Identifying Chemical Names in Biomedical Text:

An Investigation of the Substring Co-occurrence Based Approaches

Alexander Vasserman
Department of Computer and
Information Science
University of Pennsylvania
Philadelphia, PA 19104
avasserm@seas.upenn.edu

Abstract

We investigate various strategies for finding chemicals in biomedical text using substring co-occurrence information. The goal is to build a system from readily available data with minimal human involvement. Our models are trained from a dictionary of chemical names and general biomedical text. We investigated several strategies including Naïve Bayes classifiers and several types of N-gram models. We introduced a new way of interpolating N-grams that does not require tuning any parameters. We also found the task to be similar to Language Identification.

1 Introduction

Chemical names recognition is one of the first tasks needed for building an information extraction system in the biomedical domain. Chemicals, especially organic chemicals, are one of the main agents in many processes and relationships such a system would need to find. In this work, we investigate a number of approaches to the problem of chemical names identification. We focus on approaches that use string internal information for classification, those based on the character co-occurrence statistics within the strings that we would like to classify. We would also like not to spend much time and effort to do manual annotation, and hence use readily publicly available data for training all the models. Because of that, we would be satisfied with only moderate results. In the course of this investigation, we have found that N-gram methods work best given these restrictions on the models.

Work has been done on a related task of named entity recognition (Bikel et al., 1999, Riloff, 1996, Cucerzan, 1999, and others). The aim of the named entity task is usually set to find names of people, organizations, and some other similar entities in text. Adding features based on the internal substring patterns has been found useful by Cucerzan et al., 1999. For finding chemicals, internal substring patterns are even more important source of information. Many substrings of chemical names are very characteristic. For example, seeing "methyl" as a substring of a word is a strong indicator of a chemical name. The systematic chemical names are constructed from substrings like that, but even the generic names follow certain conventions, and have many characteristic substrings.

In this work, character co-occurrence patterns are extracted from available lists of chemicals that have been compiled for other purposes. We built models based on the difference between strings occurring in chemical names and strings that occur in other words. The use of only string internal information prevents us from disambiguating different word senses, but we accept this source of errors as a minor one.

Classification based solely on string internal information makes the chemical names recognition task similar to language identification. In the language identification task, these patterns are used to detect strings from a different language embedded into text. Because chemicals are so different, we can view them as a different language, and borrow some of the Language Identification techniques. Danning, 1994 was able to achieve good results using character N-gram models on language identification even on short strings (20 symbols long). This suggests that his approach might be successful in chemical names identification setting.

N-gram based methods were previously used for chemicals recognition. Wilbur et al., 1999 used all substrings of a fixed length N, but they combined the training counts in a Bayesian framework, ignoring non-independence of overlapping substring. They claimed good performance for their data, but this approach showed significantly lower performance than alternatives on our data. See the results section for more details. The difference is that their data is carefully constructed to contain only chemicals and chemicals of all types in the test data, i.e. their training and testing data is in a very close correspondence.

We on the other hand tried to use readily available chemical lists without putting much manual labor into their construction. Most of our training data comes from a single source - National Cancer Institute website - and hence represents only a very specific domain of chemicals, while testing data is coming from a random sample from MEDLINE. In addition, these lists were designed for use by human, and hence contain many comments and descriptions that are not easily separable for the chemical names themselves. Several attempts on cleaning these out have been made. Most aggressive attempts deleted about half the text from the list. While deleting many useful names, this improved the results significantly.

While we found that N-grams worked best amoung the approaches we have tried, other approaches are also possible. We did not explore the possibility of using substring as features to a generic classification algorithm, such as, for example, support vector machines (Burges, 1998).

2 Available Data

In order to train a statistical model for recognizing chemicals a list of about 240 thousands entries have been download from National Cancer Institute website (freely available at dtp.nci.nih.gov). Entries are unique names of about 45 thousands unique chemicals. Each entry includes a name of a chemical possibly followed by alternative references and some comments. This additional information had to be deleted in order to compute statistics from chemical names only. While there were no clean separators between chemical names and the additional materials, several patterns were designed to clean up the list. Applying those patterns shrunk each entry on average by half. This cleaning step has not produced perfect results in both leaving some unusable material in and deleting some useful strings, yet it improved the performance of all methods dramatically. Cleaning the list by hand might have produced better results, but it would require more expertise and take a lot of time and would contradict the goal of building the system from readily available data.

We used text from MEDLINE abstracts to model general biomedical language. These were available as a part of the MEDLINE database of bibliographical records for papers in biomedical domain. Records that had non-empty abstracts have been extracted. From those 'title' and 'abstract' fields were taken and cleaned off from remaining XML tags.

Both the list of chemical names (LCN) and the text corpus obtained from the MEDLINE database (MED) were tokenized by splitting on the white spaces. White space tokenization was used over other possible approaches, as the problem of tokenization is very hard for chemical names, because they contain a lot of internal punctuation. We also wanted to avoid splitting chemical names into tokens that are too small, as they would contain very little internal information to work with. The counts of occurrences of tokens in LCN and MD were used in all experiments to build models of chemical names and general biomedical text.

In addition, 15 abstracts containing chemical names were selected from the parts of MEDLINE corpus not used for the creation of the above list. These abstracts have been annotated by hand and used as development and test sets.

3 Classification Using Substring Importance Criteria

3.1 Classification Approach

Most obvious approach to this problem is to try to match the chemicals in the list against the text and label only the matches, i.e. chemicals that are known from the list. This approach is similar to the memory-based baseline described by Palmer et al., 1997, where instead of using precompiled list they memorized all the entries that occurred in a training text.

A natural extension of matching is a decision list. Each classification rule in the list checks if a substring is present in a token. Matching can be viewed as just an extreme of this approach, where the strings selected into the decision list are the complete tokens from the LCN (including token boundary information). Using other substrings increases recall, as non-exact matches are detected, and it also improves precision, as it decreases the number of error coming from noise in LCN.

While decision list performs better than matching, its performance is still unsatisfactory. Selecting only highly indicative substrings results in high precision, but very low recall. Lowering the thresholds and taking more substrings decreases the precision without improving the recall much until the precision gets very low.

The decision list approach makes each decision based on a single substring. This forces us to select only substrings that are extreemly rare outside the chemical names. This in turn results in extremely low recall. An alternative would be to combine the information from multiple substrings into a single decision using Naive Bayes framework. This would keep precision from dropping as dramatically when we increase the number of strings used in classification.

We would like to estimate the probability of a token being a part of a chemical name given the token (string) *p(c|s)*. Representing each string as a set of its substrings we need to estimate $p(c|s_1...s_n)$. Using Bayes Rule, we get

$$p(c|s_1...s_n) = p(s_1...s_n|c)p(c)/p(s_1...s_n) \qquad (1)$$

Assuming independence of substrings $s_1...s_n$ and conditional independence of substrings $s_1...s_n$ given c, we can rewrite:

$$p(c|s_1...s_n) = p(c)p(s_1...s_n|c)/p(s_1...s_n)$$
$$= p(c)\prod_{i=1}^{n} p(s_i|c)/\prod_{i=1}^{n} p(s_i) \qquad (2)$$

Now notice that for most applications we would like to be able to vary precision/recall tradeoff by setting some threshold t and classifying each string *s* as a chemical only if

$$p(c \mid s) = p(c)\prod_{i=1}^{n} p(s_i|c)/\prod_{i=1}^{n} p(s_i) > t \qquad (3)$$

or

$$\prod_{i=1}^{n} p(s_i|c)/\prod_{i=1}^{n} p(s_i) > t / p(c) = t' \qquad (4)$$

This allows us to avoid estimation of *p(c)* (estimating *p(c)* is hard without any labeled text). We can estimate $p(s_i|c)$ and $p(s_i)$ from the LCN and MED respectively as

$$p(s_i) = \#(\text{tokens containg } s_i)/\#(\text{tokens}) \qquad (5)$$

3.2 Substring Selection

For this approach, we need to decide what set of substring $\{s_i\}$ of *s* to use to represent *s*. We would like to select a set of non-overlapping substrings to make the independence assumption more grounded (while it is clear that even non-overlapping substrings are not independent, assuming independence of overlapping substrings clearly causes major problems). In order to do this we need some measure of usefulness of substrings. We would like to select substrings that are both informative and reliable as features, i.e. the substrings fraction of which in LCN is different from the fraction of them in MED and which occur often enough in LCN. Once this measure is defined, we can use dynamic programming algorithm similar to Viterbi decoding to select the set of non-overlapping substrings with maximum value.

Kullback-Leibler divergence based measure

If we view the substring frequencies as a distribution, we can ask the question which substrings account for the biggest contribution to Kullback-Leibler divergence (Cover et al, 1991) between distribution given by LCN and that given by MED. From this view it is reasonable to take $p(s_i|c)*log(p(s_i|c)/p(s_i))$ as a measure of value of a substring. Therefore, the selection criterion would be

$$p(s_i \mid c)\log(p(s_i \mid c)/p(s_i)) > t \qquad (6)$$

where *t* is some threshold value. Notice that this measure combines frequency of a substring in chemicals and the difference between frequencies of occurrences of the substring in chemicals and non-chemicals.

A problem with this approach arises when either $p(s_i|c)$ or $p(s_i)$ is equal to zero. In this case, this selection criterion cannot be computed, yet some of the most valuable strings could have $p(s_i)$ equal to zero. Therefore, we need to smooth probabilities of the strings to avoid zero values. One possibility is to include all strings s_i, such that $p(s_i)=0$ and $p(s_i|c)>t'$, where $t'<t$ is some new threshold needed to avoid selecting very rare strings. It would be nice though not to introduce an additional parameter. An alternative would be to reassign probabilities to all substrings and keep the selection criterion the same. It could be done, for example, using Good-Turing smoothing (Good 1953).

Selection by significance testing

A different way of viewing this is to say that we want to select all the substrings in which we are confident. It can be observed that tokens might contain certain substrings that are strong indicators of them being chemicals. Useful substrings are the ones that predict significantly different from the prior probability of being a chemical. I.e. if the frequency of chemicals among all tokens is *f(c)*, then s is a useful substring if the frequency of chemicals among tokens containing s *f(c|s)* is significantly different from *f(c)*. We test the significance by assuming that *f(c)* is a good estimate for the prior probability of a token being a chemical *p(c)*, and trying to reject the null hypothesis, that actual probability of chemicals among tokens that contain s is also *p(c)*. If the number of tokens containing s is *n(s)* and the number of chemicals containing *s* is *c(s)*, then the selection criterion becomes

$$\frac{c(s) - n(s)f(c)}{\sqrt{n(s)f(c)(1-f(c))}} > 1.65 = z_{.95} \qquad (7)$$

This formula is obtained by viewing occurrences of s as Bernoulli trials with probability *p(c)* of the occurrence being a chemical and probability *(1-p(c))* of the occurrence being non-chemical. Distribution obtained by *n(s)* such trials can be approximated with the normal distribution with mean *n(s)p(c)* and variance *n(s)p(c)(1-p(c))*.

4 Classification Using N-gram Models

We can estimate probability of a string given class (chemical or non-chemical) as the probability of letters of the string based on a finite history.

$$p(c \mid S) = p(S \mid c) p(c) / p(S)$$
$$= \frac{\prod_i p(s_i \mid s_{i-1} ... s_0, c)}{\prod_i p(s_i \mid s_{i-1} ... s_0)} p(c) \qquad (8)$$

where S is the string to be classified and s_i are the letters of S.

The N-gram approach has been a successful modeling technique in many other applications. It has a number of advantages over the Bayesian approach. In this framework we can use information from all substrings of a token, and not only sets of non-overlapping ones. There is no (incorrect) independence assumption, so we get a more sound probability model. As a practical issue, there has been a lot of work done on smoothing techniques for N-gram models (Chen et al., 1998), so it is easier to use them.

4.1 Investigating Usefulness of Different N-gram Lengths

As the first task in investigating N-gram models, we investigated usefulness of N-grams of different length. For each n, we constructed a model based on the substrings of this length only using Laplacian smoothing to avoid zero probability.

$$p(s_i \mid s_{i-1} ... s_0, c) \approx \frac{n_{c\,i-N+1}^{\;i} + d}{n_{c\,i-N+1}^{\;i-1} + dB}$$
$$p(s_i \mid s_{i-1} ... s_0) \approx \frac{n_{i-N+1}^{i} + d}{n_{i-N+1}^{i-1} + dB} \qquad (9)$$

where N is the length of the N-grams, $n^i_{i\text{-}N+1}$ and $n_c{}^i_{i\text{-}N+1}$ are the number of occurrences of N-gram $s_i s_{i\text{-}1} ... s_{i\text{-}N\text{-}1}$ in MEDLINE and chemical list respectively, d is the smoothing parameter, and B is the number of different N-grams of length N.

The smoothing parameter was tuned for each n individually using the development data (hand annotated MEDLINE abstracts). The results of these experiments showed that 3-grams and 4-grams are most useful. While poor performance by longer N-grams was somewhat surprising, results indicated that overtraining might be an issue for longer N-grams, as the model they produce models the training data more precisely. While unexpected, the result is similar to the conclusion in Dunning '94 for language identification task.

4.2 Interpolated N-gram Models

In many different tasks that use N-gram models, interpolated or back-off models have been proven useful. The idea here is to use shorter N-grams for smoothing longer ones.

$$p(s_i \mid s_{i-1} ... s_0, c)$$
$$\approx l_N \frac{n_{c\,i-N+1}^{\;i}}{n_{c\,i-N+1}^{\;i-1}} + l_{N-1} \frac{n_{c\,i-N+2}^{\;i}}{n_{c\,i-N+2}^{\;i-1}} + ... + l_1 \frac{n_{c\,i}^{\;i}}{m_c}$$
$$p(s_i \mid s_{i-1} ... s_0) \qquad (10)$$
$$\approx l_N \frac{n_{i-N+1}^{i}}{n_{i-N+1}^{i-1}} + l_{N-1} \frac{n_{i-N+2}^{i}}{n_{i-N+2}^{i-1}} + ... + l_1 \frac{n_i^i}{m}$$

where l_j's are the interpolation coefficients, m and m_c are the total number of letters in MEDLINE and chemical list respectively. l_j can generally depend on $s_{i\text{-}1} ... s_{i\text{-}N+1}$, with the only constraint that all l_j coefficients sum up to one. One of the main question for interpolated models is learning the values for l's. Estimating N different l's for each context $s_{i\text{-}1} ... s_{i\text{-}N+1}$ is a hard learning task by itself that requires a lot of development data. There are two fundamentally different ways for dealing with this problem. Often grouping different coefficients together and providing single value for each group, or imposing some other constraints on the coefficients is used to decrease the number of parameters. The other approach is providing a theory for values of l's without tuning them on the development data (This is similar in spirit to Minimal Description Length approach). We have investigated several different possibilities in both of these two approaches.

4.3 Computing Interpolation Coefficients: Fixed Coefficients

Equation (10) can be rewritten in a slightly different form:

$$p(s_i \mid s_{i-1} ... s_0) \approx (1 - l_{N-1}) \frac{n_{i-N+1}^{i}}{n_{i-n+1}^{i-1}}$$
$$+ l_{N-1} \left(\begin{array}{l} (1 - l_{N-2}) \frac{n_{i-N+2}^{i}}{n_{i-N+2}^{i}} + \\ l_{N-2} \left(... \left((1 - l_1) \frac{n_{i-1}^{i}}{n_{i-1}^{i-1}} + l_1 \frac{n_i^i}{m} \right) \right) \end{array} \right) \qquad (11)$$

This form states more explicitly that each N-gram model is smoothed by all lower models. An extreme of the grouping approach is then to make all l_j's equal, and tune this single parameter on the development data.

4.4 Computing Interpolation Coefficients: Context Independent Coefficients

Relaxing this constraint and going back to the original form of equation (10), we can make all l_j's independent of their context, so we get only N parameters to tune. When N is small, this can be done even with relatively small development set. We can do this by exploring all possible settings of these parameters in an N dimensional grid with small increment. For larger N we have to introduce an additional constraint that l_j's should lie on some function of j with a smaller number of parameters. We have used a quadratic function (2 parameters, as one of them is fixed by the constraint that all l_j's have to sum up to 1). Using higher order of the function gives more flexibility, but introduces more parameters, which would require more development data to tune well. The quadratic function seems to be a good trade off that provides enough flexibility, but does not introduce too many parameters.

4.5 Computing Interpolation Coefficients: Confidence Based Coefficients

The intuition for using interpolated models is that higher level N-grams give more information when they are reliable, but lower level N-grams are usually more reliable, as they normally occur more frequently. We can formalize this intuition by computing the confidence of higher level N-grams and weight them proportionally. We are trying to estimate $p(s_i|s_{i-1}...s_{i-N+1})$ with the ratio $n^i_{i-N+1}/n^{i-1}_{i-N+1}$. We can say that our observation in the training data was generated by n^{i-1}_{i-N+1} Bernoulli trials with outcomes either s_i or any other letter. We consider s_i to be a positive outcome and any other letter would be a negative outcome. Given this model we have n^i_{i-N+1} positive outcomes in n^{i-1}_{i-N+1} Bernoulli trials with probability of positive outcome $p(s_i|s_{i-1}...s_{i-N+1})$. This means that the estimate given by $n^i_{i-N+1}/n^{i-1}_{i-N+1}$ has the confidence interval of binomial distribution approximated by normal given by

$$I = \frac{\sqrt{c^3 z_a^{\ 2} + c^2 z_a^{\ 4}}}{2c(c + z_a^{\ 2})} \qquad (12)$$

where $c = n^{i-1}_{i-N+1}$.

Since the true probability is within I of the estimate, the lower level models should not change the estimate given by the highest-level model by more than I. This means that l_{N-1} in the equation (11) should be equal to I. By recursing the argument we get

$$l_j = I_j = \frac{\sqrt{c_j^{\ 3} z_a^{\ 2} + c_j^{\ 2} z_a^{\ 4}}}{2c_j(c_j + z_a^{\ 2})} \qquad (13)$$

where $c_j = n^{i-1}_{i-j+2}$ for $j > 1$, and $c_1 = m$.

5 Evaluation and Results

We performed cross validation experiments on 15 hand-annotated MEDLINE abstracts described in section "Available Data". Experiments were done by holding out each abstract, tuning model parameters on 14 remaining abstracts, and testing on the held out one. Fifteen such experiments were performed. The results of these experiments were combined by taking weighed geometric mean of precision results at each recall level. The results were weighted according to the number of positive examples in each file to ensure equal contribution from each example. Figure 1 shows the resulting precision/recall curves.

As we can see, the N-gram approaches perform better than the other ones. The interpolated model with quadratic coefficients needs a lot of development data, so it does not produce good results in our case. Simple Laplacian smoothing needs less development data and produces much better results. The model with confidence based coefficients works best. The graph also shows the model introduced by Wilbur et al., 1999. It does not perform nearly as well on our data, even though it produces very good results on clean data they have used. This (as well as some experiments we performed that have not been included into this work) suggests that quality of the training data has very strong effect on the model results.

6 Conclusions and Future Work

We have investigated a number of different approaches to chemical identification using string internal information. We used readily available training data, and a small amount of human annotated text that was used primarily for testing. We were able to achieve good performance on general biomedical text taken from MEDLINE abstracts. N-gram models showed the best performance. The specific details of parameter

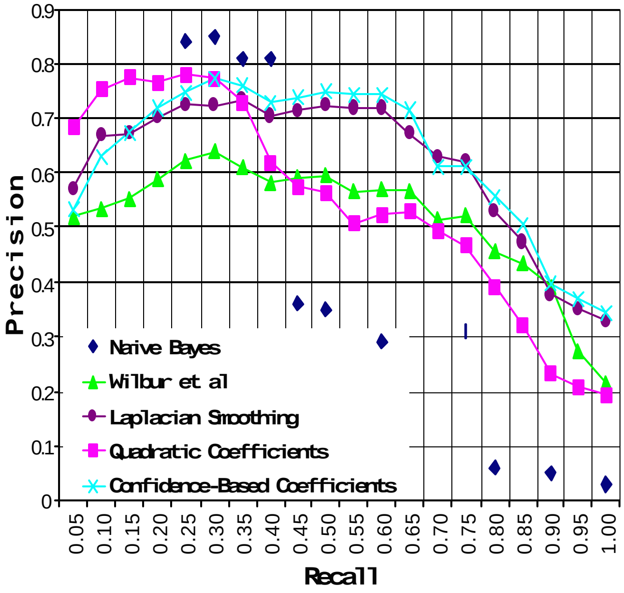

Fig. 1. Precision/Recall curves for Naïve Bayes and N-gram based models

tuning for these models produced small variations in the results. We have also introduced a method for computing interpolated N-gram model parameters without any tuning on development data. The results produced by this method were slightly better than those of other approaches. We believe this approach performed better because only one parameter - the length of N-grams - needed to be tuned on the development data. This is a big advantage when little development data is available. In general, we discovered many similarities with previous work on language identification, which suggests that other techniques introduced for language identification may carry over well into chemicals identification.

As a short term goal we would like to determine N-gram interpolation coeficients by usefulness of the corresponding context for discrimination. This would incorporate the same techinque as we used for Naive Bayes system, hopefully combining the advantage of both approaches

There are other alternatives for learning a classification rule. Recently using support vector machines (Burges 1998) have been a popular approach. More traditionally decision trees (Breiman et al, 1984) have been used for simmilar tasks. It would be interesting to try these aproaches for our task and compare them with Naive Bayes and N-gram approaches discussed here.

One limitation of the current system is that it does not find the boundaries of chemicals, but only classifies predetermind tokens as being part of a chemical name or not. The system can be improved by removing prior tokenization requirment, and attempting to identify chemical name boundaries based on the learned information.

In this work we explored just one dimention of possible features usefull for finding chemical names. We intent to incorporate other types of features including context based features with this work.

References

T. Dunning. 1994. "Statistical identification of language". Technical Report MCCS 94-273, New Mexico State University.

S. F. Chen and J. Goodman. 1998. “An Empirical Study of Smoothing Techniques for Language Modeling,” TR-10-98, Computer Science Group, Harvard Univ., 1998.

W. John Wilbur, George F. Hazard, Jr., Guy Divita, James G. Mork, Alan R. Aronson, Allen C. Browne. 1999. "Analysis of Biomedical Text for Chemical Names: A Comparison of Three Methods". Proceedings of AMIA Symposium 1999:181-5.

Daniel M. Bikel, Richard Schwartz and Ralph M. Weischedel. 1999. "An Algorithm that Learns What's in a Name", Machine Learning

Ellen Riloff. 1996. "Automatically Generating Extraction Patterns from Untagged Text", Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), pp. 1044-1049

Silviu Cucerzan, David Yarowsky. 1999. "Language Independent Named Entity Recognition Combining Morphological and Contextual Evidence". Proceedings of 1999 Joint SIGDAT conference on EMNLP and VLC, University of Maryland, MD.

D. D. Palmer, D. S. Day. 1997. "A Statistical Profile of Named Entity Task". Proceedings of Fifth ACL Conference for Applied Natural Language Processing (ANLP-97), Washington D.C.

I. Good. 1953. "The population frequencies of species and the estimation of population parameters". Biometrika, v. 40, pp. 237-264

C.J.C. Burges, 1998. "A Tutorial on Support Vector Machines for Pattern Recognition," Data Mining and Knowledge Discovery, 2(2), pp. 955-974

T. Cover and J. Thomas, 1991. “Elements of Information Theory”, Wiley, New York.

L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, 1984. "Classification and Regression Trees," Chapman & Hall, New York.