their relations to the texts or products that they produce. This is essentially an extension of the concepts in stylometrics and author identification. We have applied this perspective in a number of ways across the problems.

The third, which is a category of methods of extreme importance in real forensic work, is to identify individuals who wish to hide by taking advantage of mistakes that they have made while seeking to hide. In the dataset of the KDD Challenge, this seems to be the case with a number of authors whose names were deleted or replaced in the list of author names, but whose names continue to appear in a field which contains the address of the corresponding author. Needless to say, we expect that using this additional piece of information which is quite determinative in some cases where it appears should result in improved performance.

[We are aware that it was not the real intention of the task, but since we found that we had inadvertently performed a number of such calculations before realizing that it might be a problem, and contacting the Challenge Organizers, we include the material here.]

Labeling the rows of the table by the general method of approach as described above, we can also label the columns of the table according to the approaches that are used to deal with the problem. We also divide these into three categories, which we think of as 'our tools', 'other available software', and 'special purpose heuristics'.

By 'our tools', we refer to algorithms and software that we have developed in our work on the KDD project. Foremost among these are our Bayesian Binary Regression models. These are models which can directly address the question 'how probable is it that a particular individual a is the author of a particular work product d, which we think of as a text, but which in fact carries with it information about co-authors, the address from which the paper was submitted and key word information that has been added by expert indexers. In addition, we have applied, in two different ways, a tool called 'combinatorial Principal Component Analysis' (cPCA), developed in the first year of our KDD project, which provides a direct solution to the problem of clustering. We have applied this particularly to problem ER1a, for which we did not expect any gold standard truth to be known.

Most of our calculations using 'other software' have been done with variants of the CLUTO clustering software, developed by Karypis. We have used several variants of this in dealing with the problems ER1. In addition, in ER2 we have made use of the Maximum Entropy Toolkit developed by Le [7].

Finally, under the heading 'special purpose heuristics', we report various things that we tried which are not particularly a result of the development of our software, or of other packages that we found, but which seemed either to provide reasonable baseline efforts for the problem, or to take advantage of information inadvertently left in the files by the individuals trying to hide from us.

2 The Role of the Evaluation Measures

In our work, we quickly discovered that the choice among methods depends on several factors which are more or less undetermined for us. The first, which arises in the problem of clustering authors in ER1b, is that the question of which method appears better depends strongly on whether the size of the clusters to be discovered are roughly equal (which we refer to as the 'unskewed' case), or are quite unequal ('skewed'). We therefore invested some energy in asking whether the results of our clustering efforts themselves, and in particular differences among the apparent goodness of fit of several methods, might give us any indication of which of these situations is present in a given dataset.

More troublesome is the fact that all measures that are applied to problems of clustering or classification represent different possible interpretations of what 'really matters' to the eventual client of the work. This is most clear when specific unit costs are attached to specific types of error such as (a) wrongly assigning a particular paper to a particular author, (b) wrongly concluding that two particular authors are the same, (c) wrongly concluding that

Level	Our Methods	Other Software	Special Purpose Heuristics
Author-Author	cPCA; Affinity	CLUTO	
Author-Text	BBR Models; Docsim	MaxEnt	
Errors of hiding			AAAN

Table 1: DIMACS array of levels of analysis by method

there are K authors, when in fact there are K+1 and so forth. Because of the schedule of posting the methods, we had many opportunities to experiment with these effects.

What we find, which is, in the final analysis, not surprising, is that, depending on the choice of evaluation method, different measures will appear to be superior. It is for this reason that we are submitting a rather large number of result sets for each of the problems. In most cases the justification for submitting a result set is that it was the best, or nearly the best, for some particular choice of the evaluation measure that we examined. In the case of the variants of CLUTO, each of them actually corresponds to a specific heuristic for assessing the goodness of a clustering, although in some cases they did not perform particularly well on any of the metrics that we were using to compare to ground truth.

The influence of the choice among metrics, on preference among methods, is made more complex by the question of how total scoring for a collection of 100 problems is to be done. For each of the measures considered, there are a variety of ways in which that might have been done and, again, we were to a considerable degree, 'flying blind' in our training exercises.

With all of these caveats in mind, we now review the specific approaches that were applied to each problem, placing them in the appropriate cells of the matrix shown in Table 1.

3 Data Sets and Representation

3.1 Data Sets

For task ER1 we used the prepared materials. These were represented by selecting a range of terms that are neither too rare nor too frequent in the collections (for binary representations we required that the document frequency (number of documents in which the term appears) be at least 2 in the collection for the specific task). When using a 'tf.idf' representation we included any terms that appear in at least two documents in the entire BioBase collection (approximately 150,000 documents), with the prevalence of the term which appears in idf computed from the entire collection as well. In addition, we prepared a selected set of papers apparently by the single author T. Suzuki from the MedLine Collection, and spent some time trying to assess ground truth for that set. The resulting distribution seemed to have a very large number of 'authors' who wrote only one paper. This does not seem to be the case with the actual KDD collection, making it difficult to transfer what we have learned.

3.2 Representations and Metrics of Two Kinds

Our representation, in terms of the basis described above, is either binary or 'tf.idf'. In the binary representation the component corresponding to term t of the vector representing the document d is $b(d,t) = 1$ if t appears in d; 0 otherwise.

For the tf.idf representation the corresponding component of the vector v(d,t) is 0 if the term does not appear in the document and is:

$$v(d,t) = (1 + \ln f(d,t)) \frac{N+1}{1+N(t)}, \tag{1}$$

where $f(d,t)$ is the number of times that the term t (after stemming by the LEMUR package) appears in the document d; $N(t)$ is the number of documents (in the collection) in which the term t appears at all; and N is the total number of documents in the collection.

We have explored a number of ways of computing the similarity between documents (that is, metrics in document space) which are described below in detail.

We have also considered several possible 'metrics' for the evaluation of solutions, including an array of clustering metrics used within the CLUTO software (Karypis, 2003). Methods for comparing a solution to a ground truth include simple error counts, methods based on the number of pairs of papers that are labeled correctly (i.e. as having or not having an author in common) and the more subtle rank-based methods that are to be used in scoring the challenge.

4 Specific Problems and Methods: ER1a

4.1 cPCA

The combinatorial Principal Component Analysis (cPCA) method [1, 3] (for relevant background, see [6][1]) is based on results from the theory of quasi-convex functions defined on the power set of a given set, in this case the set of entries in a matrix whose entries all represent similarities. The method can be applied to matrices of document-document similarities, or to matrices of terms-document similarities. In the former case it provides a unique clustering of the documents. In the latter case it simultaneously clusters the documents and selects, for each cluster, the terms that are most indicative of that cluster. In practice the method is computable in $O(N^3)$ time. It first finds the 'most uniformly tight cluster', called the kernel. It then removes those documents from the collection and repeats the process.

The representation is to consider as a basis all terms (including author names appearing in author fields, and in address fields; keywords; and terms in the abstract and the title) that appear at least 2 times, and not more that 30,000 times in the full corpus for the challenge (approximately 150,000 documents).. We consider both a tf.idf' representation, and a Boolean representation, for each feature.

[1]http://mms-03.rutgers.edu/~dfradkin/papers/Monotone.pdf

We have tried many variants of the procedure, differing in the metric used to compute similarity. The notation for method names is as follows. In all cases, cPCA is used for clustering. Documents are represented by vectors that either are binary (b) or are based on term frequencies (tf.idf). The similarity is completed using the cosine measure for the norm L_p, represented as pp in the file name. The method was applied either to the document similarity matrix, called s, or to the document-by-term matrix v. Thus the code cPCA.p4.s.b uses document similarities, a binary representation, and the L_4 norm. The submission files include the organization name and the entity resolution problem, but do not specify cPCA. For example, the file named DimacsER1a.df.L1.tf contains results from the method cPCA.p1.v.tf

In addition, we performed a simple fusion of methods. The result of each clustering was used to produce a similarity score. This is done in three steps:

1. each cluster C is represented by the value of the clustering score function on that cluster:
$$\min_{a \in C} \sum_{x \in C-a} Sim(a, x) \tag{2}$$
2. all scores are normalized by dividing by the largest score.
3. a linear transform is used to set the lowest score above a nominal baseline (5%). This transformation is expected to have no effect on scoring by rank-based methods, and ROC, but would of course affect any scores based on likelihood.

The full array of these basic methods reported is as follows:

- *CPCA.p0.s.b*: Document similarities, binary representation; no normalization
- *CPCA.p0.s.tf*: Document similarities, TF representation; no normalization
- *CPCA.p0.v.b*: Documents x features, binary representation; no normalization
- *CPCA.p0.v.tf*: Document x features, TF representation; no normalization
- *CPCA.p1.s.b* : Document similarities, binary representation, L_1 norm
- *CPCA.p1.s.tf*: Document similarities, TF representation, L_1 norm
- *CPCA.p1.v.b*: Documents x features, binary representation, L_1 norm
- *CPCA.p1.v.tf* : Document x features, TF representation, L_1 norm
- *CPCA.p2.s.b*: Document similarities, binary representation, Euclidean norm
- *CPCA.p2.s.tf*: Document similarities, TF representation, Euclidean norm
- *CPCA.p2.v.b*: Documents x features, binary representation, Euclidean norm

- *CPCA.p2.v.tf*: Document x features, TF representation, Euclidean norm
- *CPCA.p4.s.b*: Document similarities, binary representation, L_4 norm
- *CPCA.p4.s.tf*: Document similarities, TF representation, L_4 norm
- *CPCA.p4.v.b*: Documents x features, binary representation, L_4 norm
- *CPCA.p4.v.tf*: Document x features, TF representation, L_4 norm

In addition, we did two data fusion experiments in which we simply added the scores assigned to pairs of documents as a result of clustering. This was done separately for all the clusterings produced by analysis of the document similarity matrix, and of the document-by-term relation matrix. The average gives all methods equal weights.

- *CPCA.s.fus*: Document similarities, average of cPCA.*.s.* results
- *CPCA.v.fus*: Document similarities, average of cPCA.*.v.* results

4.2 Direct document similarity

Finally, we have produced submissions based directly on the computed similarity between documents, as would clearly be appropriate under the scoring proposed for task ER1b. In this case the similarity itself, which is the cosine of the angle between two non-negative vectors, is treated as if it were a probability. These methods do not use cPCA, and can be thought of as a baseline. Using a notation as above they are described as shown in Table 3.

- *DimacsER1a.df.L1.tf*: cPCA.p1.v.tf documentXfeatures, TF representation, L_1 norm
- *DimacsER1a.dd.L2.tf*: cPCA.p2.s.tf document similarities, TF representation, Euclidean norm
- *DimacsER1a.dd.L2.b*: cPCA.p2.s.b document similarities, binary representation, Euclidean norm
- *DimacsER1a.dd.fus*: cPCA.s.fus document similarities, average of cPCA.*.s.* results
- *DimacsER1a.df.fus*: cPCA.v.fus document similarities, average of cPCA.*.s.* results
- *DimacsER1a.fus*: cPCA.fus average of all cPCA results
- *DimacsER1a.sim.L2.b*: sim.p2.b inner product, binary representation, L_2 norm

The first three methods were chosen because they performed best on a model set of papers we prepared from the MedLine data base, authored by any individual using the name 'T. Suzuki'. In particular, when applied to the ER1b tests, they yielded a very small number of authors (two or three) with most documents clustered to a single author (i.e. the second and, sometimes, third authors, had only one or two documents assigned to them).

The reason there are three fusion results is because of the rather significant differences between the results of cPCA on the feature data (i.e. documents by features) and on the document similarity matrix. These results seem to yield distinct groups, that don't correlate very well with each other. For instance, working with the document similarity matrix, yields on average 5 authors per problem, while the feature data matrix yields about 8.

5 Specific Problems and Methods: ER1b

The separation of artificially merged clusters of aggregate authors was studied by several methods from the CLUTO package [5]. Models for forming clusters are agglomerative, graph-cut, and recursive bisection. Details of these methods are given in the software manual [5]. Agglomeration may be done either by a single link method, or by a complete link method, and the result sets are named accordingly. We have submitted only agglomerative models based on the tf.idf representation, as they gave superior performance on our training materials. Similarly, random bisection was applied only to this representation. Results from random bisection are rather sensitive to the heuristic used to measure the clustering as it proceeds. The CLUTO methods called g1, h2, and i2 were applied. Definitions are given in the manual for the software.

We also approached the problem in another way, simply computing the similarity between documents and reporting those numbers (normalized) for the off diagonal elements of the matrix. These are reported as docsim and were computed for both the tf.idf and the binary representations of the documents.

Finally, for completeness, we applied a model averaging or data fusion procedure to results obtained both with the binary and the tf.idf representations. These are reported as modeavg. In the model averaging runs, we summed the probabilities for all document pairs as given by the different models and normalized by the number of models. Specifically, 11 CLUTO models and 1 Document Similarity method contributed to the model averaging.

For this task we also explored the effect of 'errors in hiding'. A model that makes use of 'author name information' from the 'author address' field can, in principle, be quite an advantage in detecting hidden authors. In fact, it does not appear to have a large effect in our work on this specific task (though it may on the ER2 tasks). But we consider both cases. When such information is used the string 'aaan' appears in the run name. When it is not used, the string 'noaaan' appears in the run name.

The complete set of ER1b results is shown in below:

- *Dimacs-er1b-agglo-clink-tfidf-noaaan.txt*
- *Dimacs-er1b-agglo-slink-tfidf-noaaan.txt*

- *Dimacs-er1b-docsim-binary-noaaan.txt*
- *Dimacs-er1b-docsim-tfidf-noaaan.txt*
- *Dimacs-er1b-graph-cut-tfidf-noaaan.txt*
- *Dimacs-er1b-modeavg-binary-noaaan.txt*
- *Dimacs-er1b-modeavg-tfidf-aaan.txt*
- *Dimacs-er1b-modeavg-tfidf-noaaan.txt*
- *Dimacs-er1b-rb-g1-tfidf-noaaan.txt*
- *Dimacs-er1b-rb-h2-tfidf-noaaan.txt*
- *Dimacs-er1b-rb-i2-tfidf-aaan.txt*
- *Dimacs-er1b-rb-i2-tfidf-noaaan.txt*

6 ER2 Problems: Introduction and Models

The three ER2 problems required detecting deletions or replacements of authors in BioBase abstracts. All of our approaches to the ER2 tasks were based on supervised training of statistical models that captured whether a particular author was likely to be an author of a given test record.

Four types of models were trained as described in the remainder of this section, with some of the models applied to more than one of the ER2 subtasks. The training sets and text representations varied somewhat for the different model types.

6.1 Smoothed Frequency Model for Probability of Authorship

The first approach treated $p(y_k = 1|D)$, the probability that author k was an author of test document D, as equal for all documents. It used a simple smoothed estimate of that probability, $p(y_k = 1|D) = (1 + n_k)/(1 + n)$, where n is the number of training documents, and n_k is the number of training documents with author k. Probabilities were estimated from the $n = 152,561$ BioBASE records that were not in the ER2a, ER2b, or ER2c test set, and for all 300,961 authors in that set of records. As we later discovered, this means we included the ER1b documents (with their corrupted authors), and the 405 documents with no <author> element (which presumedly are also in error, or at least are not representative of test documents).

This approach was applied to problems ER2a, ER2b, and ER2c.

6.2 Binary Logistic Regression Model for Probability of Authorship

The second approach used a separate binary logistic regression model to compute a test document-specific estimate $p(y_k = 1|D)$ for each author k and test document D. Logistic regression models were trained for all authors that appeared in test documents for ER2a, ER2b, or ER2c and all authors that appeared on the list of potentially deleted authors for ER2c: a total of 14,151 authors. In addition, we trained models for an additional 22,777 authors that appeared in 5 or more of the 152,561 BioBASE records, for a grand total of 36,928 authors.

Training was done using our Bayesian logistic regression package, BBR [2] to find a MAP (maximum a posteriori) setting of the parameter values. A Laplace prior was used to favor sparse models. Note that because we use an informative prior, we can train models even with no positive examples, though of course such models are not very good predictors.

The potential training data pool used was the 127,217 records that contained one or more of these 36,928 authors, minus the 4000 ER2 test records, for a total of 123,217 potential training examples. However, for efficiency reasons, we trained the binary logistic regression model for author k only on the positive training examples available for that author, plus approximately 5000 negative examples for the author, randomly selected from the all negative examples among the examples.

The undersampling of negative examples is slightly less problematic than it sounds. If one assumes that a logistic regression model exists which perfectly estimates the probability of class membership for all examples, then in the asymptotic limit (i.e. as training set size approaches infinity) all parameters of the fitted logistic regression model except the intercept will identical under both random sampling and the above biased sampling scheme. Further, the true value of the intercept will be related to fitted value as follows:

$$\beta_{true} = \beta_{fitted} - \ln \frac{t_0}{t_1} \tag{3}$$

where t_0 is the fraction of all negative examples that we let into the training sample, and t_1 is the fraction of all positive examples that we let into the training sample (1.0 in our case).

While this correction is strictly correct only asymptotically, it is also sometimes used with finite samples as well. After training our logistic regression models, we therefore adjusted the fitted intercept values using the above formula, with $t_1 = 1.0$ and $t_0 = 5000/(127,217 - n_k)$, where n_k was the number of positive examples of author k in the set of 127,217 records.

We actually trained two separate sets of logistic regression models:

1. The first set of models was trained on examples represented by a vector of unnormalized binary feature values. All feature types were used, including all author id features except the feature for the author being predicted. The number of features with non-default values on the training data, d, varied from model to model depending on the particular training examples. Test examples were represented in the same fashion.

The hyperparameter (variance) for the Laplace prior was BBRs training set-dependent default, d/m, where m is the mean 2-norm of the training examples. For the training sets used here, typical prior variance values were around 700.

2. The second set of models was trained on examples represented in the same fashion, except that AAAN features were omitted. Also, because simulated ER2b problems suggested that the first set of models had overfit, we used a fixed prior variance of 1.0 when training this second set of models.

Time limitations and the small number of positive examples for some authors precluded the use of cross-validation to choose prior variance.

These two sets of models were used on ER2a, 2b, and 2c.

6.3 Polytomous Logistic Regression Model for Probability of Authorship

Our third approach was to build a single polytomous logistic regression model whose dependent variable ranged over all possible deleted authors. The polytomous approach was applied only to ER2c, so the dependent variable had 3051 possible values, corresponding to the evaluation-supplied list of authors that could be deleted in ER2c.

Records were represented by unnormalized binary feature values. All feature types were used, including all author ID features appearing in the 99,454 training vectors (see below), for a total of 302,913 features. In an oversight, we included features for the 3051 target authors, making the probabilistic semantics of the model inconsistent, but having little practical impact.

We used two variants on this representation in different runs:

1. The full set of 302,913 features.

2. From the set of 302,913 features, we chose for each of the 3051 target authors the five features that had the highest values of $F1 = (2 * TP)/(2 * TP + FP + FN)$ between the feature and the class (omitting the author ID feature for the target itself). Here TP is the number of training records (out of 99,454) in which the feature has the value 1.0 and the record has the target author as an author, FP is the number of records where the feature is 1.0 and the target is not an author, and FN is the number of records where the feature is 0.0 and the target is an author. A total of 12,549 distinct features were in this representation.

The records available for training were the 61,156 records that contained one or more of the 3501 potentially deleted authors, and did not occur in the ER1b, ER2a, ER2b, or ER2c test sets. If a training record contained multiple authors from the list of 3501, we replicated the record once for each such author, resulting in a total of 99,454 training vectors. The copy used as a positive example for author k had author k omitted as a predictor feature.

With our largest feature set, the resulting polytomous logistic regression model has 3051*(302,913+1) or about 9.24×10^8 formal parameters. While the fitted model with a Laplace prior would have vastly fewer nonzero parameters, our Bayesian polytomous logistic regression software, BMR [2], currently allocates space for all formal parameters. We were thus unable to apply BMR to ER2c, though we are developing a version that will scale to problems of this size.

We therefore used a different implementation of polytomous logistic regression, Zhang Le's Maximum Entropy Toolkit [7]. This software finds a MAP fit of a polytomous logistic regression model with a Gaussian prior. The key difference from BMR is that the MET only fits parameters for feature/class combinations where the feature takes on a nonzero value in at least one training examples of the class, thus vastly reducing the number of parameters. This results in slightly different parameter fits, but typically little or no impact on effectiveness, and arguably produces a more trustworthy and understandable model. We used with a prior variance of 1.0 and 30 iterations of optimization. We would have preferred to use a smaller prior variance, but that turned up numerical problems in the optimizer.

6.4 A Heuristic Information Theoretic Author Affinity Method

In addition to the probabilistic models described above, we also tested a heuristic information theoretic method. We defined the affinity between two authors a and b to be the pointwise mutual information between them:

$$\text{Aff}(a, b) = \log \frac{p(a, b)}{p(a)p(b)} \tag{4}$$

where $p(x)$ is the probability that a randomly selected document will have x as an author, and $p(x, y)$ is the probability that a randomly selected document will have both x and y as authors. These probabilities were estimated by maximum likelihood on the 152,156 documents not in the ER2 test sets. (If a pair (a, b) did not occur in the training set, then we set $\text{Aff}(a, b) = 0$.) There were a total of 300,961 authors and 793,827 author/document pairs observed on this training data.

The affinity between an author a and a document D was then defined to be the sum of the affinities of the author with all other authors of the document:

$$\text{Aff}(a, D) = \sum_{b \in A_d, b \neq a} \text{Aff}(a, b) \tag{5}$$

A high affinity between an author and a document was suggestive that the author was one of the original authors of the document, and vice versa.

We adapted this affinity approach for ER2b and ER2c as described later.

7 Task ER2a

The test data for ER2a were 2000 records, an unknown proportion of which had had one of their authors replaced with some incorrect author. The goal was to predict for each record the probability that it was one of the corrupted ones.

7.1 Effectiveness Measures for ER2a

The stated effectiveness measures for the task were 1) area under the ROC curve produced by ranking the 2000 examples by predicted probability of corruption, and 2) cross entropy, i.e. negated mean loglikelihood the model assigns to the correct decision (corrupt vs. noncorrupt) for each test example. Cross-entropy cannot be used as a measure if some predictions are 0 or 1, so the sponsors also reported results for squared error between the probability predicted and a 1/0 target label for corrupt/noncorrupt. The cross-entropy and squared error measures meant there was a premium on making accurate probability estimates, not just ranking the test records in a good order.

7.2 Applying Authorship Models to ER2a

All our ER2a runs made use of probabilistic models of authorship—the ad hoc affinity method was not used.

Preparation of the ER2a testset involved the selection of some subset of the test documents for corruption, with those documents having one original author deleted and one new author added. We treated the set of authors of a document as a binary vector y of length r, where r is the number of distinct authors that we created models for. A value of 1 at coordinate k ($y_k = 1$) indicates that author k is an author of the document, while a value of 0 indicates that they are not.

We made the following assumptions about the corruption process:

1. Each test record had an equal probability, $p_{corrupt}$, of being chosen for corruption,

2. If a record were chosen for corruption, each author had an equal chance of being chosen to be deleted, and

3. The author chosen to replace a deleted author was drawn uniformly at random from the set of all available authors.

Assumption 3 is a bit vague, since the set of "available" authors is not well-defined. In practice, we treated the set of authors for which we built author-level models using a particular technique as the available authors. This set of authors varied from technique to technique as described in Sections 6.1 to 6.4.

Our approach to ER2a was to model the probability of seeing a particular vector of authors with and without corruption, and then use Bayes rule to invert this model. Suppose that a test record has h authors in an arbitrary order. We let $z = g$, $g = 1, \ldots, h$, indicate that the original g'th author of the record was deleted and replaced by an incorrect author. We use $z = 0$ to indicate no corruption occurred.

By Bayes rule then have

$$p[z=0|y,D] = \frac{p[y|D,z=0]p[z=0]}{p[y|D,z=0]p[z=0]+p[y|D,z\neq 0]p[z\neq 0]} \tag{6}$$

and similarly for $p[z\neq 0|y,D]$. It is convenient to work with the ratio of these two, i.e. the odds ratio:

$$R = \frac{p[z=0|y,D]}{p[z\neq 0|y,D]} = \frac{p[y|D,z=0]p[z=0]}{p[y|D,z\neq 0]p[z\neq 0]} \tag{7}$$

Given R it is easy to compute the desired probability of corruption:

$$p[z\neq 0|y,D] = \frac{1}{1+R} \tag{8}$$

We had no knowledge of the actual probability of corruption, so we assumed $p[z=0] = p[z\neq 0] = 0.5$. Computing R then required estimating $p[y|D,z=0]$ and $p[y|D,z\neq 0]$. We chose to treat the authors in the uncorrupted case as conditionally independent of each other, given the values of all other features. This gives:

$$p[y|D,z=0] = \prod_{k=1}^{\gamma} c_k \tag{9}$$

where

$$c_k = p[y_k|D,z=0] \tag{10}$$

is the probability of seeing the observed value (0 if author k is not present in the test document, 1 if they are) of y_k.

The trickier case is computing $p[y|D,z\neq 0]$, the probability of seeing a particular vector of authors, if the record has been corrupted. We must sum over all possible original records that could have been converted to the observed record by corruption. Assuming the observed document has h authors, the original document must have h authors as well. Any of them may have been the one replaced, so:

$$p[y|D,z\neq 0] = \sum_{g=1}^{k} p[y|D,z=g] \tag{11}$$

where $p[y|D,z=g]$ is the probability of getting our observed authorship vector y given that the gth original author of the record was the one replaced. This probability is

$$p[y|D,z=g] = \sum_{a\in(A-A_D)} \frac{1}{h(r-h)} p[y(a\leftrightarrow u[g])|D,z=0]. \tag{12}$$

Here:

- $u[g]$, for $g=1,\ldots,h$, is the index in authorship vectors corresponding to the g'th author of the observed record D.

- $A = 1, \ldots, r$ is the set of indices for als possible authors, and A

1. *dimacs-er2a-qr*: Estimates c_a values using smoothed author frequencies. AAAN features do not affect this approach.

2. *dimacs-er2a-single*: Estimates c_a values using $r = 36,928$ binary logistic regression models. AAAN features are used.

3. *dimacs-er2a-NAN-single*: Same as dimacs-er2a-single, but AAAN features are not used.

All runs assumed the probability a document would be selected for corruption was $p[z = 0] = 0.5 = p[z \neq 0]$.

There were several potentially problematic aspects of our strategy:

1. Our model of the selection of documents for corruption (with fixed probability 0.5), of authors for deletion (fixed probability $1/h$), and of authors for replacement (with fixed probability $1/(r - h)$) was extremely simplistic. More importantly, it was not tuned to data, and so we were at the mercy of bad guesses about the evaluation design.

2. Due to lack of time, we used limited training sets and no cross-validation in training our logistic regression models, leading to overfitting and poor effectiveness.

3. The assumption of conditional independence made with the logistic regression models led to a formally invalid probabilistic model for the authorship vectors (in the sense that sum over all possible authorship vectors is not guaranteed to be 1.0). This is because the author ID being predicted by one binary logistic regression model is an input feature to the other binary logistic regression models. Taken together, our set of 36,928 logistic regression models trained can be viewed as an *inconsistent dependency network* [4]. Heckerman, et al present a heuristic algorithm, an *ordered pseudo-Gibbs sampler*, for generating a normalized joint distribution from an inconsistent dependency network, but we did not have time to implement and run this strategy.

4. Our assumption that document had had at most one deletion and replacement ignored the overlap of 352 documents between the ER1b and ER2a test sets. This meant that some documents (75 by our count) actually had two or more deletions and replacements.

7.3 Adjustments for AAAN Information in ER2a

While author IDs were replaced in forming ER2a test cases, occurrences of author names in the `<author_address>` elements (AAAN's) were not. In AAAN runs we compared the AAAN for a record with all author names in the entire data set. If there were zero or multiple matches, we ignored the AAAN. If there was exactly one match, we adjusted the above model as follows:

- If the AAAN author exactly matches one of the test document authors, we assume that author must be one of the correct authors, not a replacement for a deleted author. We therefore omit the AAAN author when summing over authors that may have been replacements, thus summing over $h - 1$ authors instead of h.
- If the AAAN author does not match one of the test document authors, we assume that the AAAN author was one of the original authors, but was replaced with some other author. We assign a probability of 0.999999 that the document has been corrupted.

Our two ER2a runs using this strategy were

4. *dimacs-er2a-qr-X*: Same as dimacs-er2a-qr, but using the above AAAN-based adjustments.

5. *dimacs-er2a-single-X*: Same as dimacs-er2a-single, but using the above AAAN-based adjustments.

Unfortunately, we did not notice that even in uncorrupted documents the AAAN is not necessarily one of the document's authors! (We have not yet studied what proportion of documents this is true for, or why it occurs.) Setting the probability of corruption to 0.999999 in the case of a clear lack of match was therefore overconfident.

8 ER2b

In ER2b, all test documents were known to have had an author deleted and replaced by another another. The task was to estimate the probability that each of the listed authors was the replacement.

8.1 Effectiveness Measures for ER2b

The stated effectiveness measures for the task were 1) accuracy of identifying the replacement author (under the assumption that the highest probability author is the one being predicted as being the replacement), and 2) cross entropy, i.e. negated mean loglikelihood the model assigns to the correct decision (corrupt vs. noncorrupt) for each test example. Cross-entropy cannot be used as a measure if some predictions are 0 or 1, so the sponsors also reported results for squared error between the probability predicted and a 1/0 target label for corrupt/noncorrupt. The cross-entropy and squared error measures meant there was a premium on making accurate probability estimates, not just ranking the test records in a good order.

8.2 Applying Authorship Models to ER2b

We submitted ER2b runs based both on probabilistic models of authorship and the ad hoc affinity method.

8.2.1 Applying Probabilistic Models of Authorship to ER2b

Our model of the document corruption process was the same as for ER2a. The difference is that we know that every document has been corrupted, and are interested in determining which author is the replacement. As in ER2a, we use Bayes Rule, this time explicitly treating all the possibilities for which author was replaced:

$$p[z = g|D, y] = \frac{p[y|D, z = g]p[z = g]}{\sum_{g'=1}^{k} p[y|D, z = g']p[z = g']}. \tag{18}$$

We assumed all authors had an equal chance of being replaced, so this reduces to:

$$p[z = g|D, y] = \frac{p[y|D, z = g]}{\sum_{g'=1}^{k} p[y|D, z = g']}. \tag{19}$$

Substituting the expression we previously obtained for $p[y|d, z = g]$ (Equation 12) gives

$$p[z = g|D, y] = \frac{\sum_{a \in (A - A_D)} \frac{1}{h(r-h)} p[y|D, z = 0] \frac{1-c_a}{c_a} \frac{1-c_{u[g]}}{c_{u[g]}}}{\sum_{g'=1}^{k} \sum_{a \in (A - A_D)} \frac{1}{h(r-h)} p[y|D, z = 0] \frac{1-c_a}{c_a} \frac{1-c_{u[g']}}{c_{u[g']}}} \tag{20}$$

Eliminating common terms gives the very simple form:

$$p[z = g|D, y] = \frac{\frac{1-c_{u[g]}}{c_{u[g]}}}{\sum_{g'=1}^{k} \frac{1-c_{u[g']}}{c_{u[g']}}} \tag{21}$$

where as before $c_{u[g]} = p[y_{u[g]}|D, z = 0]$ is the probability that the label for author $u[g]$, the g'th observed author of the document, would have the value seen in this document, if the document was not corrupted. Note that we only need to take into account the authors observed in the document, so all the probabilities are actually of the form $c_{u[g]} = p[y_{u[g]} = 1|D, z = 0]$.

Our submitted runs were:

1. *dimacs-er2b-qr-ddl*: Estimates $c_{u[g]} = p[y_{u[g]} = 1|D, z = 0]$ for each listed author of the test document using the smoothed author frequencies, then uses Equation 21 to compute $p[z = g|D, y]$ values to submit. (AAAN features do not affect this approach.)

2. *dimacs-er2b-qr-lin*: Estimates $p[y_{u[g]} = 1|D, z = 0]$ for each listed author of the test document using the smoothed author frequencies. Makes an initial estimate $p'[z = g|D, y] = p[y_{u[g]} = 0|D, z = 0] = 1 - p[y_{u[g]} = 1|D, z = 0]$ that each $u[g]$ is the corrupt author. Then divides those estimates by their sum to get a set of $p[z = g|D, y]$ values that sum to 1.0. (AAAN features do not affect this approach.)

3. *dimacs-er2b-single-ddl*: Estimates $c_{u[g]} = p[y_{u[g]} = 1|D, z = 0]$ for each listed author of the test document using binary logistic regression models and applies Equation 21. AAAN features are used.

4. *dimacs-er2b-NAN-single-ddl*: Same as dimacs-er2b-single-ddl, but AAAN features are not used.

5. *dimacs-er2b-single-lin*: Estimates $p[y_{u[g]} = 1|D, z = 0]$ for each observed author of the test document using binary logistic regression models. Linearly scales values of $1 - p[y_{u[g]} = 1|D, z = 0]$ to sum to 1.0 as in dimacs-er2b-qr-lin. AAAN features are used.

6. *dimacs-er2b-NAN-single-lin*: Same as dimacs-er2b-single-lin, but AAAN features are not used.

8.2.2 Applying the Affinity Model to ER2b

For ER2b run dimacs-er2b-pair-pk we used this ad hoc formula to give authors with high affinity to a document a lower probability of being the replacement author, while forcing the resulting "probabilities" to sum to 1.0:

$$p[z = g|D] = \frac{\sum\limits_{a \in A_D, a \; nequ[g]} \mathrm{Aff}(a, D)}{\sum\limits_{f=1}^{k} \sum\limits_{a \in A_D, a \; nequ[f]} \mathrm{Aff}(a, D)} = \frac{\sum\limits_{a \in A_D, a \; nequ[g]} \mathrm{Aff}(a, D)}{(h-1) \sum\limits_{a \in A_D} \mathrm{Aff}(a, D)} \tag{22}$$

Our submitted runs were

7. *dimacs-er2b-pair-pk*: An affinity-based method. AAAN features are not used.

8. *dimacs-er2b-pair-ddl*: An affinity-based method. AAAN features are not used. This used a different ad hoc normalization technique than described above. This run was not particularly effective and will not be further discussed.

8.3 Adjustments for AAAN Information in ER2b

As for ER2a, our AAAN runs compared the AAAN for a record with all author names in the entire data set. If there were zero or multiple matches, we ignored the AAAN. If there was exactly one match, and if the AAAN author exactly matches one of the test document authors, we replace our original estimate of $p[z = g|D, y]$ with the value 0.000001. We then rescale the other probabilities linearly (in proportion to their original values) so that the sum of probabilities is 1.0. If there was exactly one match of the AAAN to the set of author names, but that author name was not one present in the test document, we did not attempt to modify our results, since this did not provide much new information.

Runs using these adjustments were:

9. *dimacs-er2b-qr-ddl-X*: Same as dimacs-er2b-qr-ddl, but using the AAAN-based adjustment.

10. *dimacs-er2b-qr-lin-X*: Same as dimacs-er2b-qr-lin, but using the AAAN-based adjustment.

11. *dimacs-er2b-single-ddl-X*: Same as dimacs-er2b-single-ddl, but using the AAAN-based adjustment.

12. *dimacs-er2b-single-lin-X*: Same as dimacs-er2b-single-lin, but using the AAAN-based adjustment.

9 ER2c

In ER2c, all test documents were known to have had an author deleted, and that the deleted author was one of a list of 3051 possibilities. The task was to estimate, for each document, the probability that each of the 3051 listed authors was the deleted author.

9.1 Effectiveness Measures for ER2c

The stated effectiveness measures for ER2c were 1) accuracy at rank 1, i.e. accuracy of identifying the deleted author under the assumption that the highest probability author is the one being predicted as deleted; 2) average rank of the deleted author (with ranks below 100 treated as 100); and 3) cross entropy, i.e. negated mean loglikelihood the model assigns to the choice of the correct missing author. The cross-entropy measure favored accuracy probability estimates, while for the other measures only the relative ordering of scores mattered.

9.2 Applying Authorship Models to ER2c

We submitted ER2c runs based on several probabilistic models of authorship plus the ad hoc affinity method.

9.2.1 Applying a Polytomous Model of Authorship to ER2c

Polytomous logistic regression was a straightforward technique for ER2c: we simply had the output variable range over the 3051 possible deleted authors. Training was as described in Section 6.3, and we simply submitted the probabilities output by the model. Two runs of this sort were submitted:

1. *dimacs-er2c-mx-big*: A polytomous logistic regression model is trained using Zhang Lis Maximum Entropy Toolkit [7]. All textual features (including AAAN features) and all ER2 training data authors were used as predictors.

2. *dimacs-er2c-mx-fs5*: Same as dimacs-er2c-mx-big, but predictor features were limited to approximately 15000 features. The five features with the highest F1 measure on the training set was chosen for each class.

9.2.2 Applying Binary Models of Authorship to ER2c

Since we had already constructed binary models of authorship for use in ER2a and ER2b we attempted to apply them to ER2c as well. This required renormalizing the predicted probabilities of authorship so that they summed to 1.0 over the 3051 possible authors. Two normalization schemes were used. Linear scaling added up all the predicted probabilities and divided each by the sum. Exponential scaling uses the logistic transform, replacing initial probability estimates $p_1, ..., p_k, ...p_r$ with

$$p_k^{\text{new}} = \frac{\frac{p_k}{1-p_k}}{\sum_{k'=1}^{r} \frac{p_{k'}}{1-p_{k'}}} \tag{23}$$

The submitted runs were

3. *dimacs-er2c-qr-exp*: Estimates $p(y_k = 1|D)$ values for each of the 3051 possible deleted authors using smoothed author frequencies, then normalizes them to sum to 1.0 using exponential scaling. (AAAN features do not affect this approach.)

4. *dimacs-er2c-qr-lin*: Estimates $p(y_k = 1|D)$ values for each of the 3051 possible deleted authors using smoothed author frequencies, then normalizes them to sum to 1.0 using linear scaling. (AAAN features do not affect this approach.)

5. *dimacs-er2c-single-exp*: Estimates $p(y_k = 1|D)$ values for each of the 3051 possible deleted authors using binary logistic regression models, then normalizes them to sum to 1.0 using exponential scaling. AAAN features are used.

6. *dimacs-er2c-NAN-single-exp*: Same as dimacs-er2c-single-exp, but AAAN features are not used.

7. *dimacs-er2c-single-lin*: Estimates $p(y_k = 1|D)$ values for each of the 3051 possible deleted authors using binary logistic regression models, then normalizes them to sum to 1.0 using linear scaling. AAAN features are used.

8. *dimacs-er2c-NAN-single-lin*: Same as dimacs-er2c-single-lin, but AAAN features are not used.

The runs using smoothed author frequency had dismal effectiveness. Runs using binary logistic regression models were somewhat better, but on most measures not as good as the polytomous and affinity runs.

9.2.3 Applying the Affinity Model to ER2c

For ER2c, a high affinity was suggestive that the author was the one deleted, so in run dimacs-er2c-pair-pk we used the heuristic of simply linearly scaling affinities to sum to 1.0:

$$p[y_k = 1|D] = \frac{\text{Aff}(k, D)}{\sum\limits_{l \in A_{delete}} \text{Aff}(l, D)} \tag{24}$$

A different ad hoc normalization was used in dimacs-er2c-pair-ddl. It was not as effective and will not be further discussed.

9. *dimacs-er2c-pair-ddl*: An affinity-based method. AAAN features are not used.

10. *dimacs-er2c-pair-pk*: An affinity-based method. AAAN features are not used.

9.3 Adjustments for AAAN Information in ER2c

As for ER2a and ER2b, our AAAN runs compared the AAAN for a record with all author names in the entire data set. If there were zero or multiple matches, we ignored the AAAN. If there was exactly one match, and if the AAAN author exactly matches one author, and that author is one of the 3051 possible deleted authors, we replaced the original estimate of probability that author was deleted to 0.999999. We then rescaled the set of probability estimates linearly to sum to 1.0. Otherwise, no AAAN information was used.

Runs using this technique were:

11. *dimacs-er2c-qr-exp-X*: Same as dimacs-er2c-qr-exp, but using the AAAN-based adjustments.

12. *dimacs-er2c-qr-lin-X*: Same as dimacs-er2c-qr-lin, but using the AAAN-based adjustments.

13. *dimacs-er2c-single-exp-X*: Same as dimacs-er2c-single-exp, but using the AAAN-based adjustments.

14. *dimacs-er2c-single-lin-X*: Same as dimacs-er2c-single-lin, but using the AAAN-based adjustments.

10 Summary of the Results

Here we briefly describe the methods that gave best results on each task.

On ER1a submission DimacsER1a.sim.L2.b had the best Area Under ROC curve (AUC) result over all submissions. In this submission the similarity is computed as an inner product with L_2 norm (Cosine measure) applied to binary representations. Comparing with the other submissions, the L_2 norm seems to provide the best results with respect to AUC, though using L_4 norm also gave a high AUC score. Using tf.idf representation led to worse results than using binary representation. The results weren't as high using squared error or accuracy measures (i.e. measures dependent on exact values rather than just on the ordering of scores).

On ER1b several of our submissions topped the rankings based on accuracy (the only measure used for this task). The best submission was dimacs-er1b-modelavg-tfidf-aaan. There the probabilities for all document pairs as given by 11 CLUTO and 1 Document Similarity models were summed up and normalized by the number of models. (the vectors included AAAN info). Submission dimacs-er1b-modelavg-tfidf-noaaan, had the second best score (and was close to the first) without using AAAN info in the vectors. Finally, in the third place was also a submission (dimacs-er1b-modelavg-binary-noaaan) produced by combining multiple models, except those based on binary representation.

On ER2a the measures used for evaluation of results were accuracy, squared error, roc area (AUC) and cross entropy. Submission dimacs-er2a-single-X was ranked 3rd by accuracy and 4th by AUC. This submission was generated by training binary logistic regression, using binary vectors (with AAAN info). The probability of "no replacement" is the product of conditional probabilities for individual authors. There was also additional post-processing using AAAN info. Not using AAAN post-processing (dimacs-er2a-single) led to somewhat worse results (4th by accuracy and 6th by AUC).

Submission dimacs-er2a-NAN-single was 4th by cross entropy, though 6th and 8th by accuracy and AUC. It also used binary logistic regression on binary vectors but without AAAN info in the vectors. It did not use AAAN post-processing. Surprisingly, a submission based only on author frequency in the corpus (dimacs-er2a-qr-X) and with AAAN post-processing was 5th based on AUC.

None of our methods did particularly well on ER2b task (evaluated based on accuracy of the top prediction, squared error and cross entropy with target 1). The best was dimacs-er2b-single-ddl, ranked 10th, 7th and 6th respectively. It used binary logistic regression models from binary vectors (include AAAN info). The scores were used to compute $p(y_k|D)$, probability of correctly listed author, for each author a_k listed for document D.

On ER2c the measures of performance were accuracy of the top prediction, average rank of the correct prediction and cross entropy with target 1.

Submission dimacs-er2c-pair-pk used counts of pair-wise author co-occurrences to estimate a probability that a particular author is "the best fit" given an author list of a paper. it was ranked 4th and 3rd in accuracy measures, but only 13th on cross-entropy. A rather different submission, dimacs-er2c-mx-fs5, produced by multinomial logistic regression model trained on binary vectors (projected on the union of top 5 features by F1 measure for each class), was ranked 5, 8 and 6 respectively. Using all features (dimacs-er2c-mx-big) resulted in ranks 7, 6 and 3.

Several conclusions can be drawn from these results. First, for the clustering tasks ER1a and ER1b, combining output of multiple methods resulted in much better predictions that using any one of those methods separately.

On the ER2 tasks, logistic regression has shown itself to be a powerful method, but the results were also strongly affected by the choice of representation and different uses of

AAAN features. Finally, it is important to note that a “social” approach based on author co-occurrences produced good results on ER2c, suggesting that such methods deserve further consideration.

11 Conclusions

The KDD Challenge exercise provided a challenging simulation of critical real world data mining tasks. We congratulate the sponsors on their work.

The problem ER1b is in some sense ‘completely attackable’, because we were able, we felt, to construct realistic models of this problem based on the data provided, so that we could really study the relative effectiveness of many different approaches. It is unfortunate that the number of possible approaches (representations, metrics, models) grows multiplicatively, so that it was by no means possible to explore them all. Among the directions that could not be explored fully we are particularly intrigued by two.

The first is to explore methods for developing relative weights for the several parts of the feature set. Rough exploration showed that, in problems with known solution, up-weighting some fields, such as author name, improved the results. But we were not able to couple this (grid search) exploration with our more sophisticated methods for selecting and weighting features, growing from Bayesian modeling.

The second is to explore more thoroughly the potential of models in which the text (including keywords and title words) is taken as the primary probe for identifying authors. This approach, which is somewhat dual to a straight networking approach, seems to have great potential. This is somewhat surprising to us since author identification per se has proved to be quite difficult. It leads us to suspect that among the methods represented in KDD, those with multi-layer generative models may prove effective for this problem.

In sum, our work on ER1 did not fully address the problems of term selection or pruning, and of term weighting. In our efforts on ER1 the first of these problems was handled with ad hoc rules (although cPCA does provide a subsequent term selection in some sense). The second problem was not really addressed at all, as we did only preliminary investigations of the effect of differential term weighting on the results of clustering.

On the other hand, we are guardedly optimistic about our attacks on ER2 because it seems that our BBR methods, when used in this computationally intensive fashion, should provide good solutions to the two problems of selection and weighting.

Finally, this has been important opportunity to examine our specific tools (such as cPCA, and BBR) and our ways of dissecting the problem, which are shaped by substantial experience in information retrieval and text classification, in the context of the full KDD effort. We are looking forward to a scientific exchange of ideas at the wrap-up meeting which should make it possible to identify and bring together the best from several teams, for refinement and transition to the working environment.

12 Acknowledgments

This work was supported by a KDD Challenge contract 2003 S518700 00 to DIMACS/Rutgers from IBM. We would like to thank Jordanna Chord and Melissa Mitchell - summer REU students whose work supported development of several top-ranked methods for clustering authors, and resolving author identity.

References

[1] A. Anghelescu and I. Muchnik. Combinatorial pca and svm methods for feature selection in learning classifications (applications to text categorization). In *Proceedings of KIMAS 2003*. http://www.datalaundering.com/download/cpcacsvm.pdf

[2] A. Genkin, D. D. Lewis, and D. Madigan. Large-scale bayesian logistic regression for text categorization. Software: http://mms-01.rutgers.edu/ ag/BBR/.

[3] A. V. Genkin and I. B. Muchnik. Fixed points approach to clustering. *Journal of Classification*, 10:219–240, 1993. http://www.datalaundering.com/download/fixed.pdf

[4] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. M. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *"Journal of Machine Learning Research*, 1:49–75, 2000.

[5] G. Karypis. Cluto a software package for clustering high dimensional datasets. http://www-users.cs.umn.edu/˜karypis/cluto/, 2003.

[6] E. I. Kuznecov, I. Muchnik, and L. Shvartzer. Monotonic systems and their properties. pages 29–57. 1985. Translated from Russian by D. Fradkin. Available at: http://mms-03.rutgers.edu/˜dfradkin/papers/Monotone.pdf. http://www.datalaundering.com/download/monsysp.pdf

[7] Z. Le. Maximum entropy modeling toolkit for python and c++. http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html, 2005.