Text Categorization using N-grams and Hidden-Markov-Models

Thomas Mathew

tam52@georgetown.edu
Department of Linguistics
Georgetown University,
Washington DC 20057-1232

December 8 2006

Abstract: In this paper I discuss an approach for building a soft text classifier based on a Hidden-Markov-Model. The approach treats a multi-category text classification task as predicting the best possible hidden sequence of classifiers based on the observed sequence of text tokens. This method considers the possibility that different sections of a large block of text may hint towards different yet related text categories and the HMM predicts such a sequence of categories. The most probable such sequence of categories can be estimated using the Viterbi algorithm.

Keywords: Text Categorization, N-gram, Hidden-Markov-Model, HMM, Viterbi Algorithm

1. Introduction

Text categorization is the process of assigning topic(s) or theme(s) to a document. Categorization can be *hard* when a document can get at most one category or *soft* when a document can belong to multiple categories with perhaps varying degree of significance. Soft classifiers can be argued to have a greater significance from a linguistics perspective given the indeterministic and possible multi-class result of the same categorization task performed by a human. Applications of text categorization include filtering a stream of news for a particular interest group, binary categorization (discard, pass-through) for spam filtering, catalog web pages.

A variety of methods have been previously evaluated for text categorization. Manning & Schutze (2003) discuss using machine learning algorithms such as Decision Trees, Perceptrons, k Nearest Neighbor for text categorization tasks. Nigam, McCallum, Thrun & Mitchell (1999) discuss a Naïve-Bayes text classifier and suggest use of EM algorithm for situations lacking sufficient training data. These approaches treat documents as a 'bag of tokens' i.e. a unigram model with no token inter-dependency. Despite this naïve assumption, unigram algorithms do perform well and Manning & Schutze (2003) report classification accuracy of up to 96% on the Reuters corpus. N-gram models have also been applied in text categorization (Peng & Schuurmans 2003). N-gram binary Spam-filtering categorization is discussed by Keselj, Milios, Tuttle, Wang & Zhang (2005). N-grams have also been applied towards classification of byte sequences (Maloof 2004).

Hidden-Markov-Models have been used in situations where hidden states are assumed to be generating an observable output sequence. In the case of a document, the observable output sequence can be the words. In case of a multi-section document, each section can be considered to be the observable state in an HMM as discussed by Frascone, Soda, Vullo (2002). In linguistics, HMM has been predominantly applied towards speech-processing and part-of-speech tagging.

This paper evaluates the feasibility of combining an N-gram model and HMM to create a soft classifier. The observable states of the HMM is considered to be a sliding-window N-gram over the document text. As the window moves across the whole document text, we can determine the most probable sequence of predicted categories. For evaluation purposes of this approach, the Reuters corpus has been used.

2. N-gram Models

Word order can be argued to have some significance in text categorization. For example in a bigram model the phrase '*foreign exchange*' may strongly hint at a *money-fx* topic. In a unigram model, word frequencies of the independent tokens *foreign* and *exchange* may also hint towards a *money-fx* topic. This may give undue weight towards the *money-fx* category for the sentence fragment '*exchange of notes with the Chinese Minister of Foreign Affairs*'.

More than two tokens can be used to build a model. However in an N-gram model the parameters influencing the model grow exponentially with N and hence a 5-gram model may not be practical. Larger N also suggests requiring a larger training set since the patterns tend to be more sparsely distributed across documents. To minimize parameters, this paper shall be restricted to analysis unigram, bigram and trigram models. The parameters influencing a model with larger N can be reduced by dropping high-frequency 'noise' words such as determiners (*a, the, our, my*) and prepositions (*to, in, for, under*). Manning & Schutze (2003) discuss a single HMM model which includes unigram, bigram and trigram observable states with varying weights. The weights are adjusted based on the training data. The figure below shows N-gram counts extracted from a random segment of the Reuters corpus. The deterioration of token counts with increase in N can be observed.

Topic	Document Count	Unigram		Bigram		Trigram	
		Token	Count	Token	Count	Token	Count
crude	27	oil	52	mln dlrs	42	dlrs a barrel	6
		platform	29	oil platform	15	attack oil platform	3
money-fx	27	rates	36	interest rates	26	shortterm interest rates	5
		interest	25	currency stability	7	on currency stability	5
acq	70	company	76	common stock	25	said it acquired	8
		dlrs	73	tender offer	11	shares common stock	7

Table 1: N-gram analysis of most common topics in the Reuters corpus

3. Hidden-Markov-Model

Hidden-Markov-Models (HMM) have been successfully applied in speech recognition, part-of-speech taggers and other categorization tasks where ordering of observable features in a dataset is relevant to the task. A standard HMM model uses a discrete hidden state at time *t* to summarize all the information before *t* and thus the observation at any time only depends on the current hidden state (Zhong 2004). A standard HMM is usually denoted as a model (π, *A*, *B*).

$\pi = \{\pi_i\}$
is the prior probability distribution of hidden states at time $t = 0$

$A = \{a_{ij}\}$
is the transition probability distribution between hidden states *i* and *j*

$B = \{b_{jn}\}$
is the emission probability distribution of an observation *n* given a hidden state *j*. (Note that this is for a *state-emission HMM*)

In this particular problem of text categorization, the visible states of the HMM can be considered to be a sequence of N-grams. The hidden states can be considered to be the categories that are can be assigned to the text. A block of text can have more than one category and as a consequence can have more than one hidden state. We shall also have *unknown* categories to indicate an indeterminate category state.
This particular implementation of an HMM assumes that every N-gram in an incoming sequence predicts multiple categories with a varying probability distribution as shown in the figure below. The highlighted path is the most probable category states which predict the observable sequence given the model (π, *A*, *B*). A key assumption in this paper is the creation of *unknown* hidden category states shown as C_J.

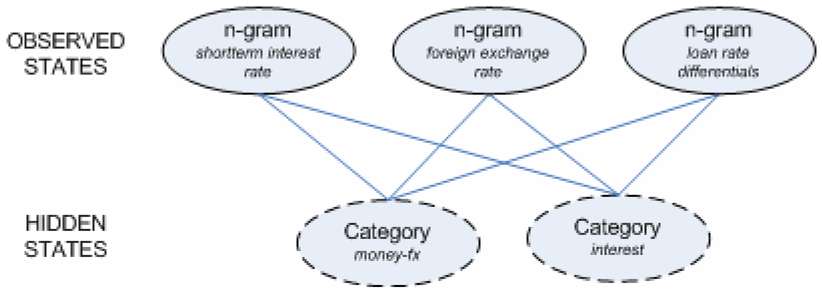

Figure 1: HMM states for Text Categorization

The categories predicted by this model for the N-gram sequence $\{N_1, N_2 \ldots N_K\}$ would be distinct categories extracted from the most probable path through the model. The most probable path can be determined using the Viterbi algorithm. A different way at looking at N-gram sequence would be to have an HMM sequence on single words but where the horizon of each state is based on the preceding N states.

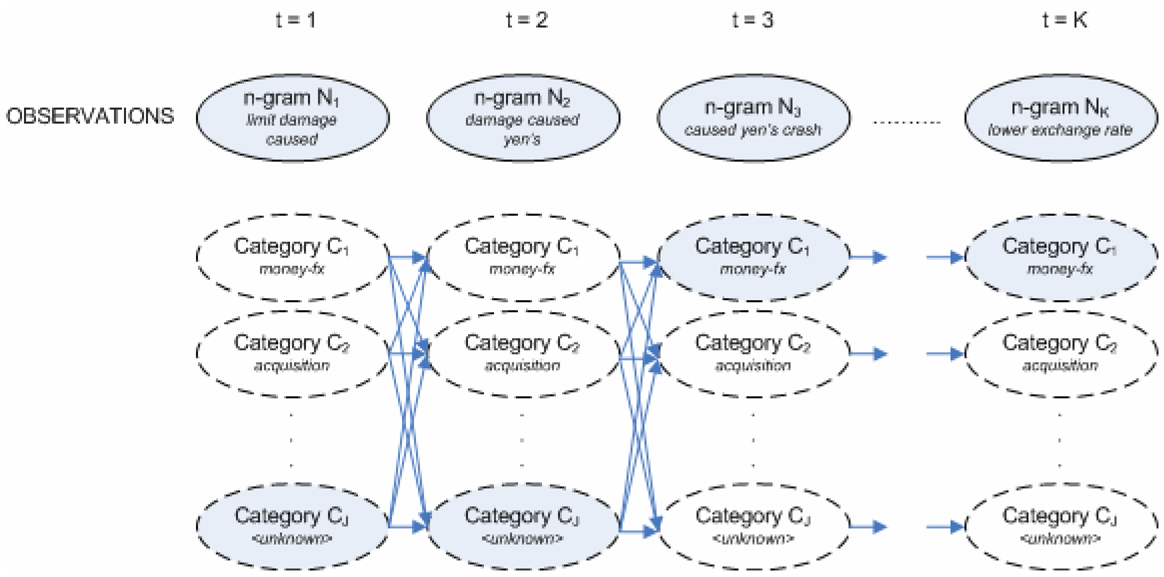

Figure 2: Category paths and most predicted category path through an HMM

4. Viterbi Algorithm

The Viterbi algorithm is an efficient means of computing the most probable path through hidden states for a given an HMM model (π, *A*, *B*). This involves the computation of the variable

$$\delta_j(t) = \max_{C_1..C_{t-1}} [\, P(C_1 .. C_{t-1}, N_1 .. N_{t-1}, C_t = j \mid \pi, A, B)\,] \qquad 1 \le t \le K$$

$\delta_j(t)$ identifies the probability of being in state *j* at time *t*. This is based on the prior hidden categories $C_1 .. C_{t-1}$ and the prior observed N-grams $N_1 .. N_{t-1}$. The computation of this algorithm is done recursively at starting at the initial state at time $t = 0$ and looping till $t = K$.

$$\delta_j(1) = \pi_j$$

this represents the initial state of the HMM

$$\delta_j(t+1) = \max_{i=1..J} [\, \delta_j(t) a_{ij} b_{jn} \,] \qquad 1 \le j \le J$$

$$\Psi_j(t+1) = \operatorname{argmax} [\, \delta_j(t) a_{ij} b_{jn} \,] \qquad 1 \le j \le J$$

i = 1..J
this represents most likely category at time *t* given N-gram *n* and category *j* at time *t + 1*.
Ψ_j is simply a marker for tracing back the most probable path

The above variables are evaluated for every time *t* in the observed sequence. For underflow reasons, the above variables are best implemented in logarithmic scale. The most probable path is then evaluated backwards starting at the final state and reading through $\Psi_j(t + 1)$ decreasing *t* till the initial state is reached. The algorithm is described in more detail in Manning & Schutze (2003). The values of $log(\delta_j(t))$ is shown below for the sentence '*the statement was a reaction to rises in foreign exchange rates*' with the highlighted path showing the most likely path through the model. Interestingly for the triggering point for the change in topic for this sentence was the bigram *a reaction*.

t	Bigram Token	Unknown log(δ)	money-fx log(δ)	interest log(δ)	acq log(δ)
0	the statement	-0.11	-6.09	-6.09	-6.09
1	statement was	-5.97	-9.88	-9.88	-9.88
2	was a	-9.84	-10.43	-15.04	-14.34
3	a reaction	-16.11	-11.80	-19.15	-20.03
4	reaction to	-17.80	-12.74	-19.40	-25.19
5	to rises	-19.86	-14.11	-21.27	-27.68
6	rises in	-30.26	-25.91	-27.96	-34.18
7	in foreign	-32.84	-26.69	-34.45	-40.27
8	foreign exchange	-32.68	-27.63	-34.29	-40.50
9	exchange rates	-34.33	-29.28	-34.33	-42.16

Table 2: Viterbi algorithm applied to HMM of bigrams

5. Approach

5.1. Datasets

The approach was tested against the Reuters newswire corpus which is a collection of SGML documents some of which have been classified by hand. A block of text can be assigned more than one text category. The Reuters corpus has a predetermined set of text categories with varying extent of coverage. Based on a sampling of one section of text, the classification task was restricted to the categories *acq*, *interest*, *crude*, *ship*, *money-fx* and *earn* to assure sufficient data coverage.

5.2. Text Preprocessing

A sample news article from Reuters is shown below –

```
<REUTERS  TOPICS="YES"  LEWISSPLIT="TEST"  CGISPLIT="TRAINING-
SET" OLDID="20430" NEWID="21007">
<DATE>19-OCT-1987 15:30:22.56</DATE>
<TOPICS><D>acq</D></TOPICS>
...
<TITLE>BROWN   DISC   TO   BUY   RHONE-POULENC   &lt;RHON.PA>
UNIT</TITLE>
<DATELINE>    COLORADO SPRINGS, Colo., Oct 19 - </DATELINE>
<BODY>Brown  Disc  Products  Co  Inc,  a  unit  fo  Genevar
Enterprises Inc, said it has purchased the ongoing business,
trademarks and certain assets of Rhone-Poulenc's Brown Disc
Manufacturing unit, for undisclosed terms. Rhone-Poulenc is a
French-based chemical company. Under the agreement, Rhone-
Poulenc will supply magnetic tape and media products to Brown
Disc Products.
</BODY></TEXT></REUTERS>
```

Figure 3: Sample document from the Reuters corpus

The SGML content was pre-processed by applying the following steps for every block of text within a `<REUTERS>` tag and `</REUTERS>` tag

- Extract the text from within `<TOPIC>` and `</TOPIC>` tags and identify all topics if any. Documents without topics are discarded
- Extract the text from within `<BODY>` and `</BODY>` tags
- Eliminate special characters `&`, `(`, `)`, `#`, `‘`, `“` using a regex based search-and-replace
- Replace all proper names with a generic token `#NAME#`. e.g. `Brown Disc Products Co Inc` would be replaced with `#NAME#`.
- Replace all numbers with a generic token `#NUMBER#`.
- Assume all words ending with `–ly` are adjectives and eliminate from the text
- Remove all determiners (*the, a, an, this, these, his, her* and variations) from the text
- Remove all conjunctions (*and, or*)
- Remove all prepositions (*in, under, above, from, upto, about, to, in* and variations)
- Remove all tense words (*be, is, am, shall, are, will, would, has, had* and derivations)

The documents were split into independent collections of training and testing data.

5.3. HMM Hidden State ‘unknown’

A key assumption in this approach is the creation of a dummy hidden state *unknown* which is added to the list of possible categories that can be assigned to a block of text. At the initial state of the HMM ($t = 0$) the *unknown* category is assumed to be the most likely category that can be assigned.

5.4. Initializing the HMM model (π, A, B)

This paper approaches initialization of the model based on a training dataset rather than using random values. For the initial state π, I have assumed a probability of 0.9 that the category is ‘*unknown*’ and distributed the other categories uniformly distributed over a probability of 0.1.

The state transition probability distribution *A* can be computed by counting actual examples in the training set where categories co-exist for the same text. The maximum probability is assigned to category *j* at time *t* predicting category *j* at time *t+1*. A dummy state transition probability to and from category *‘unknown’* is also created.

Since there is no notion of category sequence in the Reuters corpus, a special case of HMM’s called *state-emission HMM* is used. In a *state-emission HMM*, the N-gram emitted at time *t* depends just on the category at time *t* (Manning & Schutze 2003). Given that, the emission probability distribution *B* can be computed by counting actual examples in the training set where a particular N-gram in the training data is predicted by a category. A minimal dummy emission probability for category *‘unknown’* is also created for every N-gram.

5.5. Evaluating categories

Categories for a block of text are determined by applying the Viterbi algorithm to the N-grams that can be determined from the text. The most likely path through the HMM is calculated and the categories are determined to be the unique collection of all categories from the path. The dummy category *unknown* is discarded.

5.6. Evaluating performance

As discussed above, due to limited coverage on other categories, the performance evaluation was restricted to precision and recall over categories *acq, interest, crude, ship, money-fx* and *earn*. For the testing data, the following metrics are captured –

$Count_{docs}$	=	Number of documents to be classified
$Count_{docscorrect}$	=	Number of documents where at least one topic was assigned correctly
$Count_{topics}$	=	Number of topics to be classified
$Count_{correct}$	=	Number of correctly assigned topics
$Count_{incorrect}$	=	Number of categories which were incorrectly assigned by the HMM
$Count_{not\text{-}recalled}$	=	Number of categories which were not predicted by the HMM

$$Precision_{docs} = \frac{Count_{docscorrect}}{Count_{docs}}$$

$$Precision_{topics} = \frac{Count_{correct}}{Count_{correct} + Count_{incorrect}}$$

$$Recall_{topics} = \frac{Count_{correct}}{Count_{correct} + Count_{not\text{-}recalled}}$$

6. Experimental Results

Three experimental tests were run to evaluate the feasibility of this approach and on the impact of varying different parameters to gain some insight into impact on performance. Ten-fold Cross Validation was used in the capture of all metrics. N-grams with high N and limited pruning are very memory intensive and for purposes of this experiment, only documents of length < 1000 characters were used for training to prevent machine memory overflows. It should be kept in perspective that a 75% precision figure would give an error rate of 1 per 4 documents. This may be unacceptable is a lot of categorization applications such as spam-filtering.

6.1. *Varying N*

The data below shows comparative results of an HMM model with varying N from 1 to 3. In all cases, pre-processing rules were applied. The results are shown in Table 3, Table 4 and Table 5. Figure 4 shows the clustering of the different precision and recall measures for all ten executions.

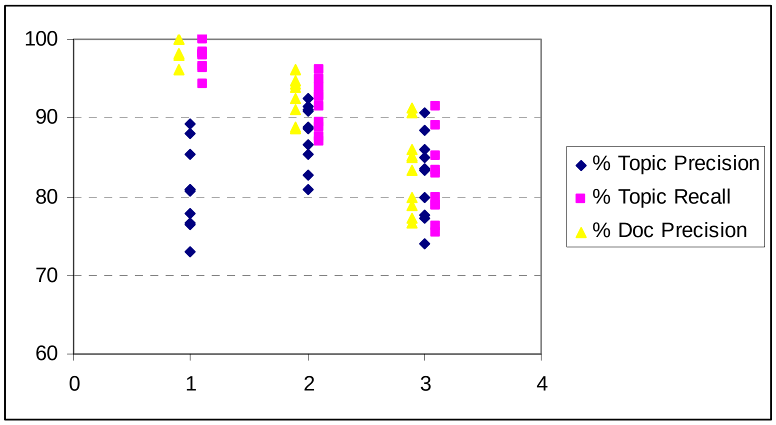

Figure 4: Experimental results from varying N for a particular HMM

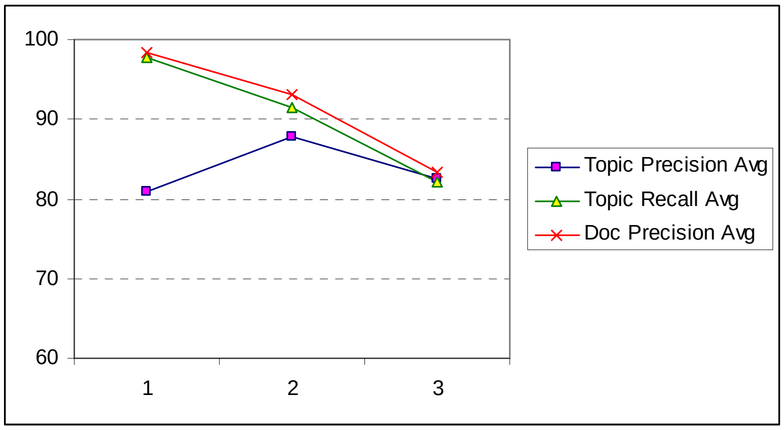

Figure 5: Experimental results from varying N for a particular HMM

N	# Docs	# Docs 1 Topic Correct	# Docs Topic Last Correct	# Topics	# Topics Correct	# Topics Wrong	# Topics Not Recalled	% Topic Precision	% Topic Recall	% Doc Precision
1	54	53	50	54	53	15	1	77.9	98.1	98.1
1	54	52	49	55	53	9	2	85.4	96.3	96.2
1	52	51	46	52	51	12	1	80.9	98.0	98.0
1	54	53	47	58	56	17	2	76.7	96.5	98.1
1	56	56	44	57	57	21	0	73.0	100.0	100.0
1	58	58	53	59	59	8	0	88.0	100.0	100.0
1	50	49	43	50	49	15	1	76.5	98.0	98.0
1	57	57	54	59	58	7	1	89.2	98.3	100.0
1	53	53	46	56	55	13	1	80.8	98.2	100.0
1	53	51	47	54	51	12	3	80.9	94.4	96.2

Table 3: Experimental results for an HMM model with N=1 with pre-processed text

N	# Docs	# Docs 1 Topic Correct	# Docs Topic Last Correct	# Topics	# Topics Correct	# Topics Wrong	# Topics Not Recalled	% Topic Precision	% Topic Recall	% Doc Precision
2	54	48	47	54	48	10	6	82.7	88.8	88.8
2	54	51	50	55	51	5	4	91.0	92.7	94.4
2	52	50	50	52	50	4	2	92.5	96.1	96.1
2	54	50	49	57	50	5	7	90.9	87.7	92.5
2	56	51	46	57	51	12	6	80.9	89.4	91.0
2	58	55	53	59	56	7	3	88.8	94.9	94.8
2	50	47	45	50	47	6	3	88.6	94.0	94.0
2	57	54	53	59	54	5	5	91.5	91.5	94.7
2	53	51	47	56	52	8	4	86.6	92.8	96.2
2	53	47	46	54	47	8	7	85.4	87.0	88.6

Table 4: Experimental results for an HMM model with N=2 with pre-processed text

N	# Docs	# Docs 1 Topic Correct	# Docs Topic Last Correct	# Topics	# Topics Correct	# Topics Wrong	# Topics Not Recalled	% Topic Precision	% Topic Recall	% Doc Precision
3	54	46	46	54	46	9	8	83.6	85.1	85.1
3	54	49	49	55	49	5	6	90.7	89.0	90.7
3	52	41	40	52	41	12	11	77.3	78.8	78.8
3	54	45	45	57	45	9	12	83.3	78.9	83.3
3	56	43	43	57	43	15	14	74.1	75.4	76.7
3	58	53	52	59	54	7	5	88.5	91.5	91.3
3	50	40	40	50	40	10	10	80.0	80.0	80.0
3	57	49	49	59	49	8	10	85.9	83.0	85.9
3	53	41	41	55	42	12	13	77.7	76.3	77.3
3	53	45	45	54	45	8	9	84.9	83.3	84.9

Table 5: Experimental results for an HMM model with N=3 with pre-processed text

N	# Docs	# Docs 1 Topic Correct	# Docs Topic Last Correct	# Topics	# Topics Correct	# Topics Wrong	# Topics Not Recalled	% Topic Precision	% Topic Recall	% Doc Precision
2	54	51	49	54	51	15	3	77.2	94.4	94.4
2	54	51	49	55	52	9	3	85.2	94.5	94.4
2	52	50	44	52	50	14	2	78.1	96.1	96.1
2	54	51	48	57	52	9	5	85.2	91.2	94.4
2	56	55	45	57	56	24	1	70.0	98.2	98.2
2	58	58	52	59	59	12	0	83.0	100.0	100.0
2	50	49	44	50	49	10	1	83.0	98.0	98.0
2	57	57	53	59	57	10	2	85.0	96.6	100.0
2	53	50	44	56	52	17	4	75.3	92.8	94.3
2	53	52	47	54	52	14	2	78.7	96.2	98.1

Table 6: Experimental results for an HMM model with N=2 without preprocessing

Figure 5 shows topic recall and document precision rates dropping with N. Topic precision rates is maximum for a bigram model.

6.2. *Significance of text pre-processing*
This test assesses the significance of pre-processing of text by comparing performance with and without the pre-processing rules discussed earlier. The test is done for a bigram model. The accuracy of the pre-processed text is slightly higher than on text that has not been pre-processed. However the reverse holds true for recall. It was observed that the memory requirements drastically increased without pre-processing. This is understandable given that the number of word pairs would be significantly higher given the larger vocabulary of words that would be found.

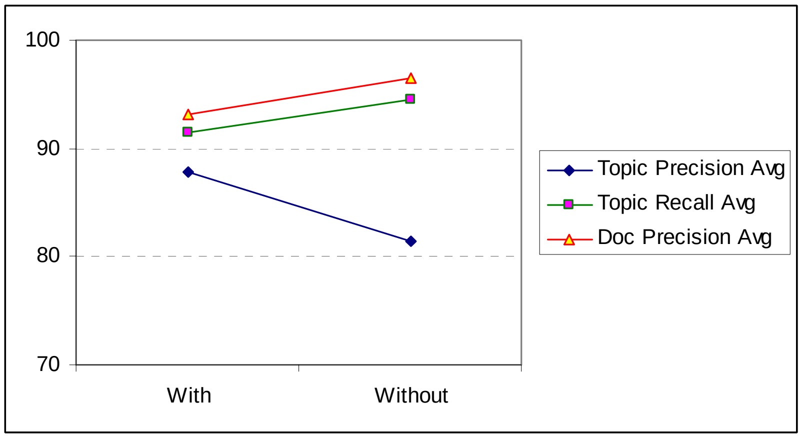

Figure 6: Experimental results with and without pre-processing for a particular HMM with bigram

7. Conclusion

For the Reuters corpus, we have shown that text categorization can be applied with a Hidden Markov model with observable states based on N-grams. It appears that larger N results in a decrease in performance which is expected as patterns tend to become sparser. The approach does not reach the levels of performance claimed by Manning & Schutze (2003) for k-NN and Decision Trees on the same Reuters corpus however it should be noted that the results reported was based on training on 50-60 documents and results may improve on training on larger sets. Higher levels of accuracy may be obtained by extending the approach with the EM algorithm. It should be noted that this approach is memory intensive.

7.1. Baum-Welch EM Algorithm
Also known as the forward-backward algorithm, the Baum-Welch EM algorithm is a means for determining an HMM model (π, A, B). The algorithm starts with initializing the model parameters to some random amounts keeping a restriction in place that the sum over a probability distribution should be 1. The value of the initial model is assessed over a training dataset and the model is adjusted until a local maximum is reached. The algorithm is described in more detail in Manning & Schutze (2003). I intend to apply the Baum-Welch algorithm to incrementally determine an alternate model (π_1, A_1, B_1) which describes a validation set more accurately. The model can be initialized as was described earlier and the model parameters can be adjusted till a local maximum performance peak is reached.

7.2. Acknowledgements
I would like to acknowledge my advisor Markus Dickinson for his suggestions and to Mark Maloof for the opportunity to write this paper.

8. References

Manning, C., & Schutze, H. (2003). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1999). Text Classification from Labeled and Unlabeled Documents using EM. In *Kluwer Academic Publishers*.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Zhong, S. (2004). Semi-supervised Sequence Classification with HMMs. In *International Journal of Pattern Recognition and Artificial Intelligence*.