Words, tokens, *n*-grams and Language Models

J.-C. Chappelier

Laboratoire d'Intelligence Artificielle
Faculté I&C

Objectives of this lecture

- Where to start NLP processing chain from? Words?
- *n*-gram models
- Example usage of *n*-grams: Language Identification
- Out-of-Vocabulary forms

Lexical level

What is the input of a NLP system? Where to start from?

☞ it's a sequence of characters

Characters however seems a bit too low-level to play the role of the atomic constituents of the language

☞ lack of generalization

What should then be the atomic entities of NLP?
What should basic core information be related to?

☞ This is a difficult question! (Still open?)

(phonological words? syntactic words? concepts?)

However, a general agreement is to focus on **words**.

It's precisely the role of the **lexical level**:
first to identify, and then associate required information with the words.

What is a word??

The notion of "correct word" is difficult to define, especially out of context/application:

"*credit card*", "*San Fransisco*", "*co-teaching*": 1 or 2 words?

Is "*John's*" from "*John's car*" one single word?
Or are they two words? Is "*'s*" a word?

Similarly, what about "*I'm*", "*isn't*", ...?

And it's even worse for languages having *aglutinative morphology* (e.g. German), or languages without explicit delimiter (e.g. Thai).

And what about: "*I called SC to ask for an app.*", or "*C U*"

☞ definition of words **depends on the application/context**
Should carefully think about it!

If your goal is to build a lexicon as portable/universal as possible:
choose minimal *tokens* and let a *properly designed* **tokenizer** (or even further modules) glue these tokens in a way that fits each specific application.

J.-C. Chappelier

Word vs. tokens

Tentative definitions (may change here and there):

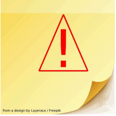
- **Word** (also sometimes called “**type**”): an element of the vocabulary;
  i.e. we *a priori* **define** what words have to be.
  Reminder: **depends on the application**!

- **Token**: ambiguously defined as (definition may vary):
  1. either
     a (continuous) sequence of non-separator characters
     - requires the definition of “separator”
     - is a separator a token in itself? (may vary)
     - does not fundamentally solve the former problems, only postpone them
  2. or
     either an instance of a type or a (continuous) sequence of non-separator characters
     - this confuses the problem even more.

We’d prefere to stick to definition 1 (and conceptually separate words from tokens).
Anyway: don’t bother so much about an (impossible?) absolute definition
but **be aware of the problem**!

Practice: *M. O’Connel payed $12,000 (V.T.A. not included) with his credit card.*

Key points



1. The notion of words is (inherently?) ambiguous
   and **depends on the application**.

2. Tokens are more useful in practice
   but may also **depend on the application**
   - ☞ **!!!** be sure all your NLP modules do indeed share the <u>same</u> definition of what tokens are!!!
     (otherwise, it's really a way to shoot yourself in the foot)

Language models

Back to start:
What is the input of a NLP system? Where to start from?

☞ it's a sequence of characters $\longrightarrow$ sequence(s?) of tokens $\longrightarrow$ sequence(s?) of words

How to choose among sequences (of characters/tokens)?
How to decide **which sequence is the best** (e.g. comparing two)?

Examples:

- language identification: *rendez-vous* vs. *gestalt* (*n*-grams of *characters*)
- spelling-error correction: *errro* vs. *error* (*n*-grams of *characters*)
- collocations: *real car wheel* vs. *real estate market* (*n*-grams of *tokens*)
- tokenization: *fullcapacitytocarryon* (coming from OCR) vs. *full capacity to carry on* (*n*-grams of *characters* / *n*-grams of *tokens*)

One approach: **probabilities**: *n*-grams of characters and *n*-grams of tokens
(for such an approach: "the best" = the more probable)

Notes:

- all modern neural NLP techniques actually focus on *n*-grams, estimating various kinds of related probabilities
- probabilization of *n*-grams of tokens a.k.a. "*language model*"

J.-C. Chappelier
EPFL

Probabilities: Notation (abuse)

X, Y, ..., X_1, ...: (discreate) random variables

x, y, ..., x_1, ...: values $\quad x \in X$: values for X

$P(x)$: same as $P(X = x)$ when X is clear by context

$P(X)$: distribution (set of all $P(x)$ for all $x \in X$)

(**Note:** for *continuous* variables, $P(X)$ denotes in fact the density function $\mathrm{d}P(X)$)

$P(x|y)$: same as $P(X = x|Y = y)$ when X and Y are clear by context

$P(X|y)$: distribution knowing $Y = y$ (set of all $P(X = x|Y = y)$ for all $x \in X$)

$P(X|Y)$: shouldn't make much sense

$P(x, y)$: same as $P(X = x, Y = y)$ when X and Y are clear by context, typically $P(X_1 = x, X_2 = y)$

Notice: $P(X_1 = x_1, X_2 = x_2)$ is truly the same as $P(X_2 = x_2, X_1 = x_1)$, whereas $P(x_1, x_2)$ is <u>not</u> the same as $P(x_2, x_1)$:
$P(x_1, x_2)$ is $P(X_1 = x_1, X_2 = x_2)$, whereas $P(x_2, x_1)$ is $P(X_1 = x_2, X_2 = x_1)$!

Probabilities: quick (and dirty) reminder

$$\sum_{x_1 \in X_1, \ldots, x_N \in X_N} P(x_1, \ldots, x_N) = 1 \qquad \text{(and } P(x_1, \ldots, x_N) \geq 0\text{)}$$

Additivity (a.k.a. marginalization): ($M < N$)

$$P(x_1, \ldots, x_M) = \sum_{x_{M+1} \in X_{M+1}, \ldots, x_N \in X_N} P(x_1, \ldots, x_M, x_{M+1}, \ldots, x_N)$$

Conditional probabilities: (for $P(y_1, \ldots, y_N) \neq 0$)

$$P(x_1, \ldots, x_M | y_1, \ldots, y_N) = \frac{P(x_1, \ldots, x_M, y_1, \ldots, y_N)}{P(y_1, \ldots, y_N)}$$

Note: thus $\displaystyle\sum_{x_1 \in X_1, \ldots, x_M \in X_M} P(x_1, \ldots, x_M | y_1, \ldots, y_N) = 1$

Chain rule:

$$P(x_1 \cdots x_N) = P(x_1) \cdot \prod_{i=2}^{N} P(x_i | x_1 \cdots x_{i-1})$$

Bayes' rule: (for $P(x) \neq 0$ and $P(y) \neq 0$)

$$P(x|y) = \frac{P(x) \cdot P(y|x)}{P(y)}$$

n-gram approach

Consider a sequence of xs (characters, tokens, ...)

Make use of $(n-1)$-order Markov assumption: $P(x_i|x_1\cdots x_{i-1}) = P(x_i|x_{i-n+1}\cdots x_{i-1})$

to end up with (S: size of the input):

$$P(x_1\cdots x_S) = P(x_1\cdots x_n)\cdot\prod_{i=n+1}^{S} P(x_i|x_{i-n+1}\cdots x_{i-1})$$

Use this value as a score to compare sequences ($n \geq 2$):

$$\frac{\prod_{i=1}^{S-n+1} P(x_i\cdots x_{i+n-1})}{\prod_{i=2}^{S-n+1} P(x_i\cdots x_{i+n-2})}$$

$\boxed{P(x_i\cdots x_{i+n-1})\text{: paramaters estimated on some corpus}}$

these are "the n-grams (probabilities)"

Reminder: $P(x_i\cdots x_{i+n-2}) = \sum_x P(x_i\cdots x_{i+n-2}\,x)$

n-gram approach: example

Assume $n = 3$ (trigrams):

$$P(errro) = P(err) \cdot P(r|rr) \cdot P(o|rr)$$
$$= P(err) \cdot \frac{P(rrr)}{P(rr)} \cdot \frac{P(rro)}{P(rr)}$$

$$P(error) = P(err) \cdot \frac{P(rro)}{P(rr)} \cdot \frac{P(ror)}{P(ro)}$$

Parameters: trigrams probabilities: $P(aaa)$, ..., $P(err)$, ..., $P(rro)$, ..., $P(zzz)$

Bigrams probabilities are simply sums of trigrams' : $P(ro) = \sum_x P(rox)$

Caveat!

Don't compare probabilities of sequences of different sizes!!

$P(x_1, ..., x_M)$ and $P(x_1, ..., x_N)$ usually DO NOT COMPARE ($M \neq N$)

They are in **two different** probabilized spaces:

$$\sum_{x_1 \in X_1, ..., x_N \in X_N} P(x_1, ..., x_N) = 1$$

for a given N: in fact, $P(x_1, ..., x_N)$ is a $P(x_1, ..., x_N | \text{size} = N)$

For instance, do not compare $P($*real estate*$)$, $P($*real estate market*$)$ and $P($*real estate market increase*$)$

Then how compare them if we have to?

☞ put (all of them) in a broader model in which they make sense

Note: we here made the assumption that $P(x_1, ..., x_{\max(N,M)})$ is not a decent "broader" model
(for instance that $P($*real estate market*$) = \sum_w P($*real estate market* $w)$ is of no interest for the considered application [since: the shorter, the higher])

Estimation (= model learning)

Where do the values $P(x_i \cdots x_{i+n-1})$ come from?

☞ from a learning corpus

Simplest estimate: **maximum-likelihood estimate**:

$$\widehat{P}(x_1 \cdots x_n) = \frac{\#(x_1 \cdots x_n)}{N_n}$$

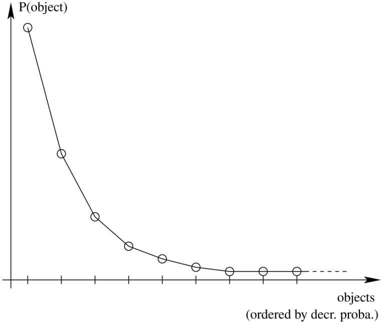
where $\#(y)$ is the count of y (= the number of times y appears in the corpus) and N_n is the size of the corpus = the total number of n-grams in that corpus:
$N_n = \sum_{x_1,\ldots,x_n} \#(x_1 \cdots x_n)$

Better estimates (1/2)

Maximum-likelihood estimates (MLE) are the simplest ones
but suffer from unseen events:
unseen rare events have a 0 frequency, thus a 0 probability MLE (☞ **overfitting**)

That could be OK in domains where the number of zeros isn't huge (e.g. maybe for categorization),
but is **not** for language modeling.

Reminder: power laws



☞ which 0 are "real zeros" and which ones are simply unseen, but possible, events?
Difficult question!

Better estimates (2/2)

Several approaches to better estimate unseen rare events (a.k.a **smoothing methods**):

- introduce a “prior” (a.k.a. “additive smoothing”)
  - ☞ leads to special cases known as *Lidstone smoothing*, *Laplace smoothing* or *add-one smoothing*
- add a new word (e.g. “`<UNKNOWN>`”) and estimate (held-out) probabilities accordingly
- backoff smoothing: fall-back on smaller n: increase the chance to observe events by decreasing the context-size
- interpolation: mix n-grams with $(n-1)$-grams, $(n-2)$-grams, etc.
  the mixing coefficients can be fixed or adaptative (learned on held-out data)
- Good-Turing smoothing: use the count of hapaxes (events seen only once) to improve estimates of probabilities of unseen events
- **Kneser-Ney smoothing**: considered as the **most-effective** for n-grams;
  it's a mixture of discounting and interpolation

☞ let's in-depth explain the first one

Additive smoothing (properly explained; 1/2)

n-grams is a probabilistic model, the parameters θ of which are the probabilities of the various *n*-grams (i.e. θ is a constrained vector of dimension $D = |X|^n$, with $|X|$ the number of possible values for X)

A *partially* Bayesian view on learning θ from a corpus $\mathscr{C}$ leads to estimating θ as:

$$\widehat{\theta} = \operatorname*{argmax}_{\theta} P(\theta|\mathscr{C}) = \operatorname*{argmax}_{\theta} P(\theta)P(\mathscr{C}|\theta)$$

$P(\mathscr{C}|\theta)$ (the likelihood of a corpus, represented here as a "bag-of-*n*-grams", i.e. by its *n*-grams counts) follows a multinomial law (the parameters of which are θ).

It's conjugate prior is the Dirichlet distribution; so let's model $P(\theta)$ by a Dirichlet distribution (it's indeed a probability density function on probabilities!):

$$P(\theta|\alpha) = \Gamma(\sum_{i=1}^{D} \alpha_i) \cdot \prod_{i=1}^{D} \frac{\theta_i^{\alpha_i - 1}}{\Gamma(\alpha_i)} \qquad (\alpha_i > 0)$$

where $\Gamma()$ represents the "gamma function".

Additive smoothing (properly explained; 2/2)

Thus the posterior $P(\theta|\mathscr{C})$ is itself a Dirichlet distribution, which is maximized (MAP) for

$$\widehat{P}(x_1 \cdots x_n) = \frac{\#(x_1 \cdots x_n) + \alpha_{x_1,\dots,x_n} - 1}{N_n + \left(\sum_{x_1,\dots,x_n} \alpha_{x_1,\dots,x_n}\right) - D}$$

In a "more Bayesian view", however, the *expected* value of $\theta_{x_1 \cdots x_n} = P(x_1 \cdots x_n)$ (under *posterior* Dirichlet distribution) is:

$$\widetilde{P}(x_1 \cdots x_n) = E_{\theta|\mathscr{C},\alpha}[\theta_i] = \frac{\#(x_1 \cdots x_n) + \alpha_{x_1,\dots,x_n}}{N_n + \sum_{x_1,\dots,x_n} \alpha_{x_1,\dots,x_n}}$$

and moreover (predictive distribution):

$$P(x_1 \cdots x_n|\mathscr{C},\alpha) = E_{\theta|\mathscr{C},\alpha}[\underbrace{P(x_1 \cdots x_n|\theta)}_{=\theta_{x_1 \cdots x_n}}] = \widetilde{P}(x_1 \cdots x_n) = \frac{\#(x_1 \cdots x_n) + \alpha_{x_1,\dots,x_n}}{N_n + \sum\limits_{x_1,\dots,x_n} \alpha_{x_1,\dots,x_n}}$$

Example (bigrams among two letters)

$X = \left\{\mathtt{a}, \mathtt{b}\right\}$, $n = 2 \longrightarrow D = |X|^n = 2^2 = 4$:

$$\theta = \Big(P(\mathtt{ab}), P(\mathtt{ba}), P(\mathtt{aa}), P(\mathtt{bb})\Big)$$

Consider $\mathscr{C} = \mathtt{ababaababababaabab} = \Big\{ (\mathtt{ab}, 7), (\mathtt{ba}, 6), (\mathtt{aa}, 2), (\mathtt{bb}, 0) \Big\}$

MLE:

$$P(\mathtt{ab}) = \frac{7}{15} \qquad P(\mathtt{ba}) = \frac{6}{15} \qquad P(\mathtt{aa}) = \frac{2}{15} \qquad P(\mathtt{bb}) = 0$$

Predictive distribution with uniform Dirichlet prior $\alpha_i = 0.5$ for all $i \in \Big\{ \mathtt{ab}, \mathtt{ba}, \mathtt{aa}, \mathtt{bb} \Big\}$:

$$P(\mathtt{ab}|\mathscr{C}, \alpha) = \frac{7.5}{17} \qquad P(\mathtt{ba}|\mathscr{C}, \alpha) = \frac{6.5}{17} \qquad P(\mathtt{aa}|\mathscr{C}, \alpha) = \frac{2.5}{17} \qquad P(\mathtt{bb}|\mathscr{C}, \alpha) = \frac{0.5}{17}$$

Additive smoothing = Dirichlet prior

So additive smoothing techniques

$$P(x_1 \cdots x_n | \mathscr{C}, \alpha) = \frac{\#(x_1 \cdots x_n) + \alpha_{x_1,\ldots,x_n}}{N_n + \sum_{x_1,\ldots,x_n} \alpha_{x_1,\ldots,x_n}}$$

result from a Bayesian predictive distribution with a Dirichlet-prior assumption:

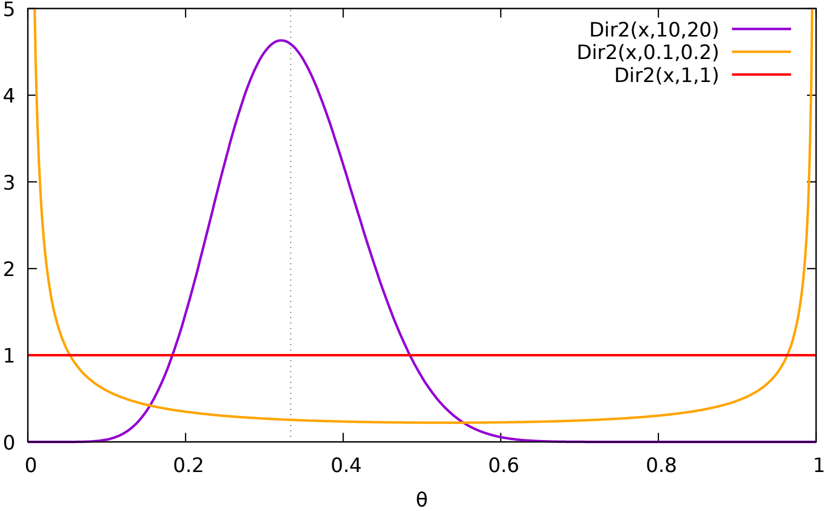
- MLE : "$\alpha_i = 0$" (not possible in this framework)
- $\alpha_i = 1$: "Laplace smoothing", a.k.a. "add-one smoothing"
  - ☞ **don't use** that for linguistic corpora (see next slides and reference [7])
- $\alpha_i < 1$: makes sense with power laws (a priori θ lies "close to the borders")

But what does α_i actually represent (intuitively)?

The components of α represent the relative importance of each component of θ.
For α_i smaller than 1, the distribution tends to "sharply increase" (in other words, to discretize) to the maximum α_i values.
More details in appendix for those interested.

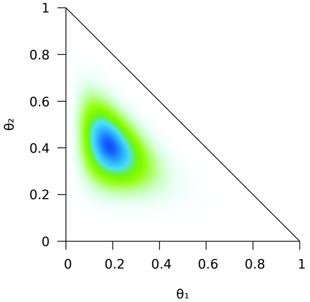
Examples of α parameter in 2D

For $D = 2$ (i.e. only 1 free parameter; $n = 1$, $|X| = 2$)

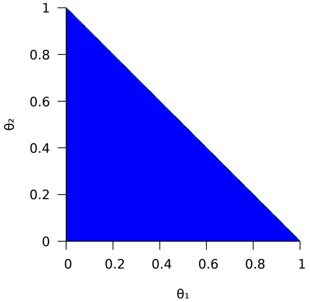


Examples of α parameter in 3D

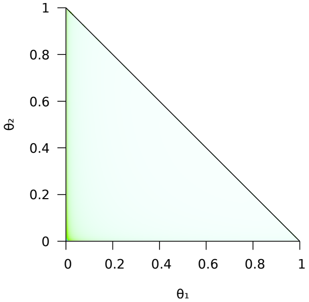
For $D = 3$ (i.e. 2 free parameters; $n = 1$, $|X| = 3$)



$\alpha = (6, 12, 12)$



$\alpha = (1, 1, 1)$



$\alpha = (0.6, 0.7, 0.8)$

J.-C. Chappelier

Example usage of *n*-gram: Language Identification

Example of *tasks* making use of language models: **Language Identification**

Goal: identification of the source language
Input: some textual document (or part of it)
Output: (name of the) language it was (mostly) written in

Two main techniques (which are combined):

- most frequent words
- Decomposition into *n*-grams of characters

Example: for trigrams

dribble $\rightarrow$ (dri,rib,ibb,bbl,ble)

In practice: *n* varies from 2 to 4

From reference corpora, estimate the likelihood of a word to belong to a given language.

Example for trigrams:

$$P(\text{dribble}|L) = P(\text{dri}|L) \cdot \frac{P(\text{rib}|L)}{P(\text{ri}|L)} \cdot \ldots \cdot \frac{P(\text{ble}|L)}{P(\text{bl}|L)}$$

Trigrams for French, English, German and Spanish: $\simeq$ 90% discrimination accuracy

Likelihood vs. Posterior probability

In the former slide, why make use of the **likelihood** $P(w|L)$ rather than the **posterior probability** $P(L|w)$?

- They are **both hard** to accurately model without any further assumptions (w belongs to a huge set!)

  but no further simplification can be made on $P(L|w)$: w is fixed (and there is nothing to gain "simplifying" L!)

  On the other hand, $P(w|L)$ can be further simplified making assumptions on w
- Using the Bayes' rule:

$$\underset{L}{\text{argmax}}\, P(L|w) = \underset{L}{\text{argmax}}\, P(w|L) \cdot P(L)$$

  ☞ introduces the likelihood *anyway*! (which could then be simplified further)
- **If you can accurately estimate $P(L)$, sure, make use of it!**
- Otherwise, the least biaised hypothesis (maximum entropy) is to *a priori* assume that all languages are **all equally possible**: maximizing posterior probability is then the same as maximizing likelihood

Out of Vocabulary forms

Out of Vocabulary (OoV) forms matter:
they occur quite frequently (e.g. $\simeq$ 10% in newspapers)

What do they consist of?

- spelling errors: *foget*, *summmary*, *usqge*, ...
- neologisms: *Internetization*, *Tacherism*, ...
  - ☞ morphology (and may also later become part of the lexicon)
- borrowings: *gestalt*, *rendez-vous*, ...
  - ☞ shall be included in the lexicon at some point
- forms difficult to exhaustively lexicalize:
  (numbers,) proper names, abbreviations, ...
  - ☞ ad-hoc regular expressions, Named-Entity Recognition

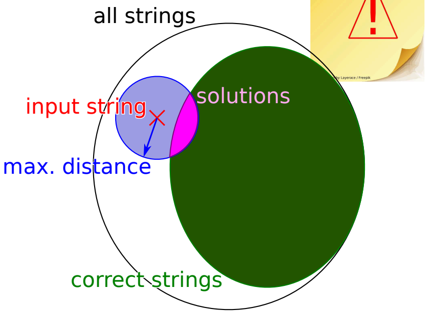
Spelling error correction

Input:

- incorrect form (OoV),
- lexicon,
- threshold (e.g. max. distance)

Output:

- all correct words ($\in$ lexicon) within threshold from input string



Two approaches:

	Exact lexicon-based	**Probabilistic**
correct forms:	lexicon	lexicon or any string (ordered with probabilities)
metric:	edit distance	probability

In this lecture:

- only a few words about the probabilistic approach

Probabilistic approach summarized (1/2)

Make (one more time!) use of n-grams (both levels, characters and tokens, are combined)

w: OoV token to be corrected (input string)

c: candidate correction, out of $\mathscr{C}(w)$, set of possible candidates for w

$$\underset{c \in \mathscr{C}(w)}{\operatorname{argmax}} P(c|w) = \underset{c \in \mathscr{C}(w)}{\operatorname{argmax}} P(c) \cdot P(w|c)$$

$P(c)$: language model (**n-grams of tokens/words**; $n = 1$ here, but could easily be extended to neighboring tokens ($n > 1$ then))

$P(w|c)$: error model: edit distance and/or **m-grams of characters**

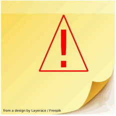
Probabilistic approach summarized (2/2)

A usual (unexplicit?) assumption is that $P(w|c)$ is many orders of magnitude higher for smaller edit distance (than for higher): thus closer candidate are considereds first, leading to this simple algorithm,
where $\mathscr{C}_d(w)$ is the set of candidates at distance d from w:

- if $\mathscr{C}_1(w)$ is not empty, return $\operatorname*{argmax}_{c \in \mathscr{C}_1(w)} P(c)$;
- (else) if $\mathscr{C}_2(w)$ is not empty, return $\operatorname*{argmax}_{c \in \mathscr{C}_2(w)} P(c)$;
- etc...

For more details: see `http://norvig.com/spell-correct.html`

Keypoints



- Tokenization may be difficult and should be properly designed/defined
- *n*-gram approach (both on chars and on tokens) is a really effective tool for many tasks
- Smoothing techniques for *n*-gram probabilities estimates

References

[1] C. D. Manning & H. Schütze, *Foundations of Statistical Natural Language Processing*, chapters 4.2, 5 and 6, MIT Press, 1999 (6th printing 2003).

[2] D. Jurafsky & J. H. Martin, *Speech and Language Processing*, chapters 2, 3 and 4, Prentice Hall, 2009 (2nd ed.).

[3] H. Ney, U. Essen and R. Kneser, *On structuring probabilistic dependences in stochastic language modelling*", Computer Speech & Language. 8 (1): 1–38, , jan. 1994.

[4] W. Gale & K. Church, *What's Wrong with Adding One?*, in N. Oostdijk & P. de Haan (eds.), Corpus-Based Research into Languge: In honour of Jan Aarts, pp. 189-200, Rodopi (1994).

[5] S. F. Chen & J. Goodman, *An empirical study of smoothing techniques for language modeling*, Computer Speech and Language 13, 359–394 (1999; first published in ACL proceedings in 1996).

Appendix: more about Dirichlet distribution (1/3)

A D-dimensional Dirichlet distribution parametrized by α (a D-sized vector, the components of which are all strictly positive) is a distribution over the simplex with dimension $D-1$ such that:

$$P(\theta|\alpha) = \Gamma(\sum_{i=1}^{D} \alpha_i) \cdot \prod_{i=1}^{D} \frac{{\theta_i}^{\alpha_i - 1}}{\Gamma(\alpha_i)}$$

The components of α represent the relative importance of each component of θ, the average point being $\overline{\theta} = \frac{1}{S}\alpha$.

Their sum $S = \sum_{i=1}^{D} \alpha_i$ (inversely) influences the variance around this average point:

$$\mathbf{Var}(\theta) = (\mathbf{diag}(\overline{\theta}) - \overline{\theta}\,\overline{\theta}')/(S+1)$$

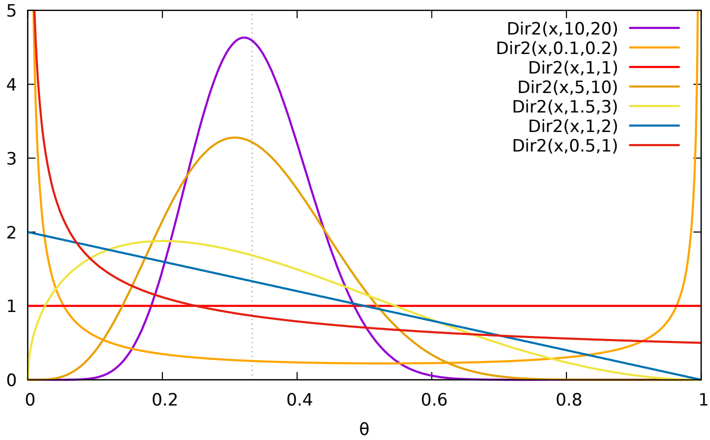
For (all) α_i bigger than 1, when *some* of the α_i approaches 1, the corresponding θ-components approaches 0 (unless all the α_i are equal to 1).
For α_i smaller than 1, the distribution tends to “sharply increase” (in other words, to discretize) to the maximum α_i values.
When α_i is larger than 1, the mode (i.e. the most probable point) is given by:

$$\widehat{\theta} = \frac{1}{S-D}(S\overline{\theta} - 1) = \frac{1}{S-D}(\alpha - 1)$$

J.-C. Chappelier
EPFL

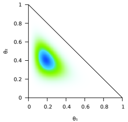
more about Dirichlet distribution (2/3)

Several probability densities of a single Dirichlet dimension ("beta law") corresponding to different parameters α: $(11, 22)$, $(5, 10)$, $(\frac{3}{2}, 3)$, $(1, 2)$, $(\frac{1}{2}, 1)$, $(\frac{1}{10}, \frac{1}{5})$, and $(1, 1)$. Note how the $S = \alpha_1 + \alpha_2$ parameter (inversely) influences the concentration of the probability density and how, when the components are lower than 1, the distribution tends to "sharply increase" at the edges.

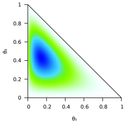


more about Dirichlet distribution (3/3)

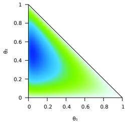
Several Dirichlet probability densities on the 2-simplex (smaller left triangle) corresponding to different α parameters. Bluer zones indicate higher values.
Note how the $S = \alpha_1 + \alpha_2 + \alpha_3$ parameter (inversely) influences the concentration of the probability density. It should also be noticed how when one of the α components approaches 1 the corresponding density tends to 0 and when the components are smaller than 1 the distribution “sharply increases” (on (0,0) in the bottom right figure, in other words concentrates on $\theta = (0,0,1)$).



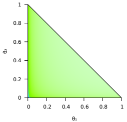
$$\alpha = (6,12,12) = 30\,(.2,.4,.4)$$
$$\widehat{\theta} = (.18,.41,.41)$$



$$\alpha = (2,4,4) = 10\,(.2,.4,.4)$$
$$\widehat{\theta} = (.14,.43,.43)$$



$$\alpha = (1.1,2.2,2.2) = 5.5\,(.2,.4,.4)$$
$$\widehat{\theta} = (.04,.48,.48)$$



$$\alpha = (0.6,0.8,0.99) = 2.39\,(.25,.33,.41)$$
$$\widehat{\theta} = (0,0,1)$$