Chinese (Restaurant) Menu Translation
Ting Liao

1 Description of the problem

When you goes to a Chinese restaurant, especially when traveling in Asia, you will have no idea what the menu says even if there is an English translation on it. You will see names from "burn the spring chicken" to "cowboy meat". Even the menu at Olympic Village during the 2008 Beijing Olympics has similar problems. The main reason is that a normal translator cannot generally handle the name of a dish due to the lack of sentence structure. When you simply group a set of nouns together, as in most of the cases in names of Chinese dishes, it would be hard for the translator to decide what to do when the words have multiple meanings.

If we want to build a general translator that can handle food translation, it would be very hard. However, if we create a translator that is specifically for food translation, the problem becomes much easier to tackle. First of all we do not have to worry about the context, we know it has to do with food, and knowing that the numbers of useful vocabularies is easier to handle. This paper describes a supervised learning algorithm that trains the computer to translate the name of a dish in Chinese to English.

2 The idea

In the movie "Terminal", Tom Hanks learned the English language by comparing a booklet in English and the same booklet in the fictional Krakozhian. The same idea applies here. The training set contains entries consisted of the Chinese name of a dish and the corresponding English name of the dish. The challenge to is break up to Chinese name such that we can find the Chinese word(s) that corresponds to a certain English word(s) in the name. Unlike the English Language, Chinese words are not made up by letters, but instead they are made up with characters. A Chinese word can be a single character or a multi-character one. (however a word with 5 or more characters is extremely rare). Another difference between the languages is that Chinese words are not separated by space. So when you see two Chinese characters together, it could be one word or it could be two, and we will have to account for that.

3 Model

We assume that there is a correct objective translation $F(C) \rightarrow + E$, which means whenever we see the word C, we can translate it to multiple E's, the different E's should only happen when the word C is used for different meanings. By objective I mean if two C's have the same meaning, they cannot return a different E. Consider a training set of size m (we have m pairs), the Chinese word $C^{(i)}$ is the Chinese translation of $E^{(i)}$. $C^{(i)} = C_1^{(i)} C_2^{(i)} .... C_n^{(i)}$ where $C_j^{(i)}$ is the j-th character for the Chinese word with length n and $E_j^{(i)}$ is the j-th word in the English word. The goal is to create a set of translation rules F such that when we see a Chinese word C, we can apply $F(C) \rightarrow E$ for the English translation E.

Obviously, if we choose F that makes. $F(C^{(i)}) \rightarrow E^{(i)}$, then we will get a 100% match for the training set. But as you can imagine, if we get anything that is not in the set, we will fail to translate it, so this is not very useful. If we know the word "chicken", we should be able to use this knowledge in other context. So if there is a word Cj we see multiple times in different context, and their English translations contain the same specific sequence E, the rule $F(Cj) \rightarrow E$ will be a useful one because the more appearances of this translation in the training set could mean that there is a higher probability we see this rule used in words outside of this training set. We assign a score to each translation guess. The score should reflect

two things :
1. The more this guess appears, the higher the score should be
2. A rule that translates an n-length C is preferred to a translation rule for a shorter C. Otherwise we will always translate only the substring of C because if the translation for F(C) → E has x appearance, the rule F(substring of C) → must have at least x appearance. But it would seem that F(C) → E should be considered if the number of appearance does not differ by too much.

The model I used (which is a bit arbitrary) is for each rule F(C) → E, where the length of C is n, we give it a -n score, but we will treat the set of rules $F(C^{(i)}) \rightarrow E^{(i)}$ as the starting point, meaning it has 0 points. Now if we find a rule F(C)->E with a length n C that appears x times, then these n * x characters can be represented by 1 rule, I thus treat this rule as scoring n * (x-1) points. We can see that this scoring system fits well with the 2 requirements I set if we use a greedy algorithm to try to maximize the score. (each $C_j^{(i)}$ can only be used to help find one rule, after it is used, we delete that character. See below).

4 Training

The ideal case here would be maximizing the score, which is find a set of rules F such that
each F(C) → E for a C with length n that appears x times has score n * (x-1)
find the set of non overlapping (we cannot use two different F for the same $C_j^{(i)}$ when keeping score) F's that will maximize the score for this training set.

The problem is, the total number of possibility of F is enormous because of the multi-character and multi-word translation. So it would be hard to find the "real" maximum. Instead I used a greedy algorithm to find F. We went through the whole set and find out the single rule F that creates the highest scores. (in my dataset, the highest score rule is for the word "rice", which has score 87.)

The way to do that is simply going through all the words with different lengths, then find out how many times the word has appeared in the training set, then we find out the English translation for these words and find out which English pattern has appeared most (and x = the number of times the English pattern appears). Once we have found the F(C) → E with the highest score at this point, we can remove C and E from the training pairs if C is in the Chinese name and E is in the English name of the training pair. We replace C and E with *, to make sure we won't check for the word AC after B is removed from ABC. We then repeat the process until we cannot find rule (from the left over words) F such that it scores higher than 0. This is the first phase of the algorithm

The second phase is to train something I called the "unbreakable". Which represents a Chinese word that cannot be broken up into substring that makes sense, it happens when there is a special name of a dish. Therefore we simply add $F(C^{(i)}) \rightarrow E^{(i)}$ as one of the translation rules. The third phase is to find out all the useless words. Sometimes on a menu, there are some useless words (in the sense of translations) like "wonderful" and "tasty". So after removing all our guesses. If the Chinese part of the left over pair is not empty but the English part is empty, we make a guess that these Chinese words are useless. That means F(C)-> "" . If at this point there are pairs that has only one consecutive sequence of characters in the Chinese part and only one sequence of words in the English part, we basically link these sequence of Chinese words to the sequence of English words. Since we do not have a great basis for these guesses, we simply give them a score of 0. There may be still some pairs that are still not completely mapped, so we will have to give up and find out what the pair originally looked like (before replacing words with *) and make a direct mapping from the Chinese name to the English name.

5 Testing

When given a Chinese name C, we try to find out what combinations of F(Cj)->E give the highest scores. This method inherently has some problems. For a word that has multiple meaning, if one meaning dominates by having a much higher score, the other meaning does not have much chance of showing up (unless it can squeeze into the top 3, which means the rest of the words do not allow multiple meanings), even though it may be the correct meaning. We tested these translations on the original training set, trying to see what is the average hit rate (a word in the guess that is in the original English word), and the result turned out to be 90%. I tested it on different subsets of the training set and the result varies between 75% to 98%.

6 Problem

I) Positions

The biggest problem I have is with the position of the words. We may get F(C)->E right for a substring C, but where should E be put in our final guess? That is a major problem right now. I have a method in mind but I am not sure whether it will work. Given each F(C)->E, in addition to giving it a score, we assign to it a position score, which is the average "index of E / the length of the whole word" of all words where this rule F(C)->E applies. And when we are making the final guess, we sort the substring by their position score. This part I have not implemented.

II) Word inconsistency

The other problem is the inconsistency in words. For example, when to use "fried" and "deep fried"? They could mean the same thing most of the time. But using my algorithm, we will always translate the Chinese word to "fried". That is not that big a problem by itself, the translation is reasonable, but when we are in the later phases of the training, it depends on the first phase removing the words accurately and in this case, "deep" is left over and may lead to unexpected result. For example a "useless" word will now be translated to "deep" instead of the empty string "". That is not the only problem with inconsistency. We have different tenses and we have plural and singular forms. So is it meat slice? Or is it sliced meat? If C is the Chinese word for meat slice, F(C) → meat is the most likely result. Although we give weights to the length of the Chinese words, we do not really give weight to the length of the English word. Inconsistencies simply confuse the algorithm.

Perhaps what we can do here is do a reverse matching, meaning that we will use a similar algorithm to find a set of rules G(E)->C, which is supposed to be F-inverse. Training G will put weights on the length of the English word such that one mistake (e.g. typo) is not going to change F(C)->meat slice to F(C)->slice (if one of the set has "meet slice" instead). This is actually a quite important fix. But the problem is that this is not a 1 to 1 mapping. If it were, running this algorithm both ways would be a great way to remove inconsistencies. But since F is a one to many mapping, G will have to be many to one if it is F-inverse. That would be hard to achieve, so the most likely G will end up to be a many to many mapping. We can then try each of these rules in G(E)-> C and create a corresponding F(C)->E and see how the rule scores in the training set.

III)One to many mapping

I guess the worst thing about this algorithm is that if the F(C)->E is a one-to-many mapping, they we get multiple results. As I have explained above, a less likely translation could be the right translation, but my algorithm does not know when is that the case. If somehow the translation got you a choice between "beef rice" and "pork rice", you can't really decide even with the help of the translator.

8 Application

One of the main point of this algorithm is that we do not need to know much about the languages. We do need some minimal knowledge however such as the Chinese words are not separated by space but the English language is etc. If we have a perfectly objectively consistently translated menu (between languages), we can simply feed it to the machine, then the algorithm will learn whatever it can and be used to translate other menu. Provided we have proper character recognition software on cell phones, we can take pictures of the menu and find out the guessed meaning of the name, which would be interesting.

9 Conclusion

The algorithm itself is not an especially efficient one, for the training data I have 451 training pair (which by the way made me really hungry when I worked on this) it took about 30 minutes (after quite a bit of optimization).. The slowness is due to the fact that I do not know how to break up the words, which lead me to a lot of guessing and thus delaying the result. I got most of my data from restaurants around the bay area. But the problem remains : these menus are badly translated and extremely inconsistent (and creative), so unfortunately I have to put in my own translated names for the program to work. It is unfortunate because my goal is to feed the algorithm with menus that I do not understand and have the machine learn it for me. The algorithm can possibly be applied to other fields where the context does not involve too many words and where the grammar is not too important in the translation (I think “pork rice black bean sauce” is much better than “bone rice”, so even with bad grammar the translation here I think is quite useful to some people). I think the translations are helpful to a certain degree with a lot of room for improvement, but since it may be related to what you eat and thus maybe your health, use with care. Attached is a list with F's that has the highest score.

Training Data obtained from :
Hong Kong Restaurant
Washington Bakery and Restaurant
Tam Cafe

飯	rice	87
牛肉	beef	74
雞	chicken	70
炒	fried	51
麵	noodle	46
海鮮	seafood	38
湯	soup	37
菜	vegetable	31
牛	beef	31
煲	clay pot	30
豆腐	bean curd	28
雲吞	wonton	28
XO醬	xo sauce	27
三文治	sandwich	27
椒	pepper	26
焗	baked	26
豬扒	pork chop	22
意粉	spaghetti	22
蛋	egg	21
咖哩	curry	20
汁	sauce	18
魚	fish	18
通粉	macaroni	18
蝦	shrimp	17
排骨	spare rib	16
豉	black bean	15
絲	shredded	15
粥	porridge	15
麵	mein	14
公仔	instant	14
鹽	salt	13
肉	pork	13
米	rice	12
薑蔥	ginger	12
沙爹	satay	12
斑球	cod	12
炸	fried	11
河	chow fun	11
蝦	prawn	10
粟米	corn	10
茄	tomato	10
粉絲	vermicelli	10
宮保	kung pao	10
多士	toast	10
肉	meat	9
叉燒	pork	8
茄子	egg plant	8
瑤柱	scallop	8
菇	mushroom	8
沙茶	satay sauce	8
腩	stew	8
黑	black	8
蠔油	oyster sauce	8
火腿	ham	8
烏冬	udon	8
撈	braised	8
豆	pea	7
粒	diced	7
鮮魷	squid	6
煎	chow	6
忌廉	cream	6
芥菜	mustard	6
鴨	duck	6
招牌	special	6
午餐	luncheon	6
腸粉	rice	6
腰肝	kidney	6
水餃	dumpling	6
瀨粉	lai fun	6
牛仔骨	short rib	6
什	haslet	5
蒜	garlic	5
式	style	5
味	chinese	5
柳	fillet	5
蘭	broccoli	5
伊	yee	5
生蠔	oyster	4
菜	tender	4
羹	soup	4
節瓜	chinese melon	4
上湯	broth	4
蜆	clam	4
牛	ox	4
XO	xo sauce	4
腰果	cashew	4
蛋	ball	4
炆	braised	4
家鄉	country style	4
河	noodle	4
豬	pork	4
丸	ball	4
免治	minced	4
芋頭	taro	4
三絲	assorted meat	4
油	butter	3
蒸	steamed	3
酸	sour	3
鮑	abalone	3
扒	steak	3
雪	snow	3
醬	sauce	3
味	preserved	3
炒	sauteed	3
炒	sauce	3
咸	salted	3

fig. 1 translation rules with the highest scores