Application Of Information Extraction To Automate Research On Chinese Corruption

Shidan Xu, Tingtao Zhou

December 14, 2015

1 Abstract

This project is an attempt to extract structured information from Chinese news articles reporting corruption cases, as an input for subsequent research on Chinese corruption. In particular, for each article, we are interested in knowing the culprit, the crime and punishment, the job of culprit, and time and location of crime and the corresponding relations.

We used regex matching, logistic modeling classfier, Boson NLP(commercial software), and conditional random field(CRF) for Named Entity Recognition. We then did word alignment and occupation filtering, and utilized distance metrics to aid us in performing relation matching.

In this paper, we discuss the approaches and compromises we made to achieve a 38% recall and 39% precision score (for the entire pipeline), and 70% and 68% normalized scores (for the relation mapping part only), respectively for this relation based extraction problem.

Our work can be found in the Github repository https://github.com/shidanxu/corruption.

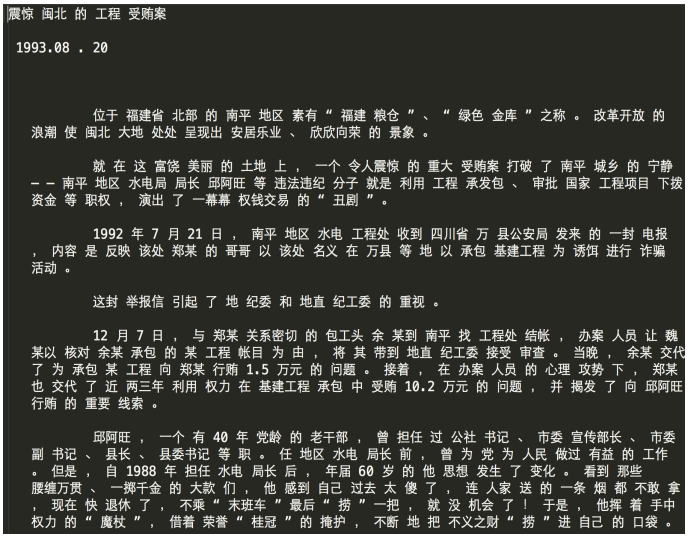
2 Introduction

Corruption has been one of the major concerns in Chinese government. Frequently news agencies report incidence of corruption involving bribery and money laundering. A typical newsreader needs to read through the entire article to get information on who committed what crime and for how much money. Given a news article, it would be useful to automatically extract the crucial information, such as the culprit and the crime. This project was suggested to us by Prof. Regina Barzilay to supplement Prof. Karen Zheng's ongoing research at the Sloan School of Management.

3 Dataset

We had 460 segmented text corpus from People's Daily, a Chinese newspaper agency. Each text corpus is a news article related to financial crimes. A UROP student annotates each news article by extracting the culprit, the location of crime, the amount of crime, the time the crime was discovered, and the punishment. The student also identifies which attributes correspond to whom. The selection of text corpus from a larger corpus base of People's Daily articles is done by an automated script, and for each corpus, sentences are presented to us with readily word segmented text using Stanford's NLP parser.

Our task was to automatically produce the annotation files just as a student would. Figure 1 shows an example annotation file.



震惊 闽北 的 工程 受贿案

1993.08 . 20

位于 福建省 北部 的 南平 地区 素有 “ 福建 粮仓 ” 、 “ 绿色 金库 ” 之称 。 改革开放 的 浪潮 使 闽北 大地 处处 呈现出 安居乐业 、 欣欣向荣 的 景象 。

就 在 这 富饶 美丽 的 土地 上 ， 一个 令人震惊 的 重大 受贿案 打破 了 南平 城乡 的 宁静 — — 南平 地区 水电局 局长 邱阿旺 等 违法违纪 分子 就是 利用 工程 承发包 、 审批 国家 工程项目 下拨 资金 等 职权 ， 演出 了 一幕幕 权钱交易 的 “ 丑剧 ” 。

1992 年 7 月 21 日 ， 南平 地区 水电 工程处 收到 四川省 万 县公安局 发来 的 一封 电报 ， 内容 是 反映 该处 郑某 的 哥哥 以 该处 名义 在 万县 等 地 以 承包 基建工程 为 诱饵 进行 诈骗 活动 。

这封 举报信 引起 了 地 纪委 和 地直 纪工委 的 重视 。

12 月 7 日 ， 与 郑某 关系密切 的 包工头 余 某到 南平 找 工程处 结帐 ， 办案 人员 让 魏某以 核对 余某 承包 的 某 工程 帐目 为 由 ， 将 其 带到 地直 纪工委 接受 审查 。 当晚 ， 余某 交代了 为 承包 某 工程 向 郑某 行贿 1.5 万元 的 问题 。 接着 ， 在 办案 人员 的 心理 攻势 下 ， 郑某 也 交代 了 近 两三年 利用 权力 在 基建工程 承包 中 受贿 10.2 万元 的 问题 ， 并 揭发 了 向 邱阿旺 行贿 的 重要 线索 。

邱阿旺 ， 一个 有 40 年 党龄 的 老干部 ， 曾 担任 过 公社 书记 、 市委 宣传部长 、 市委 副 书记 、 县长 、 县委书记 等 职 。 任 地区 水电 局长 前 ， 曾 为 党 为 人民 做过 有益 的 工作 。 但是 ， 自 1988 年 担任 水电 局长 后 ， 年届 60 岁 的 他 思想 发生 了 变化 。 看到 那些 腰缠万贯 、 一掷千金 的 大款 们 ， 他 感到 自己 过去 太 傻 了 ， 连 人家 送 的 一条 烟 都 不敢 拿 ， 现在 快 退休 了 ， 不乘 “ 末班车 ” 最后 “ 捞 ” 一把 ， 就 没 机会 了 ！ 于是 ， 他挥 着 手中 权力 的 “ 魔杖 ” ， 借着 荣誉 “ 桂冠 ” 的 掩护 ， 不断 地 把 不义之财 “ 捞 ” 进 自己 的 口袋 。

Figure 1: Part of $L\_R\_1993\_4280.txt$ The news raw data is word segmented.

```
T1  1Position 746 757   安徽省 机械厅 副厅长
T2  1Province 548 551   安徽省
T3  1Person 758 762 刘 玉山
T4  1Crime 763 765  贪污
T5  1Money_Person 766 772   7.5 万元
T6  2City 777 780   阜阳市
T7  2Province 548 551   安徽省
T8  2Position 777 785   阜阳市 工商银行
T9  2Person 786 789 顾荣胜
T10 2Crime 790 792  贪污
T11 2Money_Person 793 798   49 万元
T12 2Crime 801 805  挪用公款
T13 2Money_Person 806 812   680 万元
T14 3Position 820 828   省 电力局 干部
T15 3Money_Person 829 831   李津
T16 3Crime 832 834  贪污
T17 3Money_Person 835 840   49 万元
T18 3Province 548 551   安徽省
```

Figure 2: $L\_R\_1993\_4280.ann$ The 1 in 1Crime tells us person 1 committed "corruption".

4 Methods & Approaches

Multiple aspects of project are unique due to its scope and challenges. First, summarization task is one of the most difficult aspects of natural language processing. Second, not only did we need to extract all the relevant information, we also had to correctly identify the relations. Finding name attribute reference is in general, a difficult task as it introduces ambiguity. Third, Chinese language has no spaces in its natural form. We can use various existing APIs to do word segmentation, but different parameters and different APIs introduce noise. In practice, we found that Stanford NLP parser's performance is lower than other commercial available software. Extra dirty work was necessary to align the outputs.

We had several possible approaches to tackle this problem. We chose the one that was most results-oriented as opposed to trying out the coolest tricks. For each step, we had a clear thought on what our next largest margin would be, and tackled right away.

With only 460 data points, training a neural network would be less promising, if not fruitless. Given Chinese has over 50 thousand[1] characters, training a bigram or trigram model is sparse. Each article has on average 300 characters for a total of 140 thousand bigram training data in a 2.5 million long one hot vector.

Hence we decided to use our knowledge to help us with the feature vectors. In particular, Prof. Zheng provided us two lists of all financial crimes and lawful punishments from the law enforcement documents, respectively. She also provided us an excel file with all cities in China, along with their provinces. Because of this added dataset, we were able to reach a better performance than word embedding.

4.1 Pipelined Procedure

- Named Entity Recognition (NER)
  - We extract people's names and locations using Boson NLP[2]
  - We extract crimes, punishments, location, job position, money amount using feature vector matching.
  - We extract crimes, punishments, location, job position, money amount, time crime started, and time of discovery using a CRF.
  - We extract time crime started, time of discovery of crime by training a logistic regression model.
- Word Alignment
  - To combine the Boson and Stanford returned values
- Occupation Filtering
  - Remove irrelevant persons by occupation features
- Correspondence
  - Using a distance metric, we capture the relationship
  - Syntactic parsing to parse parallelization.

4.2 NER

Since we are interested in relation based data at the end, the first step is to extract the human's names. We used a language model (LM) provided by Boson NLP. Boson is a commercial software for Chinese. We experimented with various parameters that are more loose vs. strict at recognizing NERs. We decided to use 2 out of [1, 2, 3, 4, 5] for our NER. 1 means highest recall(lenient and finds more entities but can introduce false positive), 5 means highest precision (strict and finds less entities but found entities are accurate).

Prof. Zheng provided us a combined list of crimes from several law enforcement documents. Given that People's Daily is a paper that is mainly managed by the Party, the writing style is very formal. We hence decided to use directly the list as a feature vector. We were able to reach 66% precision and 40% recall for crimes extractions alone. A large proportion of the misses were due to inconsistency in annotations. The human annotations were done by multiple students and some write terse some copy paste the entire sentence. Hence when we match word for word to give a score, there is some extra info / info missing.

Prof. Zheng also provided us a list of cities in China. However this list only includes relatively big cities and do not include smaller divisions such as counties. In a typical article, a county can occur without the city name. Boson NLP does a better job of capturing the counties. Hence we used both feature vector matching and language model for this part. For Boson NLP, the parameter of 2 out of 5 is mostly for this part we want to detect more cities. In retrospect, we lost some precision on the name identification from here.

For job position, punishments, money amount, time of crime, we used several different approaches.

In general, there were two approaches. 1. Use our own knowledge of what constitutes a job position / punishment etc, and look for the exact string in sentence. 2. Use machine learning to aid us in finding the weights of the features, where features are loosely what these keyword are plus some additional ideas. We used a conditional random field (CRF) approach for machine learning part. We reached a much higher precision with the regex extraction as expected. The recall for most fields were higher with the CRF approach.

For job position, punishments, we did two approaches. The first approach is first to use a small script to extract the job positions and punishments in the first 40 annotation files. With those information, we wrote a regex matching algorithm that have some subset of the job positions and punishments found in the script, along with additions as we find appropriate by intuition. We previously had some idea of what job positions and punishments are available. We included almost everything that happened in the annotation file, along with some specific word patterns that humans can detect from these annotations. For example, we have city mayors as an occupation. Likewise we can deduce that we should also have county mayors, associate city mayors, etc. The second approach is to train a CRF, which will be discussed later.

For money amount, for first approach, we looked at 10 example output files and extracted the basic regex string that rep-

resents a money amount. We were able to reach 46% precision with this extraction. The recall number is high because we were able to extract most of the numerical values and that gave score. The precision is lower because there are various ways to represent money in Chinese that were overlooked by the model. The second approach also uses CRF.

For time when crime started, and time when crime was discovered, there is extra difficulty in ambiguity. We can identify a time entry from regex matching, but knowing whether it is discovery time or the beginning of crime is difficult. Both of us (native Chinese speakers) had no clear idea what features dominate the decision. Hence we decided to train a logistic regression model, in additional to the CRF approach. We gave the following features:

- **isStartOfArticle** This feature detects whether the time appears at the beginning 10% of the article. We noticed that at the beginning of each article, there's always a time to specify when this article was written. This time has nothing to do with the crime time.
- **nearCrime** This feature detects whether the complete sentence the time occurs in includes an incidence of crime, as detected by our language model. We believe that seeing an incidence of crime makes the time more likely to be either a crime discovery time or a crime start time.
- **earliestTime, latestTime** These features check whether this time stamp is the latest or earliest of all times in the news. We know that the latest time is usually the time the article is written. Because of the nature of law enforcement articles, it is common courtesy to wait till the person is convicted to publish news article. Therefore the latest time in the article is unlikely the crime times. The earliestTime is possibly the start of the crime.
- **nearCaughtKeywords, nearReportKeywords** We have several tens of features for those keywords. The feature just checks whether a specific word is in the same sentence as the time. Due to neural net training data size limit, we specified the list of keywords by outputting the highest frequency keywords in sentences where time is found for the 40 training articles.

With these features, the training happened. For each time we found in the article, we compared with the annotation files and gave it one of three tags: $y =$ "None", "Year_Disc", or "Year_Crime". In testing we take the $w^T x$ to get the outputs. We allow for multiple tags for each $y_i = 1$. Not to our surprise but worth noting, the testing results were disastrous. We had 4% precision and recall for time tags, respectively. This shows us that no features that we used were dominating.

For culprit - attribute matching, we used naive nearest distance. So if A is a person in sentence 1 and B is a crime in the following sentence, they would be attributed together. This does not take account of sentence structure.

With these features we have our baseline. For all numerical results please refer to results section.

CRF Training

For each tag, we also trained a CRF model. The CRF model takes account of the context and is useful in POS tagging for capturing the transitions. Here we extend it to do NER. Our training data consisted of 400 training files generated from each article in the following way: For every sentence in the article, if there is a relevant (nonzero) tag in the article, we add the entire sentence, in separated words format, to the training file. Otherwise we add the sentence with a probability λ. For example, the sentence "James/0 was/0 sentenced/punish to/punish death/punish" will be added to the training file as 5 rows, whereas "She/0 finished/0 her/0 homework/0" will not be added unless $random.random() < \lambda$.

Our feature vector was of length 88. It involves mostly keywords matching, and certain minimal additional information such as the position in the sentence. The features were similar of those used in the regex matching, as we would like to compare the performance of the two.

After training with the *pycrfsuite* package using BFGS, we evaluate our performance on the training set(with all sentences) and the remaining test data sets.

This usage of λ produces a very interesting point. The reason why we used λ is due to data scarcity. Initially we used the entire document as training data, and the output is unanimously 0. Because of the imbalance of 0 and other tags, everything is more likely a 0. In the original document, we have many words that are tagged "0". These words do not help in distinguishing the useful information. When we include those irrelevant information into the training data, it will cause bias in our transition matrix for the CRF. The overabundance of $0- > 0$ state transitions cause our matrix to assign large values to $x_{0,0}$, leaving minimal probability for the remaining entries of the same row and column as each row/column sums to 1.

After realizing this point, we decided to only use the sentences that have nonzero tags. With this, we see a significant increase in predicting other tags. However, because the number of zeroes is now underrepresented in the population, we sacrificed precision. The imbalance of data now says we need more zeroes.

So we calculated the total number of words that have 0 as tags, with those who have other tags. Approximately 6% of all words have nonzero as tags. As we make the training, we'd like to evaluate some model close to the reality. Hence we decided to randomly add sentences that have all 0 tags to the training data with probability λ. We used $\lambda = [0.01, 0.02, ...0.10]$.

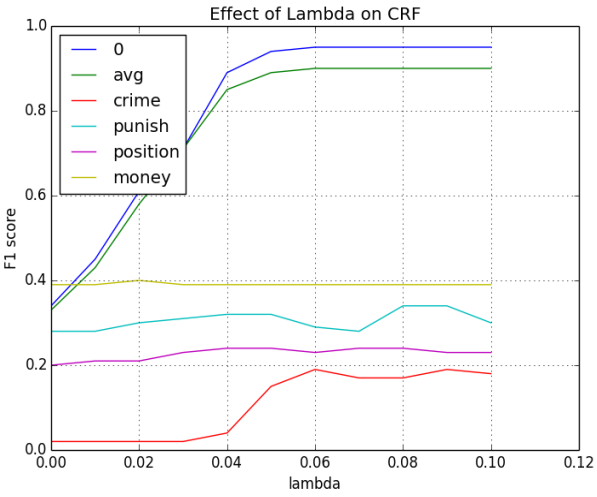


Figure 3: How different λs affected other training data. With no added 0's, recognition of crime is impossible. The low f1 score for 0 is largely due to a low recall at low λs.

Figure 3 illustrates this point. Initially with no added 0s ($\lambda = 0$), we have much lower F1 scores on crime and 0. With increased λs, we include more sentences with no tags to balance out the lack of zeroes, and hence reached better performance for both crime detection and 0 recall. The sensitive zone is roughly $5, 6\%$, in accordance to the 6% nonzero tags we calculated earlier.

4.3 Word Alignment

Perplexed, we examined the output to see where the largest margins of improvement are. Turns out some results were misaligned due to the use of multiple APIs. Consistently we had a off-by-2 position error. This compromised our scores. We implemented a linear search algorithm that, given the Boson returned output, finds the corresponding (and possibly mismatched because segmented word may be different) match in the original text file, segmented by Stanford NLP.

Our word alignment results more than doubled our recall, but had minimal increment on the precision.

4.4 Occupation Filtering

With a overall recall of 0.17, we are extracting some relevant text. But why is precision extremely low? By exploring some machine annotation files, we realized that the culprit - attribute relation is the hindrance. For instance, we can have a situation such as: The reporter A learned that he's been stealing money for the past 5 years. Here "he" is doing the stealing, but we are associating the stealing with the reporter.

We have two approaches here: 1. Resolve who he is by coreference. 2. Filter out the good occupations. Approach 1 is an open research area and in general a hard problem[4]. We decided to shun away from it after discussing with the TAs.

Because People's Daily is a article managed mostly by the Party, certain occupations are less likely to be accused of crimes. Reporters, judges, police officers, and other law enforcement officials represent the protection for society. The People's Daily is unlikely to post a news where a reporter or police officer commits a crime. We decided that whenever we see those occupations associated with a name, we deem that person good, and filter him/her out for the calculation of nearest name, etc. City mayors, accountants, secretaries who are closer to power are more likely to commit a financial crime.

This brilliant observation gave us the largest margin. We were able to reach 39% precision and 38% recall after filtering out the good occupations.

4.5 Correspondence

Lastly we try to fix correspondence issues. Instead of using a distance metric that finds the nearest name by the closest word, we now look only in the current, previous, and next sentences. This resolves the issue of one particular attribute being extremely far away gets assigned to a name one paragraph ago.

We have higher precision but sacrificed recall.

Language Technology Platform - cloud (LTP)[3] was the language model we used for syntactic parsing. This is another language model for Chinese open source on Github. We were not able to complete the testing by this deadline, as final reports for 6.864 are due on Monday. Essentially, we target the case where there are multiple culprits and multiple punishments in the same sentence. For instance, "A and B were each charged with a sentence of 10 years, 5 years, respectively." Here by using nearest name matching, B will be matched with two punishments, but in reality it is a parallel "respectively" situation. By using syntactic parsing, we can detect where parallelization happens and assigns accordingly.

However, we do not expect our performance to increase by much because such case is rare.

5 Evaluations

Here we present the baseline, later evaluation metrics. Since evaluations are nonstandard.

We treat the 460 human annotated files as gold standard. For NLP tasks, we are mainly concerned with balancing precision and recall. As previously defined, precision = true positive / (true positive + false positive). Recall = true positive / (true positive + false negative).

Because Prof. Zheng wants the annotations to be accurate and **terse**, we were harsh on ourselves defining the scores.

Precision: For every entry the machine found, we check the **maximum continuous substring** match in the gold standard *under the same person.* Given the maximum substring match length, we divide by length of the machine found entry. Hence if our machine found entry is too long, it will receive a low score. This is a measure of both the relation and the extraction.

Recall: For every entry in the gold standard, we check the **maximum continuous substring** match in the machine annotation *under the same person*. Given the maximum substring match length, we divide by length of the gold standard entry. So if our machine found a long entry but contains the gold standard, it receives a score of 1. This is a measure of both the relation and the extraction.

Information Extraction Precision: Same as precision but without the condition *under the same person*. This is a measure of information extraction.

Information Extraction Recall: Same as recall but without the condition *under the same person*. This is a measure of information extraction.

Normalized Precision: Precision / Information Extraction Precision. Loosely speaking, this is a measure of only the relation precision.

Normalized Recall: Recall / Information Extraction Recall. Loosely speaking, this is a measure of only the relation recall.

6 Results

Here we give the results.

For each individual tag

	Precision	Recall	F1
Job Position CRF	0.20	0.31	0.25
Job Position Combined	0.76	0.30	0.48
Punishment CRF	0.31	0.37	0.34
Punishment Combined	0.66	0.16	0.32
Crime CRF	0.27	0.15	0.20
Crime Regex	0.66	0.40	0.51
Money Amount CRF	0.26	0.81	0.46
Money Amount Regex	0.46	0.63	0.54
Time Regex	0.21	0.24	0.22
Time LR Train	0.04	0.05	0.04
Time LR Test	0.02	0.03	0.02
Time CRF	0.14	0.48	0.26
Location Boson	0.82	0.77	0.79

For CRF the training set is the first 400 articles whereas the testing set is the remaining 60 articles. Combined means regex matching, to combine the features found in 40 annotation files, along with human interpretation to complete as a simple regex. As seen from the table, most trained features were outperformed by a regex. The time feature training is non-surprisingly especially bad because we have no short phrases to help us distinguish between Year_disc and Year_crime. When the researchers do not have good features, we do not expect logistic regression to shine light on us. A randomly assigned time achieves 21% and 24%. Trained data is worse than random. The CRF performed best for detecting time. Here human intuition were terrible, which is why CRF outperformed by large margins. Here if we had more data, a neural net could potentially be useful.

Another important point to make is that CRF achieves worse results than our regex in general in terms of performance. In most cases other than crime detection, CRF achieves a higher recall. For common POS tagging algorithms, each word is tagged with some information, unlike here the majority of words carry "0" as information. The lack of information was mistreated by the algorithm as some form of information, therefore downgrading the performance. The very interesting thing is for both models to detect time, CRF outperformed the simple logistic regression vastly.

This gives us confidence that the use of machine learning is useful when we have high level features of the data. If we are very clear which features will decide the tag, we are better off just using those values for precision. However, if we are unsure what exactly causes the tag, such as the ambiguity in time, we can utilize machine learning to help us distinguish the important factors.

From these numbers, we discover that crime, punishment, job position and location are relatively easy to detect. We tend to have low recall by using regex matching as clearly we are not omnipotent in knowing the features. The low precision in money amount regex matching is due to multiple alternative orders to represent a currency in Chinese. In general, the F1 scores are higher for regex matching, which means we still have room for improvement in the amount of data, and the depth of data. We can increase that score by having a more sophisticated CRF that tags each individual word with some information, for instance make each tag two dimensional: POS and our crime tags.

For pipeline final results:

Approach	Precision	Recall	EPrecision	ERecall
Baseline	0.03	0.11	N/A	N/A
Word Alignment	0.09	0.17	0.34	0.48
Occup. Filtering	0.38	0.39	0.54	0.53
Correspondence	0.43	0.36	0.60	0.52

Approach	Normalized Precision	Normalized Recall
Baseline	N/A	N/A
Word Alignment	0.26	0.35
Occup. Filtering	0.67	0.65
Correspondence	0.71	0.68

For the baseline some evaluation data were not available. The greatest margin of improvement comes from occupational filtering. Correspondence has not been completely implemented, but we expect the each individual case to be rather spread out and it may further improve precision but hurt recall.

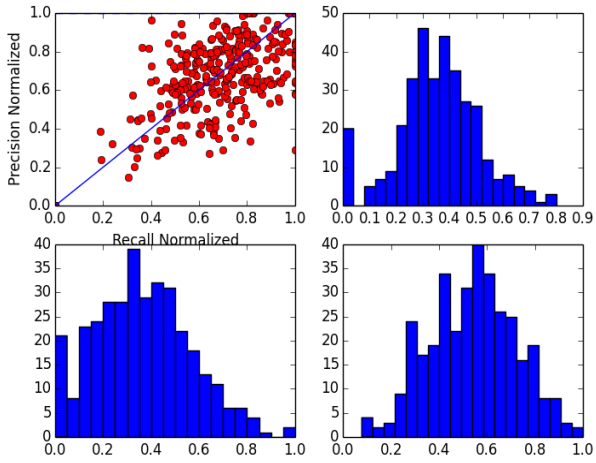


Figure 4: Clockwise: Distribution of scores, each dot represents an article. Distribution of precision. Distribution of normalized precision. Distribution of recall. All for occupational filtering step.

We ended our research here because we do not see any large margin. The distribution of normalized scores is roughly linear. The distribution of average precision, recall, and normalized precision are all approximately Gaussian. This means that other than systematic errors, other sources of error are independent. There is no single dominant criterion that we are not capturing that is causing the score distributions. There may be multiple factors that affect the scores, but these factors tend to be independent of each other. Hence by law of large numbers we approach a Gaussian distribution at the end.

An example of a bad output is

```
(0.0, 0.0, 0.5666666666666667, 0.2435185185185185)
WORKING ON FILE:  corruption annotated data/L_R_2004_9085.a
type of document corruption annotated data/L_R_2004_9085.an
T1      1City 322 325   晋城市
T2      1Province 89 91 山西
T4      1Person 431 434 张震有
T5      1Crime 435 442  挪用 专项资金
T3      1Position 423 430       市 科委 主任
T6      1Event 307 308  2
T7      1Event 461 462  。
```

Figure 5: Gold Standard for 2004_9085.

```
WORKING ON FILE:  corruption annotated
type of document corruption annotated
1Person 杨林林
1Money_Person 873 875   255.8万元
1Position 3 4   局长
1Position 30 31 局长
1Position 63 64 记者
1Position 79 80 主任
1Position 138 139       主任
1Position 159 160       市委书记
1Position 168 169       市长
1Position 197 198       人大常委会
1Position 198 199       主任
1Position 207 208       人大常委会
1Position 222 223       市长
1Position 232 233       市长
1Position 235 236       秘书长
1Position 294 295       人大常委会
1Position 305 306       局长
1Position 349 350       人大常委会
1Position 367 368       局长
1Position 392 393       局长
1Position 412 413       人大常委会
1Position 424 425       党组书记
1Position 473 474       厅长
1Position 611 612       主任
1Position 709 710       主任
1Position 758 759       书记
1Position 761 762       纪委书记
1Position 764 765       记者
1Position 833 834       主任
1Position 853 854       主任
1Position 947 948       主任
1Position 1092 1093     人大常委会
1Position 1097 1098     市长
1Position 1102 1103     人大常委会
1Position 1122 1123     局长
1Time 19 25     2003年12月29日
1location 26 27 山西晋
2Person 张震
2Position 1133 1134     人大常委会
2Position 1134 1135     主任
2Position 1140 1141     市长
2Position 1142 1143     市委书记
```

Figure 6: Machine annotation for 2004_9085..

Here the problem is we found too many positions. In retrospect, these positions are there because there are many people who were not mentioned by name due to privacy. Because they are not mentioned by name, the machine found the nearest name, which there is only 1. So one person was incorrectly identified to 20 positions.

Another example of a bad output is

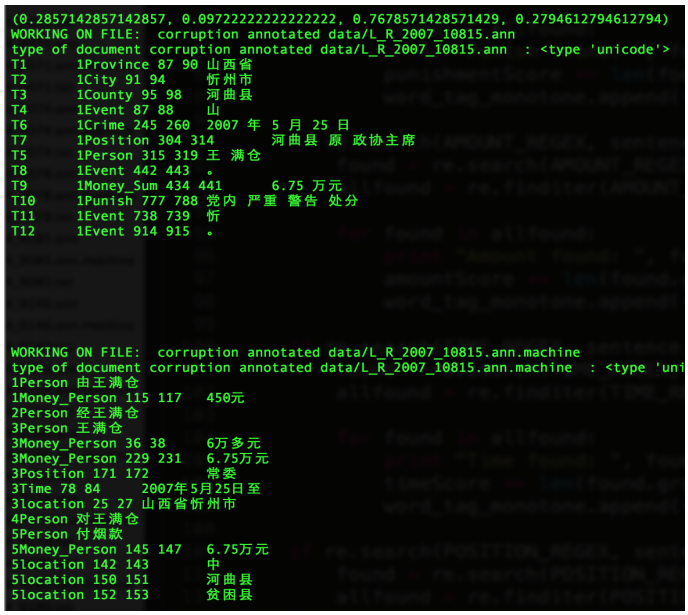


Figure 7: Gold standard, machine annotation, respectively, for 2007_10815.

Here the issue is that Boson NLP returned wrong named entities. If you observe closely at the machine translation, the 4 different persons all share three same characters. The three common characters are the person's name. Hence they are the same person but the API recognized them as different names due to incorrect segmentation. This has detrimental affect on relation precision as the name matching is simply wrong.

An example of a good output is

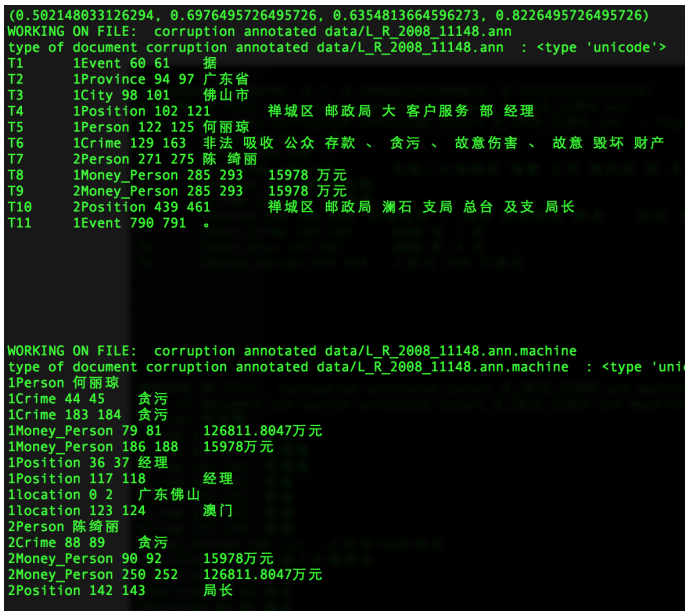


Figure 8: Gold standard, machine annotation, respectively, for 2008_11148.

Here we correctly identified the culprits, the crime type, the location, and one of the money amount involved, and the positions.

Based on our discovery, the errors are individual and independent. There is no single fix that can easier increase the performance by whopping amounts.

7 Discussion

This real world project really changed my opinion on machine learning and natural language processing. We have a difficult task at a high level, and we used a very practical approach. We were not surprised that training may not be as efficient as direct feature matching. Two things we are especially thankful for: 1. The obsolete formal sentence structure of People's Daily. 2. Our great feature intuition as native speakers. The giant alphabet of Chinese also makes training difficult. As opposed to pure academia, data is always incomplete and dirty.

One particular challenge in data processing was the encoding of Chinese languages in python. In python 2, Chinese languages need to be encoded as unicode strings. This introduces extra layer of difficulty as we were not previously experienced. In retrospect, instead of worrying about encoding in the middle of a function, we should convert all text files to unicode strings as preprocessing once and for all.

Shidan's advisor, Prof. Gerald Sussman once asked him what his name was. In perplexity, Shidan responded with delay. Sussman said something that struct Shidan, "It took you 0.5 seconds to respond. That's only enough time for 50 neurons firing." Hence a model cannot be overly complex. In testing out the bi-grams and trigram models, especially with Chinese and its overabundance of characters, I find it less credible that such an approach is how humans think. Such a bashing approach seems fruitless. Neural net, in essence, is to wait for a kid to turn to 20, and then feed him all the human knowledge, instead of training him as he grows up.

8 Future works

For future works, we'd like to explore more syntactic parsing. The juxtaposition of culprits, relation disambiguity with dependency parsing are two immediate approaches that come to mind. While these are clear cases the current algorithm does not account for, we are not sure how often these situations happen. They can probably increase the score by single digit percentages.

Another direction we can take is to extend this application to other datasets. Initially, the project idea was that for experienced judges, by reading the court transcript, they can decide, with high accuracy, whether the defendant is guilty or not. We switched because of lack of data. Nevertheless, such an application can benefit from our information extraction, and use the information we extracted as feature vectors. This is when we can train a more elaborate model for predicting a single tag guilty or not.

9 Distribution of Work

Shidan Xu and Tingtao Zhou worked on this project. Shidan worked on the custom evaluation methods, implemented the CRF model, worked out the feature extraction and regex matching, and struggled with parsing Chinese text into python. Tingtao

worked on indexing the paragraphs, figuring out the distance metric, occupation filtering, connecting all the APIs to generate correct tagging, and the logistic regression for time. Both researchers spent a lot of time on word alignment. We spent more than 70 working hours each on this project.

10 Acknowledgements

The authors would like to thank Prof. Regina Barzilay for introducing them to such an interesting real world problem. The authors thank Prof. Karen Zheng for providing the documentations and asking the UROPs to proofread annotated data in a very timely fashion. The authors thank Kiva and Tao for suggesting approaches and APIs for the problem. We wanted the final product to be useful in aiding Prof. Zheng's research, hence we took a more results oriented approach. We sincerely believe that given the timeframe, the approach in this paper gives the highest score.

11 References

1. Chinese Characters. http://www.bbc.co.uk/languages/chinese/real_chinese/mini_guides/characters/characters_howmany.shtml

2. Boson NLP, http://bosonnlp.com/

3. Language Technology Platform, http://www.ltp-cloud.com/

4. Fader, Anthony. Identifying Relations for Open Information Extraction. https://homes.cs.washington.edu/ soderlan/Fader-emnlp11.pdf

5. Okurowski, Mary Ellen. INFORMATION EXTRACTION OVERVIEW. http://www.aclweb.org/anthology/X93-1012