Language and Computation

LING 227 01 / 627 01 / PSYC 327 01

Interim summary

(in lieu of the lecture cancelled due to snow storm)

February 13, 2014

1 Introductory remarks

The following pages summarize the topics covered in the last few lectures and prepares the floor for the following ones. It is meant as a replacement for the lecture of 02/13, cancelled due to the snow storm. While reading the seemingly redundant repetition of earlier topics, please also pay close attention to the minor remarks added here and there.

In the previous lectures, I introduced the basics of **machine learning**, which is *the* major focus of contemporary computational linguistics. We used text classification as an example. Texts (or documents) were characterized by frequencies: either with word or letter frequencies, or with **n-gram** frequencies. Please remember that n-grams can be defined both on the character level (for instance, pairs of characters) and on the word level (for example, triplets of words). In most applications, we fix both n and the unit (character or word) using which n-grams (a.k.a. n-tuples) are collected.[1] One also has to make clear whether n-grams overlap (as almost always), or they don't.

Introducing the notion of n-grams has paved the way to **Markov Models** (to be defined below), a concept extremely useful in many NLP applications. Following the textbook, we shall discuss Markov Models with two applications in mind: parts-of-speech tagging (POS-tagging) and speech recognition (ASR).

Beside (re-)introducing these and further concepts, an additional goal of mine is to explain how the popular concept of **probability** is used (or, sometimes, misused) in computational linguistics. The turn toward probabilistic models in the 1990s has resulted in an unprecedented explosion in the field, a consequence of which is that language technology has broken the barriers of the theoretical labs, and has penetrated our everyday life.

1. A future homework might involve a method that makes use of n-grams generated with several n values at the same time: pairs, triplets etc. are considered together. Yet, such an approach is rare. Please remember the terms *unigram*, *bigram* and *trigram*, standing for n-grams with $n = 1$, $n = 2$ and $n = 3$, respectively. The term *n-tuple* generalizes over the terms *single*, *double* (or *pair*), *triple*, *quadruple*, *quintuple*, etc. Finally, note that word frequencies and character frequencies are simply unigram frequencies on the word and character levels, respectively. Therefore, speaking about n-grams is just a convenient shorthand to cover many similar options: unigrams and non-unigrams, with either characters or words as units.

2 The notion of *probability*

2.1 Two approaches: frequentist and measure-theoretic

The mathematical concept of *probability* developed in the seventeenth and early eighteenth century from the concept of *frequency*: the probability of an event corresponds to the frequency of that event, when the experiment is repeated many times. For instance,

> the probability of throwing a 6 with a dice is 1/6 *because* if you throw a dice many times, you will obtain a 6 in approximately 1/6 of the cases.

However, this naive, **frequentist approach** has been recast into a formal theory in the nineteenth and twentieth century, resulting in a **measure-theoretic approach**. Here is a summary that is slightly less abstract than what you would find in mathematical textbooks:

1. Let Ω be a set, called a *sample space.* You can think of this set as the set of elementary outcomes in an experiment, such as the six possibilities when rolling a dice. However, Ω can be any set, not necessarily finite or enumerably infinite. In a few pages, when defining probability over words (or n-grams), the set Ω will correspond to the set of words (or of n-grams).
2. A subset A of Ω is called an *event.* Events (as any subset) can be singletons (containing a single elementary event), empty ("nothing happens"), or contain more elements ("either this or that outcome"; e.g., "throwing an even number"). Ω is also an event ("something happens").
3. **Probability distribution** P is defined as a *measure* on Ω that is normed to 1. For details, refer to maths books. In a nutshell, that means:
4. For each even A (each subset of Ω), $P(A)$ is a real number between 0 and 1, both endpoints of the interval included. That is, $P : \mathcal{P}(\Omega) \rightarrow [0, 1]$.
5. The measure (probability) of Ω (as a subset of itself) is 1 (that is, $P(\Omega) = 1$). This fact is referred to as the probability must be normed to 1.
6. The measure of the empty set is $P(\emptyset) = 0$. ("The probability of nothing happening when something happens in an experiment is 0".) Note however that other events might also have zero probability, such as impossible events. In a continuous set, even possible events may have zero probability. Similarly, proper subsets of Ω may also have a probability of 1.
7. Events A and B are *disjoint* if their intersection is empty ($A \cap B = \emptyset$): they cannot occur simultaneously, no "outcome of the experiment" (element of Ω) can belong to both. In such a case, $P(A \cup B) = P(A) + P(B)$ must hold. (Writing $A \cup B$ corresponds to the set theoretic approach to events. But events can also be viewed as propositions, and so I could have used logical disjunction, $A \vee B$, to refer to the complex proposition of either A or B happening. The latter notation is used more often.)
8. It follows that for any two events, $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.
9. Two events are *independent* if $P(A \cap B) = P(A) \cdot P(B)$.

10. *Conditional probability* $P(A|B)$ is the probability of event A given that event B has occurred. It is defined as $P(A|B) = \frac{P(A \cap B)}{P(B)}$. If A and B are independent events, as defined above, then the probability of A remains the same even if we are told that event B has occurred: $P(A|B) = P(A)$.
11. It also follows that $P(A \cap B) = P(B) \cdot P(A|B)$. This means that the probability of events A and B occurring simultaneously can be calculated as the probability of B occurring multiplied with the probability of 'A given B'. This idea will help understand Markov Models and their friends.
12. Since both $P(A \cap B) = P(B) \cdot P(A|B)$ and $P(A \cap B) = P(A) \cdot P(B|A)$ hold, it follows that

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

This fact, known as **Bayes' theorem**, plays a central role in contemporary computational linguistics.

The last few items have been added to this list so that you have a relatively complete summary (or refresher) of the basics of probability theory. Familiarity with them is essential for the rest of the course. In order to understand how this measure-theoretic approach defines the probability of obtaining a 6 with a dice, we return to the earlier part of the list:

> When rolling a dice, there are six elementary outcomes. Therefore, let $\Omega = \{1, 2, 3, 4, 5, 6\}$. The dice being fair (not loaded) means that $P(\{x\}) = P(\{y\})$ for any two elements $x, y \in \Omega$. In words: any two sides of the dice are associated with the same probability. If $x \neq y$, then $\{x\} \cap \{y\} = \emptyset$, and so by the axioms above $P(\{x, y\}) = P(\{x\}) + P(\{y\})$. Similarly, $\sum_{x=1}^{6} P(\{x\}) = P(\Omega)$. Since the axioms also require that $P(\Omega) = 1$, it follows that $P(\{x\}) = 1/6$ for any elementary outcome $x = 1, 2, \ldots 6$.

Please compare this train of thought to the earlier, frequentist argument at the beginning of this subsection. How do they relate to each other?

We can ask how often we expect to obtain a 6 if we repeat the experiment (throw the dice) n times. In order to answer this question, we suppose that experiments are *independent* of each other, as defined above: the probability of throwing a 6 in experiment k is unaffected by the outcome of experiment j. We also have to define in mathematical terms what the "expected number of" 6s is. Having done so, we can show that out of n independent experiments, the expected number of experiments with an outcome of 6 is $n/6$. This is the point where the measure-theoretic approach finally meets the frequentist approach.

2.2 What about computational linguistics?

Why is this digression relevant to computational linguistics? Because the two approaches to the notion of probability are very often confounded.

Since Noam Chomsky in the late 1950s, many (theoretical) linguists have criticized the probabilistic approach to language: for instance, in what sense do we speak about the ‘probability’ of a sentence or of a document? Certain sentences (*I was born in New York*) might be more frequently uttered than other sequences (*I was born in New Haven*), and yet, they do not differ in grammaticality. Moreover, what is the “experiment” in which we consider the probability of that sentence? What is the “corpus” in which we consider the frequency of that sentence?

And yet, contemporary computational linguistics could not be imagined without the notion of probability. In my interpretation, this notion is not used by computational linguists with a frequentist interpretation, but as a mathematical construct (as defined by the measure-theoretic approach). The abstract mathematical tool assigns different **scores** to different possible parses, sentences, documents, etc. The goal being to choose the best parse, etc., this score drives our choice. In a moment, our task will be to choose between language tags for a given document, and so we shall choose the tag that results in the highest score. But since the score is based on probability theory, we happen to call that score *probability*—maybe a misnomer. In fact, the score thus obtained is often referred to as **likelihood** instead, to avoid frequentist interpretations.

However, the story is more complicated than that. Even when we argue that the scores thus calculated must not be understood as ‘probability’, possibly with a frequentist connotation, the calculation will be based on a frequentist approach: the data entering the computation are corpus frequencies, and the way we combine them into a score is again based on such a perspective.

Let us now turn to an example.

3 Text classification with word frequencies

3.1 Machine learning: recap with some new remarks

To recapitulate, the task of text categorization, as an example of **machine learning**, can be summarized as follows:

> X is the set of possible documents, Y is the set of possible tags (henceforth, languages, but it could also be genres, authors, topics, etc.). We are given a *training set* of n documents ($x_i \in X$), each coming with its correct label ($y_i \in Y$). Our task is to develop a method that will assign a label to any (possibly unseen) document (in X). This method should be as correct as possible, as compared to the *golden standard* (e.g., human judgement) on a *test set* (usually disjunct from the training set).

Note two features of this summary, which will be gradually given up in techniques we shall learn about later in the course. First, now we suppose that the labels provided for the training set are correct, that is, they correspond to the **golden standard**. In fact, data may also be **noisy**, and more advanced techniques in machine learning take into account the fact that some of the y_is may be erroneous, different from what we expect the system to return.

Second, now we focus on *supervised learning*, that is, each x_i comes with some y_i. In **unsupervised learning**, however, learning data are unlabeled. As we shall soon see it in the case of Hidden Markov Models, the learner will not only have to assign labels to unseen data, but it also will have to find the most probable y_i for each x_i among the learning data (based, for instance, on the statistical patterns observable in the representative learning data).

Very often, the solution will be iterative: given some hypothesis, the learner assigns labels to the learning data, and then, these (data, label) pairs serve as learning data to revise the hypothesis. Such a so-called **bootstrapping** technique is also very often conjectured to drive child language acquisition. Children, in fact, face several chicken-or-egg problems, such as: how to learn syntactic structures without reference to the meaning of the learning data, and how to derive meaning without being able to recover syntactic structures?

3.2 Scores for comparing documents

Suppose that the set X contains documents in English, French and Spanish. The training set is a subset of X, also containing texts in all three languages. One possible solution to the text classification problem is to use the training set for creating a *language model* for each of the three languages. In this context, I shall keep referring to the "language L section of the training corpus".

We should pose ourselves the following three questions:

1. Each "language model" should contain relevant *features* of that language. What features should we consider? Given that both French and Spanish use special characters not used in the other languages, relevant features for

the current task may be character frequencies. Other feature, maybe more useful for other tasks, include word frequency, character bigram frequency, word trigram frequency, etc. By the way, why not consider punctuation mark frequencies? (Remember Spanish ¿ and French ≪ guillemets ≫.)

2. How do we want to compare an (unseen) document to a language model learned from the training set? What "scores" should be introduced to "measure" the similarity or distance of the document to be classified to each of the language models?
3. How can we guarantee that language models do not reflect the training set, but the languages they ought to represent? A recurrent topic in machine learning literature is how to avoid what is known as **overfitting** or **overtraining**: a model that is too accurate in fitting the learning data in the training set may prove much less accurate in predicting new data. One may wish to reduce the system's performance on the training set, if that change may improve performance on the test set. (More on that in the context of *smoothing*).

For instance, a simple solution may be this: the features considered are the presence (not even the frequencies) of "funny" characters. If a document contains ¿ or ñ, then it is in Spanish. If it contains vowels with acute accents, grave accents, circumflexes and tremas, as well as the ç character, then it is in French. Otherwise, it is in English. Such a solution is called a *decision tree* in machine learning.

Yet, this proposed solution poses several problems: what if a text, for whatever reason, contains both an ñ and a ç? What if a Spanish or French text (e.g., a few keywords in the search field of a search engine) is too short to contain any special character? What if the languages to be compared do not contain so clear characteristics? And, finally, observe that this solution is based on a human's intuition observing a specific case. It is an *ad hoc* handcrafted solution, while the success of language technology in the last decade is due to, and motivates the automatic processing of large amount of data, and the portability of solutions to new situations (e.g., text categorization with many different languages).

So let us return to our favorite counting of n-grams (possibly unigrams; either of characters, or of words). A model of the language y (where $y \in Y$) will be a *frequency table.* Note that the most efficient way of implementing such a table is using a hash table (a *dictionary* in Pythonese), mapping each observed n-gram onto its count or frequency, and not requiring memory for unobserved n-grams. An alternative would be to define an n-dimensional array, for instance a $26 \times 26 \times 26$ matrix for English letter trigrams. While such an array can easily be handled by 21st century computers, one can easily imagine what its size would be for 5-grams of words. However, most of the cells in this huge table would be zero anyway, either because a combination is impossible (word n-gram prohibited by syntax, character n-gram prohibited by phonotactics), or because this combination simply happens not to occur in our corpus with restricted length. At this point emerges what is called the **sparse data problem**: how do we know whether an n-gram combination has 0 observed frequency because it is impossible, or because of coincidence? In the former case, the language model should maintain the 0 value. In the latter case, however, the language model, which is meant to generalize beyond the training corpus, should allow for a low frequency.

Depending on what we use the frequencies for, such a divergence between the ideal language model and the language model derived from our specific training set may have or not have serious consequences. Earlier in class we discussed the *cosine measure* for similarity, an approach that is not very sensitive to having a 0 instead of a low frequency for some words in the reference language models. In this case, the models of the languages were unit vectors created from the frequencies observed in the training corpus. The document to be classified is also transformed into a unit vector. By taking the dot product of the document vector with each reference vector, we obtain a measure of similarity. This score being a value between 0 and 1 looks *as if it were* a probability value: if it is 1 or close to it, "it is very probable" that the document is written in the language modelled by the reference vector, and if the score is 0 or close to it, then the same is "extremely improbable". And yet, nothing (to my knowledge) in the mathematical construct of the cosine measure warrants such a probabilistic interpretation: the probability of what experiment should it be?

3.3 A new measure (to be called *likelihood* later)

Now, let me suggest a different similarity measure, replacing the cosine measure. Let c_i denote the count (absolute frequency) of the word (n-gram) w_i in the training corpus (that is, in the language L section of the training corpus). Here w_i is a type, with c_i tokens in the corpus. Then, we calculate $P(w_i|L) = c_i/N_L$, where N_L is the length of the language L section of the corpus (number of tokens). Please forget for a moment that the letter P is also used for probability. $P(w_i|L)$ simply stands for the relative frequency of type w_i in corpus L, a value that was already employed to build the vectors for the cosine measure. Now take the document to be classified, and replace each of its words (tokens) with the corresponding $P(w_i)$, and multiply these numbers. The product is a number between 0 and 1, because it is the product of numbers between 0 and 1.

Why would this product give us another measure of similarity between the language L section of the training corpus and our new document? A "real" measure of similarity would return a value close to 1 for high similarity, and close to 0 for low similarity; while this is clearly not the case for our new measure. The score thus obtained will be extremely low. The longer the document to be classified (that is, the more $P(w_i|L)$ to be multiplied), the lower the product. If you copy-paste your document twice, the measure of this doubled document will be the square of the original document, while you probably do not want to change its similarity to the language model.

It is however true that a given document will receive its highest score for the language L it resembles the most. When you calculate this score for your document D, you have one factor in the product for each word (token) of D. If you perform this computation for different languages, the $P(w_i|L)$ values will change, but not the number of factors in the product. In order to get a higher score, you have to multiply higher numbers. Consequently, the score is higher if document D contains words that are frequent in the corpus of language L.

Is the value returned by this second measure more informative than the value of the cosine metric? We have just seen it is not. Does text classification works better with this second metric than with the cosine metric? Maybe, I have not tried it out. Why have I introduced this second metric, then? Because, unlike the cosine metric, it can be interpreted as probability.

3.4 Creating a document from a bag of words

As you might have guessed, $P(w_i|L) = c_i/N_L$ is a probability. Imagine we cut out the words in the language L section of the training corpus and we throw them into a bag. There will be c_i copies of the word w_i in this bag, and the total number of words will be $\sum_i c_i = N$, the total number of words in the corpus. Drawing a word from the bag at random, the probability of word w_i is c_i/N.

Now take a document D of length n, a sequence of words $d_1, d_2, \ldots d_n$. Here d_ks are tokens, and maybe d_k and d_l belong to the same type w_i. Document D can be the original corpus out of which the content of the bag was created, or it can also be the document to be classified. Then, imagine the following experiment: we draw a word from this bag, write down that word, return it to the bag, and we repeat this procedure n times. What is the probability that the text thus obtained is D? The answer is the product of the $P(w_i)$'s:

$$P(D|L) = \prod_{k=1}^{n} P\left(t[d_k]|L\right) \tag{1}$$

where $t[d_k]$ is the type of the token d_k. If d_k is the word token `apple` (or the character trigram `abc`), then $t[d_k]$ is the type *apple* (or the trigram *abc*), one of the possible w_is, and $P(w_i|L)$ provides its relative frequency in the language L section of the training corpus.

Notation $P(D|L)$ sounds like "the probability of document D, provided we employ language L". To be more precise: the probability of producing D using the random procedure described above, if the bag-of-words has been created with the language L section of the corpus. It must never be interpreted as "the probability that document D is written in language L". And yet, this is the interpretation we have intuitively in mind [2] when we suggest a text classifier

2. A Baysian approach, to be discussed later in the course, can clarify our intuition. Our question is: which language is the most probable guess for document D? In other words, our aim is to maximize $P(L|D)$, the probability of language L, given document D. Bayes' theorem, introduced earlier in this note, suggests that $P(L|D) = P(L) \cdot P(D|L)/P(D)$. Obviously, we do not know what the probability of a language and the probability of a document are. Yet, D is given, and so $P(D)$ cannot be varied in order to maximize $P(L|D)$. Moreover, we suppose that the so-called *prior* probabilities $P(L)$ are equal for all languages: we have no reason to suppose *a priori* that a document is more likely to be written in a certain language than in another language. Consequently, maximizing $P(L|D)$ is equivalent to maximizing $P(D|L)$. Yet, the latter is easier to compute, and so we aim to maximize the latter, not the former.

based on this measure ("classify D as L_1, if $P(D|L_1)$ is greater than $P(D|L)$ for any other L" [3]).

Focusing on a single language L, the values $P(w_i)$ can be seen as a *probability distribution* on vocabulary $\mathcal{V}$. Vocabulary $\mathcal{V}$ is a set of words (word types) that contains all the words in the training set, and possibly also further ones (unseen words, which might show up in the test set). Indeed, $c_i \leq N_L$, and so $0 \leq P(w_i) \leq 1$. Moreover—and this is the real test for anything being claimed to be some sort of probability!—they sum up to 1:

$$\sum_{w \in \mathcal{V}} P(w) = \sum_{w \in \mathcal{V}} \frac{\text{\# of tokens of } w \text{ in training corpus}}{\text{total \# of tokens in traing corups}} = 1$$

That is, the vocabulary $\mathcal{V}$ plays here the role of Ω in the general definition of a probability distribution. That is the set on which probabilities sum up to 1.

As a mathematical concept, $P(w)$ works as any probability should. Its frequentist interpretation, however, is problematic. The $P(w)$ values originate in "frequency", namely, in corpus frequency. It is certainly not *the* frequency of w in the language. The imaginary "drawing from a bag of words" experiment makes it comparable to the probabilities in the not so imaginary "throwing the dice" experiment. One might also speculate that the training corpus is representative for the entire language, and so $P(w)$ is a good approximation of the frequency of w in the entire language. But what is "the entire language"?

And in what sense is $P(D|L)$ a probability? It is the probability of producing D in this imaginary experiment. Summing up $P(D|L)$ for all the outcomes D of this imaginary experiment—all texts of length n—will result in 1. But this is quite an artificial story. The sum of $P(D|L)$ for all texts (of any length) in language L is the sum for the texts of length 1 plus the sum for the texts of length 2, and so on, resulting in infinity. On a different note, summing up the $P(D|L)$ of a specific D for all possible languages will give an arbitrary number, depending on how many languages are included.

To sum up, the score $P(D|L)$ is very useful for determining how a language model derived from the language L section of the training corpus fits document D. It is also "probability-like": derived from frequencies, it can be understood as the probability of an outcome in an imaginary and artificial experiment. It fits into a mathematical framework built of concepts in probability theory. It is, however, not *the* probability of document D in any realistic sense. That is the reason why this measure is often called **likelihood**, and not probability.

3.5 The *likelihood* of a text

We now tackle the topic from a different angle. Let $\mathcal{V}$ be a vocabulary. (It could also be an alphabet, if the units are characters and not words. But forget about $(n > 1)$-grams for a while.) We introduce a probability P on $\mathcal{V}$ (that is, $\mathcal{V}$ plays the role of Ω). We can use any probability distribution, as long as

3. It is extremely unlikely that two different languages return the same score for D.

$$\sum_{w \in \mathcal{V}} P(w) = 1 \tag{2}$$

Let D be a text, a series of tokens of words from the vocabulary. Let c_i denote the count (number of tokens) of the word $w_i \in \mathcal{V}$. The value $c_i = 0$ corresponds to a word in the vocabulary that does not occur in D. The length of D is $N = \sum_i c_i$. If we denote the size of $\mathcal{V}$ as V (*vocabulary size*), then the indices i are the integer elements of the $[1, V]$ interval. Note that we do not need any reference to the tokens (denoted as d_i earlier), nor to their order in D.

Then, the **likelihood** of text D for the probability distribution P is defined as the product of the Ps of all tokens in D. By grouping the c_i tokens of the same type w_i together, we turn eq. (1) into

$$L(D, P) = \prod_{i=1}^{V} P(w_i)^{c_i} \tag{3}$$

Different probability distributions over $\mathcal{V}$ will result in different likelihood values for the same text D. Which is the distribution that maximizes the likelihood? As I shall demonstrate it in the next subsection, $L(D, P)$ is maximal if the probability distribution P reflects the frequencies in D. This is why the following probability distribution is called the **maximum likelihood estimate**:

$$P_{\mathrm{MLE}}(w_i) = \frac{c_i}{N} \tag{4}$$

This fact can be interpreted as follows: notwithstanding how low $L(D, P)$ is, the highest likelihood for a given text D is given by having P correspond to the frequencies in D.

Turning back to our text classification problem: the highest score (now called *likelihood*) for a given D is obtained when the language L section of the training corpus has exactly the same word frequencies as the text D to be classified. (As the absolute counts are irrelevant, the training corpus may be n times longer than D, and the order of the words do not matter either.) Interestingly, this is exactly the case when the cosine measure would return a similarity value of 1. If P originates in a text with a similar, though not equal frequency distribution to D, then the likelihood score will still be high, though not maximal. And if P originates in a training set representing a different language, then it will contain very different word frequencies, resulting in a much lower likelihood score.

To summarize, *likelihood* is a useful score to gauge how well P and D fit. It is derived from a probability distribution on a vocabulary—this probability distribution being a mathematical concept (measure-theoretic approach).[4] But likelihood is not the probability of D in any (frequentist) sense corresponding to anything in the "real world". Nor is likelihood a measure of similarity in the sense that full identity would result in a fix value (usually 1).

4. So far, this probability distribution on $\mathcal{V}$ corresponds to corpus frequencies in the Maximum Likelihood Estimate. In this sense, the connection to the frequentist approach is still there. Yet, this will not be true anymore for the smoothened probability distributions.

3.6 Appendix: Proving the Maximum Likelihood Estimate

Now we demonstrate that it is the probability distribution $P(w_i) = c_i/N$ that maximizes the likelihood $L(D, P)$ for a given D. The following proof will help some of you better understand the concepts, while other people can safely skip this subsection.

In this subsection, p_i will be used as a shorthand for $P(w_i)$. The likelihood of D given P becomes

$$L(c_1, \ldots, c_V; p_1, \ldots, p_V) = \prod_{i=1}^{V} p_i^{c_i} \tag{5}$$

Maximizing this function is equivalent to maximizing its logarithm. **Log-likelihood** is in fact very frequently used in computational linguistics:

$$LL(c_1, \ldots, c_V; p_1, \ldots, p_V) = \log L(c_1, \ldots, c_V; p_1, \ldots, p_V) = \sum_{i=1}^{V} c_i \log p_i \tag{6}$$

Our aim is to maximize LL, for a given combination of $c_1, \ldots, c_V$, and varying $p_1, \ldots, p_V$, subject to the following constraint:

$$\sum_{i=1}^{V} p_i = 1 \tag{7}$$

In order to find the (local) maximum of the function LL subject to equality constraint (7), we employ the method of *Lagrange multipliers.* The Lagrangian is

$$\Lambda(c_1, \ldots, c_V; p_1, \ldots, p_V; \lambda) = \sum_{i=1}^{V} c_i \log p_i + \lambda \cdot \left(\sum_{i=1}^{V} p_i - 1 \right) \tag{8}$$

While parameters c_i are fixed for a given text D, and the partial derivative in respect to the Lagrange multiplier λ returns the constraint, we have to make the partial derivatives in respect to each p_i equal to zero (suppose natural logarithm for the sake of simplicity):

$$\frac{\partial}{\partial p_i} \Lambda(c_1, \ldots, c_V; p_1, \ldots, p_V; \lambda) = \frac{c_i}{p_i} + \lambda = 0 \tag{9}$$

Hence, $p_i = -c_i/\lambda$ for each i. In order to satisfy constraint (7),

$$\sum_{i=1}^{V} p_i = -\sum_{i=1}^{V} \frac{c_i}{\lambda} = 1$$

we must have $\lambda = -\sum_i c_i = -N$, with N being the length of document D. Consequently,

$$p_i = \frac{c_i}{N}$$

holds in the local optimum of the log-likelihood function. One should now check that this is indeed a local maximum, and this function has no other local optima. Since logarithm is a monotonic function, the likelihood function also has its maximum at the same place.

4 Smoothing

To summarize, the **Maximum Likelihood Estimate** is the relative frequency of each word type in text D:

$$P_{\mathrm{MLE}}(w_i) = \frac{c_i}{N} \tag{10}$$

Given our training corpus (a language L section of the training corpus), these $P_{\mathrm{MLE}}(w_i)$ values can be used as a model of language L—either as the vector components in a vector space model with the cosine measure, or as the probability distribution for a likelihood measure. In order to categorize any document, we only have to calculate its cosine measure or likelihood measure to the reference language models, and the language that maximizes the measure will determine how to classify that document.

But we have not checked yet whether the **maximum likelihood estimate** indeed defines a probability measure on $\mathcal{V}$. In order to do so, we have to check that $\sum_i P_{\mathrm{MLE}}(w_i) = 1$. However, remembering that $\sum_i c_i = N$, the proof is straightforward.

Using the Maximum Likelihood Estimate, the likelihood of the training set itself will be maximal. However, our goal is not necessarily to maximize the "likelihood" of the training corpus being correctly categorized, but the "likelihood" (in an informal sense) of the test set. The language model derived from the training corpus should be a generizable model of the language, and not a model of the training corpus.

If a word in the vocabulary does not occur in the training corpus ($c_i = 0$ for some $w_i \in \mathcal{V}$), its MLE will be 0. It might, however, be a word in the language, and occur in the test set, even if very rarely. In the case of the cosine measure, if two vectors are "quite similar", but one has a zero component where the other has a small positive value, then their similarity may still be quite high. However, if likelihood is used, then a single word unseen in the training corpus will reduce the likelihood of the document to be classified to zero. [5] A zero likelihood score will be interpreted as the document *not* belonging to that language, resulting in a Type II error, a false negative.

In order to avoid such a Type II error, various **smoothing techniques** have been introduced. They slightly increase the $P(w_i)$ of the words not observed in the corpus, taking away this probability mass from other words, and using different strategies for doing so. Therefore, these smoothing techniques slightly reduce the likelihood of the training corpus with respect to the Maximum Likelihood Estimate (hereby possibly slightly increasing the chances of Type I errors), but they much more radically decrease the chances of Type II errors.

5. The MLE calculated from the training corpus gives $P(w_i) = 0$. The word count of w_i in the document to be classified gives a $c_i > 0$. Therefore, the product defining the likelihood will contain a factor $P(w_i)^{c_i} = 0$. Observe, however, that if the word does not occur in the document either ($c_i = 0$), then the corresponding factor will be $0^0 = 1$.

4.1 Laplace Smoothing

Laplace smoothing is introduced in section 4.5.1 of JM. The basic idea is to increase the count of each word in $\mathcal{V}$ by one, as if we appended the vocabulary to the training corpus. This way, unseen words become hapaxes, hapaxes turn into twice-occurring words, and so forth. For details, refer to the textbook.

The new probability distribution, replacing MLE, eq. (10) is:

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V} \tag{11}$$

In order to have a better grasp of this distribution, let us show that it indeed sums up to 1:

$$\begin{aligned} \sum_{i=1}^{V} P_{\text{Laplace}}(w_i) &= \sum_{i=1}^{V} \frac{c_i + 1}{N + V} \\ &= \sum_{i=1}^{V} \frac{c_i}{N + V} + \sum_{i=1}^{V} \frac{1}{N + V} \end{aligned} \tag{12}$$

Remembering that $\sum_{i=1}^{V} c_i = N$ and $\sum_{i=1}^{V} 1 = V$, we indeed get what we have hoped for:

$$\sum_{i=1}^{V} P_{\text{Laplace}}(w_i) = \frac{\sum_{i=1}^{V} c_i + \sum_{i=1}^{V} 1}{N + V} \tag{13}$$

$$= \frac{N + V}{N + V} = 1 \tag{14}$$

4.2 Good-Turing Smoothing

Introduced in section 4.5.2 of your textbook, **Good-Turing Smoothing** uses the following probability distribution:

$$P_{\text{GT}}(w) = (c + 1)\frac{N_{c+1}}{N N_c} \tag{15}$$

where c is the count (absolute frequency) of the word w, N is the corpus size and N_c is the **frequency of frequency** c (refer to JM for details).

Regarding the frequency of frequencies, observe that

$$\sum_{c} c \cdot N_c = N \tag{16}$$

This is true because of the following: c is the number of tokens of a word with frequency c, and there are N_c such words. Therefore, $c \cdot N_c$ is the count of all word tokens in the corpus that correspond to the types with frequency c. When summing up $c \cdot N_c$ for all possible frequencies c in (16), we count all word tokens in the corpus: first the hapaxes, then the tokens of the types with count 2, then the tokens of the types with count 3, and so forth.

Why does (15) make sense? JM proposes a background “philosophy”. But in order to really understand that formula, and to understand why the basic idea has not led to a slightly different formula, we would like to show that this is the approach that sums up to 1:

$$\sum_{w \in \mathcal{V}} P_{\mathrm{GT}}(w) = 1 \tag{17}$$

In order to do so, let us first sum up the probability of the words that occur c times in the corpus. There are N_c such words, and so the probability mass they represent is $N_c(c+1)\frac{N_{c+1}}{NN_c} = (c+1)\frac{N_{c+1}}{N}$.

Now, let us sum up $(c+1)\frac{N_{c+1}}{N}$ for all c, from $c = 0$ to $c =$ the frequency of the most frequent word in the corpus If $c =$ the frequency of the most frequent word in the corpus, then $N_{c+1} = 0$. Therefore, this sum equals to the sum of $c\frac{N_c}{N}$ for all c, from $c = 1$ to $c =$ the frequency of the most frequent word in the corpus. But remember (16), and so the sum will be equal to 1:

$$\sum_{c=0}^{\max}(c+1)\frac{N_{c+1}}{N} = \sum_{c=0}^{\max -1}(c+1)\frac{N_{c+1}}{N} = \sum_{c=1}^{\max} c\frac{N_c}{N} = \frac{N}{N} = 1$$

4.3 Summary

Given a training corpus in some language, we create a model of that language. While other types of models will be discussed later in the course, our current approach is a *bag-of-words model*: a probability distribution $P(w)$ over all words in the vocabulary $\mathcal{V}$. (An analogous approach is a probability distribution $P(\sigma)$ over an alphabet of characters Σ.) This approach is extremely naive from a linguistic perspective, but it reflects many lay people’s understanding of what a language consists of: a language is characterized by its words (and by their frequencies).

In a first approximation, $\mathcal{V}$ is the set of words (types) in the training corpus. In a second approximation, we allow some words of the language (included in $\mathcal{V}$) not to appear in the corpus by coincidence. These words will have a zero count in the corpus, but should not be excluded from the language. In other words, we estimate $P(w)$ using word frequencies in the corpus, as a first approximation; but then, we would like to revise it (“smoothen” it), so that it reflects the language, in general, and not the corpus only. Our goal is to avoid *overtraining* (a.k.a. *overfitting*), and we have seen two different ways of smoothing the $P(w)$ distribution.[6]

6. In a third approximation, we could include all words in all languages in $\mathcal{V}$. A word w is assigned $P(w) = 0$ even after smoothing, if it does not belong to that language. But can we differentiate between words belonging to the language but missing from the corpus, and words not belonging to the language, in an automatized, corpus-driven process? Moreover, foreign words—most frequently proper nouns, and occasionally code switching—are likely to appear in texts of many genres.

The various smoothened $P(w)$ distributions are probability distributions in the *measure-theoretic sense*: they are functions on some set (the role of Ω being played by $\mathcal{V}$) that take their values in the $[0, 1]$ interval and satisfy further criteria. The most important such criterion is that $\sum_{w \in \mathcal{V}} P(w) = 1$. However, the various versions of $P(w)$ have not much to do with a *frequentist view* of probability.

We could prove, using the toolkit of advanced mathematics, that the *Maximum Likelihood Estimate* happens to be the corpus frequencies. Smoothened probabilities are more remotely related to frequencies. True, among many others, we also use corpus frequencies when calculating them. And true, they are interpreted as better estimates for the frequency of the word in the language, in general. However, what does "in the language, in general" mean? Is GT or Laplace smoothing closer to the frequencies "in the language, in general"? I suggest that you view them as abstract mathematical constructs that are useful in various applications.

In this case, which smoothing technique should one use? There is no simple answer. At this point, your task is to understand the concepts, the challenges being raised and to know what kind of answers are available (and where to look them up). Whenever you face a specific problem (for instance, text classification), you will have to keep these challenges in mind and look for a solution using these concepts. In practice, you will either use the smoothing technique employed by most of your peers (as discussed in the literature that you will have been able to look up), or you will experiment with some alternatives and end up using the one producing the best results.

5 From n-grams to Markov Models

5.1 Intro to Markov Models—approach 1

Probably one of the major aims of computational linguistics can be summarized as *creating a computational (formal, mathematical) model of natural language (a.k.a. human language)*. Such a model is able to characterize each language in some sense, for instance by telling us how to generate a sentence or text.

If you arrive from linguistics, then characterizing languages is the goal of your life, and so computational linguistics should provide further tools for you. If you arrive from cognitive science, then you view the brain/mind as an information processing system, and so a computational characterization of language(s) will contribute to your understanding of the "software" in the human brain. If you are interested in developing new software, then computationally characterizing a language is a means toward solving some task. Please keep in mind all these three different approaches, when reading about any issue in CL.

A *finite-state automaton* was such a model, because it told us how to accept or generate a string belonging to the language it described. It was also able to characterize (to define) a class of formal languages. But the problem was that natural languages did not fall in this class.

The *bag-of-words model* is also certainly such a model. Each language is characterized by a probability distribution over a vocabulary. It also tells us how to generate a sentence: you randomly pick words from the bag, with the probabilities assigned to the words. Aside from probability, this procedure also defines a regular language, simply $\mathcal{V}^*$. The problem is that we know human languages are way more complicated than that.

A bag-of-words model is superior to an FSA model in that the former also takes probabilities into account. An FSA is, however, superior to a bag-of-words model in that an FSA can take into account the "interaction" between neighboring elements: what character (and what word) can follow what other character (or word)? An FSA has many limitations, though, and many of these will be overcome by CFGs (context-free grammars) in a week or two.[7] Now we turn to Markov Models, which combine the advantage of using probabilistic techniques with the advantage of being able to refer to local interactions. Markov Models can be seen as probabilistic extensions of FSAs,[8] and similarly, CFGs will later also be extended to probabilistic context-free grammars (PCFG).

7. It has been argued that natural languages should be characterized as belonging to a class broader than context-free languages, although narrower than the class of context-sensitive languages. For an introduction to these *mildly context-sensitive languages*, you are referred to the *Formal Foundations of Linguistic Theories* course (LING 224a/624a) by Robert Frank.

8. In fact, *Probabilistic Finite-State Automata* are defined slightly differently from Markov Models. But the difference is not significant.

5.2 Intro to Markov Models—approach 2

An important detail of the bag-of-words model was that each word is replaced to the bag before we pick a subsequent word. Consequently, each draw is supposed to be independent from the rest. The result of the first draw does not influence the probabilities in the second draw, etc. That is why we multiplied the probabilities of each word in a text D composed of the tokens $d_1, d_2, \ldots, d_k$:

$$P(D) = P(d_1, d_2, \ldots, d_k) = P(d_1) \cdot P(d_2) \cdot \ldots \cdot P(d_k) \tag{18}$$

Suppose event A is that the first draw returns d_1, and event B is that the second draw returns d_2. The independence assumption, as introduced at the beginning of this note, is that $P(A \wedge B) = P(A) \cdot P(B)$. Conditional probabilities are equal to the unconditional ones: $P(B|A) = P(B)$. Similarly, if A_j represents the event that we pick word d_j in draw j, then the complex event $A_1 \wedge A_2 \wedge \ldots \wedge A_k$ is the event that the experiment including k draws will return text D. The probability of this complex event can, in turn, be calculated as done in eq. (18), provided the independence assumption is true.

But let us be realistic! If there were a competition for the most absurd hypotheses ever taken seriously in the history of science, then this independence assumption would have a good chance to win.

So let us go to the other extreme. The event $A_1 \wedge A_2 \wedge \ldots \wedge A_k$ is a conjunction of $A_1 \wedge A_2 \wedge \ldots \wedge A_{k-1}$ and A_k. Therefore, by the definition of conditional probabilities,

$$P(D) = P(d_1, d_2, \ldots, d_k) = P(A_1 \wedge A_2 \wedge \ldots \wedge A_{k-1}) \cdot (A_k | A_1 \wedge A_2 \wedge \ldots \wedge A_{k-1})$$

or, simply

$$P(D) = P(d_1, d_2, \ldots, d_k) = P(d_1, d_2, \ldots, d_{k-1}) \cdot (d_k | d_1, d_2, \ldots, d_{k-1})$$

To calculate the probability of the text (whatever that means!), we first calculate the probability of the text without its last word, and then we multiply it with the conditional probability of appending the last word to this chunk. The probability of the last word d_k may be influenced by all previous words. This formulation allows for zero influence, as well as for a combined effect of syntactic, semantic, pragmatic and non-linguistic influences.

We can continue this train of thought: [9]

$$\begin{aligned}
P(d_1, d_2, \ldots, d_{k-1}) &= P(d_1, d_2, \ldots, d_{k-2}) \cdot (d_{k-1} | d_1, d_2, \ldots, d_{k-2}) \\
P(d_1, d_2, \ldots, d_{k-2}) &= P(d_1, d_2, \ldots, d_{k-3}) \cdot (d_{k-2} | d_1, d_2, \ldots, d_{k-3}) \\
&\text{etc.} \\
P(d_1, d_2) &= P(d_1) \cdot P(d_2 | d_1)
\end{aligned}$$

9. In fact, we should have added a "beginning-of-text" symbol before d_1. So $P(d_1)$ might not be the "probability" of d_1 in isolation, but the probability that a text begins with d_1. But let me ignore this detail here.

It follows that the probability of the text can be computed as a series or chain of conditional probabilities:

$$\begin{aligned} P(D) = P(d_1, \ldots, d_k) &= P(d_1) \cdot P(d_2|d_1) \cdot P(d_3|d_1, d_2) \cdot P(d_4|d_1, d_2, d_3) \cdot \\ &\quad \cdot \ldots \cdot P(d_k|d_1, \ldots, d_{k-1}) \end{aligned} \tag{19}$$

This is certainly a language model that is free of absurd assumptions. And yet, two problems still should be mentioned. The first question is what we mean by the probability of a document: e.g., how frequently it can be found in public libraries?? My answer to this question is that I do not know, but it is still a useful score in many applications. By trying to maximize the "probability" of the solution, we can develop solutions to tasks that will satisfy the users. Let us not call it *probability*, but rather *likelihood*. We are developing a mathematical framework that is based on probability theory (with a measure-theoretic foundation), and we ignore possible frequentist interpretations.

The second question is how to compute the conditional probabilities in eq. (19). I stop philosophizing about the meaning of these conditional probabilities, and I rather raise a pragmatic question: even if I do not know what they mean, I would like to compute them. One can think of two solutions. We either develop a theoretical framework providing those conditional probabilities, or we empirically rely on corpora.

The first solution is what people hoped for for decades in the second half of the twentieth century, but it turned out to be an unsolvable problem. We simply cannot include all syntactic and semantic constraints into a model providing those conditional probabilities, not to speak about pragmatic constraints, facts in the outside world, etc. Moreover, even if we were able to do so for a language, we still would need to handcraft solutions for other languages. These two reasons jointly motivated the field's move toward using large corpora in the last two decades, when such large corpora became available and computers became able to handle them.

So the second solution is to turn to large corpora, and use them to estimate these probabilities. First, by definition of the conditional probabilities, [10]

$$P(d_j|d_1, \ldots, d_{j-1}) = \frac{P(d_1, \ldots, d_{j-1}, d_j)}{P(d_1, \ldots, d_{j-1})}$$

And, second, we can use j-gram frequencies in the training corpus to estimate $P(d_1, \ldots, d_j)$ of all j-grams for all $j = 1, \ldots, k$. Consequently, we can easily calculate all necessary conditional probabilities. In fact, j-gram frequencies provide the *Maximum Likelihood Estimates* of $P(d_1, \ldots, d_j)$, whereas smoothing might be necessary to refine these estimates. [We will not enter details of smoothing n-gram frequencies. By now, you should understand the problem, and you should

10. Let event A be d_j in position j of D. Let event B be d_1 in position 1, d_2 in position 2,..., and d_{j-1} in position $j-1$ of D. Then, event $A \cap B$ is the event $d_1, d_2, \ldots, d_{j-1}, d_j$. Remembering that $P(A|B) = P(A \cap B)/P(B)$, you obtain the above equation.

also possess the skills to understand by yourself a train-of-thought similar to the one we have discussed earlier with respect to word frequencies.]

But wait a moment! We need an estimate of $P(d_1, \ldots, d_{j-1}, d_k)$ in order to calculate $P(d_k|d_1, \ldots, d_{j-1}, d_{k-1})$, which in turn will be used to calculate $P(D) = P(d_1, \ldots, d_{j-1}, d_k)$. But then, why on earth do we need all of this hocus-pocus to finally get $P(d_1, \ldots, d_{j-1}, d_k)$? Second, our corpus is extremely unlikely to contain a significant amount of tokens of these n-grams with very large n. Unless document D is part of the training corpus, it is actually improbable that the training corpus will contain any instances of these n-grams. It follows that we very quickly run into the **sparse data problem** as n grows.

Therefore, we need a different approach. Markov Models offer an assumption (the *Markov assumption*) that represents a compromise between the absurd independence assumption of the bag-of-words model and the absurd assumptionlessness of this latter approach. In an nth order Markov Model, we suppose that the probability of the next draw only depends on the outcome of the previous n draws:

$$P(d_j|d_1, \ldots, d_{j-1}) = P(d_j|d_{j-n}, \ldots, d_{j-1})$$

In the simplest case (first order Markov Model), only the outcome of the last draw influences the next draw:

$$P(d_j|d_1, \ldots, d_{j-1}) = P(d_j|d_{j-1})$$

In a second order Markov Model, the last two draws may influence the outcome:

$$P(d_j|d_1, \ldots, d_{j-1}) = P(d_j|d_{j-2}, d_{j-1})$$

Whereas our old good bag-of-words model is a zeroth order Markov Model:

$$P(d_j|d_1, \ldots, d_{j-1}) = P(d_j)$$

In a Markov Model, the chain equation (19) is simplified due to these last equations. The factors in the chain are simpler, and hence, simpler to calculate using n-grams with a small n-value. For instance, in a first order Markov Chain,

$$P(D) = P(d_1, \ldots, d_k) = P(d_1) \cdot P(d_2|d_1) \cdot P(d_3|d_2) \cdot P(d_4|d_3) \cdot \ldots \cdot P(d_k|d_{k-1})$$

Using either the Maximum Likelihood Estimates or some kind of smoothing, unigram and bigram frequencies will provide some estimates for $P(d_j)$ (including $P(d_1)$), and for $P(d_j, d_l)$. Subsequently, we obtain the conditional frequencies by the formula

$$P(d_j|d_{j-1}) = \frac{P(d_j, d_{j-1})}{P(d_{j-1})}$$

Simple, isn't it? Two questions are raised, though: (1) is this an adequate model of natural language, and (2) can we nevertheless make use of it in NLP? The answers will be a definite no (cf. Chomsky 1957) and a definite yes, respectively.

This is the point where the course continues next week.