Hidden Markov Based Mathematical Model dedicated to Extract Ingredients from Recipe Text

Zied Baklouti^{*†}

August 2019

Abstract

Natural Language Processing (NLP) is a branch of machine learning that gives the machines the ability to decode human languages. Part-of-speech tagging (POS tagging) is a preprocessing task that requires an annotated corpora. Rule-based and stochastic methods showed great results for POS tag prediction. On this work, I performed a mathematical model based on Hidden Markov structures and I obtained a high level accuracy of ingredients extracted from text recipe which is a performance greater than what traditional methods could make without unknown words consideration.

1 Introduction

My primary intention was to analyze the nutrition value of Tunisian recipes after crawling these information from food websites. Many previous work were interested in analyzing cuisine recipes, for example Sina Sajadmanesh ([Sajadmanesh et al., 2017](#)) presented an analysis of the ingredients diversity around the word using an ingredient-based classifier to differentiate between recipes around the word based on its geographical identity. Sina Sajadmanesh ([Sajadmanesh et al., 2017](#)) studied the diversity of ingredients in dishes with introduction of global diversity (the ability to have diversified ingredients between recipes) and local diversity (the ability to have diversified ingredients within a recipe). Other related work for culinary habits is Yong-Yeol Ahn paper ([Ahn, Ahnert, Bagrow, & Barabási, 2011](#)) who introduced the notion of Flavor Network and tried to verify the Food Paring hypothesis introduced on the 90's by Heston Blumenthal and Francois Benzi. Flavor network as described by Ahn is a graph where the nodes are the ingredients extracted from recipes and weights are shared flavors between nodes. Food paring hypothesis is an indicator calculated after forming

^{*}Corresponding author: ziedbaklouti.ing.math@gmail.com

[†]Engineer in Applied Mathematics ,Tunis University

the Flavor Network in order to show if in a country or in a geographical part of the word we have tasty recipes or the ingredients do not have similar molecules. Tiago Simas (Simas, Ficek, Diaz-Guilera, Obrador, & Rodriguez, 2017) introduced the notion of food bridging formed with semi-metric distances.

20 All previously cited researches on cuisine recipes need information extraction from text recipe in order to use it on graphical visualization and statistical analysis. Information extraction can be used manually by extracting ingredients indicated on recipes or automatically. The problem in automatic extraction is that information should be precise in order to have also precise analysis, for
 25 example some ingredients take only one word and others can take two and three words. Another problem on automatic information extraction is that some ingredients that take one word have in common some words with other ingredients that have more than one word which make automatic information extraction more difficult. I tried to develop a mathematical model dedicated to extract
 30 ingredients from text recipe written in Arabic language with highest precision than traditional methods. According to Cutting (Cutting, Kupiec, Pedersen, & Sibun, 1992), a Tagger must be robust that should deal with unknown words, efficient that can deal with large corpora, accurate that can tag with high accuracy, tunable that can deal with different corpora and reusable that take small
 35 efforts to re-target a new corpus. There are three types of POS Tagger: Taggers based on stochastic models, Taggers based on rules and Taggers Based on neural networks. On this work we will use Taggers based on HMM models. The use of POS tags as external features to solve NER problems was experimented by Zhou (Zhou & Su, 2002) but it was discarded because it showed bad results but
 40 our methodology and experiments demonstrate that using POS tags as external features is not a bad idea. This could be explained by the different definitions between our tokens and Zhou’s tokens: tokens as defined by Zhou is a pair of word-feature and in our model token is only a word on the corpus.

2 Training Dataset

45 Hidden Markov Models are used in previous work to resolve named entity recognition (NER) problems. In our case we have a NER problem with one Named Entity to extract (Ingredients) and with various boundaries (one word, two words, three words). Our sentences belong to only one category of phrases on Arabic language : The noun phrases ”الجملة الاسمية”. It is a kind of phrases
 50 which don’t begin with verbs (phrase which don’t describe an action). Noun phrases are constituted with 2 parts: The beginning of a sentence ”المبتدا” contains the object of the information , and the end of a sentence ”الخبر” containing the core information. Table1 shows how the dataset is trained: When an ingredient is found on a corpus, we attribute 1 to the concerned token. If the
 55 information we want to extract takes more than one word we attribute 2 to the extra word to complete the information. 0, 1, 2, 3 is the new list of tags used on the HMM tagger in the second phase of our ingredient extractor system. Our word dictionary is constituted with 809 words. The number of trained sentences

Translation in english	Tokens in arabic	Label	detected ingredients
a little of	رشة	C	0
salt	ملح	D	1
and	و	J	0
black (اسود)	فلفل	E	1
pepper (فلفل)	اسود	F	2
.	.	.	0

Table 1: Example of trained sentence on our trained corpus

is 1973. The number of POS tags used in the first phase of our ingredient extractor is 14 POS tags. The sentences are describing ingredients constituting a cuisine recipe. Table1 shows an example of a trained sentence. These sentences are extracted from cuisine websites using scrapy framework.

3 Methodology

Our Ingredient extractor is constituted with two parts. In the first part we are going to predict the part of speech tags of our sequences extracted as explained on the previous section. On the second part we are going to use predicted Part of Speech as a grammatical features in order to predict ingredients state.

3.1 Grammatical Features Collection

Figure 1: Methodology Diagram of our Ingredient Extractor

In our Model we have observation sequences which are sentences containing information we want to extract (Ingredients). The first part of our Ingredient Extractor will try to collect grammatical features by resolving the POS tagging

problem. At the end of this part we want to obtain the grammatical role for each token on each sequence. We experimented both first order HMM and second order HMM to predict ingredients state. We obtained strangely better results with first order HMM. This could be explained by the fact that in Arabic language a POS tag is organized in bi-gram manner. For example if we have an E tag **منعوت** we are certain to have an F tag **نعت** because in Arabic a word tagged as an E tag **منعوت** could be tagged as a noun but if the intention of the writer is to describe that noun with an adjective coming after it the exact tag to be used is an E tag **منعوت**.

POS Tag (arabic)	POS Tag (english)	Label
إسم	Noun	A
رقم	Number	B
إسم معرف	Known Noun	C
إضافة	Noun after Known Noun	D
منعوت	Noun before the Adjective	E
نعت	Adjective	F
حرف جر	Preposition	G
إسم مجرور	Noun after preposition	H
وحدة قياس	unit of measure	I
واو العطف	AND	J
فعل مبني للمجهول	Passive Verb	K
المفعول المطلق	Superlative	L
أداة عطف غير واو العطف	OR	M
.	.	.

Table 2: POS Tag-set used to annotate our corpus

The elements needed to define a first order HMM in order to perform a POS tag task are:

- N=14, the number of POS tags each token can take as shown in Table2 we have 14 POS tag. We represent the sequence of POS tags by the sequence $T = \{t_1, t_2, \dots, t_{14}\}$.
- M=809, is the sequence of symbols generated by the stochastic procedure of the HMM model. In our case it is the list of tokens (words) constituting our lexicon.
- Matrix A containing the state transition probability distribution. In our case it is the probability to move from one POS tag to another POS tag.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

$$a_{ij} = P(T_r = t_j / T_{r-1} = t_i)$$

- Matrix B containing the observation symbol probability distribution. In our case it is the probability to emit a word v_k given that the system is emitting one of the state of the sequence $T = \{t_1, t_2, \dots, t_{14}\}$.

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix}$$

$$b_{jk} = P(V_r = v_k / T_r = t_j)$$

- π a vector containing the probability distributions that a sentence begins with one of the elements of T

$$\pi = [\pi_1 \quad \pi_2 \quad \pi_3 \quad \dots \quad \pi_{14}]$$

3.2 Predicting Sequences of Ingredients State

90 3.2.1 Prediction Using First Order HMM

Suppose that we want to use first order HMM to predict ingredients state on the second part of our Ingredient Extractor without using grammatical features estimated on previous section, in this case the elements needed to define this HMM according to Rabiner([Rabiner, 1989](#)):

- 95 • N=4, the number of distinct state each token can take. In our case it is one element of the set $\Gamma = 0, 1, 2, 3$
- M=809, is the sequence of symbols generated by the stochastic procedure of the HMM model. In our case it is the list of tokens (words) constituting our lexicon.
- Matrix A containing the state transition probability distribution. In our case it is the probability to move from one of the state on the set Γ to another.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

$$a_{ij} = P(\Gamma_r = \gamma_j / \Gamma_{r-1} = \gamma_i)$$

- Matrix B containing the observation symbol probability distribution. In our case it is the probability to emit a word v_k given that the system is emitting one of the state of the sequence Γ .

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix}$$

$$b_{jv_k} = P(V_r = v_k / \Gamma_r = \gamma_j)$$

- π a vector containing the probability distributions that a sentence begins with one of the elements of Γ

$$\pi = [\pi_1 \quad \pi_2 \quad \pi_3 \quad \pi_4]$$

100 This model described above is a first-order hidden Markov model, we can call it also a bi-gram model to predict ingredients state.

3.2.2 Prediction Using Full Second Order HMM

Scott M.Thede([Thede & Harper, 1999](#)) used a full second-order hidden Markov model for his POS tagger. To constitute this trigram tagger from the previous
105 bigram tagger Scott M.Thede([Thede & Harper, 1999](#)) replaced bi-gram transition probability a_{ij} with a trigram transition probability a_{ijk} and replaced bi-gram lexical probability b_{jv_k} with a trigram lexical probability b_{ijv_k} . If we want to use a full second-order hidden Markov model as described by Scott M.Thede([Thede & Harper, 1999](#)) the element needed to define this HMM are :

- 110 • N=4, the number of distinct states each token can take.
- M=809, number of tokens.
- Matrix A shown on [figure2](#) containing the state transition probability distribution. For a trigram model this matrix is of dimensions $N \times N \times N$.
- 115 • Matrix B shown on [figure3](#) containing the observation symbol probability distribution. For trigram model this matrix is of dimensions $N \times N \times M$
- π a vector containing the probability distributions that a sentence begins with one of the elements of Γ

$$\pi = [\pi_1 \quad \pi_2 \quad \pi_3 \quad \pi_4]$$

				$a_{1,1,4}$	$a_{1,2,4}$	$a_{1,3,4}$	$a_{1,4,4}$
				$a_{2,1,4}$	$a_{2,2,4}$	$a_{2,3,4}$	$a_{2,4,4}$
		$a_{1,1,3}$	$a_{1,2,3}$	$a_{1,3,3}$	$a_{1,4,3}$	$a_{3,3,4}$	$a_{3,4,4}$
		$a_{2,1,3}$	$a_{2,2,3}$	$a_{2,3,3}$	$a_{2,4,3}$	$a_{4,3,4}$	$a_{4,4,4}$
	$a_{1,1,2}$	$a_{1,2,2}$	$a_{1,3,2}$	$a_{1,4,2}$	$a_{3,3,3}$	$a_{3,4,3}$	
	$a_{2,1,2}$	$a_{2,2,2}$	$a_{2,3,2}$	$a_{2,4,2}$	$a_{4,3,3}$	$a_{4,4,3}$	
$a_{1,1,1}$	$a_{1,2,1}$	$a_{1,3,1}$	$a_{1,4,1}$	$a_{3,3,2}$	$a_{3,4,2}$		
$a_{2,1,1}$	$a_{2,2,1}$	$a_{2,3,1}$	$a_{2,4,1}$	$a_{4,3,2}$	$a_{4,4,2}$		
$a_{3,1,1}$	$a_{3,2,1}$	$a_{3,3,1}$	$a_{3,4,1}$				
$a_{4,1,1}$	$a_{4,2,1}$	$a_{4,3,1}$	$a_{4,4,1}$				

$$a_{ijk} = P(\Gamma_r = \gamma_k / \Gamma_{r-1} = \gamma_j, \Gamma_{r-2} = \gamma_i)$$

Figure 2: Transition matrix used in full second order HMM

				$b_{1,1,m}$	$b_{1,2,m}$	$b_{1,3,m}$	$b_{1,4,m}$
				$b_{2,1,m}$	$b_{2,2,m}$	$b_{2,3,m}$	$b_{2,4,m}$
		$b_{1,3,2}$	$b_{1,4,2}$	$b_{3,2,m}$	$b_{3,3,m}$	$b_{3,4,m}$	
		$b_{2,3,2}$	$b_{2,4,2}$	$b_{4,2,m}$	$b_{4,3,m}$	$b_{4,4,m}$	
		$b_{3,3,2}$	$b_{3,4,2}$				
		$b_{4,3,2}$	$b_{4,4,2}$				
$b_{1,1,1}$	$b_{1,2,1}$	$b_{1,3,1}$	$b_{1,4,1}$				
$b_{2,1,1}$	$b_{2,2,1}$	$b_{2,3,1}$	$b_{2,4,1}$				
$b_{3,1,1}$	$b_{3,2,1}$	$b_{3,3,1}$	$b_{3,4,1}$				
$b_{4,1,1}$	$b_{4,2,1}$	$b_{4,3,1}$	$b_{4,4,1}$				

$$b_{ij}(v_k) = P(V_r = v_k / \Gamma_r = \gamma_j, \Gamma_{r-1} = \gamma_i)$$

Figure 3: Lexical matrix used in full second order HMM

3.2.3 Prediction with the introduction of collected Grammatical Features

At this point of the research, grammatical features are still not introduced on the model. When we observed the elements constituting the full second-order hidden

Markov model contextual probabilities and lexical probabilities are formed to include second order informations. What we tried to do on this stage is to keep the trigram property of the model but modifying the parameters of the contextual and lexical probabilities to introduce grammatical features. The changes made are: the trigram transition probability $a_{ijk} = P(\Gamma_r = \gamma_k / \Gamma_{r-1} = \gamma_j, \Gamma_{r-2} = \gamma_i)$ is replaced by the probability $a_{ijk} = P(\Gamma_r = \gamma_k / \Gamma_{r-1} = \gamma_j, T_{r-1} = \tau_i)$ and the trigram lexical probability $b_{ij}(v_k) = P(V_r = v_k / \Gamma_r = \gamma_j, \Gamma_{r-1} = \gamma_i)$ is replaced by the probability $b_{ij}(v_k) = P(V_r = v_k / \Gamma_r = \gamma_j, T_r = \tau_i)$. The elements of the HMM becomes:

- N=4, the number of distinct element of the set Γ .
- M=809, number of tokens.
- K=14, the number of distinct POS tags.
- Matrix A containing the state transition probability distribution. This matrix is of dimensions $N \times N \times K$ as shown on [figure4](#).
- Matrix B containing the observation symbol probability distribution. This matrix is of dimensions $M \times N \times K$ as shown on [figure5](#).
- π a vector containing the probability distributions that a sentence begins with one of the elements of Γ

$$\pi = [\pi_1 \quad \pi_2 \quad \pi_3 \quad \pi_4]$$

4 Calculating Probabilities for Unknown Words

One common problem in part of speech tagging processing is to predict the hidden state of an unknown word. Andrei Mikheev (Mikheev, 1996) constructed an automatic technique for learning English part-of-speech guessing rules. Scott M. Thede (Thede & Harper, 1999) used a Second-Order Hidden Markov Model for POS tagging by estimating the probability $P(\text{word has suffix } s_k / \text{tag is } t_j)$ when he tried to predict the hidden state of an unknown word. Scott M. Thede (Thede & Harper, 1999) used a trigram tagger respecting the elements that constitute an HMM explained in details by Lawrence R. Rabiner (Rabiner, 1989). On this work we tried to create a first-order Hidden Markov Model POS Arabic Tagger and we choose the prefix of the word as a word feature rather than word suffix as described by Thede. The Viterbi algorithm becomes Figure??.

After training the corpus we can estimate the probability $P(v_i / t_i)$ by counting the number of time the hidden state t_i was attributed to the word v_i . For example we had a corpus constituted of 4 words as represented on the Table 3.

We trained the corpus with 3 hidden states and we obtain the discrete state-dependent word probabilities table :

	2	البرتقال	الاكلیل	أو
B	1	0	0	0
C	0	0.5	0.5	0
M	0	0	0	1

Table 3: The discrete state-dependent word probabilities table for a mini-corpus

For example the probability $P(w_i = \text{البرتقال} / t_i = C)$ is estimated by counting the number of times the word البرتقال appears with the tag C divided by the number of times C appears on the corpus. In Arabic as in English, words are constituted morphologically based on its grammatical role on a sentence. This is why suffix and prefix of a word can help us to predict it's tag. Andrei Mikheev (Mikheev, 1996) presented a list of morphological rules for English to predict unknown words. For example the word we obtain by adding the prefix "un" to a word that was tagged with the tags (VBD) and (VBN) can play the role of an adjective on a sentence. Such morphological rule can be also applied to Arabic Language. For example adding "ال" to a word that played the role of a noun "إسم" or a noun that comes before the adjective "منعوت" can be estimated as a Known Noun "إسم معرف". Scott M.Thede (Thede,1998) was inspired by morphological rules described by Mikheev to construct his unknown word predictor. He first created a lexicon to estimate $P(w_i / t_i)$, then he used prefix and suffix of words on this lexicon to predict possible tags. This method can be used to our corpus as in Arabic word feature extraction can be done by the prefix of the words. For example the discrete state-dependent word probabilities table for the previous mini-corpus can be represented depending on word prefixes:

	2	ال	ال	أو
B	1	0	0	0
C	0	0.5	0.5	0
M	0	0	0	1

Table 4: The discrete state-dependent word probabilities table for a mini-corpus (prefix-based approach)

We considered the two first characters of a word on this work if a word counts more than two characters and the first character of a word if the word is constituted with only one character. After that a probability distribution for each affix is created by adding the probabilities $P(v_i / t_i)$ for words with the same tag. For example $P(prefix = "ال" / t_i = C) = P(v_i = \text{البرتقال} / t_i = C) + P(v_i = \text{الاكلیل} / t_i = C)$ the Table4 becomes:

	2	ا	و
B	1	0	0
C	0	1	0
M	0	0	1

Table 5: The discrete state-dependent word probabilities table for a mini-corpus after adding the probabilities $P(v_i/t_i)$ for words with the same tag (prefix-based approach)

[Table5](#) constitute the matrix C used on the Hidden Markov Model algorithm to replace the matrix B when a word is unknown.

	Accuracy	F1 score
First order Hidden Markov Model applied to tokens to predict ingredients	96.61 %	70.06 %
Second order Hidden Markov Model applied to tokens to predict ingredients	95.82 %	68.18 %
First Order Hidden Markov Model applied to tags to predict ingredients	79.31 %	54.33 %
Second Order Hidden Markov Model applied to tags to predict ingredients	87.73 %	58.53 %
Our ingredient extractor with 100 % accuracy on the first layer with $\lambda = 4$	98.43 %	81.43 %
Our ingredient extractor with 90.26 % accuracy on the first layer with 82.01 % F1 score on the first layer with $\lambda = 4$	97.02 %	73.8 %

Table 6: Comparison of performances between our ingredient extractor and other HMM taggers using the trained corpus as a testing dataset

Figure 6: Variations of accuracy of our Ingredient Extractor depending on λ and accuracy score on the first layer

Figure 7: Variations of F1 score of our Ingredient Extractor depending on λ and accuracy score on the first layer

Fold	accuracy	f1 score	accuracy for unknown words	accuracy for known words	number of unknown words	unknown words percentage
0	93.59 %	66.37 %	65.35 %	95.63 %	201	24.84 %
1	95.26 %	68.67 %	71.81 %	97.06 %	208	25.71 %
2	95.08 %	68.39 %	68.13 %	97.18 %	209	25.83 %
3	94.77 %	67.92 %	68.68 %	96.53 %	165	20.39 %
4	94.16 %	67.06 %	67.68 %	95.9 %	137	16.93 %
5	94.08 %	67.15 %	65.35 %	96.12 %	223	27.56 %
6	95.53 %	69.11 %	72.32 %	97.3 %	203	25.09 %
7	94.97 %	68.36 %	73.05 %	96.83 %	242	29.91 %
8	94.99 %	68.41 %	69.92 %	96.82 %	213	26.32 %
9	93.53 %	66.62 %	66.86 %	96.39 %	334	41.28 %
Avg	94.59 %	67.8 %	68.91 %	96.57 %	213	26.32 %

Table 7: Performances of first order HMM used to predict ingredients state using 10 fold cross-validation

Fold	accuracy	f1 score	accuracy for unknown words	accuracy for known words	number of unknown words	unknown words percentage
0	93.14 %	63.83 %	66.92 %	94.64 %	201	24.84 %
1	93.37 %	64.77 %	58.4 %	95.53 %	208	25.71 %
2	93.71 %	65.75 %	60.07 %	95.8 %	209	25.83 %
3	93.32 %	64.28 %	65.15 %	94.84 %	165	20.39 %
4	86.51 %	61.17 %	65.24 %	87.62 %	137	16.93 %
5	93.11 %	63.97 %	62.14 %	94.87 %	223	27.56 %
6	87.93 %	63.26 %	66.05 %	89.27 %	203	25.09 %
7	93.72 %	65.32 %	63.17 %	95.8 %	242	29.91 %
8	93.53 %	65.25 %	60.9 %	95.45 %	213	26.32 %
9	92.19 %	62.95 %	64.14 %	94.57 %	334	41.28 %
Avg	92.07 %	64.05 %	63.22 %	93.83 %	213	26.32 %

Table 8: Performances of second order HMM used to predict ingredients state using 10 fold cross-validation

Fold	accuracy	f1 score	accuracy for unknown words	accuracy for known words	number of unknown words	unknown words percentage
0	91.75 %	63.18 %	54.72 %	93.87 %	201	24.84 %
1	93.12 %	64.9 %	55 %	95.47 %	208	25.71 %
2	92.98 %	64.79 %	50.91 %	95.6 %	209	25.83 %
3	93.12 %	64.9 %	55 %	95.47 %	165	20.39 %
4	95.1 %	68.38 %	63.41 %	96.77 %	137	16.93 %
5	93.84 %	64.98 %	61.07 %	95.69 %	223	27.56 %
6	94.23 %	66.76 %	62.36 %	96.17 %	203	25.09 %
7	93.24 %	65.15 %	53.89 %	95.92 %	242	29.91 %
8	94.49 %	66.75 %	59.39 %	96.56 %	213	26.32 %
9	91.58 %	63 %	69.08 %	93.48 %	334	41.28 %
Avg	93.34 %	65.27 %	58.48 %	95.5 %	213	26.32 %

Table 9: Performances of our ingredient extractor with 100 % accuracy on the first layer and $\lambda = 4$ using 10 fold cross-validation

Fold	First Layer first order HMM				Second Layer second order HMM			
	accuracy	f1 score	accuracy unknown words	accuracy known words	accuracy	f1 score	accuracy unknown words	accuracy known words
0	84.29 %	71.76 %	51.96 %	86.63 %	91.19 %	63.58 %	51.57 %	93.47 %
1	84.81 %	69.51 %	57.04 %	86.95 %	92.54 %	63.79 %	55.9 %	94.8 %
2	85.28 %	70.71 %	58.6 %	87.35 %	92.98 %	64.07 %	55.67 %	95.3 %
3	87.06 %	75.05 %	58.1 %	89.6 %	92.29 %	66.32 %	54.94 %	94.9 %
4	87.5 %	76.9 %	59.75 %	89.33 %	94.53 %	69.77 %	63.41 %	96.16 %
5	86.46 %	73.65 %	58.57 %	88.44 %	92.71 %	64.82 %	57.5 %	94.7 %
6	86.59 %	73.37 %	59.04 %	88.69 %	93.13 %	66.63 %	63.09 %	94.96 %
7	84.63 %	69.66 %	56.88 %	86.99 %	92.78 %	63.81 %	58.08 %	95.15 %
8	85.3 %	71.52 %	54.88 %	87.53 %	93.68 %	65.51 %	60.9 %	95.61 %
9	83.54 %	71.47 %	57.33 %	86.33 %	90.76 %	62.53 %	63.99 %	93.02 %
Avg	85.54 %	72.36 %	51.24 %	87.78 %	92.65 %	65.08 %	58.5 %	94.8 %

Table 10: Performances of our ingredient extractor with various results accuracy in the first layer and $\lambda = 4$ using 10 fold cross-validation

	known	unknown	overall	F1 score
First order Hidden Markov Model applied to tokens	96.57 %	68.91 %	94.59 %	67.8 %
Second order Hidden Markov Model applied to tokens	93.83 %	63.22 %	92.07 %	64.05 %
Our ingredient extractor with 100 % accuracy on the first layer with $\lambda = 4$	95.5 %	58.48 %	93.34 %	65.27 %
Our ingredient extractor with 85.54 % accuracy on the first layer with 72.36 % F1 score on the first layer with $\lambda = 4$	94.8 %	58.5 %	92.65 %	65.08 %

Table 11: Comparison of performances using test dataset independent from training dataset and using 10-fold cross validation with 213 as an average number of unknown words

5 Experimental Results

First we tested the Modified Viterbi algorithm used in our Ingredient Extractor on the same trained corpus to avoid using matrix C described in Table 4. Our Ingredient Extractor shows the highest performance and F1 score even when accuracy on the first layer don't overcome 91 % . When we vary the value of λ depending of different values of accuracies obtained on the first layer , it is clear that the accuracy and F1 score obtained for the second layer reach a maximum value at $\lambda = 4$. This could be explained by the fact that the lexical probability used in our ingredient extractor $P(V_r = v_k / \Gamma_r = \gamma_j, T_r = \tau_i)$ can contribute more precisely to the estimation of ingredient state than contextual probabilities on the same model.

The utility of using grammatical features represented on the model by the lexical probability $P(V_r = v_k / \Gamma_r = \gamma_j, T_r = \tau_i)$ is essential especially when we want to extract entities with various boundaries that could have in common one or multiple tokens. For example the extracted entity **فلفل زينة** or **فلفل** have different boundary that causes an ambiguity problem because the program could stop extracting only a single word especially when he was trained frequently extracting a single word of that entity but with the use of grammatical features our model can overcome that problem and fix the great boundary for the extracted entity.

Second we tested the Modified Viterbi algorithm used in our Ingredient Extractor on a testing dataset different from the trained dataset and we compare the results with the results obtained for a first order HMM and a second order HMM using 10 fold cross-validation. Table11 contain a comparison of different

HMM methods results with our Ingredient Extractor. Our method show 58.48
210 % accuracy for unknown words with 100 % accuracy for predicted tags on the
first layer and 58.5 % accuracy for unknown words for a 85.54 % accuracy on
the first layer. Best performance in unknown word prediction is realised by
first order HMM model with 68.91 % accuracy. It is clear that our Ingredient
extractor system is not more performant than first and second order HMM for
215 unknown words.

6 Conclusion

Our Ingredient Extractor algorithm showed great results for ingredients extrac-
tion. It is based on HMM methods. We realized it by training two layers:
first we trained tokens by tagging POS tags and second we trained tokens by
220 extracting the ingredients. Our HMM model needed modifications in iteration
step because we didn't get a square transition probability matrix or a square
lexical probability matrix after the training step. A detailed iterations of our
method is illustrated in appendix A.

A Detailed Iterations of Modified Second Order Viterbi Algorithm Used in our Ingredient Extractor

225

1. Predicting Tags for the first Layer:

We use for predicting tags a first order hidden Markov model as this method have a better accuracy than second order hidden Markov model.

230

2. Predicting Ingredients state for the second Layer:

The variables:

- $\delta_l(i, j) = \max_{\tau_1 \dots \tau_{l-1}} P(\tau_1 \dots \tau_l = [t_i, \gamma_j] / v_1 \dots v_l, t_1 \dots t_l), l = 1 \dots L$
- $\psi_l(i, j) = \argmax_{\tau_1 \dots \tau_{l-1}} P(\tau_1 \dots \tau_l = [t_i, \gamma_j] / v_1 \dots v_l, t_1 \dots t_l), l = 1 \dots L$

The procedure:

(1) Initialization step:

$$\delta_1(i, j) = \begin{cases} \pi_j b_{ij}(v_1) & v_1 \text{ is known } i=1..14 \text{ } j=1..4 \\ \pi_j c_{ij}(v_1) & v_1 \text{ is unknown } i=1..14 \text{ } j=1..4 \end{cases}$$

$$\psi_1(i, j) = 0 \text{ } i=1..14 \text{ } j=1..4$$

(2) Iteration step:

$$\delta_l(i, j) = \begin{cases} \max_j [\delta_{l-1}(i, j) a_{ijk}] b_{ij}(v_l) & v_l \text{ is known , } l=2..L \\ & \text{length } j=1..4, i=1..14 \\ & k=1..4 \\ \max_j [\delta_{l-1}(i, j) a_{ijk}] c_{ij}(v_l) & v_l \text{ is unknown , } l=2..L \\ & j=1..4, i=1..14 \\ & k=1..4 \end{cases}$$

$$\psi_l(i, j) = \argmax_j [\delta_{l-1}(i, j) a_{ijk}] j=1..4, i=1..14, k=1..4$$

(3) Termination:

$$\tau_L^* = \argmax_{i=y(L), j=1..4} \delta_L(i, j)$$

where $y(L)$ is the tag calculated at previous layer in position L

(4) Backtracking:

$$\tau_l^* = \psi_l(y(l), \tau_{l+1}^*) \quad l=L-1..2,1$$

where $y(l)$ is the tag calculated at previous layer in position l

235 3. Predicting Ingredients state for the second Layer in Log-space and introduction of the hyper-parameter λ

The variables:

- $\delta_l(i, j) = \max_{\tau_1 \dots \tau_{l-1}} P(\tau_1 \dots \tau_l = [t_i, \gamma_j] / v_1 \dots v_l, t_1 \dots t_l), l = 1 \dots L$
- $\psi_l(i, j) = \operatorname{argmax}_{\tau_1 \dots \tau_{l-1}} P(\tau_1 \dots \tau_l = [t_i, \gamma_j] / v_1 \dots v_l, t_1 \dots t_l), l = 1 \dots L$

240 **The procedure:**

(1) Initialization step:

$$\delta_1(i, j) = \begin{cases} \log(\pi_j) + \log(b_{ij}(v_1)) & v_1 \text{ is known } i=1..14 \text{ } j=1..4 \\ \log(\pi_j) + \log(c_{ij}(v_1)) & v_1 \text{ is unknown } i=1..14 \text{ } j=1..4 \end{cases}$$

$$\psi_1(i, j) = 0 \quad i=1..14 \text{ } j=1..4$$

(2) Iteration step:

$$\delta_l(i, j) = \begin{cases} \max_j [\delta_{l-1}(i, j) + \log(a_{ijk})] + \lambda_{\max} \log(b_{ij}(v_l)) & v_l \text{ is known } l=2..L \\ & j=1..4, i=1..14 \\ & k=1..4 \\ \max_j [\delta_{l-1}(i, j) + \log(a_{ijk})] + \lambda_{\max} \log(c_{ij}(v_l)) & v_l \text{ is unknown} \\ & l=2..L \text{ } j=1..4 \\ & i=1..14 \text{ } k=1..4 \end{cases}$$

$$\psi_l(i, j) = \operatorname{argmax}_j [\delta_{l-1}(i, j) + \log(a_{ijk})] \quad j=1..4, i=1..14, k=1..4$$

(3) Termination:

$$\tau_L^* = \operatorname{argmax}_{i=y(L), j=1..4} \delta_L(i, j)$$

where $y(L)$ is the tag calculated at previous layer in position L

(4) Backtracking:

$$\tau_l^* = \psi_l(y(l), \tau_{l+1}^*) \quad l=L-1..2,1$$

where $y(l)$ is the tag calculated at previous layer in position l

1. Initialization step:

$$\left[\begin{array}{cccc} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{array} \right] \quad [pi_1 \quad pi_2 \quad pi_3 \quad pi_4] \quad \left[\begin{array}{cccc} b_{1,1,x[1]} & b_{1,2,x[1]} & b_{1,3,x[1]} & b_{1,4,x[1]} \\ b_{2,1,x[1]} & b_{2,2,x[1]} & b_{2,3,x[1]} & b_{2,4,x[1]} \\ b_{3,1,x[1]} & b_{3,2,x[1]} & b_{3,3,x[1]} & b_{3,4,x[1]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[1]} & b_{14,2,x[1]} & b_{14,3,x[1]} & b_{14,4,x[1]} \end{array} \right]$$

$$d_{1,1,1} = pi_1 \times b_{1,1,x}[1]$$

$$\begin{bmatrix} d_{1,1,1} & \textcolor{red}{d_{1,1,2}} & d_{1,1,3} & d_{1,1,4} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{14,1,1} & d_{14,1,2} & d_{14,1,3} & d_{14,1,4} \end{bmatrix} \quad [pi_1 \quad \textcolor{red}{pi_2} \quad pi_3 \quad pi_4] \quad \begin{bmatrix} b_{1,1,x[1]} & \textcolor{red}{b_{1,2,x[1]}} & b_{1,3,x[1]} & b_{1,4,x[1]} \\ b_{2,1,x[1]} & b_{2,2,x[1]} & b_{2,3,x[1]} & b_{2,4,x[1]} \\ b_{3,1,x[1]} & b_{3,2,x[1]} & b_{3,3,x[1]} & b_{3,4,x[1]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[1]} & b_{14,2,x[1]} & b_{14,3,x[1]} & b_{14,4,x[1]} \end{bmatrix}$$

$$d_{1,1,2} = p_{i_2} \times b_{1,2,x[1]}$$

$$\left[\begin{array}{cccc} d_{1,1,1} & d_{1,1,2} & \textcolor{red}{d}_{1,1,3} & d_{1,1,4} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{array} \right] \quad [pi_1 \quad pi_2 \quad \textcolor{red}{pi}_3 \quad pi_4] \quad \left[\begin{array}{cccc} b_{1,1,x[1]} & b_{1,2,x[1]} & \textcolor{red}{b}_{1,3,x[1]} & b_{1,4,x[1]} \\ b_{2,1,x[1]} & b_{2,2,x[1]} & b_{2,3,x[1]} & b_{2,4,x[1]} \\ b_{3,1,x[1]} & b_{3,2,x[1]} & b_{3,3,x[1]} & b_{3,4,x[1]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[1]} & b_{14,2,x[1]} & b_{14,3,x[1]} & b_{14,4,x[1]} \end{array} \right]$$

$$d_{1,1,3} = pi_3 \times b_{1,3,x[1]}$$

$$\begin{bmatrix} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{bmatrix} \begin{matrix} \leftarrow \\ \\ \\ \\ \leftarrow \end{matrix} \begin{matrix} [p_{i_1} & p_{i_2} & p_{i_3} & p_{i_4}] \\ \\ \\ \\ \end{matrix} \begin{bmatrix} b_{1,1,x[1]} & b_{1,2,x[1]} & b_{1,3,x[1]} & b_{1,4,x[1]} \\ b_{2,1,x[1]} & b_{2,2,x[1]} & b_{2,3,x[1]} & b_{2,4,x[1]} \\ b_{3,1,x[1]} & b_{3,2,x[1]} & b_{3,3,x[1]} & b_{3,4,x[1]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[1]} & b_{14,2,x[1]} & b_{14,3,x[1]} & b_{14,4,x[1]} \end{bmatrix}$$

$$d_{1,1,4} = pi_4 \times b_{1,4,x[1]}$$

$$\left[\begin{array}{cccc} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ \textcolor{red}{d}_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{array} \right] \quad \begin{array}{c} \textcolor{red}{pi}_1 \quad pi_2 \quad pi_3 \quad pi_4 \end{array} \quad \left[\begin{array}{cccc} b_{1,1,x[1]} & b_{1,2,x[1]} & b_{1,3,x[1]} & b_{1,4,x[1]} \\ \textcolor{red}{b}_{2,1,x[1]} & b_{2,2,x[1]} & b_{2,3,x[1]} & b_{2,4,x[1]} \\ b_{3,1,x[1]} & b_{3,2,x[1]} & b_{3,3,x[1]} & b_{3,4,x[1]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[1]} & b_{14,2,x[1]} & b_{14,3,x[1]} & b_{14,4,x[1]} \end{array} \right]$$

$$d_{1,2,1} = pi_1 \times b_{2,1,x[1]}$$

⋮

$$\begin{bmatrix} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{bmatrix} \begin{matrix} [pi_1 \quad pi_2 \quad pi_3 \quad \textcolor{red}{pi_4}] \end{matrix} \begin{bmatrix} b_{1,1,x[1]} & b_{1,2,x[1]} & b_{1,3,x[1]} & b_{1,4,x[1]} \\ b_{2,1,x[1]} & b_{2,2,x[1]} & b_{2,3,x[1]} & b_{2,4,x[1]} \\ b_{3,1,x[1]} & b_{3,2,x[1]} & b_{3,3,x[1]} & b_{3,4,x[1]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[1]} & b_{14,2,x[1]} & b_{14,3,x[1]} & \textcolor{red}{b_{14,4,x[1]}} \end{bmatrix}$$

$$d_{1,14,4} = pi_4 \times b_{14,4,x[1]}$$

2. Iteration step:

$$\begin{bmatrix} \textcolor{red}{d_{2,1,1}} & d_{2,1,2} & d_{2,1,3} & d_{2,1,4} \\ d_{2,2,1} & d_{2,2,2} & d_{2,2,3} & d_{2,2,4} \\ d_{2,3,1} & d_{2,3,2} & d_{2,3,3} & d_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{2,14,1} & d_{2,14,2} & d_{2,14,3} & d_{2,14,4} \end{bmatrix} = \max_i \left(\begin{bmatrix} \textcolor{red}{d_{1,1,1}} & \textcolor{red}{d_{1,1,2}} & \textcolor{red}{d_{1,1,3}} & \textcolor{red}{d_{1,1,4}} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{bmatrix} \cdot \begin{bmatrix} \textcolor{red}{a_{1,1,1}} & \textcolor{red}{a_{1,2,1}} & \textcolor{red}{a_{1,3,1}} & \textcolor{red}{a_{1,4,1}} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} & a_{2,4,1} \\ a_{3,1,1} & a_{3,2,1} & a_{3,3,1} & a_{3,4,1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,1} & a_{14,2,1} & a_{14,3,1} & a_{14,4,1} \end{bmatrix} \cdot \begin{bmatrix} \textcolor{red}{b_{1,1,x[2]}} & b_{1,2,x[2]} & b_{1,3,x[2]} & b_{1,4,x[2]} \\ b_{2,1,x[2]} & b_{2,2,x[2]} & b_{2,3,x[2]} & b_{2,4,x[2]} \\ b_{3,1,x[2]} & b_{3,2,x[2]} & b_{3,3,x[2]} & b_{3,4,x[2]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[2]} & b_{14,2,x[2]} & b_{14,3,x[2]} & b_{14,4,x[2]} \end{bmatrix} \right)$$

$$d_{2,1,1} = \max(d_{1,1,1} \times a_{1,1,1}, d_{1,1,2} \times a_{1,2,1}, \dots, d_{1,1,4} \times a_{1,4,1}) \times b_{1,1,x[2]}$$

$$\begin{bmatrix} \textcolor{red}{p_{2,1,1}} & p_{2,1,2} & p_{2,1,3} & p_{2,1,4} \\ p_{2,2,1} & p_{2,2,2} & p_{2,2,3} & p_{2,2,4} \\ p_{2,3,1} & p_{2,3,2} & p_{2,3,3} & p_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ p_{2,14,1} & p_{2,14,2} & p_{2,14,3} & p_{2,14,4} \end{bmatrix} = \argmax_i \left(\begin{bmatrix} \textcolor{red}{d_{1,1,1}} & \textcolor{red}{d_{1,1,2}} & \textcolor{red}{d_{1,1,3}} & \textcolor{red}{d_{1,1,4}} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{bmatrix} \cdot \begin{bmatrix} \textcolor{red}{a_{1,1,1}} & \textcolor{red}{a_{1,2,1}} & \textcolor{red}{a_{1,3,1}} & \textcolor{red}{a_{1,4,1}} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} & a_{2,4,1} \\ a_{3,1,1} & a_{3,2,1} & a_{3,3,1} & a_{3,4,1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,1} & a_{14,2,1} & a_{14,3,1} & a_{14,4,1} \end{bmatrix} \right)$$

$$p_{2,1,1} = \argmax(d_{1,1,1} \times a_{1,1,1}, d_{1,1,2} \times a_{1,2,1}, \dots, d_{1,1,4} \times a_{1,4,1})$$

$$\begin{bmatrix} d_{2,1,1} & d_{2,1,2} & d_{2,1,3} & d_{2,1,4} \\ \textcolor{red}{d_{2,2,1}} & d_{2,2,2} & d_{2,2,3} & d_{2,2,4} \\ d_{2,3,1} & d_{2,3,2} & d_{2,3,3} & d_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{2,14,1} & d_{2,14,2} & d_{2,14,3} & d_{2,14,4} \end{bmatrix} = \max_i \left(\begin{bmatrix} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ \textcolor{red}{d_{1,2,1}} & \textcolor{red}{d_{1,2,2}} & \textcolor{red}{d_{1,2,3}} & \textcolor{red}{d_{1,2,4}} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{bmatrix} \cdot \begin{bmatrix} \textcolor{red}{a_{1,1,1}} & \textcolor{red}{a_{1,2,1}} & \textcolor{red}{a_{1,3,1}} & \textcolor{red}{a_{1,4,1}} \\ \textcolor{red}{a_{2,1,1}} & \textcolor{red}{a_{2,2,1}} & \textcolor{red}{a_{2,3,1}} & \textcolor{red}{a_{2,4,1}} \\ a_{3,1,1} & a_{3,2,1} & a_{3,3,1} & a_{3,4,1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,1} & a_{14,2,1} & a_{14,3,1} & a_{14,4,1} \end{bmatrix} \cdot \begin{bmatrix} b_{1,1,x[2]} & b_{1,2,x[2]} & b_{1,3,x[2]} & b_{1,4,x[2]} \\ \textcolor{red}{b_{2,1,x[2]}} & b_{2,2,x[2]} & b_{2,3,x[2]} & b_{2,4,x[2]} \\ b_{3,1,x[2]} & b_{3,2,x[2]} & b_{3,3,x[2]} & b_{3,4,x[2]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[2]} & b_{14,2,x[2]} & b_{14,3,x[2]} & b_{14,4,x[2]} \end{bmatrix} \right)$$

$$d_{2,2,2} = \max(d_{1,2,1} \times a_{2,1,2}, d_{1,2,2} \times a_{2,2,2}, \dots, d_{1,2,4} \times a_{2,4,2}) \times b_{2,2,x[2]}$$

$$\begin{bmatrix} p_{2,1,1} & p_{2,1,2} & p_{2,1,3} & p_{2,1,4} \\ p_{2,2,1} & \textcolor{red}{p_{2,2,2}} & p_{2,2,3} & p_{2,2,4} \\ p_{2,3,1} & p_{2,3,2} & p_{2,3,3} & p_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ p_{2,14,1} & p_{2,14,2} & p_{2,14,3} & p_{2,14,4} \end{bmatrix} = \operatorname{argmax}_i \left(\begin{bmatrix} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ \textcolor{red}{d_{1,2,1}} & \textcolor{red}{d_{1,2,2}} & \textcolor{red}{d_{1,2,3}} & \textcolor{red}{d_{1,2,4}} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{1,14,1} & d_{1,14,2} & d_{1,14,3} & d_{1,14,4} \end{bmatrix} \cdot \begin{bmatrix} a_{1,1,2} & a_{1,2,2} & a_{1,3,2} & a_{1,4,2} \\ \textcolor{red}{a_{2,1,2}} & \textcolor{red}{a_{2,2,2}} & \textcolor{red}{a_{2,3,2}} & \textcolor{red}{a_{2,4,2}} \\ a_{3,1,2} & a_{3,2,2} & a_{3,3,2} & a_{3,4,2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,2} & a_{14,2,2} & a_{14,3,2} & a_{14,4,2} \end{bmatrix} \right)$$

$$p_{2,2,2} = \operatorname{argmax}(d_{1,2,1} \times a_{2,1,2}, d_{1,2,2} \times a_{2,2,2}, \dots, d_{1,2,4} \times a_{2,4,2})$$

⋮

$$\begin{bmatrix} d_{2,1,1} & d_{2,1,2} & d_{2,1,3} & d_{2,1,4} \\ d_{2,2,1} & d_{2,2,2} & d_{2,2,3} & d_{2,2,4} \\ d_{2,3,1} & d_{2,3,2} & d_{2,3,3} & d_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{2,14,1} & d_{2,14,2} & d_{2,14,3} & \textcolor{red}{d_{2,14,4}} \end{bmatrix} = \max_i \left(\begin{bmatrix} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ d_{1,2,1} & d_{1,2,2} & d_{1,2,3} & d_{1,2,4} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ \textcolor{red}{d_{1,14,1}} & \textcolor{red}{d_{1,14,2}} & \textcolor{red}{d_{1,14,3}} & \textcolor{red}{d_{1,14,4}} \end{bmatrix} \cdot \begin{bmatrix} a_{1,1,2} & a_{1,2,2} & a_{1,3,2} & a_{1,4,2} \\ a_{2,1,2} & a_{2,2,2} & a_{2,3,2} & a_{2,4,2} \\ a_{3,1,2} & a_{3,2,2} & a_{3,3,2} & a_{3,4,2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,2} & a_{14,2,2} & a_{14,3,2} & a_{14,4,2} \end{bmatrix} \cdot \begin{bmatrix} b_{1,1,x[2]} & b_{1,2,x[2]} & b_{1,3,x[2]} & b_{1,4,x[2]} \\ b_{2,1,x[2]} & b_{2,2,x[2]} & b_{2,3,x[2]} & b_{2,4,x[2]} \\ b_{3,1,x[2]} & b_{3,2,x[2]} & b_{3,3,x[2]} & b_{3,4,x[2]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[2]} & b_{14,2,x[2]} & b_{14,3,x[2]} & \textcolor{red}{b_{14,4,x[2]}} \end{bmatrix} \right)$$

$$d_{2,14,4} = \max(d_{1,14,1} \times a_{14,1,2}, d_{1,14,2} \times a_{14,2,2}, \dots, d_{1,14,4} \times a_{14,4,2}) \times b_{14,4,x[2]}$$

$$\begin{bmatrix} p_{2,1,1} & p_{2,1,2} & p_{2,1,3} & p_{2,1,4} \\ p_{2,2,1} & p_{2,2,2} & p_{2,2,3} & p_{2,2,4} \\ p_{2,3,1} & p_{2,3,2} & p_{2,3,3} & p_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ p_{2,14,1} & p_{2,14,2} & p_{2,14,3} & \textcolor{red}{p_{2,14,4}} \end{bmatrix} = \operatorname{argmax}_i \left(\begin{bmatrix} d_{1,1,1} & d_{1,1,2} & d_{1,1,3} & d_{1,1,4} \\ \textcolor{red}{d_{1,2,1}} & \textcolor{red}{d_{1,2,2}} & \textcolor{red}{d_{1,2,3}} & \textcolor{red}{d_{1,2,4}} \\ d_{1,3,1} & d_{1,3,2} & d_{1,3,3} & d_{1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ \textcolor{red}{d_{1,14,1}} & \textcolor{red}{d_{1,14,2}} & \textcolor{red}{d_{1,14,3}} & \textcolor{red}{d_{1,14,4}} \end{bmatrix} \cdot \begin{bmatrix} a_{1,1,2} & a_{1,2,2} & a_{1,3,2} & a_{1,4,2} \\ a_{2,1,2} & a_{2,2,2} & a_{2,3,2} & a_{2,4,2} \\ a_{3,1,2} & a_{3,2,2} & a_{3,3,2} & a_{3,4,2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,2} & a_{14,2,2} & a_{14,3,2} & a_{14,4,2} \end{bmatrix} \right)$$

$$p_{2,14,4} = \operatorname{argmax}(d_{1,14,1} \times a_{14,1,2}, d_{1,14,2} \times a_{14,2,2}, \dots, d_{1,14,4} \times a_{14,4,2})$$

$$\begin{bmatrix} d_{3,1,1} & d_{3,1,2} & d_{3,1,3} & d_{3,1,4} \\ d_{3,2,1} & d_{3,2,2} & d_{3,2,3} & d_{3,2,4} \\ d_{3,3,1} & d_{3,3,2} & d_{3,3,3} & d_{3,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{3,14,1} & d_{3,14,2} & d_{3,14,3} & d_{3,14,4} \end{bmatrix} = \max_i \left(\begin{bmatrix} d_{2,1,1} & d_{2,1,2} & d_{2,1,3} & d_{2,1,4} \\ d_{2,2,1} & d_{2,2,2} & d_{2,2,3} & d_{2,2,4} \\ d_{2,3,1} & d_{2,3,2} & d_{2,3,3} & d_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{2,14,1} & d_{2,14,2} & d_{2,14,3} & d_{2,14,4} \end{bmatrix} \cdot \begin{bmatrix} a_{1,1,3} & a_{1,2,3} & a_{1,3,3} & a_{1,4,3} \\ a_{2,1,3} & a_{2,2,3} & a_{2,3,3} & a_{2,4,3} \\ a_{3,1,3} & a_{3,2,3} & a_{3,3,3} & a_{3,4,3} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,3} & a_{14,2,3} & a_{14,3,3} & a_{14,4,3} \end{bmatrix} \cdot \begin{bmatrix} b_{1,1,x[3]} & b_{1,2,x[3]} & b_{1,3,x[3]} & b_{1,4,x[3]} \\ b_{2,1,x[3]} & b_{2,2,x[3]} & b_{2,3,x[3]} & b_{2,4,x[3]} \\ b_{3,1,x[3]} & b_{3,2,x[3]} & b_{3,3,x[3]} & b_{3,4,x[3]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[3]} & b_{14,2,x[3]} & b_{14,3,x[3]} & b_{14,4,x[3]} \end{bmatrix} \right)$$

$$d_{3,1,1} = \max(d_{2,1,1} \times a_{1,1,3}, d_{2,1,2} \times a_{1,2,3}, \dots, d_{2,1,4} \times a_{1,4,3}) \times b_{1,1,x[3]}$$

$$\begin{bmatrix} p_{3,1,1} & p_{3,1,2} & p_{3,1,3} & p_{3,1,4} \\ p_{3,2,1} & p_{3,2,2} & p_{3,2,3} & p_{3,2,4} \\ p_{3,3,1} & p_{3,3,2} & p_{3,3,3} & p_{3,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ p_{3,14,1} & p_{3,14,2} & p_{3,14,3} & p_{3,14,4} \end{bmatrix} = \operatorname{argmax}_i \left(\begin{bmatrix} d_{2,1,1} & d_{2,1,2} & d_{2,1,3} & d_{2,1,4} \\ d_{2,2,1} & d_{2,2,2} & d_{2,2,3} & d_{2,2,4} \\ d_{2,3,1} & d_{2,3,2} & d_{2,3,3} & d_{2,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{2,14,1} & d_{2,14,2} & d_{2,14,3} & d_{2,14,4} \end{bmatrix} \cdot \begin{bmatrix} a_{1,1,3} & a_{1,2,3} & a_{1,3,3} & a_{1,4,3} \\ a_{2,1,3} & a_{2,2,3} & a_{2,3,3} & a_{2,4,3} \\ a_{3,1,3} & a_{3,2,3} & a_{3,3,3} & a_{3,4,3} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,3} & a_{14,2,3} & a_{14,3,3} & a_{14,4,3} \end{bmatrix} \right)$$

$$p_{3,1,1} = \operatorname{argmax}(d_{2,1,1} \times a_{1,1,3}, d_{2,1,2} \times a_{1,2,3}, \dots, d_{2,1,4} \times a_{1,4,3})$$

$$\begin{bmatrix} d_{L,1,1} & d_{L,1,2} & d_{L,1,3} & d_{L,1,4} \\ d_{L,2,1} & d_{L,2,2} & d_{L,2,3} & d_{L,2,4} \\ d_{L,3,1} & d_{L,3,2} & d_{L,3,3} & d_{L,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{L,14,1} & d_{L,14,2} & d_{L,14,3} & d_{L,14,4} \end{bmatrix} = \max_i \left(\begin{bmatrix} d_{L-1,1,1} & d_{L-1,1,2} & d_{L-1,1,3} & d_{L-1,1,4} \\ d_{L-1,2,1} & d_{L-1,2,2} & d_{L-1,2,3} & d_{L-1,2,4} \\ d_{L-1,3,1} & d_{L-1,3,2} & d_{L-1,3,3} & d_{L-1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{L-1,14,1} & d_{L-1,14,2} & d_{L-1,14,3} & d_{L-1,14,4} \end{bmatrix} \cdot \begin{bmatrix} a_{1,1,4} & a_{1,2,4} & a_{1,3,4} & a_{1,4,4} \\ a_{2,1,4} & a_{2,2,4} & a_{2,3,4} & a_{2,4,4} \\ a_{3,1,4} & a_{3,2,4} & a_{3,3,4} & a_{3,4,4} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,4} & a_{14,2,4} & a_{14,3,4} & a_{14,4,4} \end{bmatrix} \cdot \begin{bmatrix} b_{1,1,x[L]} & b_{1,2,x[L]} & b_{1,3,x[L]} & b_{1,4,x[L]} \\ b_{2,1,x[L]} & b_{2,2,x[L]} & b_{2,3,x[L]} & b_{2,4,x[L]} \\ b_{3,1,x[L]} & b_{3,2,x[L]} & b_{3,3,x[L]} & b_{3,4,x[L]} \\ \vdots & \vdots & \vdots & \vdots \\ b_{14,1,x[L]} & b_{14,2,x[L]} & b_{14,3,x[L]} & b_{14,4,x[L]} \end{bmatrix} \right)$$

$$d_{L,14,4} = \max(d_{L-1,14,1} \times a_{14,1,4}, d_{L-1,14,2} \times a_{14,2,4}, \dots, d_{L-1,14,4} \times a_{14,4,4}) \times b_{14,4,x[L]}$$

$$\begin{bmatrix} p_{L,1,1} & p_{L,1,2} & p_{L,1,3} & p_{L,1,4} \\ p_{L,2,1} & p_{L,2,2} & p_{L,2,3} & p_{L,2,4} \\ p_{L,3,1} & p_{L,3,2} & p_{L,3,3} & p_{L,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ p_{L,14,1} & p_{L,14,2} & p_{L,14,3} & p_{L,14,4} \end{bmatrix} = \operatorname{argmax}_i \left(\begin{bmatrix} d_{L-1,1,1} & d_{L-1,1,2} & d_{L-1,1,3} & d_{L-1,1,4} \\ d_{L-1,2,1} & d_{L-1,2,2} & d_{L-1,2,3} & d_{L-1,2,4} \\ d_{L-1,3,1} & d_{L-1,3,2} & d_{L-1,3,3} & d_{L-1,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{L-1,14,1} & d_{L-1,14,2} & d_{L-1,14,3} & d_{L-1,14,4} \end{bmatrix} \cdot \begin{bmatrix} a_{1,1,4} & a_{1,2,4} & a_{1,3,4} & a_{1,4,4} \\ a_{2,1,4} & a_{2,2,4} & a_{2,3,4} & a_{2,4,4} \\ a_{3,1,4} & a_{3,2,4} & a_{3,3,4} & a_{3,4,4} \\ \vdots & \vdots & \vdots & \vdots \\ a_{14,1,4} & a_{14,2,4} & a_{14,3,4} & a_{14,4,4} \end{bmatrix} \right)$$

$$p_{L,14,4} = \operatorname{argmax}(d_{L-1,14,1} \times a_{14,1,4}, d_{L-1,14,2} \times a_{14,2,4}, \dots, d_{L-1,14,4} \times a_{14,4,4})$$

3. Termination:

$$[\tau_1 \quad \tau_2 \quad \cdots \quad \tau_{L-2} \quad \tau_{L-1} \quad \tau_L] = \underset{1 \leq i \leq 14, 1 \leq j \leq 4}{\operatorname{argmax}} \left(\begin{bmatrix} d_{L,1,1} & d_{L,1,2} & d_{L,1,3} & d_{L,1,4} \\ d_{L,2,1} & d_{L,2,2} & d_{L,2,3} & d_{L,2,4} \\ d_{L,3,1} & d_{L,3,2} & d_{L,3,3} & d_{L,3,4} \\ \vdots & \vdots & \vdots & \vdots \\ d_{L,14,1} & d_{L,14,2} & d_{L,14,3} & d_{L,14,4} \end{bmatrix} \right)$$

4. Backtracking:

$$[\tau_1 \quad \tau_2 \quad \cdots \quad \tau_{L-2} \quad \tau_{L-1} \quad \tau_L] = \psi(L, Y_L, \tau_L)$$

$$\vdots$$

$$[\tau_1 \quad \tau_2 \quad \cdots \quad \tau_{L-2} \quad \tau_{L-1} \quad \tau_L] = \psi(2, Y_2, \tau_2)$$

List of Figures

	1	Methodology Diagram of our Ingredient Extractor	3
	2	Transition matrix used in full second order HMM	7
245	3	Lexical matrix used in full second order HMM	7
	4	Transition Matrix used in our Ingredient Extractor	8
	5	Lexical Matrix used in our Ingredient Extractor	9
	6	Variations of accuracy of our Ingredient Extractor depending on λ and accuracy score on the first layer	12
250	7	Variations of F1 score of our Ingredient Extractor depending on λ and accuracy score on the first layer	12

List of Tables

	1	Example of trained sentence on our trained corpus	3
	2	POS Tag-set used to annotate our corpus	4
255	3	The discrete state-dependent word probabilities table for a mini-corpus	10
	4	The discrete state-dependent word probabilities table for a mini-corpus (prefix-based approach)	10
260	5	The discrete state-dependent word probabilities table for a mini-corpus after adding the probabilities $P(v_i/t_i)$ for words with the same tag (prefix-based approach)	11
	6	Comparison of performances between our ingredient extractor and other HMM taggers using the trained corpus as a testing dataset	11
265	7	Performances of first order HMM used to predict ingredients state using 10 fold cross-validation	13
	8	Performances of second order HMM used to predict ingredients state using 10 fold cross-validation	13
270	9	Performances of our ingredient extractor with 100 % accuracy on the first layer and $\lambda = 4$ using 10 fold cross-validation	14
	10	Performances of our ingredient extractor with various results accuracy in the first layer and $\lambda = 4$ using 10 fold cross-validation	14
275	11	Comparison of performances using test dataset independent from training dataset and using 10-fold cross validation with 213 as an average number of unknown words	15

References

- Ahn, Y.-Y., Ahnert, S. E., Bagrow, J. P., & Barabási, A.-L. (2011). Flavor network and the principles of food pairing. *Scientific reports*, 1, 196.
- Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the third conference on applied natural language processing* (pp. 133–140).

- Mikheev, A. (1996). Learning part-of-speech guessing rules from lexicon: Extension to non-concatenative operations. In *Proceedings of the 16th conference on computational linguistics-volume 2* (pp. 770–775).
- 285 Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286.
- Sajadmanesh, S., Jafarzadeh, S., Ossia, S. A., Rabiee, H. R., Haddadi, H., Mejova, Y., ... Stringhini, G. (2017). Kissing cuisines: Exploring world-wide culinary habits on the web. In *Proceedings of the 26th international*
 290 *conference on world wide web companion* (pp. 1013–1021).
- Simas, T., Ficek, M., Diaz-Guilera, A., Obrador, P., & Rodriguez, P. R. (2017). Food-bridging: a new network construction to unveil the principles of cooking. *Frontiers in ICT*, 4, 14.
- Thede, S. M., & Harper, M. P. (1999). A second-order hidden markov model
 295 for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the association for computational linguistics on computational linguistics* (pp. 175–182).
- Zhou, G., & Su, J. (2002). Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th annual meeting on association*
 300 *for computational linguistics* (pp. 473–480).