Using Hidden Markov Models to Extract Target Phrases From Text

Final Project CS224N
Spring 2003

Jenny Finkel
jrfinkel@stanford.edu
Penka Vassileva Markova
penka@stanford.edu

1. Introduction

For our project we were interested in extracting target phrases from text. This has many applications − we focused on extracting speaker, location and time information from email announcements and on extracting the type of cuisine from restaurant reviews. We tried to implements a flexible Hidden Markov Model so that we could find the best structure for the type of data being extracted. We were pleased with the performance of our system − because our training data was not hand tagged there were even instances when our system (in our opinion) pulled out phrases which were more relevant to the question than those phrases that were tagged.

2. Our Data Sets

When looking for data sets we were interested in having some variety in the types of information being extracted. We also specifically wanted data sets where there could be more than one correct answer for a target and where the target could occur more than once in the text. We were less concerned with how many types of targets where in a particular text or the possible range of target values.

2.1 Cuisine Type

We created one of the datasets ourselves using information we found on the web. We wanted restaurant reviews from which we would extract information on the type of cuisine, but the restaurant dataset we found of AFS ('Musclea−Zagat') was insufficient − there were not enough reviews and the reviews did not tend to actually mention the type of cuisine, a critical characteristic. Instead we found a list of reviews on the website for the 'Online Washingtonian' which were perfect for our needs. On the webpage for each review there was a separate list of cuisines. For most reviews, most of the cuisines that the restaurant was labeled under, where mentioned in the review. Lastly, they had a master list with links to every review so it was easy to systematically find every review. We parsed the reviews into files containing a list of possible target values (the cuisines) and the text of the review with all instances of target values tagged (for example '*and <cuisine>Turkish</cuisine> dishes'*). In the list of possible target values we only


Address http://www.washingtonian.com/dining/Profiles/kuzine.html

WASHINGTONIAN RESTAURANT REVIEW

HOME
SUBSCRIBE
RESTAURANTS & DINING
ARTS & ENTERTAINMENT
VISITORS GUIDE
TRAVEL & RECREATION
HEALTH & MEDICINE
SCHOOLS & COMMUNITY
SHOPPING & SERVICES
CAREERS & MONEY
REAL ESTATE
CLASSIFIEDS & PERSONALS
PEOPLE

Kuzine Turkish·Middle Eastern·Russian Inexpensive

Rockville
302 King Farm Blvd., Suite 110
Rockville, MD
301-963-3400
Open daily for lunch and dinner. Wheelchair accessible.

Up to now, Turkish restaurants like Nizam's (Vienna), Temel (Fairfax), and Kazan (McLean) have clustered in Northern Virginia. Kuzine, an engaging storefront cafe which opened this fall, is Rockville's (and possibly southern Maryland's) only Turkish eatery. What makes things even more interesting is that chef Aylan Hisirci, who owns the place with her sister, Selva Gunenc, cooked in Russia for more than a decade, so Kuzine offers Eastern European and Russian dishes as well.

A combo platter of rice-filled bell pepper, artichoke hearts with carrots, peas, and dill, and olive-oil-shiny eggplant done up with tomatoes, garlic, and onion makes a fine start. Or you may want to try the borscht, here a hearty stew of cabbage, potatoes, beets, and meat. Russian greatest hits like Kievski Kotlet, a.k.a. chicken Kiev, wrapped-and-fried bundles oozing cheese and butter, and a lightened-up Stroganoff with just a smidgen of sour cream, are smashing. So are Turkish mainstays like the not-often-seen karniyarik, eggplant filled with ground beef and tomato sauce, and mezze classics like zucchini fritters, crispy mussels, and sigara borek, cigar-shaped phyllo filled with feta and fried. And the place does rotisserie chicken too. Once the beer and wine license comes (the owner guessed sometime in February), the picture will be even rosier. But then you'll have to fight the hordes to get in.

— Cynthia Hacinli

February 2003

Figure 1: Sample webpage from Online Washingtonian

kept the cuisines which were actually mentioned in the review; for instance, for the sample review to the right '*Turkish'* and '*Russian'* were kept as possible target values, but *'Middle Eastern'* was not kept because it did not occur in the text of the review. This was important because the task is information *extraction*, so there has to be something to extract. If none of the specified cuisines was mentioned in the review then the review was abandoned. We ended up with a dataset containing 752 reviews. One thing we found interesting about this dataset was that one wanted to extract **all** of the different types of cuisine − just finding one was not necessarily sufficient; for instance, in the writers' opinions, there is likely to be a large difference in the quality of the sushi at a restaurant that just serves Japanese food and one which serves both Japanese and Chinese food.

2.1 Speaker, Location, Start and End Time in Email Announcements

We also created four datasets from Dayne Freitag's announcement dataset which we found at /afs/ir/data/linguistic-data/IE/RISE/freitag-sa-tagged. We separated the announcements into datasets based on location, speaker, start time and end time and reparsed them in a similar manner to the restaurant data. In these datasets it was not infrequent for a particular target to occur more than once in the text, but in different form, such as '*Professor Orloff'* and *'Professor Ann Orloff'* or *'4:30 p.m'* and *'4:30 PM'* (although sometimes the different target values actually refer to different things, such as in events with more than one speaker). In these cases it is sufficient for the model to extract **any** of the target values because the target values are, effectively, synonyms. We were also drawn to the announcement dataset because of the variety in the types of targets. Times tend to be very rigidly formatted, so prefix and suffix information is less important, while locations and names are less rigid thus increasing the value of prefix and suffix information. We were also able to produce fairly large datasets from the announcement dataset: start time: 485 instances, end time: 228 instances, speaker: 409 instances, location: 463 instances.

3. Our Hidden Markov Model (HMM)

3.1 Structure

We built a Hidden Markov Model which has a good deal of flexibility in terms of settings. Our model has four types of nodes, taken from the model in (Freitag and McCallum 2000): *target states* which emit only tokens to be extracted; *prefix states*, which emit tokens leading up to the target; *suffix states*, which emit tokens following the target; and *background states* which emit all other tokens. Our structure has some flexibility − it can handle a varying number of prefix and suffix nodes, but they must all be inline. The general structure has one background state, which can transition to either itself or the first prefix state. Each prefix state can transfer only to the next prefix state, or if it is the last prefix state to the target state. There is one target state with a self−loop an d a transition to the first suffix state. The suffix states are similar to the prefix states, in that each one can transfer only to the next one unless it is the last in which case it transitions to the original background state. The emissions from the prefix states together make the string preceding the target and the suffix states emit the string following the target.

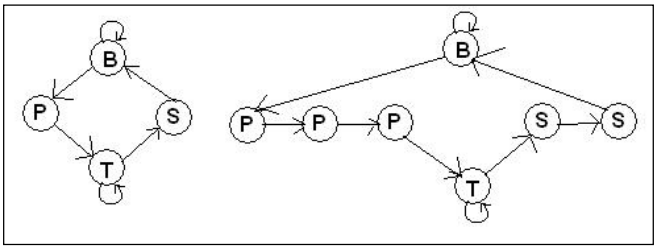


Figure 2: Sample HMM Structures

3.2 Learning the Transition and Emission Probabilities

We used maximum likelihood estimation for assigning the transition and emission probabilities. However, sometimes targets would be too close to one another in the text to make it through all of the prefix and suffix states between the targets (for example, 'the <cuisine>Italian</cusine> and <cuisine>American</cuisine> food') and in those cases we did not allow non−zero probabilities for "short−cuts" in the structure, those transitions just would not be used in estimating the transition probabilities.

3.3 Smoothing

We did not use any smoothing in determining the transition probabilities. The structures are rigid and we did not want to smooth away the zero probabilities inherent to them. Moreover, our data sets where large enough that we felt the produced transition probabilities where likely to be fairly accurate. We did use smoothing for the emission probabilities − this was imperative to because of the high likelihood of encountering new words. We used Lidstone's law for smoothing and our model allows the user to specify the value of the smoothing factor(λ). We also allow different smoothing factors to be used based on the type of node. This makes sense as target nodes should use the least smoothing (smallest λ) because target values tend to be repeated in other instances, and background nodes should use the most smoothing (highest λ) since they are meant to just sort of emit anything. We found that as the smoothing factors increased, precision also increased and recall decreased. This is discussed in more depth later in the report. One set of values which we found performed well was ($\lambda_{background} = 0.01$, $\lambda_{prefix} = 0.0005$, $\lambda_{target} = 0.00005$, $\lambda_{suffix} = 0.0005$).

3.4 Other factors

Capitalization of and in words was another factor which struck us as being important. It seems that sometimes capitalization is important − when deciding if a token was likely to be emitted from the target node, whether or not the token is capitalized is very important. All of the cuisine types were capitalized and speaker names and building names also have a strong tendency to be capitalized. For the target node only, when finding the initial emission probabilities we also found the probability that the token was capitalized. Then, when we want the probability of the target node emitting a particular token, the probability given is actually the probability of that word as discussed previously, multiplied by the probability of capitalized/not capitalized based on whether the token is capitalized. Because every cuisine is capitalized, our HMM never extracted a non−capitalized word as a cuisine target − when we turned this feature off the precision decreased substantially.

It is also important to recognize when two words are equal regardless any capitalization in the word. Therefore, when calculating the emission probability of a token as discussed in the previous section all capitalization in the word is ignored.

3.5 Picking the Best path (Viterbi)

We used a straightforward implementation of the Viterbi algorithm for picking the most likely path through the HMM. We used log−addition instead of regular multiplications to prevent the probabilities from going to zero. Because the HMM is cyclic and contains self−loops it is possible for the chosen path to contain any number of target emissions, from 0 to many. Our HMM will occasionally pick best paths that contain no target emissions. This is rare − out of 752 restaurant reviews the HMM declined to pick a target 3 times (0.40%). It tended to occur when the target value was only mentioned once in the text and in a non-helpful surrounding, such as *"inside the look turns stylish <cuisine>Mexican</cuisine> with cowhide slipper chairs and cinnamon walls."* Picking more than one

target of course makes sense. If a target occurs more than once (such as when a restaurant serves more than one type of food) than the HMM **should** pick more than one target. Frequently, if the exact same target value occurred more than once in the text the HMM would pick out all instances of it − it would then rank its outputs by how frequently they occurred.

3.6 Outputting an answer

The output of the HMM is just all of the distinct targets found in the best path of the previous section. If consecutive tokens are emitted from the target node then the tokens are concatenated into one output token (e.g. *"Middle Eastern"* or *"South American"*).

4. Performance

Overall we were pleased with the performance of our HMM. As we will discuss, the performance numbers did not always do justice to the actual performance.

4.1 Testing

We used the leave−one−out method of cross−validation when testing our system.

4.2 Evaluation Criteria

As was hinted at earlier, we consider precision and recall to be the best means of evaluating the performance of our HMM. However, there are many fine points to consider. For each data set we found the average precision and the average recall. Each iteration of testing in the cross validation trained on all but one instance and then tested the remaining instance. The precision (the proportion of output target values which were correct) and recall (the proportion of correct target values which were output) of that instance i is:

$$\text{precision}_i = {}^{\text{true positives(i)}}/_{\text{true positives(i) + false positives(i)}}$$
$$\text{recall}_i = {}^{\text{true positives(i)}}/_{\text{true positives(i) + false negatives(i)}}$$

What is the precision when nothing is output for an instance? It is ${}^{0}/_{0}$ which is undefined. We just ignored those instances when calculating the average precision. However, in the same situation, the recall is 0, not undefined, so it is still factored in. The averages are then:

$$\text{avg\_prec} = (\Sigma_{i \in \text{dataset}} ({}^{\text{true\_positives(i)}}/_{\text{true\_positives(i) + false\_positives(i)}})(?\text{is\_defined}))/_{|\text{dataset}|-|\text{undefined}|}$$
$$\text{avg\_recall} = (\Sigma_{i \in \text{dataset}} {}^{\text{true\_positives(i)}}/_{\text{true\_positives(i) + false\_negatives(i)}})/_{|\text{dataset}|}$$

It is important to note that these values are different that the overall precision and overall recall:

$$\text{overall\_prec} = ({}^{\Sigma[i \in \text{dataset}]\ \text{true\_positives(i)}}/_{\Sigma[i \in \text{dataset}]\ \text{true\_positives(i) + false\_positives(i)}})$$
$$\text{overall\_recall} = ({}^{\Sigma[i \in \text{dataset}]\ \text{true\_positives(i)}}/_{\Sigma[i \in \text{dataset}]\ \text{true\_positives(i) + false\_negatives(i)}})$$

The average assigns equal value to each instance in the training set, while the latter places more weight on instances where the target can take on more values. We felt that the former was a more valid evaluation criteria because we felt that each instance was equally important.

We also used a different equation to find the recall for a particular instance for the datasets where we considered different target values to be synonyms for the same thing. As previously discussed, if there is more than one possible value for a location of a talk then clearly these values must be synonyms so therefore it is OK to return **any** of them so recall was computed as a binary function of whether any true positives where found. We used this equation for speaker, location, start time and end time.

When computing performance we used the F measure as discussed in the test:

$$F = (\alpha P^{-1} + (1-\alpha)R^{-1})^{-1}$$

We computed performance using two different alphas. We used $\alpha = ½$, which weighs precision and recall equally and is commonly known as F1. We also computed performance with $\alpha = ¼$ which places more value on recall because we felt that in the task of information extraction recall is more important. It is better to have all of the information, along with some garbage, than to not have enough information.

4.3 Evaluation Pitfalls

First, we have some problems with the tagging, an inevitable side effect of the fact that our datasets were not hand−tagged. In some cases, the information we extract seems more correct or is in some way complimentary to the target words in the training set. For example, in one restaurant review, the target word is *"Afghan",* while our system chose *"Middle Eastern"*. In other cases, the tagging is ambiguous. For example, again in the cuisine set, the one restaurant is tagged as *"Modern American"* in the training text, while another is tagged as both *"Modern American"* and *"American*". This is contradictory, given that in real life Modern American is a subset of American, and that the decision whether both or only one of them has to be listed is a matter of principle, not of features of the restaurant and would cause harm our system's performance:

```
cuisine/1044.txt
System: [American] (1)
Actual: [Modern American, American] (2)
precision: 1.0
recall: 0.5

cuisine/1006.txt
System: [American] (1)
Actual: [Modern American] (1)
precision: 0.0
recall: 0.0
```

In other text, our system picked "Afghani" and *"Afgan"*, and *"Afghani"* was considered wrong but *"Afghan"* right, which does not make sense either.

Indeed, in many of the cases the mistake is real. For instance a review of Chinese restaurant mentioned that it is located next to a Middle Eastern, and this mislead the HMM to assume Middle Eastern to be a target, which was plain wrong. In other cases even more random words appear, for example the word *"discussion"* when we are trying to extract a location.

The controversy of what is a good way to evaluate information extraction exibits itself also in the results we get on the announcements data set. It is not always necessary to extract the full information in order for it to be useful. For instance, in the following example we obtain fairly useful information about the speakers - 2 last names:

```
speaker/p120.txt
System: [Rutenbar, Lave, Julia] (3)
Actual: [Julia Evens, Lester Lave, Rob Rutenbar] (3)
precision: 0.0
recall: 0.0
```

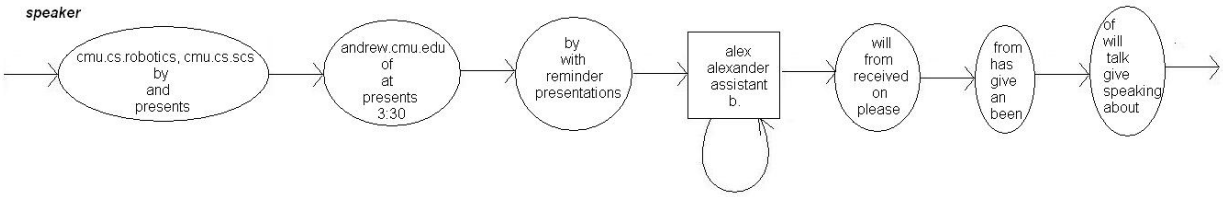
However, this has not been tagged as useful and we do not get any credit for it, with both precision and recall being 0. In a different case, one can obtain all the necessary information, but it is considered useless because it is not explicitly structured in the way that is expected:

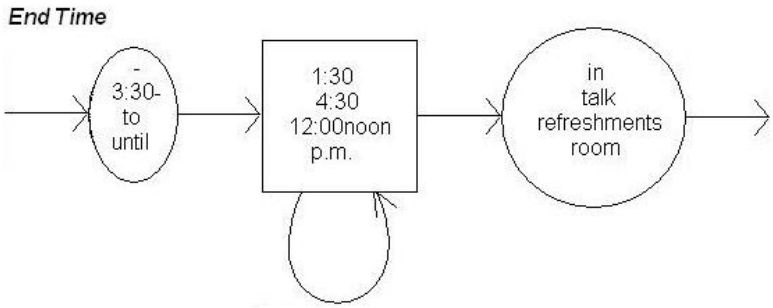
```
speaker/p235.txt
System: [Dr., Jelinek, Dr. Fred] (3)
Actual: [Fred Jelinek, Dr. Fred Jelinek] (2)
```

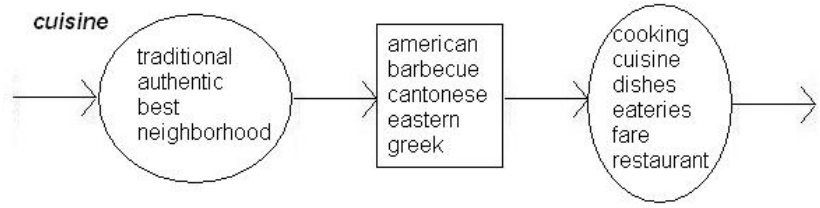
```
precision: 0.0
recall: 0.0
```

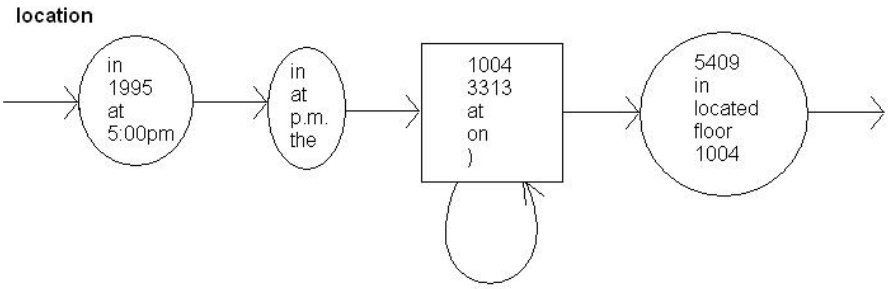
4.4 Resultant Structures

As part of the output we had the HMM print the most common emissions for all nodes that where not background nodes. This was quite interesting because you could really see the pattern matching that was going on. Below are pictures of some of the structures produced by the HMM for the assorted datasets (ovals are prefix/suffix, rectangles are target and background is not shown):









4.5 Actual Performance

We have attached at the end of this report a table containing all runs with different settings for the length of the prefix and suffix and different values for the smoothing factors. For each dataset it has been sorted according to performance (F measure with $\alpha = ¼$ as previously discussed). The sorting is helpful because it allows other patterns about the relative importance of different attributes and settings stand out. Our best performances are summed up in the following table:

dataset	prefix length	suffix length	λ_{backgr}	λ_{prefix}	λ_{target}	λ_{suffix}	average precision	average recall	F1 performance	F (α = ¼) performance
cuisine	2	1	0.002	0.0004	0.00001	0.0003	0.58921	0.917886	0.717708	0.805547
end time	2	2	0.002	0.0004	0.0001	0.0003	0.583228	0.899123	0.707517	0.791894
location	2	1	0.002	0.0004	0.00001	0.0003	0.134042	0.298056	0.184921	0.228238
speaker	1	2	0.002	0.0001	0.001	0.0003	0.199599	0.616137	0.30152	0.404895
start time	2	2	0.002	0.0004	0.00001	0.0003	0.556992	0.969072	0.707395	0.817812

4.5.1 Cuisine

For identifying cuisine the length of the prefix and suffix was very important − the shorter the better. It also appears to do better when λ_{target} is smaller, which make since because this makes it only pick out words for cuisines which it *knows* are types of cuisines, regardless of their context.

4.5.2 End Time

For determining end time the HMM performed better with length 2 for both the prefix and the suffix. It also appeared to do slightly better when the λs where larger.

4.5.3 Location

We never managed to configure a satisfactory HMM fr identifying location, but it did do better with a mid−length prefix and suffix and small λs. Tweaking the attributes caused the performance to almost double between the worst settings and the best setting!

4.5.4 Speaker

For extracting the speaker it was very clear that the HMM needed a suffix of length 2 and a (comparably) very large λ_{target}. Intuitively, this makes sense because the HMM is not likely to have seen all names and must rely much more on context. In fact, here tweaking parameters caused a 10−fold increase in performance from 0.04 to 0.4!

4.5.5 Start Time

For start time, the λs tended to be the more influential factor and for six of the top seven performing HMMs they were $\lambda_{backgr} = 0.002$, $\lambda_{prefix} = 0.0004$, $\lambda_{target} = 0.00001$, $\lambda_{suffix} = 0.0003$. The next most important factor was that the prefix and suffix both be of length 2.

5. Conclusion

We built an HMM for extracting data and it worked to varying degrees depending on the type of information being extracted and the internal settings.

6. References

C. Manning and H. Schutze, Foundations of Statistical Natural Language Processing. The MIT Press, England (1999).

D. Freitag and A. McCallum, "Information extraction with HMM structures learned by stochastic optimization," Proceedings of AAAI-2000.

D. Freitag and A. McCallum, "Information extraction using HMMs and shrinkage," Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction.