Occurrence Statistics of Entities, Relations and Types on the Web

Aman Madaan, Under the Guidance of Prof. Sunita Sarawagi
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

May 2014

Contents

1 Evaluating AIDA **4**
1.1 Introduction . . . 5
1.2 Setup . . . 5
1.2.1 Labeled data . . . 5
1.2.2 Getting annotations . . . 5
1.2.3 Annotated files . . . 6
1.2.4 Code . . . 6
1.3 Score . . . 7
1.4 Results . . . 7
1.4.1 Setup 1 . . . 7
1.4.2 Setup 2 . . . 11
1.5 Analysis . . . 12

2 Class Ratio estimation using mmd on "easy" datasets **13**
2.1 Introduction . . . 13
2.2 Twonorm Dataset . . . 13
2.2.1 About . . . 13
2.2.2 Sampling . . . 13
2.2.3 Approach . . . 13

2.2.4 Results . . . 14
2.3 Nist Digit classification data . . . 15
2.3.1 About . . . 15
2.3.2 Sampling . . . 15
2.3.3 Results . . . 15
2.3.4 RMS values of error . . . 15
2.4 Observations . . . 16

3 Multiclass SVM for entity disambiguation 22
3.1 abstract . . . 22
3.2 Features . . . 22
3.3 Finding Parameters . . . 23
3.4 Data . . . 24
3.4.1 Sampling . . . 24
3.4.2 Training Data . . . 24
3.4.3 Test Data . . . 25
3.5 Results . . . 26
3.6 Observations . . . 26

4 Structured learning for entity disambiguation 27

5 Introduction 27
5.1 Features . . . 27
5.2 Data . . . 28
5.2.1 csawLabel file . . . 28
5.2.2 aidaLabel file . . . 28
5.2.3 Label file details . . . 28
5.2.4 Data Pruning . . . 29
5.2.5 Pruned Label file details . . . 29
5.2.6 Sampling . . . 29
5.3 Training Data . . . 29
5.3.1 Test Data . . . 30
5.4 Training the model to predict entities using struct learn . . . 31
5.4.1 Code . . . 31
5.4.2 Bias Feature . . . 32
5.4.3 Weights . . . 32
5.5 Results . . . 33
5.5.1 Do the weights help at all? . . . 33

6 Applying Maximum Mean Discrepancy for obtaining Entity Occurrence Ratios 34
6.1 Features . . . 34
6.2 Data . . . 35
6.2.1 Sampling . . . 35
6.3 Training Data . . . 35
6.3.1 Test Data . . . 35

7 Code **35**
7.1 Gradient 35
7.2 Kernel 36
7.3 Neddata 36
7.4 Finding Parameters 36

8 The case of impossible ratios **36**

9 Modified optimization objective **37**

10 Rounding the fractions to obtain the "correct' ratios **37**

11 Results **37**
11.1 Plots 38

12 Results for different sets **41**

13 Error Analysis **42**

14 Conclusions **42**

Abstract

1 Evaluating AIDA

1.1 Introduction

Aim of this exercise was to benchmark AIDA by obtaining entity annotations on an already (hand) annotated dataset. A low recall is observed, caused by failure to resolve ambiguities correctly. The low recall can additionally be attributed to a large number of non named entity tags like "Loss" . For about 62% of the 548 entities which were annotated by both CSAW team and AIDA, the number of annotations exactly match. For about 87% of such entities, the number of annotations is within ± 5% of the real value.

1.2 Setup

1.2.1 Labeled data

The ground truth data was taken from `http://www.cse.iitb.ac.in/soumen/doc/CSAW/Annot/`

- 104 annotated files
- 3795 entities
- 19000 spots

The annotations are available in XML format. A sample annotation is as follows :

```
 <annotation>
<docName>ganeshTestDoc.txt</docName>
<userId>amitsingh</userId>
<wikiName>Sachin Tendulkar</wikiName>
<offset>420</offset>
<length>9</length>
</annotation>
```

1.2.2 Getting annotations

Annotations from AIDA were obtained in the same XML schema as used by the the CSAW team. The userId field was set to "AIDA".

```
 <annotation>
<docName>ganeshTestDoc.txt</docName>
<userId>AIDA</userId>
<wikiName>Mike Denness</wikiName>
<offset>17</offset>
<length>12</length>
</annotation>
```

Both the annotation files were then read into a hashmap of the form, $h(Entity) -> Count$. Maps for the two files were then scanned to generate the statistics presented in the next section.

1.2.3 Annotated files

XML file containing annotations for 104 files as above and for other 560 files is available at `www.cse.iitb.ac.in/~amanmadaan/store`

1.2.4 Code

The source code (without jar files) is hosted at `http://github.com/madaan/aida_benchmark`

1.3 Score

For a given entity, we calculate the number of times it has been annotated by the CSAW team, and the number of times it was discovered by AIDA.We define the score for an entity, e, using these counts as follows

$$score(e) = \frac{\#\text{Annotations of e by AIDA}}{\#\text{Annotations of e by CSAW}} \tag{1}$$

A score of 1 for an entity e tells that both the ground truth data and AIDA agree on all the mentions of e.

The score is undefined for entities which were annotated by only one of the parties. Various statistics on this score are then used as metric of evaluation.

1.4 Results

1.4.1 Setup 1

Details AIDA was run in *FastLocalDisambiguation* mode. The Corpus used was yago trained from 2010 version of Wikipedia. A total of 104 documents were annotated.

Statistics

Total entities annotated by AIDA	989
Total entities annotated by CSAW Team	3795
Total entities common to both	548
CSAW Entities missed by AIDA : (full list follows)	3247
AIDA Entities missed by CSAW : (full list follows)	441

Table 1: Statistics on Entities

Figure 1: Entities Annotated by CSAW and AIDA

n	548
min	0.03333333333333333
max	10.0
$mean$	1.1219961591723266
std_dev	0.9419811831930233
$median$	1.0
$skewness$	5.1513317988869085
$kurtosis$	34.719172895503725

Table 2: Statistics for $score = \frac{\text{Annotations by AIDA}}{\text{Annotations by CSAW}}$

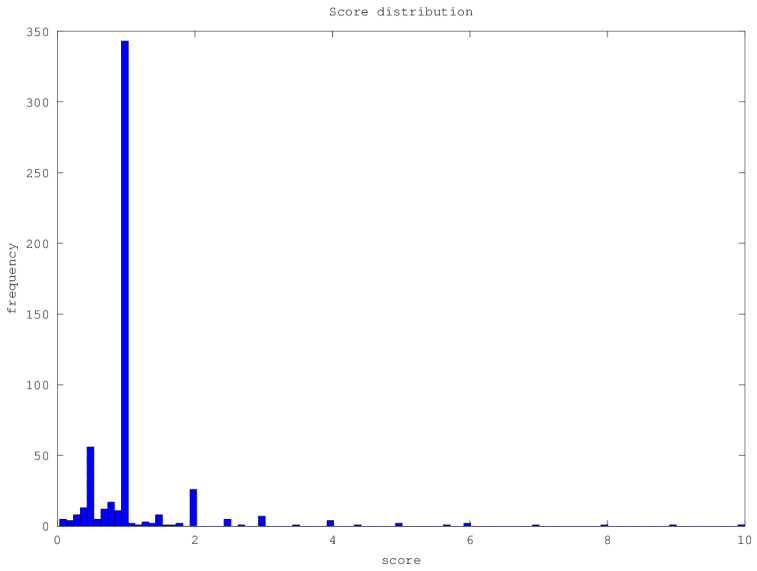

Figure 2: Distribution of scores for entities annotated by both CSAW and AIDA. CSAW and AIDA agree on 62% entities which match (score = 1)

Score	Frequency
0.03	1
0.17	1
0.20	3
0.25	7
0.31	1
0.33	7
0.50	52
0.60	1
0.67	9
1.00	342
1.50	8
2.00	26
4.00	4
4.33	1
5.00	2
5.67	1
6.00	2
7.00	1
8.00	1
9.00	1
10.00	1

Table 3: Distribution of $score = \frac{\text{Annotations by AIDA}}{\text{Annotations by CSAW}}$

1.4.2 Setup 2

Details AIDA was run in *CocktailPartyDisambiguation* mode. The Corpus used was yago trained from **2012** version of Wikipedia. A total of 104 documents were annotated.

Statistics

Total entities annotated by AIDA	911
Total entities annotated by CSAW Team	3795
Total entities common to both	510
CSAW Entities missed by AIDA : (full list follows)	3285
AIDA Entities missed by CSAW : (full list follows)	401

Table 4: Statistics on Entities

n	510
min	0.045454545454545456
max	9.0
$mean$	1.1572691785583593
std_dev	0.9019959891067724
$median$	1.0
$skewness$	4.558145927208736
$kurtosis$	27.172766135955523

Table 5: Statistics for $score = \frac{\text{Annotations by AIDA}}{\text{Annotations by CSAW}}$

1.5 Analysis

- **AIDA annotates 989 entities, but only 548 of them were also in CSAW** There can be several reasons for this :
  - **Ambiguity of tagging**. Gandhi was wrongly linked to "Indira Gandhi" and not "Mahatma Gandhi" at several places.
  - **Different Wikipedia titles for same entity**. "Reuters" can be "Thomson Reuters" or "the news agency". Since the comparison was based on the title of the Wikipedia page that is linked, such annotations fail to match.
  - **New Wikipedia pages** might have emerged since CSAW was annotated. For eg, Harald zur Hansen is present in the corpus but not tagged by CSAW team.
- out of 548 matches, 342 **(62%)** were *exactly* annotated. This means that AIDA performs as good as a human being for these cases.
- The hand annotated data missed some tags. In the file 13OctAmitSport14.txt, only 1 out of 9 instances of Steve Hodge were tagged. (Steve Hodge was mentioned by just the last name in the 8 other places)
- There were some tags to non named entities; words like Loss, Excellency, Utility, Year etc. were tagged with links to the respective Wikipedia pages. **The low recall**, $(0.2606, \frac{989}{3795})$ seems to be caused by such extraneous tags.
- The performance with both local and global disambiguation settings was almost the same.

2 Class Ratio estimation using mmd on "easy" datasets

2.1 Introduction

The aim of this excercise was to estimate class ratio using mmd for datasets on which svm performs really well. The datasets used are the Twonorm dataset (2 classes) and the NIST digit classification dataset (10 classes). Standard svm implementations give an accuracy of around 98% on both the datasets. This report outlines the methodology and the results.

2.2 Twonorm Dataset

2.2.1 About

Features	20
Feature type	Numerical
Classes	2 (0 / 1)
Instances	7400

2.2.2 Sampling

90% of the instances were used for training and the rest were used for testing. The code for sampling is located at `/mnt/a99/d0/aman/workspace/mmdsg/src/Sampler.java` The initial class ratios were 0.50 in both training and test data.

2.2.3 Approach

The training data (and hence distribution of class labels) was kept fixed and instances belonging to class 0 were deleted from the test data to obtain the results. The plots of error and the estimates predicted by mmd approach are as follows.

2.2.4

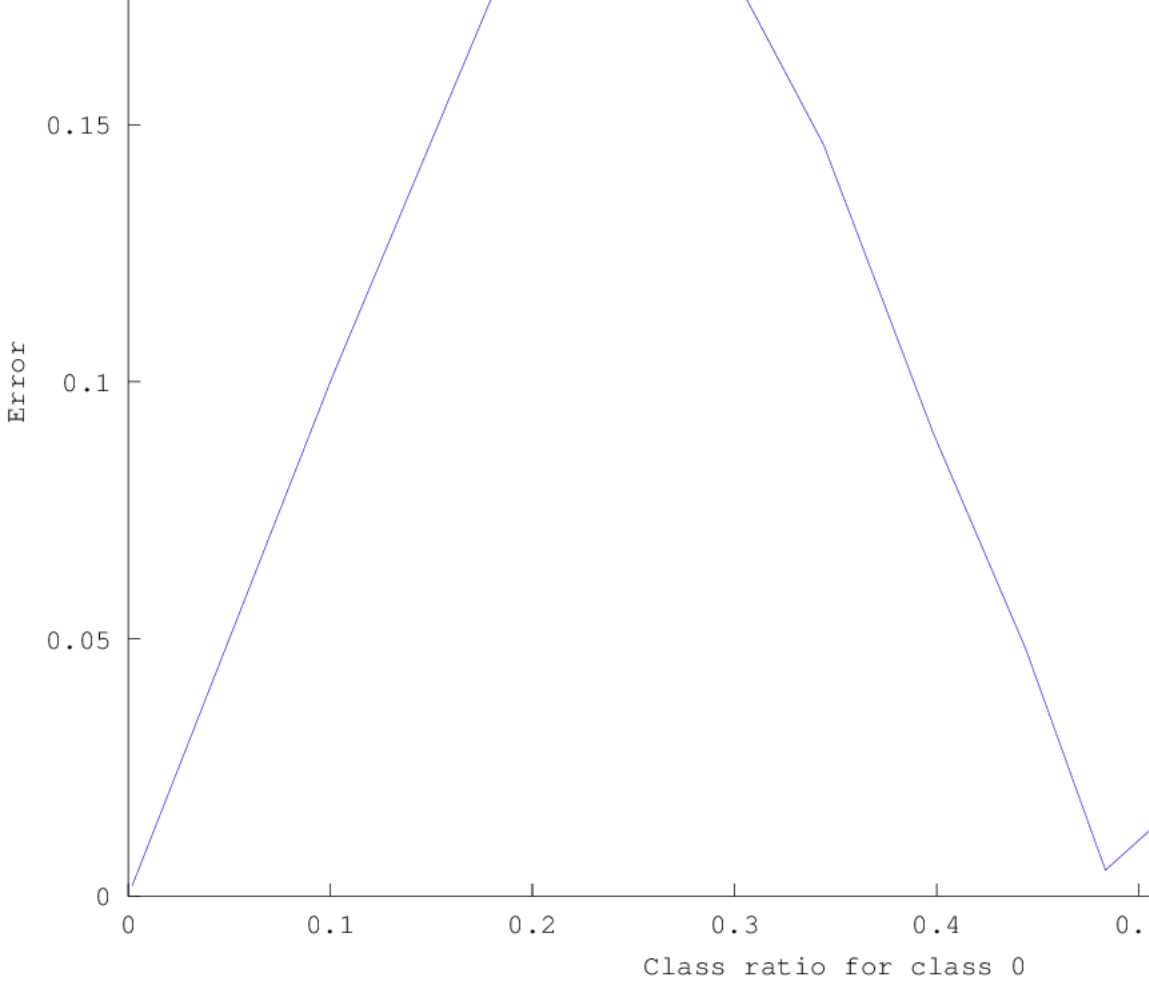

Figure 3: Error plot (abs of the difference between real and predicted class ratios)

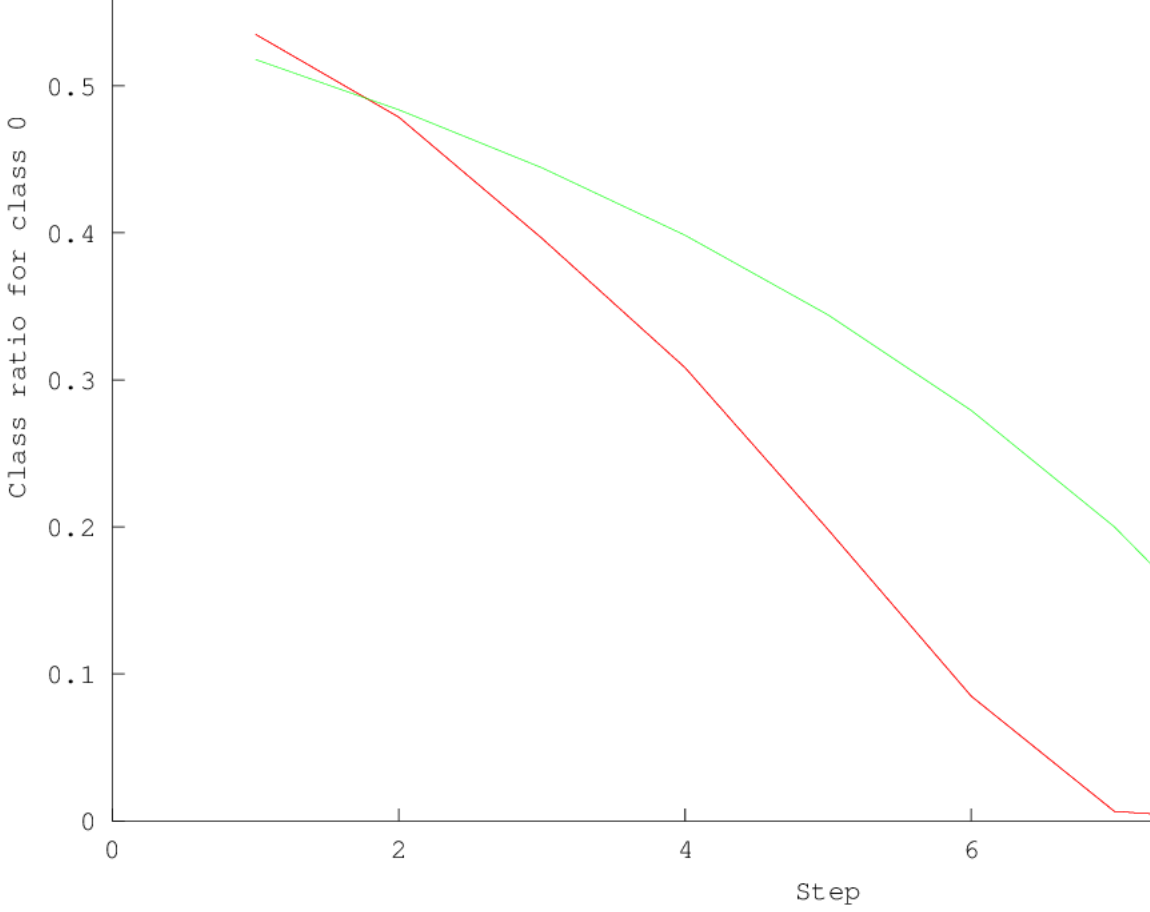

Figure 4: Class ratios predicted using mmd

2.3 Nist Digit classification data

The dataset was obtained from `http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#`

2.3.1 About

Classes	10
Samples per class	180
Samples total	1797
Dimensionality	64
Features	integers 0-16

2.3.2 Sampling

From 1797 instances, 1617 were used for training. Class ratios were obtained for the remaining test set of 180 instances and randomly sampled 130 instances.

2.3.3 Results

2.3.4 RMS values of error

Dataset	RMS SVM	RMS MMD
180 instances	0.0043033	0.034603
130 instances	0.067937	0.036054

130 test instances

REAL	SVM	MMD
0.061538	0.061538	0.103281
0.107692	0.115385	0.100152
0.084615	0.084615	0.099571
0.069231	0.069231	0.106992
0.046154	0.053846	0.105755
0.138462	0.146154	0.099571
0.123077	0.123077	0.096479
0.100000	0.092308	0.098334
0.115385	0.107692	0.093386
0.153846	0.146154	0.096479

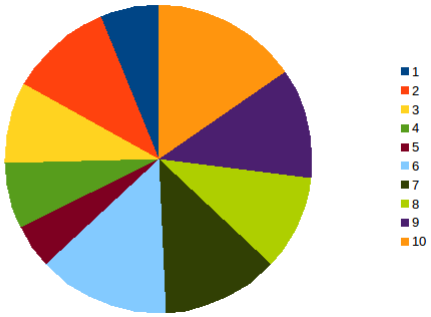

Figure 5: Real Distribution

180 instances

REAL	SVM	MMD
0.061111	0.061111	0.103282
0.111111	0.116667	0.10015
0.088889	0.088889	0.099571
0.055556	0.055556	0.106992
0.055556	0.061111	0.105755
0.116667	0.122222	0.099571
0.138889	0.138889	0.096479
0.111111	0.105556	0.098334
0.127778	0.122222	0.093386
0.133333	0.127778	0.096479

2.4 Observations

- The results improve with the size of training data.

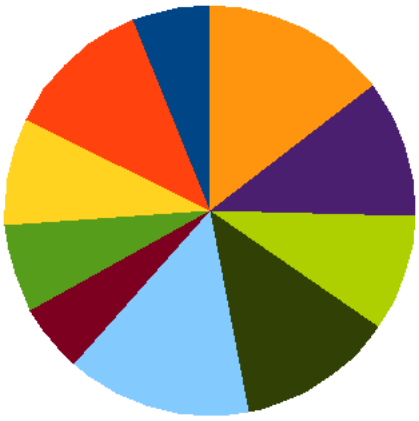

Figure 6: SVM Class ratios

- For datasets for which per instance classification yields good results may not benefit much from mmd based approach. The occam's razor takes a U turn in some sense, estimating class ratios directly becomes the harder problem.
- The mmd approach is **highly** sensitive to the value of gamma. It was observed that using SVM cross validation to choose gamma yields poorer results.

Figure 7: MMD class ratios

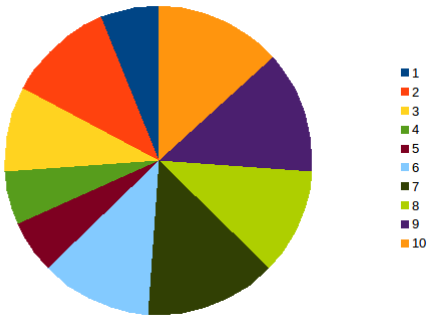

Figure 8: Real Distribution

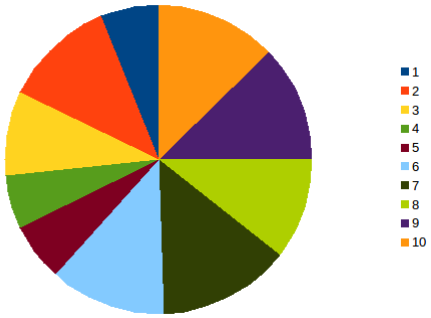

Figure 9: SVM Class ratios

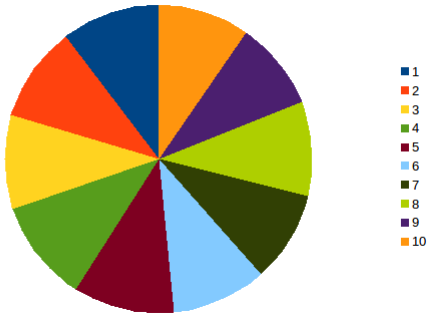

Figure 10: MMD class ratios

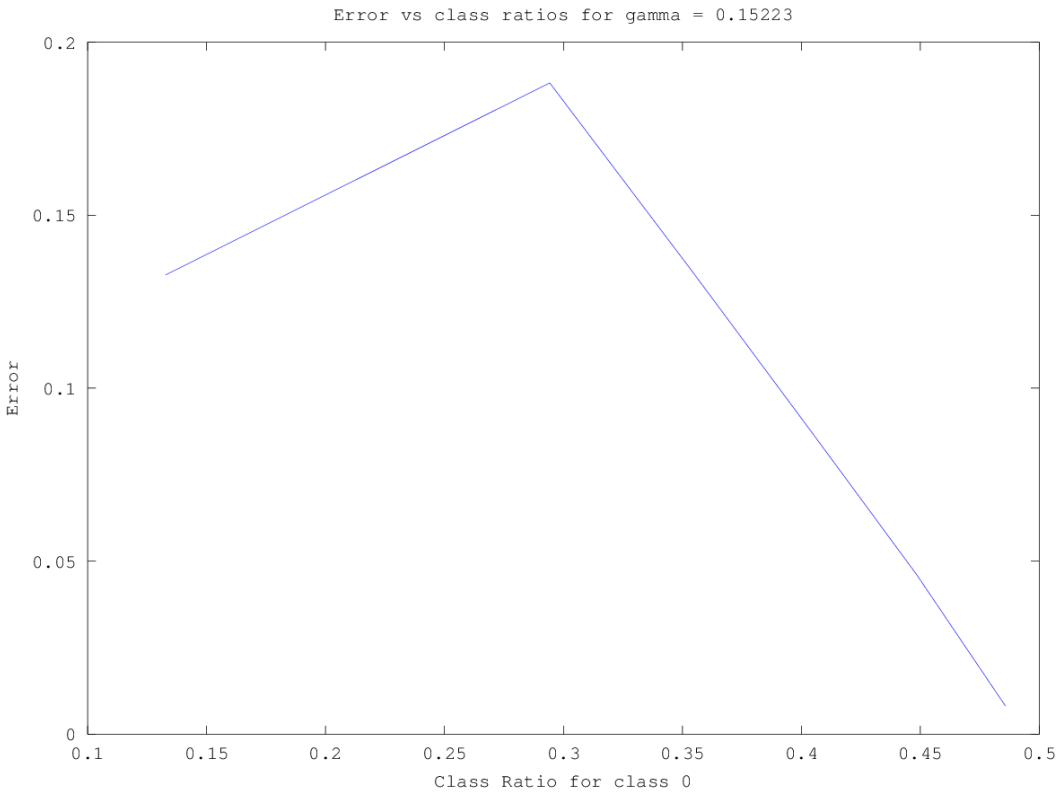

Figure 11: Gamma value choosen by SVM cross validation

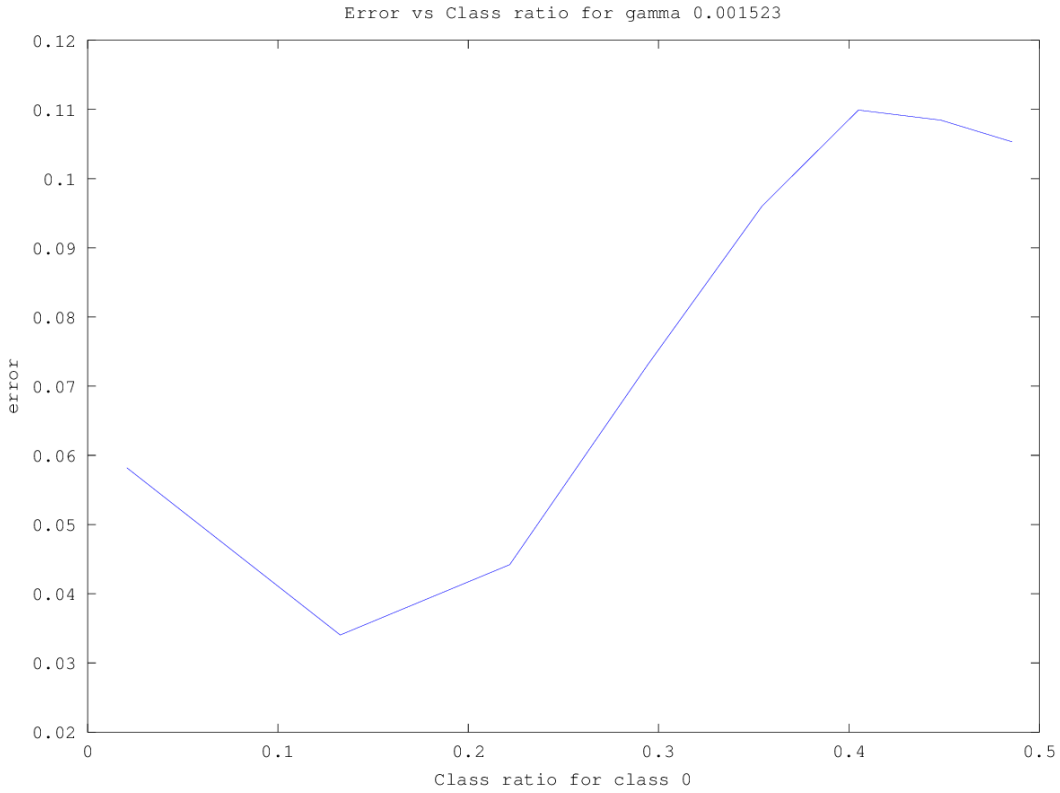

Figure 12: Another gamma value in the feasible range yields considerably better results

3 Multiclass SVM for entity disambiguation

3.1 abstract

Libsvm was used to train a multiclass SVM for disambiguating entities. The features for a mention were based on the mention entity similarity score given by AIDA. The overall accuracy achieved is 73.4%, attributed to insufficient instances per class.

3.2 Features

- For each mention, a sparse feature vector was obtained by using the mention - entity similarity score provided by AIDA.
- The score is as defined in [**?**]. The implementation is the function *public double calcSimilarity(Mention mention, Context context, Entity entity* located at `https://github.com/yago-naga/aida/blob/master/src/mpi/aida/graph/similarity/EnsembleMentionEntitySimilarity.java`.
- The score **includes** a prior for each of the possible candidates if the maximum prior is above a threshold. The prior is **added** to weighted score after scaling (Line 209 in the aforementioned file).
- A mention with true entity E_{true}, and candidates $E_1, E_2, ..., E_n$ with respective scores $W_1, W_2, ..., W_n$ is represented as :

  $$E_1 : W_1 \ E_2 : W_2 \ ... \ E_n : W_n$$

  where E_{true} is the entity E_i with the highest score (W_i).
- An example feature vector (for the mention "Kabul") thus looks like

  1021662 55061:0.006644 62673:0.001566 99131:0.009709 262564:0.006688 400332:0.020700 1021662:0.663744 4373304:0.003100 4809346:0.002034 5331197:0.023680 5566671:0.001442 9356677:0.000080 12146961:0.003348

  The first column represents the true label as required by the libsvm sparse feature vector format.

3.3 Finding Parameters

- grid.py from the libsvm tools collection was used to find the best Gamma and C.
- Range :
  - C : 2^{-5} to 2^{15} with steps of 1
  - Gamma : 2^{-15} to 2^{3} with steps of 2

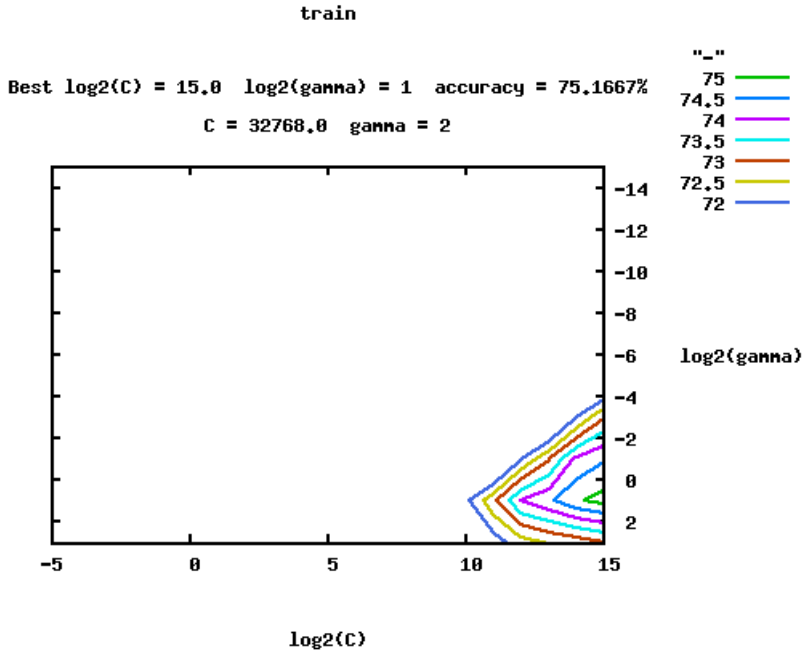

Figure 13: Grid search for SVM

3.4 Data

Total number of classes	952
Classes in Training Data	905
Classes in Test Data	156
Shared classes	109

As the following figure shows, most of the classes (544) just had 1 instance.

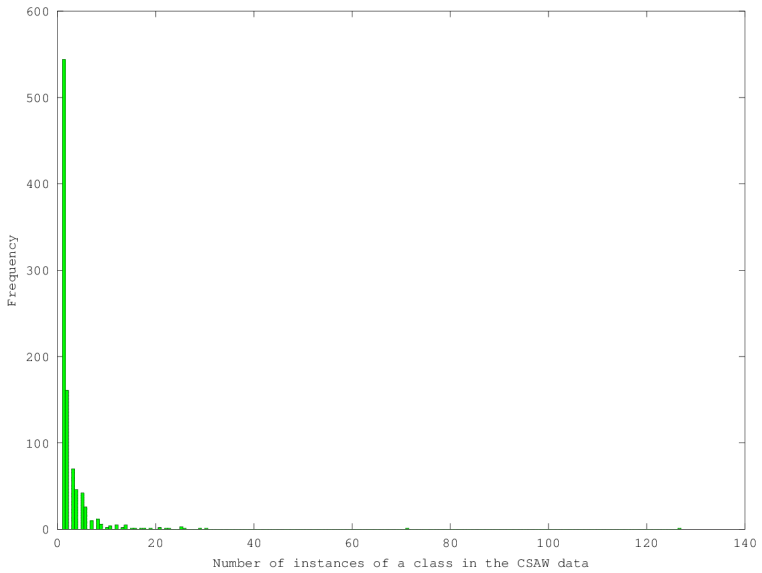

Figure 14: Frequency distribution of the number of instances per class

3.4.1 Sampling

Random sampling was used to divide the data into training and test sets. Since many classes had just 1 instance, stratified sampling could not be used. Subset.py from the libsvm tools was used for this purpose.

3.4.2 Training Data

- 2400 instances
- 905 different classes
- 2.65 instances per class
- Max 114 instances, United States

3.4.3 Test Data

- 196 instances
- 156 classes
- Max 13 instances, United States

3.5 Results

- The SVM achieved an overall accuracy of 73.46 %
- If we only consider classes whose instance was available in the training data, the accuracy shoots to 95.83%.
- As expected, out of these 52 misclassified instances, 46 had labels not which were not present in the training data.
- 45 of the 52 misclassified instances were assigned the label 1937733, Pierre Golle. Pierre Golle has only 5 instances in the training data, and just one instance in the test data.

The code, trained model, results and the data is available at
`https://github.com/madaan/aida_benchmark/tree/master/mclass`

3.6 Observations

- Training data sparsity is a big issue. The classifier understandably fails to do anything about instances it has not seen.
- Even for classes present in the corpus, the number of instance per class was around 2.5, with most of the classes having just 1 instance. While increasing the size of the corpus may increase the frequencies for popular entities, the lack of enough instances will remain a challenge for most of the tail classes.

4 Structured learning for entity disambiguation

5 Introduction

Structured learning was applied to learn the optimal weights for the different features to disambiguate entities. The model performs slightly better than AIDA, perhaps because of giving due importance to the mention prior. The test data had 80 unknown classes and thus SVM performs poorly as expected, giving an accurcy of just 34.5%.

5.1 Features

- For each mention, 3 different scores were used.
  - A. Prior score
  - B. Similarity Score
  - C. A weighted combination of (A) and (B). The weights are given by AIDA.
- The implementation for score (C) is as given in the function *public double calcSimilarity(Mention mention, Context context, Entity entity* located at `https://github.com/yago-naga/aida/blob/master/src/mpi/aida/graph/similarity/EnsembleMentionEntitySimilarity.java`.

  Functions for scores (A) and (B) were implemented on the basis of (C).
- A mention with true entity E_{true}, and candidates $E_1, E_2, ..., E_n$ with respective scores $(W_1a, W_1b, W_1c), ..., (W_na, W_nb, W_nc)$ is represented as :

  $$E_1 : W_1a, W_1b, W_1c \ ... \ E_n : W_na, W_nb, W_nc$$

  where E_{true} is the true entity E_i (Hand labeled)

Where

- Weight A : Weighted similarity and prior where the weights are given by AIDA.
- Weight B : Keyphrase based similarity score given by AIDA
- Weight C : Prior of the mention

5.2 Data

103 hand annotated files having 3100 mentions were annotated using AIDA. The AIDA pipeline was spliced to get the 3 scores for different mentions. After the annotation is complete, we get the following 3 files

- csawLabels : 3100 rows. ith row has the true entity corresponding to the ith mention. (Some of the entries here were "-1". The following subsection elaborates on this)
- aidaLabels : 3100 rows. ith row has the entity label assigned by aida to the ith mention.
- featureFile : 3100 rows. ith row has a list of candidate entities for the ith mention, with scores attached to each candidate in the format above.

5.2.1 csawLabel file

This file has ground truth labels for all the mentions, one per line.

- As stated above, csawLabel file had a lot of "-1" entires.
- A "-1" entry means one of the following :
  - There is no entity corresponding to the mention in the hand labelled data.
  - The entity tagged in the hand labelled data has a renamed wikipedia page.
- A "0" entry in the CSAW data means one of the following:
  - There is no entity corresponding to the mention in the knowledge base.
  - A wikipedia page has been created for the entity now.

5.2.2 aidaLabel file

This file has the labels given by AIDA to each of the mentions.

5.2.3 Label file details

	AIDA LABEL	CSAW LABEL
0	459	534
-1	0	1228
Others	2641	1338

5.2.4 Data Pruning

- Data corresponding to mentions that satisfy either of the following criteria was deleted:
  - Null attachments without any candidate. These are the words that are wrongly tagged by the NER as Named entities. Eg. In a sentence like : "Combine that thrilling energy", "Combine" may be tagged as a named entity
  - CSAW label is -1.

5.2.5 Pruned Label file details

After pruning as above, the label file had 1294 mentions. The details are as follows :

	AIDA LABEL	CSAW LABEL
0	0	8
-1	0	0
Others	1294	1286

5.2.6 Sampling

Top 1094 instances were used for training and the remaining 200 instances were used for testing. Since a majority of the mentions had just 1 mention, it was not possible to create the training data such that the class ratio is maintained.

5.3 Training Data

- 1094 mentions
- 460 different (entities) classes
- 2.37 mentions per entity

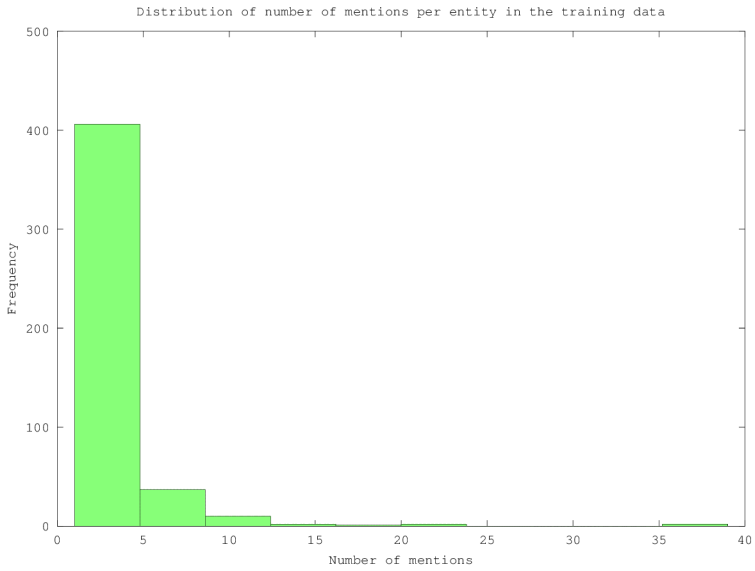

Figure 15: Distribution of Number of mentions per class

5.3.1 Test Data

- 200 mentions
- 108 different (entities) classes
- 80 entities were unknown to the test data
- 1.85 mentions per entity

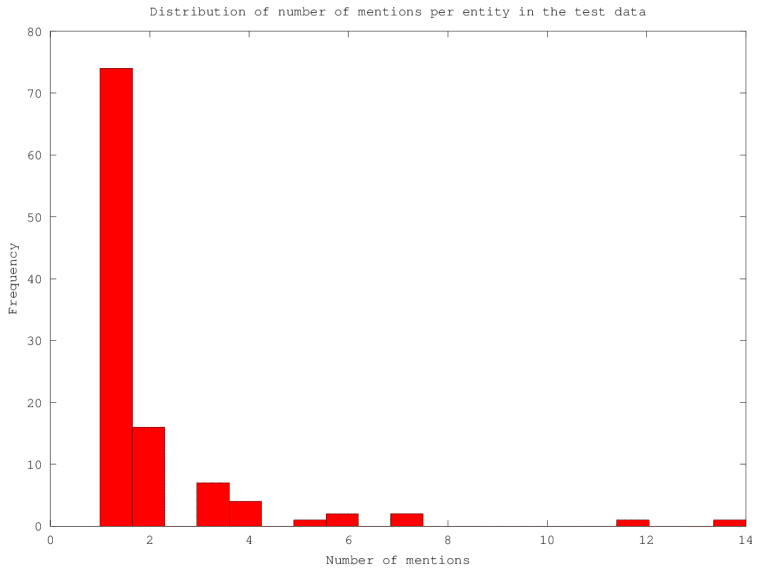

Figure 16: Distribution of Number of mentions per class

5.4 Training the model to predict entities using struct learn

5.4.1 Code

The in house structure learning package was used. The code is located at /mnt/a99/d0/aman/workspace/EntityStructLearn Brief notes on the various classes follow :

MentionFeatureSequence One instance of this class represents a mention. This class wraps the candidates and their respective scores in a hashmap from Entity Ids to a vector of doubles.

MentionDataIter Essentially an ArrayList of MentionFeatureSequence. Exposes the iterator behavior of the ArrayList to be used by the trainer.

DisambConstraintsGenerator Implements getViolatedConstraints :

```
public Vector<Constraint> getViolatedConstraints(double[] lambda,
DataSequence dataSeq, int iterNum, int numRecord,
LossScaler lossScaler)
```

Given a mention feature sequence, the function first calculates the best entity for the mention amongst the candidates given the current value of lambda. If the best entity thus found is different from the true entity, a constraint is returned.

CrossValidateC Finds the best value of C by performing a 5 fold cross validation. The range of C searched is 2^{-5} to 2^{15}

The procedure returned the value of C = 1.

5.4.2 Bias Feature

Two different methods were used to inculcate the biased feature.

- Add to the delFeature vector a value of "1" for violated constraints.
- Set the bias feature value to 1 only for null attachments.

No improvements were observed with either of these methods, owing to small number of null attachments with no candidates.

5.4.3 Weights

WeightedSimPrior	0.22196129094460354
Similarity Score	0.030758649381491955
Prior	0.2245345883216353
Bias	0.6919561243144317

5.5 Results

The test file had 200 mentions. We thus have a golden truth on the 200 labels, and a verdict on these from svm and structlearn model.

Method	Correct Labelings	Accuracy
Ground Truth	200	100%
STRUCT LEARN	176	88%
AIDA	173	86.5%
SVM	69	34.5%

5.5.1 Do the weights help at all?

3 more label files were created, by selecting the best candidate as one which gives the largest weight for a particular candidate.

- priorMaxLabel : Best candidate is the one which has the highest prior
- simMaxLabel : Best candidate is the one with the highest similarity
- bothMaxLabel : Best candidate is the one which has the highest weighted prior and sim score.

STRUCT LEARN	176	88%
PRIOR	170	85%
SIM	148	74%
BOTH	173	86.5%

These results corroborate the correctness of weights (prior and sim score have the highest weight) and also indicate that the weights do matter.

Occurrence Statistics of Entities, Relations and Types on the Web

Aman Madaan, Under the Guidance of Prof. Sunita Sarawagi
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

May 2014

6 Applying Maximum Mean Discrepancy for obtaining Entity Occurrence Ratios

The final exercise was application of Maximum mean discrepancy for learning entity occurrence ratios. We continue with the same dataset, pruned to remove all the entities for which there are only single instances, leaving us with 212 different entities. The results indicate that svm is better than mmd for estimating the class ratios. We discuss the results, possible reasons for error and suggest ways of improvement.

6.1 Features

As discussed, the features for struct learn take the following form

$$E_{true}\ E_1 : W_1a, W_1b, W_1c ... E_n : W_na, W_nb, W_nc$$

where E_{true} is the true entity E_i (Hand labeled) and the weights are as defined before.

We now flatten the features using the maximum entity id as follows :

$$E_{true}\ E_1 + 0 * MEID : W_{1a}E_1 + 1 * MEID : W_{1b}E_1 + 2 * MEID : W_{1c} ... E_n + 0 * i : W_{na}E_n + 1 * i : W_{nb}E_n + 2 * i : W_{nc}$$

where MEID is the highest entity id.

6.2 Data

Out of the pruned data points (1294), we deleted instances for which only one mention per class was available. Upon deletion of such points, we were left with 966 rows.

6.2.1 Sampling

Subset.py from the libsvm tools was used for randomly dividing the training data into training and test sets.

6.3 Training Data

- 807 mentions
- 212 different (entities) classes
- average 3.8 mentions per entity
- median 2 mentions per entity

6.3.1 Test Data

- 159 mentions
- 109 different (entities) classes
- average 1.5 mentions per entity
- median 1 mentions per entity

For this dataset, it was ensured that for every class in the test set, we have at least one instance in the training set.

7 Code

This section briefly outlines the functionality of the different packages in mmdned. The MirrorDescentOptimizer by Prof. Sunita Sarawagi was used for solving the objective.

7.1 Gradient

- MakeMatrices : Calculates matrices A and b required to calculate objective and the gradient
- NedGradientComputer : Calculates the gradient. An instance of NedGradientComputer gradient compute has to be supplied to the optimizer during its instantiation.

7.2 Kernel

Implementation of the Gaussian Kernel to handle the sparse feature representation.

7.3 Neddata

Framework for representation and handling of the sparse features.

- DataSetInfo : Stores standard facts about the dataset (Number of classes, class proportions etc)
- Feature : Represents a feature, consisting of an id and a value
- NedRow : A row consists of a true label followed by a list of features
- NedDataReader : Returns a list of NedRows with DataSetInfo filled in

7.4 Finding Parameters

We use the gaussian kernel to determine similarity between two instances.

$$K(x1, x2) = exp^{-\gamma * ||x1 - x2||^2} \tag{2}$$

Where $x1$ and $x2$ are sparse representations of the features. The only parameter to be supplied is gamma. We find the gamma by using 5 fold cross validation on the training data.

8 The case of impossible ratios

Consider the case where we have 10 unlabeled instances. In such a setting, we *cannot* have class ratios with one of the elements set to 0.99 (9.9 elements belonging to a class is not possible!). In fact, for the current case, any class proportion in which any of the elements takes a value which when multiplied by 10 does not yield an integer is impossible.

Although it is easy to see that not every class proportion is possible given the unlabeled dataset, yet the mmd formulation does not consider this fact right now. This may lead to errors, since we are looking for the solution in the complete R^n space where we *must* be looking in only a proper subset of it. Intuitively, we need to add the following constraint to the mmd objective.

> Let $|U|$ be the number of elements in the unlabeled test set. For any element θ_i of then (unknown) class proportion vector, $\theta_i * |U|$ should be an integer.

9 Modified optimization objective

The current mmd objective can be seen as an approximation to the following optimization objective.

Given an unlabeled dataset U, find the values of n_i such that

$$\Sigma_i^c n_i = N \tag{3}$$

where N is the total number of unlabeled instances, c is the number of classes and n_i is the number of instances belonging to the i^{th} class. Clearly, each of the n_i needs to be an integer. Dividing the objective above by N on both sides and rewriting n_i/N as θ_i gives us the current optimization objective. The problem is that once we do that, there is no way to restrict θ_i to only those values that are legal, i.e. fractions n_i/N where $n_i <= N$.

As an extension to the current work, we can use an IP solver to obtain the required values.

10 Rounding the fractions to obtain the "correct' ratios

A hack to convert the ratios obtained via mmd to one of the possible ratios is as follows. We first multiply each element of the ratio vector by the number of unlabeled instances. Each entry is then rounded and finally normalized to obtained the new ratio.

This boils down to the following matlab operation :

$$r' = round(r * |U|)/sum(round(r * |U|)) \tag{4}$$

Where r' are the rounded proportion vector, r is the original proportion vector and $|U|$ is the number of instances in the unlabeled dataset.

11 Results

We compare the ratio vectors obtained from the different methods. To quantify the difference between two ratio vectors, we use the L1 norm.

$$l1(x1, x2) = \Sigma_{i=0}^{n} |x1_i - x2_i| \tag{5}$$

Method	L1 Norm
SVM	0.20482
STRUCT LEARN	0.22892
MMD	0.5561

We use 3 different plots to visualize the difference between true and obtained ratios.

- **Dispersion Plot** Let k be the total number of elements in the ratio vector. Let *true* be the true ratio vector and let *model* be the obtained ratio vector. Define

$$I = [1, 2, ..., k] \tag{6}$$

$$model'_i = \frac{model_i}{true_i} * i \ \forall i \in I \tag{7}$$

We then plot $model'$ *vs* I. In the ideal case, the plot would be a straight line. Diversion from the straight line thus becomes an indicator of the degree of error of the ratios obtained.

- **Cumulative Frequency Plot** Plot of cumulative sum of the obtained ratios and the original ratios. In the ideal case, the shape of the 2 curves should match.

- **Error Plot** We plot the error of the original ratios and the obtained ratios. In the ideal case, all the points should fall on error = 0 line.

11.1 Plots

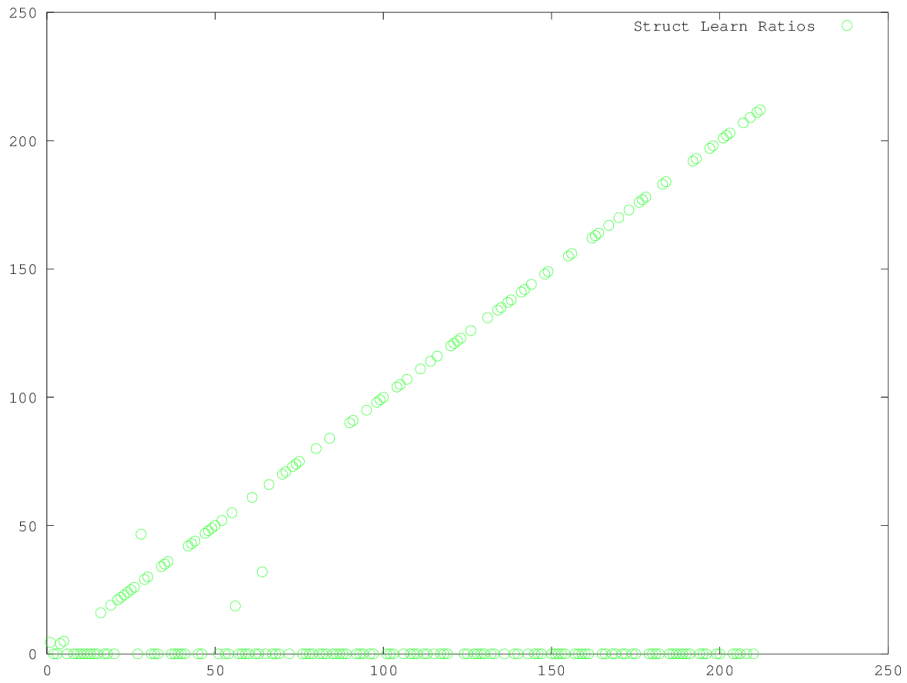

Figure 17: Ratios Obtained via Structured Learning

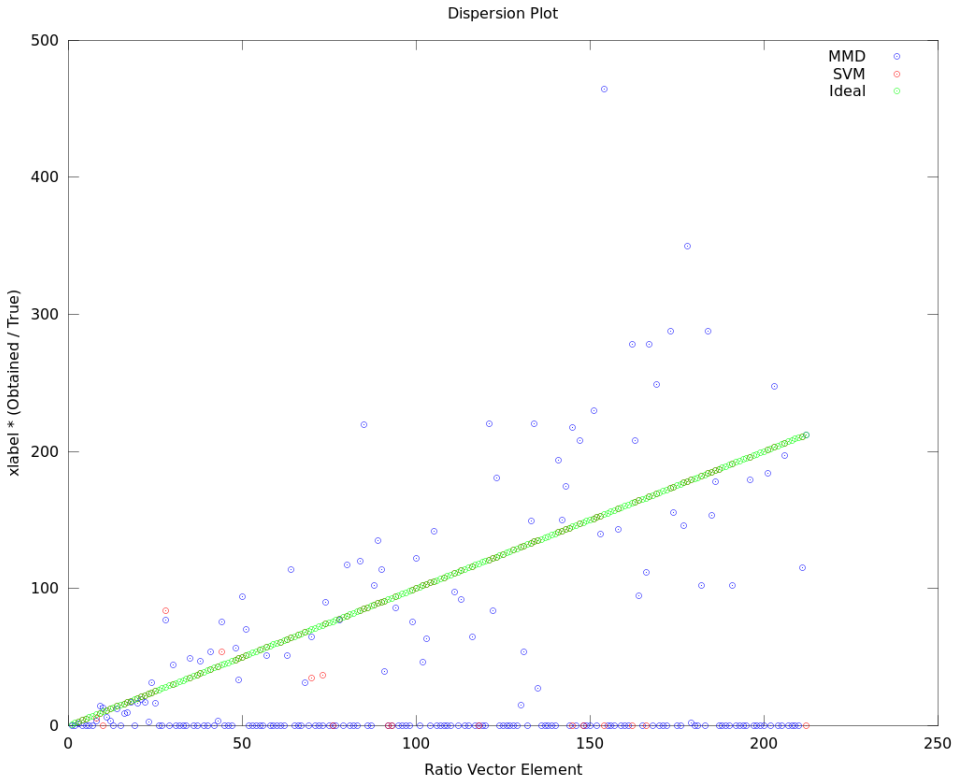

Figure 18: Digression : SVM vs MMD

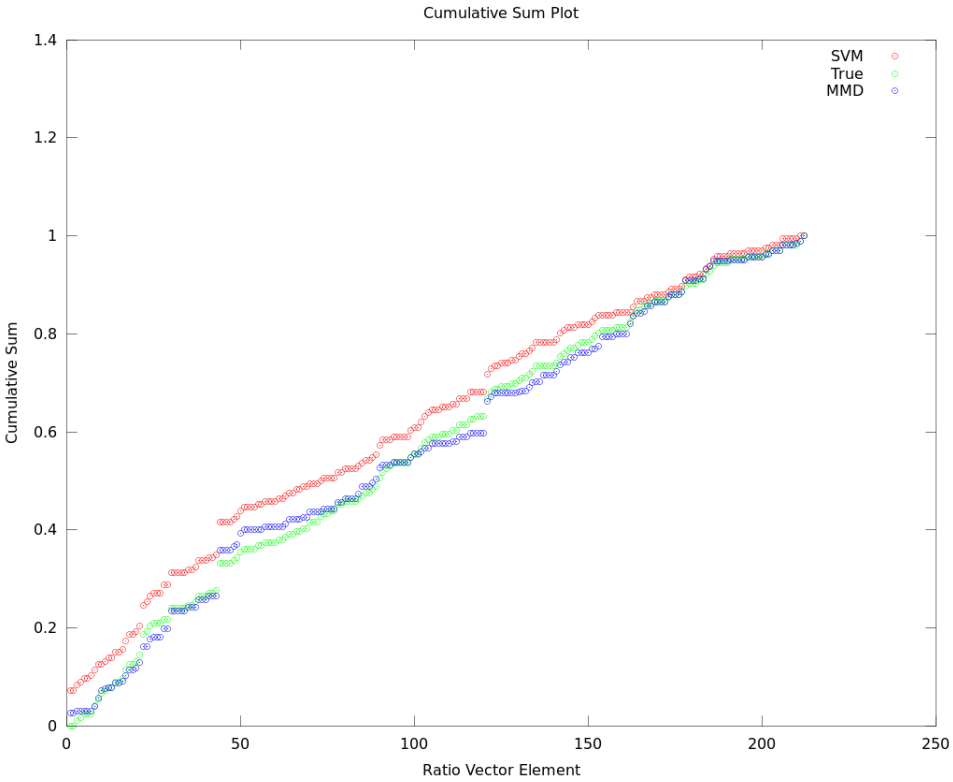

Figure 19: Cumulative Ratio

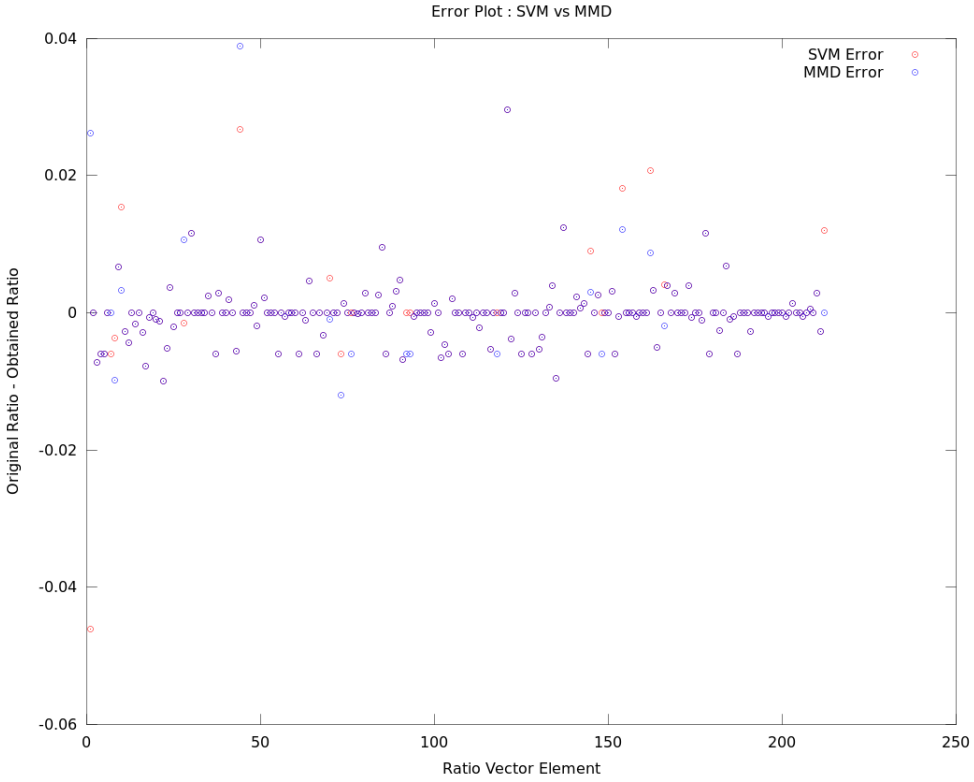

Figure 20: Per Element Error

12 Results for different sets

To test our hypothesis of the rounded ratios, we created 5 different datasets and trained svm and mmd on them. The results are as shown in table [12]. We note that although svm outperforms mmd approach, there is a consistent improvement in performance with rounding. L1 denotes the L1 error as defined above and PC denotes the per class error.

Run	SVM L1	SVM L1 PC	MMD L1	MMD L1 PC	MMD Rounded L1	MMD L1 PC
1	0.13	0.00061	0.577	0.00272	0.553	0.00261
2	0.144	0.00068	0.506	0.00239	0.47669	0.00225
3	0.1083	0.00051	0.54423	0..257	0.49288	0.00232
4	0.14458	0.00068	0.59434	0.00280	0.59036	0.00278
5	0.25301	0.00119	0.58582	0.00276	0.54217	0.00256

Please note that for these datasets, there can be unseen classes in the test set

13 Error Analysis

- **Kernel Selection** Experiments with multiclass datasets have earlier shown that mmd is highly sensitive to kernel selection. The earlier method of cross validation by fixing class proportions cannot work in the current setting because of lack of the training data. We thus use 5 fold cross validation. Improper γ learning can be a major source of error for the mmd methods.
- **Lack of training data** The training data had 4.5 mentions per entity on average. These instances may not be a good representative of an *average* mention for the candidate.

14 Conclusions

- **Restricting thetas** As mentioned, the search space for the θs can be restricted to a great extent by adding constraints to ensure that only possible values of the θs are considered while coming up with the best. This will complicate the existing mmd formulation, but our results have consistently indicated that the results will greatly benefit from such additional constraints.
- **Clustering Entities for disjoint occurrence statistics learning** Can we drive around the problem of dealing with large number of classes by just finding the ratios for each set of disambiguations? For example, can we find the distribution of all the different Michael Jordans on the web by collecting all the mentions that refer to one of these?

  Though the idea looks intuitive, there are multiple challenges involved. First of all, can we even create such disjoint subsets of mentions? Even if we can, how will we ensure that test set will have mentions that only have candidates from the set of Michael Jordans?
- **Cross validation** The existing method of obtaining γ via cross validation depends on varying the class proportions. This method cannot be applied to the case in hand because there aren't enough instances per class. To add to the problem, the number of classes is also very large.