Acronym Expansion via Hidden Markov Models

Kazem Taghva and Lakshmi Vyas
School of Computer Science
University of Nevada, Las Vegas
Kazem.Taghva@unlv.edu

Abstract- In this paper, we report on design and implementation of a Hidden Markov Model (HMM) to extract acronyms and their expansions. We also report on the training of this HMM with Maximum Likelihood Estimation (MLE) algorithm using a set of examples. Finally, we report on our testing using standard recall and precision. The HMM achieves a recall and precision of 98% and 92% respectively.

Key Words: Hidden Markov Models, HMM, supervised learning, Maximum Likelihood Estimation, MLE, Acronyms.

1. Introduction

Information Extraction (IE) is the task of automatic identification and removal of entities and relationships from unstructured objects such as text. Typically the extracted information is used to populate tables in a database setting. For example, one may be interested in finding the list of companies and their CEOs. According to [9], the extracted information can be used in a variety of applications such as:

- ***Semantic Search****: This information can be used to expand the list of metadata associated with a document and subsequently enhance the search results.*
- ***Data as a Service****: The extracted information can be cleaned up to form a set of data that can be rented or sold.*
- ***Business Intelligence****: This information can be used to aid in mining of blogs and other business data such as call centers records to address product placement advertising.*
- ***Data-driven Mashups****: The extracted information can be merged with other databases to build new data mashups.*

Historically, the database community have managed the structured data such as customer records to facilitate the enterprise information needs. The managing of the data is typically done with the standard relational databases. On the other hand, the information retrieval and natural language processing communities have dealt with the text streams to analyze and understand semantics and syntax. In a database setting, one may query the database for specific information such as the population of Las Vegas. In response, the database looks up for the answer in a table. In a search and retrieval setting such as Google, the engine will identify a collection of documents that may contain the answer. It becomes the responsibility of the user to find the specific answer in the collection of documents.

The two above mentioned approaches to information request have different roots and technologies. As our knowledge of these technologies increases, we have seen a lot of commonality in both approaches. In particular, IE seems to narrow the gaps between the two methodology.

This paper is composed of 6 sections in addition to the introduction. In Section 2, we give a background on IE and acronym expansion. Section 3 is a short description of hidden Markov Models. The detail of our HMM is given in section 4. Section 5 gives an overview of our experiments with a small collection of documents. Section 5 is the conclusion and future work.

2. Background

There are three distinct techniques in information extraction. The first method is ad hoc and has been used extensively. Good examples of this method are acronym finder and abbreviation expansion [8][2]. The second approach is the Frietag and McCallum[6] model that is based on Hidden Markov Models. The third approach is the DIPRE and other related techniques[1]. The most well known example of this is the Open IE project at the University of Washington[10]. In this paper, we want to

978-0-7695-4495-3/11 $26.00 © 2011 IEEE
DOI 10.1109/ICSEng.2011.29

develop an HMM to recognize acronyms in contrast to our early ad hoc approach.

The acronym finder was initially developed to help with error correction with Optical Character Recognition (OCR) [8]. There are many papers that have expanded on the original work for various applications. Of particular interest is the extension of acronym expansion to abbreviation expansion[2]. There are also many notable applications of the acronym finder to the biomedical areas [3][4]. It is also worth mentioning other data mining applications that has been developed to identify the acronyms[5].

The original acronym finder uses an inexact pattern matching algorithm applied to text surrounding the possible acronym. A lexicon is not used to validate words that are picked up by the program which essentially means that the spelling of the word is of little consequence. The Acronym Finder program consists of four phases namely initialization, input filtering, parsing the remaining input into words, and the application of the acronym algorithm.

The input of the algorithm is composed of several lists of words, with the text of the document as the final input stream. These inputs are:

- A list of *stopwords* – words like "the", "and", "of", that are insignificant parts of an acronym. These need to be distinguished from other words that make good matches for the acronym definitions.
- A list of *reject* words – words in the document that resemble acronyms but are words that are frequent in any document and that are known not to be acronyms. For e.g. "TABLE", "FIGURE", Roman Numerals.
- The text of the document to be searched.

The input stream is preprocessed to remove lines of text that consists of words that are all uppercase (e.g. headings, titles). Identify a candidate acronym and compare it against the list of reject words. Once it is elected as a candidate, a context window is selected around it. The text window is divided into two sub-windows, the pre-window and the post-window. The length of the sub-window is twice the number of characters in the acronym.

In order for the algorithm to work efficiently different types of words have to be identified and a priority should be assigned to each one of them. Stopwords (s) are normally ignored in traditional text but they can sometimes be part of the acronym. Precedence should be given to non-stopwords over stopwords in the matching process. Hyphenated words are a special case of words in which either the first letter of the word (H) or every first letter of the hyphenated set of words (h) correspond to the acronym. Both cases need to be tested to find the best match. Acronyms (a) can themselves be part of the definitions of other acronyms. It is therefore necessary to see if the acronym that is part of the definition is the same or a different one. Words apart from the ones that have been defined are normal words (w).

When parsing the sub-windows, two symbolic arrays are generated; the *leader* array consisting of the first letter of every word and the *type* array that is composed of the type (defined above) of every word.

The algorithm identifies the longest common subsequence of the letters of the acronym and the leader array to find a probable definition. The longest common subsequence (LCS) of any two strings is a common subsequence with the maximum length among all common subsequences. The length of the longest subsequence c[i,j]; where 'i' is the length of the prefix of a string, say X and 'j' is the length of the prefix of the comparison string, say Y; can be found recursively using the formula:

$$c[i,j] = \begin{cases} 0 \text{ if } i = 0, j = 0 \\ c[i-1, j-1] + 1 \text{ if } i, j > 0 \text{ and } X_i = Y_j \\ \max(c[i, j-1], c[i-1, j]) \text{ if } i, j > 0 \text{ and } X_i \neq Y_j \end{cases}$$

Example 2.1.

Consider the text:

The displays use arrays of Organic Light Emitting Diodes OLED to project the image onto a screen contained within the armor much like a rear projection TV.

The pre-window of the acronym OLED is:
displays use arrays of Organic Light Emitting Diodes

leader array is [d u a o o l e d]
type array is [w w w s w w w w]

The acronym finder algorithm produces all ordered arrangements of indices for all possible LCS'. The last part of the algorithm selects the appropriate definition among the LCS choices. This is done by evaluating the candidate definitions for the number of stopwords that are part of the definition, the number of words in the acronym definition that do not match the acronym, etc. There are two LCS in the case of our example, and the acronym finder selects *Organic Light Emitting Diode* as the definition as opposed to *of Light Emitting Diodes* based on the fact that nonstop words take precedence over the stop words.

3. Hidden Markov Models

A Hidden Markov Model (HMM) is a model where the stochastic process produces a sequence of observations output from states of the model but the states themselves are not seen. Consider a 3 state Markov model that models the weather of a city. The weather on a particular day can be any one of the three states mentioned below:

State 1: Snow
State 2: Rain
State 3: Sunny

The probabilities associated with the weather changing between these states can be written in a matrix as follows:

	Snow	Rain	Sunny
Snow	0.4	0.3	0.3
Rain	0.2	0.6	0.2
Sunny	0.1	0.1	0.8

Given these probabilities we can find the probability associated with a sequence of weather states such as 'sunny->sunny->snow->snow->sunny'. The probability is evaluated for the observation sequence, O = { S_3,S_3,S_1,S_1,S_3}.

P(O|Model) = P[S_3, S_3, S_1, S_1, S_3|Model]
= P[S_3] * P[S_3|S_3] * P[S_3|S_1] * P[S_1|S_1] * P[S_1|S_3]
= 0.4 * 0.8 *0.3*0.4*0.1 (assuming that initial probability of S_3 is 0.4)
= 0.00384

To someone who is oblivious to the weather conditions, because he is confined to a small closed space, it is possible to draw inferences on the weather based on the way his visitor is dressed i.e. if the guest is wearing a coat (C) or not (D). Consider the probability that the visitor wears a coat is 0.1 on a sunny day, 0.3 on a rainy day and 0.7 on the day it snows. Finding the probability of a certain type of weather q_i can be based on the observation x_i. The conditional probability $P(q_i|x_i)$ can be written according to Bayes' rule as:

$$P(q_i|x_i) = \frac{P(x_i|q_i)P(q_i)}{P(x_i)}$$

This can be easily generalized for n days, the weather sequence $Q = \{q_1, \dots, q_n\}$, as well as the sequence of observations $X = \{x_1, \dots, x_n\}$ to:

$$P(q_1, \dots, q_n|x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n|q_1, \dots, q_n)P(q_1, \dots, q_n)}{P(x_1, \dots, x_n)}$$

Using the probability $P(q_1, \dots, q_n)$ from above and $P(x_1, \dots, x_n)$ of seeing a particular sequence of coat events. The probability $P(x_1, \dots, x_n|q_1, \dots, q_n)$ can be estimated as $\prod_{i=1}^{N} P(x_i|q_i)$, when it is assumed for all i that q_i, x_i are independent of all x_j and q_j for all $j \neq i$. The probability of seeing the visitor wear a coat is independent of the weather that we like to predict, so we can disregard $P(x_1, \dots, x_n)$. This measure is now referred to as Likelihood.

Assume that the person knows that the day he entered confinement was Sunny. The visitor on the next day carries a coat with him. Using this information and the probabilities it is not difficult to analyze what the weather most likely weather condition outside is. This evaluation is done as follows:

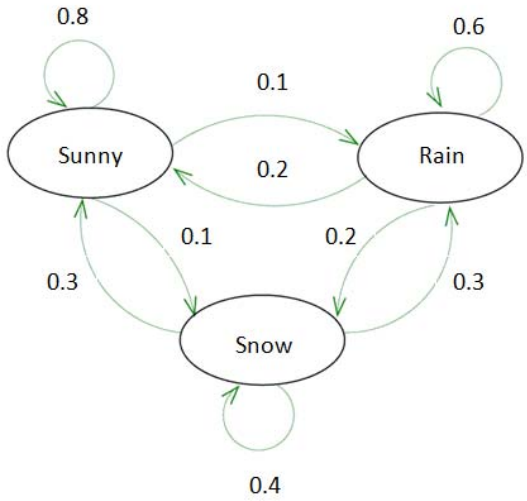
Likelihood that the second day is sunny
$= P(x_2 = C|q_2 = S_3).P(q_2 = S_3|q_1 = S_3)$
= 0.1 * 0.8 = 0.08

Likelihood that it is raining on the second day is
$= P(x_2 = C|q_2 = S_2).P(q_2 = S_2|q_1 = S_3)$
= 0.3 * 0.1 = 0.03

Likelihood that it is snowing on the second day is
$= P(x_2 = C|q_2 = S_1).P(q_2 = S_1|q_1 = S_3)$
= 0.1 * 0.7 = 0.07

The highest of these probabilities is 0.08. From the result obtained we find the highest probability is associated with Sunny weather even though the visitor brings a coat with him.

Formally said, it is possible to find the sequence of physical events when a string of observations generated by these events is available with the use of Hidden Markov models. Now that we have understood the basic idea behind HMMs, we delve into some of the specifics. The above process is typically depicted as the following figure:



An HMM is characterized by a 5-tuple (S, V, π, A, B), where

- S is a finite set of N states $\{S_1, \ldots, S_N\}$. Although the states are hidden, for many practical applications there is some significance associated to the states of the model.
- V is the set of M distinct symbols in the vocabulary of an HMM. The M observation/emission symbols correspond to the physical output of the system being modeled.
- $\pi = \{\pi_i\}$ are the initial state probabilities where $\pi_i = P[q_1 = S_i]$ and $1 \leq i \leq N$
- $A = \{a_{ij}\}$ are the state transition probabilities where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$, when $1 \leq i, j \leq N$.
- $B = \{b_j(o_k)\}$ are the emission probabilities. The observation symbol probability distribution in state i is $B = \{b_i(k)\}$, where $b_i(k) = P[\{o_k \text{ at } t | q_t = S_i\}]$ and $1 \leq i \leq N$ and $1 \leq k \leq M$.

We use $\lambda = (A, B, \pi)$ to denote the complete parameter set of the HMM. The constraints on the HMM are

$$\sum_{i=1}^{N} \pi_i = 1 \text{ for } 1 \leq i \leq N$$

$$\sum_{j=1}^{N} a_{ij} = 1 \text{ for } i = 1, 2, \ldots N$$

$$\sum_{k=1}^{M} b_i(o_k) = 1 \text{ for } i = 1, 2, \ldots N$$

For the model to be useful in real world applications, three problems need to be addressed. These problems are:

1. **Evaluation:** Evaluating the probability of an observed sequence of symbols $O = o_1 o_2 \ldots o_T$ $(o_t \in V)$, given a particular HMM, i.e. $p(O|\lambda)$.
2. **Decoding:** Finding the most likely sequence of states for the observed sequence. Let q = $q_1 q_2 \ldots q_T$ be a sequence of states. We want to find $q^* = argmax_q p(q|O, \lambda)$. This is done using the Viterbi algorithm[].
3. **Training:** Adjusting all the parameters λ to maximize the probability of generating an observed sequence, i.e., to find $\lambda^* = argmax_\lambda p(O|\lambda)$. This is done with Maximum Likelihood Estimation (MLE) algorithm[].

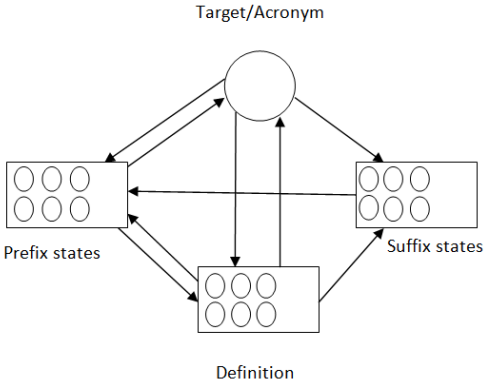
We refer the reader to the standard papers on the subject for more details.

4. Acronym HMM

The design of the HMM to extract acronyms with their definitions is similar to the design chosen by Frietag and McCallum [6] to model two data sets namely, online seminar announcements and Reuter's newswire articles on company acquisitions. The HMM model for acronyms has 4 states:

- Prefix (0)
- Acronym (1)
- Definition (2)
- Suffix (3)

The states of special consequence are the Acronym and the Definition state. These states are also called the target states; the words that are generated by them are candidate acronyms and candidate definitions respectively. To capture context certain restrictions have been placed on the transitions between states. This is illustrated in figure below:



As can be seen from the figure, the HMM is not entirely ergodic. While tagging data to train the HMM model (for calculating probabilities) it is important to keep the topology of transitions in check. In the event that acronyms occur in quick succession the target states can transition to the prefix state without traversing to suffix states. It is however not possible to reach target states from the suffix state, the transition must happen through the prefix state. The rules states above are what the model implies.

It has been stated in section 3.1 that an essential part of an HMM is a definition of its emission vocabulary. The vocabulary associated with the acronym finder HMM

consists of three symbols, each of which represents a type of word. The symbols are

- A – for the acronym that is made of all uppercase letters
- D – for possible definition of the acronym that starts with an uppercase letter
- n – for all other words in the text.

The document collection is first pre-processed before it can be used in the learning module or the decoding module. When the document collection (pre-processed) is parsed by the routines; every word in the document is translated to one of the above symbols. The routines that have been written do not capture context or semantics; they are only concerned with the sequences of symbols the words translate to.

MLE is calculated on a tagged document of words, rather symbols with states. The probabilities that are calculated in this manner are transition probabilities between all combinations of states and symbol emission probabilities associated with every state. Initial probabilities are set by hand. Let us consider the following statement:

The example explains how Maximum Likelihood Estimate (MLE) works in our paper

.

Preprocessing and tagging this statement would result in a sequence such as the one shown below:

The 0
example 0
explains 0
how 0
Maximum 2
Likelihood 2
Estimate 2
MLE 1
works 3
in 3
our 3
thesis 3

The routine that uses MLE reads this file one line at a time. The very first thing that is done by the routine is the translation of words into one of the defined emission symbols. The routine keeps track of the current state and the previous state and keeps a count of the number of times the combination is encountered in the file. In the example that is considered it can be seen that the transition from state 0 to state 0 happens 3 times and the total number of transitions from state 0 in the text is 4. The probability associated with the transition from state 0 to state 0 is therefore 0.75. In a manner similar to what is described the emission of symbol 'n' from state 0 happens 3 times in our example and the symbol 'D' is generated once from state 0. The probability associate with the state 0 for emitting symbol 'n' is the number of times symbol 'n' is emitted from state 0 divided by the total number of emissions from state 0 i.e. 0.75.

The Viterbi algorithm uses these probabilities to find the best sequence of states for the input string.

5. Experiment

A set of 200 documents were randomly collected from various articles available on the Internet. The only pre-requisite that was to be satisfied by every file was the occurrence of at least one acronym with the definition in its vicinity. The collection of documents was divided into two categories. One set of 100 documents were used to train the HMM model and the other was used to test the same model so as to evaluate the performance of the HMM model that was designed.

Before either set of documents were used further, each of the set of documents were put through a pre-process phase. Pre-processing of the training documents and the set of testing documents were conducted separately although the process involved was essentially the same. The decision to make the tradeoff between saving time and accuracy makes the results of the experiments credible. A routine was written to do two tasks; strip the document collections of any punctuation and special characters, white space characters were replaced with new line characters in order to allow easy tagging and easy scanning of the result set. The output of the pre-processing step is a file that has no punctuations, no special characters and has only one word per line. Initial probabilities are set by hand so that all states have a fair chance of being the start state. If MLE were to decide these initial probabilities, only the prefix state would be assigned a large probability while the others would have a probability closer to zero as documents start at the prefix state most often. The output of the learning phase is a file with probabilities associated with the HMM model.

The 100 documents set aside for testing use the probabilities calculated in the previous step. The output of the decoding phase is a file similar in appearance to the training file. This file is analyzed by a human observer (not necessarily aware of the design) for words that are tagged with state 1 and 2 as they are candidate acronyms and definitions respectively. The file is checked for True Positives, False Positives and False Negatives.

Standard measures of Precision, Recall and F1 measure are used to evaluate the performance of the HMM.

Precision is defined as

$$Precision = \frac{Number\ of\ True\ Positives}{Number\ of\ True\ Positives + False\ Positives}$$

Recall is defined as

$$Recall = \frac{Number\ of\ True\ Positives}{Number\ of\ True\ Positives + False\ Negatives}$$

The results obtained as shown below:

True positive	False Positive	False Negative	Precision	Recall
196	16	4	0.9245	0.98

6. Conclusion

In this paper, we reported on two methods of finding acronyms and their definitions. The second method is based on HMM and is more general that can be applied to other text mining applications. It requires a lot of machinery that may not be appropriate for some applications.

7. References

[1] Sergey Brin, Extracting Patterns and Relations from the World Wide Web, Selected papers from the International Workshop on The World Wide Web and Databases, p.172-183, March 27-28, 1998

[2] Park, Y., and Byrd, R.J., Hybrid text mining for finding abbreviation and their definitions. In Proceedings of EMNLP, 2001.

[3] Ariel S. Schwartz, Marti A. Hearst , ***A Simple Algorithm For Identifying Abbreviation Definitions in Biomedical Text*** . 2003.

4] Jeffrey T. Chang, Hinrich Schütze, Russ B. Altman, **Creating an online dictionary of abbreviations from MEDLINE** Journal of the American Medical Informatics Association, 2002.

[5] David Nadeau, Peter D. Turney , **A supervised learning approach to acronym identification** In 8th Canadian Conference on Artificial Intelligence (AI'2005) (LNAI 3501), 2005.

[6] Freitag & McCallum 1999] Freitag, D. and McCallum, A. Information extraction using HMMs and shrinakge. In Proceedings AAAI-99 Workshop on Machine Learning for Information Extraction. AAAI Technical Report WS-99-11

[7] L. R. Rabiner and B.H. Juang. An introduction to hidden markov models.

IEEE ASSP Magazine, pages 4–16, 1986.

[8] Kazem Taghva and Jeff Gilbreth, Recognizing acronyms and their definition ,191-198, 1999.

[9] Laura Chiticariu, Yunyao Li, Sriram Raghavan, and Frederick Reiss: "Enterprise Information Extraction: Recent Developments and Open Challenges". SIGMOD 2010 (tutorial)

[10] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni, Open Information Extraction from theWeb, IJCAI-07