ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

School of Computer and Communication Sciences
CS-431: Introduction to Natural Language Processing
Chappelier, J.-C. & Rajman, M.

CS–431: Introduction to Natural Language Processing

Exercises with solutions

(version 202310–2)

Contents

1 NLP levels

Exercise I.1

A company active in automatic recognition of hand-written documents needs to improve the quality of their recognizer. This recognizer produces sets of sequences of correct English words, but some of the produced sequences do not make any sense. For instance the processing of a given hand-written input can produce a set of transcriptions like: "*A was salmon outer the does*", "*It was a afternoon nice sunny*", and "*I Thomas at mice not the spoon*".

What is wrong with such sentences? NLP techniques of what level might allow the system to select the correct one(s)? What would be the required resources?

Solution

Those sentences are not "grammatically" (syntactically) correct. It should be filtered out at the *syntactic* level using a (phrase-structure) *grammar*.

2 Evaluation

Exercise II.1

① Give some arguments justifying why evaluation is especially important for NLP. In particular, explain the role of evaluation when a corpus-based approach is used.

② Many general evaluation metrics can be considered for various NLP tasks. The simplest one is accuracy.

Give several examples of NLP tasks for which accuracy can be used as an evaluation metric. Justify why.

In general, what property(ies) must an NLP task satisfy in order to be evaluable through accuracy?

③ Consider a Part-of-Speech tagger[1] producing the following output:

```
The/Determiner program/Noun can/Noun deal/Noun with/Preposition three/Number
types/Verb of/Preposition inputs/Noun ./Punctuation
```

(using your own knowledge of general English,) Compute the accuracy of the tagger.

What do you think of the performance of this system with respect to the State of the Art? Is this conclusion reliable?

④ What is the formal relation between accuracy and the error rate? In which case would you recommend to use the one or the other?

⑤ Consider the following "breaking news scanning system":
A company receives a continuous stream of information messages (newswires); each time a new message arrives, its average textual similarity score with respect to the stored collection of previously received messages is computed. If this average similarity is below a given threshold, the message is considered "breaking news" and is automatically distributed to the company personnel.

The company has carried out an evaluation of the system in place, which produced the following average figures:

- one message out of 1000 is considered to be "breaking news" by the system;
- 30% of the claimed "breaking news" messages are evaluated as not new by human judges;
- the system is missing one truly "breaking news" message every 1000 messages processed.

Use the provided figures to compute the accuracy of the system.

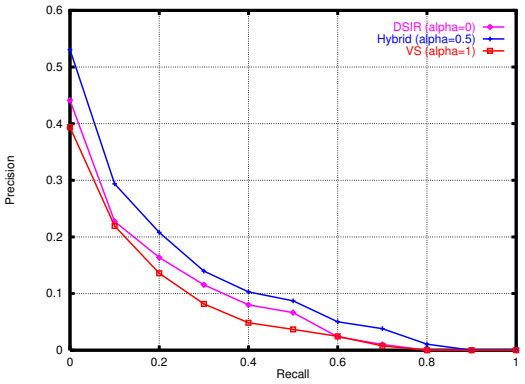
Is accuracy a good metric in this case? Justify your answer, and, possibly, propose some alternative performance score(s) and compute the corresponding value(s).

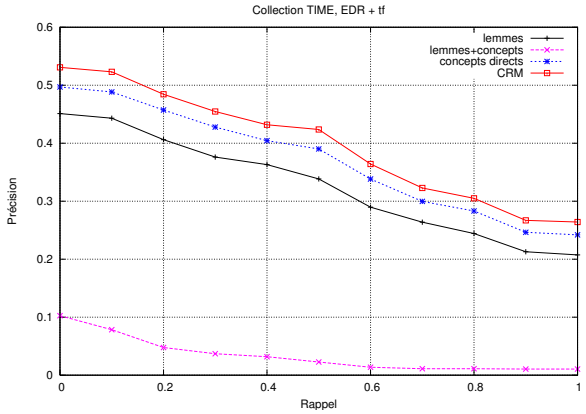
[1] Part-of-Speech tagging, which will be studied in more details later in the semester, consists in adding each word a ("Part-of-Speech") tag corresponding to its syntactic role within the sentence:.

⑥ Another very general evaluation framework concerns this kind of NLP tasks where the goal of the system is to propose a set of outputs among which some might turn to be correct, while other might not (e.g. Information Retrieval (IR)). In this type of situation, the standard evaluation metrics are the Precision and the Recall.

Give the formal definition of Precision and Recall and indicate some examples of NLP tasks (other than IR) that can be evaluated with the Precision/Recall metrics.

⑦ Consider the following Precision/Recall curves





What conclusions can one derive from such curves? Provide a detailed interpretation of the results.

⑧ It is often desirable to be able to express the performance of an NLP system in the form of one single number, which is not the case with Precision/Recall curves.

Indicate what score can be used to convert a Precision/Recall performance into a unique number. Give the formula for the corresponding evaluation metric, and indicate how it can be weighted.

⑨ Give well chosen examples of applications that can be evaluated with the single metric derived from Precision/Recall and illustrate:

- a situation where more weight should be given to Precision;
- a situation where more weight should be given to Recall.

Solutions

① a few hints:

- there is no theoretical proof nor unique optimal solution in NLP
- so as to have an objective (not subjective) quantitative (not qualitative) measure
- it helps clarifying, even specifying, the objectives to be reached
- allow to monitor variability over time (task shift, for whatever reasons, e.g. change in vocabulary)
- feed-back loop (propose clues where it can help the best)

② (a) PoS tagging, but also Information Retrieval (IR), Text Classification, Information Extraction. For the later, accuracy sounds like precision (but it depends on what we actually mean by "task" (vs. subtask)) .

(b) a reference must be available, "correct" and "incorrect" must be clearly defined

③ 7/10 (can, deal, types). It seems quite low for such a "simple" task and is indeed below State-of-the-Art. Any conclusion drawn from such a tiny example is not reliable at all.

④ (a) $err = 1 - acc$. (b) does not make any sense: they are the same (opposite, actually)

⑤ over 10'000 :

	OK ref	KO ref
OK Sys	7	3
KO Sys	10	9980

$$acc = \frac{7+9980}{10000} = 99.87\%$$

No, different priors and risks ; use recall : $\frac{7}{7+10} = 41\%$

⑥ see lecture/slides

⑦ explain what the axis are; the higher the curve the better, ideally (theoretical) top right corner, curves are decreasing by construction; left corpus is certainly much bigger than the right one (faster decreasing, very low recall).

⑧ F1, or any combination, e.g. weighted averages (F_β)

⑨ Precision is prefered when very large amount of data are available and only a few well choosen one are enough: we want to have those very early, e.g. Web search

Recall is prefered when have all the correct documents is important (implying that, if we want to handle them, they are not *that* many). Typically in legal situations.

Exercise II.2

You have been hired to *evaluate* an email monitoring system aimed at detecting potential security issues. The targeted goal of the application is to decide whether a given email should be further reviewed or not.

① Give four standard measures usually considered for the evaluation of such a system? Explain their meaning. Briefly discuss their advantages/drawbacks.

- accuracy / error rate / "overall performance": number of correct/incorrect over total number ; adv: simple ; drawback: too simple, does not take unbalancing of classes into account
- Precision (for one class): number of correctly classified emails over number of emails classified in that class by the system ; Ignores false negatives ; can be biaised by classifying only very few highly trusted emails

- Recall / true positive rate: number of correctly classified emails over number of emails classified in that class by experts (in the referential) ; Ignores false positives ; Can be biaised by classifying all documents in the most important class
- Area under ROC Curve ; Plot true positive rate vs false positive rates ; not easy to compute ;
- F score: Harmonic mean of precision and recall; balances P and R ; too simple: unary score for complex situation
- false positive rate

② For three of the measures you mentioned in the previous question, what are the corresponding scores for a system providing the following results:

email	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}
referential	C_1	C_1	C_1	C_1	C_1	C_1	C_1	C_1	C_2	C_2	C_2	C_2	C_2	C_2
system	C_1	C_2	C_1	C_2	C_1	C_2	C_1	C_1	C_2	C_1	C_2	C_1	C_2	C_2

The main point here is to discuss WHAT to compute: we don't know what neither C_1 nor C_2 are. So we have to compute either overall score (not very good) or scores FOR EACH class.

The confusion matrix is:

		system C_1	system C_2
reference	C_1	5	3
reference	C_2	2	4

from where we get: accurary=9/14, thus overall error=5/14

P/R for C_1: P=5/7 R=5/8

P/R for C_2: P=4/7 R=4/6

(note that "overall P and R" does not make any sense and are equal to accuracy)

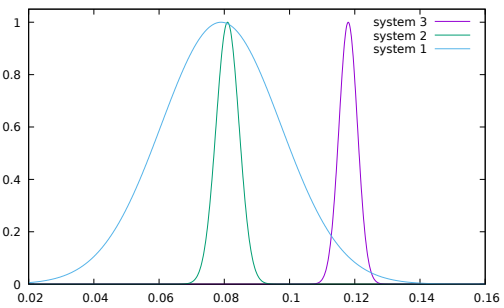
C_1: FPR = 2/7, FNR=3/7 (and vice-versa for C_2)

③ You have been given the results of three different systems that have been evaluated on the same panel of 157 different emails. Here are the classification errors and their standard deviations:

	system 1	system 2	system 3
error	0.079	0.081	0.118
std dev	0.026	0.005	0.004

Which system would you recommend? Why?

system 2: error is first criterion, then for *stastistically* ***non*** *signifiant* differences in error (which is the case for system 1 and 2), then min std dev is better (especially with such big difference as here!)



④ **Optional** (too advanced for the current version of the course)**:** What should be the minimal size of a test set to ensure, at a 95% confidence level, that a system has an error 0.02 lower (absolute difference) than system 3? Justify your answer.

We could consider at least two approaches here: either binomial confidence interval or t-test.

- binomial confidence interval: evaluation of a binary classifier (success or not) follow a binomial law with parameters (p_{error}, T), where T is the test-set size (157 in the above question; is it big enough?).

  Using normal approximation of the binomial law, the width of the confidence interval aroung estimated error probability is $q(\alpha)\sqrt{\frac{\widehat{p}(1-\widehat{p})}{T}}$, where $q(\alpha)$ is the $1-\frac{\alpha}{2}$ quantile (for a $1-\alpha$ confidence level) and $\widehat{p}$ is the estimation of p_{error}. We here want this confidence interval width to be 0.02, and have $\widehat{p} = 0.118$ (and "know" that $q(0.05) = 1.96$ from normal distribution quantile charts); thus we have to solve:

  $$(0.02)^2 = (1.96)^2 \frac{0.118 \times (1-0.118)}{T}$$

  Thus $T \simeq 1000$.

- t-test approach: let's consider estimating their relative behaviour on each of the test cases (i.e. each test estimation subset is of size 1). If the new system as an error of 0.098 $(= 0.118 - 0.02)$, it can vary from system 3 between 0.02 of the test cases (both systems almost always agree but where the new system improves the results) and 0.216 of the test cases (the two systems never make their errors on the same test case, so they disagree on $0.118 + 0.098$ of the cases). Thus μ of the t-test is between 0.02 and 0.216. And $s = 0.004$ (by assumption, same variance).

  Thus t is between $5\sqrt{T}$ and $54\sqrt{T}$ which is already bigger than 1.645 for any T bigger than 1. So this doesn't help much.

  So all we can say is that if we want to have a (lowest possible) difference of 0.02 we should have at least $1/0.02 = 50$ test cases ;-) And if we consider that we have 0.216 difference, then we have at least 5 test cases...

  The reason why these numbers are so low is simply because we here make strong assumptions about the test setup: that it is a *paired* evaluation. In such a case, having a difference (0.02) that is 5 times bigger than the standard deviation is always statistically significant at a 95% level.

4 Tokenization/Lexicons/*n*-grams

Exercise IV.1

According to your knowledge of English, split the following sentence into words and punctuation:

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

Which of these words won't usually be in a standard lexicon? Justify your answer.

Assuming separators are: whitespace, quote ('), full-stop/period (.), parenthesis,
and that separators a kept as tokens, tokenize the former sentence.

How would you propose to go from tokens to words? (propose concreat implementations)

Solution

words and punctuation: *M.*|*O'Connel*|*payed*|*$12,000*|*(*|*V.T.A.*|*not*|*included*|*)*|*with*|*his*|*credit card*|.|

Usually not in a lexicon because hard to lexicalize (too many hard-to-predict occurrences): *O'Connel*, *$12,000*

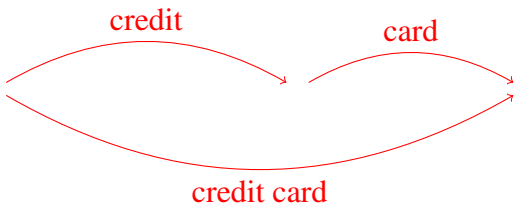
"*O'Connel*" could be in some lexicon of proper names (but not so usual), or recognized by some NER (Named-Entity Recognizer).

"*$12,000*" could be in some lexicon making use of regular expressions (e.g. a FSA), but this is also not so usual unless making use of some (other) NER.

tokens: *M*|.|_|*O*|'|*Connel*|_|*payed*|_|*$*|_|*12*|,|*000*|_|*(*|*V*|.|*T*|.|*A*|.|_|*not*|_|*included*|*)*|_|*with*|_|*his*|_|*credit*|_|*card*|.|

We could go from tokens to words by:

- agglutinating several (consecutive) tokens when the resulting word is in our lexicon
- doing so, it would be good to keep all possible solutions, for instance in the compact form of a graph/lattice; for instance:



- making use of NERs (check their input format/tokenization rules)
- add our own had-oc rules, e.g. M + period + whitespace + proper name/unknow token with capital letter $\longrightarrow$ proper noun

---

Exercise IV.2

Consider the following toy corpus:

the cat cut the hat

- How many different bigrams of characters (including whitespace) do you have in that corpus?
- How many occurences do you have in total? (i.e. including repertitions)
- Considering only lowercase alphabetical and whitespace, how many bigrams are possible?
- What are the parameters of a bigram model using the same set of characters (lowercase alphabetical and whitespace)?
- What is the probability of the following sequences, if the parameters are estimated using MLE (maximum-likelihood estimation) on the above corpus (make use of a calculator or even a short program):
  - *cutthechat*
  - *cut the chat*

  Fully justify your answer.
- What is the probability of the same sequences, if the parameters are estimated using Dirichlet prior with α having all its components equal to 0.05?

  Fully justify your answer.

Solution

- there are 12 different bigrams (denoting here the whitespace with 'X' to better see it): *Xc, Xh, Xt, at, ca, cu, eX, ha, he, tX, th, ut,*
- the corpus being 19 characters long, there are 18 bigrams in total. Here are the counts *Xc, 2; Xh, 1; Xt, 1; at, 2; ca, 1; cu, 1; eX, 2; ha, 1; he, 2; tX, 2; th, 2; ut, 1*
- $27^2 = 729$ bigrams in total

- parameters are all the 729 probabilies of the 729 bigrams ('X' = whitespace): $P(\text{XX})$, $P(\text{Xa})$, $P(\text{Xb})$, ..., $P(\text{aa})$, $P(\text{ab})$, ..., $P(\text{zz})$.

- Using MLE, the probability of the observed bigram are proportionnal to their number of occurence: *Xc: 2/18; Xh: 1/18; Xt: 1/18; at: 2/18; ca: 1/18; cu: 1/18; eX: 2/18; ha: 1/18; he: 2/18; tX: 2/18; th: 2/18; ut: 1/18*

  and all the other are 0.

  Thus the propability of any sequence containing an unseen bigram is 0 (as a product of terms, at least one of which is 0), which is the case for both sequences (bigram '*ch*' never seen)

- With a Dirichlet prior with parameter $\alpha = (\underbrace{0.05, \ldots, 0.05}_{729 \text{ times}})$ each observed bigram as a extra 0.05 to its count and the denominator is augmented by $729 \times 0.05 = 36.45$, leading thus to: *Xc: 2.05/54.45; Xh: 1.05/54.45; Xt: 1.05/54.45; at: 2.05/54.45; ca: 1.05/54.45; cu: 1.05/54.45; eX: 2.05/54.45; ha: 1.05/54.45; he: 2.05/54.45; tX: 2.05/54.45; th: 2.05/54.45; ut: 1.05/54.45*

  and all the unseen bigrams have a probability of $0.05/54.45$;

  The probability of the two sequences then becomes (in blue the bigrams seen in the learning corpus):

$$P(cutthechat) = P(cu) \cdot \frac{P(ut)}{P(u)} \cdot \frac{P(tt)}{P(t)} \cdot \frac{P(th)}{P(t)} \cdot \frac{P(he)}{P(h)} \cdot \frac{P(ec)}{P(e)} \cdot \frac{P(ch)}{P(c)} \cdot \frac{P(ha)}{P(h)} \cdot \frac{P(at)}{P(a)}$$

  and

$$P(u) = \sum_y P(uy) = P(ut) + 26 \times \frac{0.05}{54.45} = \frac{1.05}{54.45} + 26 \times \frac{0.05}{54.45} = \frac{2.35}{54.45} \simeq 4.32\%$$

$$P(c) = \sum_y P(cy) = P(ca) + P(cu) + 25 \times \frac{0.05}{54.45} = \frac{1.05}{54.45} + \frac{1.05}{54.45} + 25 \times \frac{0.05}{54.45} = \frac{3.35}{54.45} \simeq 6.15\%$$

$$P(t) = \sum_y P(ty) = P(tX) + P(th) + 25 \times \frac{0.05}{54.45} = \frac{2.05}{54.45} + \frac{2.05}{54.45} + 25 \times \frac{0.05}{54.45} = \frac{5.35}{54.45} \simeq 9.83\%$$

  and similarly for others.

  So we end up with:

$$P(cutthechat) = \frac{1.05}{54.45} \cdot \frac{1.05}{2.35} \cdot \frac{0.05}{5.35} \cdot \frac{2.05}{5.35} \cdot \frac{2.05}{4.25} \cdot \frac{0.05}{3.35} \cdot \frac{0.05}{3.35} \cdot \frac{1.05}{4.35} \cdot \frac{2.05}{3.35} \simeq 4.9 \times 10^{-10}$$

  Regarding the other sequence:

$$\begin{aligned} &P(cutXtheXchat) \\ &= P(cu) \cdot \frac{P(ut)}{P(u)} \cdot \frac{P(tX)}{P(t)} \cdot \frac{P(Xt)}{P(X)} \cdot \frac{P(th)}{P(t)} \cdot \frac{P(he)}{P(h)} \cdot \frac{P(eX)}{P(e)} \cdot \frac{P(Xc)}{P(X)} \cdot \frac{P(ch)}{P(c)} \cdot \frac{P(ha)}{P(h)} \cdot \frac{P(at)}{P(a)} \\ &= \frac{1.05}{54.45} \cdot \frac{1.05}{2.35} \cdot \frac{2.05}{5.35} \cdot \frac{1.05}{5.35} \cdot \frac{2.05}{5.35} \cdot \frac{2.05}{4.25} \cdot \frac{2.05}{3.35} \cdot \frac{2.05}{5.35} \cdot \frac{0.05}{3.35} \cdot \frac{1.05}{4.35} \cdot \frac{2.05}{3.35} \\ &\simeq 6.2 \times 10^{-8} \end{aligned}$$

Notice however that the two sequences do not have the same length, so their probabilities shall not be compared without a minimum amount of care. But in this case, since the probability of the shorter is smaller than the probability of the longer, it's conclusive: the longer is definitely the better (since, *for instance*, any substring of length 10 (=length of the shorther) of the longer will be more probable than the shorter).

5 Part-of-Speech tagging

Exercise V.1

What is the tagging of the following sentence

computers process programs accurately

with the following HMM tagger:

(part of) lexicon:

```
computers  N       0.123
process    N       0.1
process    V       0.2
programs   N       0.11
programs   V       0.15
accurately Adv     0.789
```

(part of) transitions:

```
P(N|V)=0.5        P(N|Adv)=0.12        P(V|Adv)=0.05
P(V|N)=0.4        P(Adv|N)=0.01        P(Adv|V)=0.13
P(N|N)=0.6        P(V|V)=0.05
```

Solutions

4 choices (it's a lattice):

```
computers process programs accurately
    N       N        N        Adv
            V        V
```

Differences are (skept the common factors):

```
P(N|N) P(process|N) P(N|N) P(programs|N) P(Adv|N)
P(N|N) P(process|N) P(V|N) P(programs|V) P(Adv|V)
P(V|N) P(process|V) P(N|V) P(programs|N) P(Adv|N)
P(V|N) P(process|V) P(V|V) P(programs|V) P(Adv|V)
```

i.e.:

```
    0.6    0.1    0.6    0.11   0.01
--> 0.6    0.1    0.4    0.15   0.13 <--MAX
    0.4    0.2    0.5    0.11   0.01
    0.4    0.2    0.05   0.15   0.13
```

Tagging obtained (not corresponding to the one expected by an average English reader ;-)):

```
computers process programs accurately
    N       N        V        Adv
```

Exercise V.2

We aim at tagging English texts with "Part-of-Speech" (PoS) tags. For this, we consider using the following model (partial picture):

...some picture...

Explanation of (some) tags:

Tag	English expl.	Expl. française	Example(s)
JJ	Adjective	adjectif	*yellow*
NN	Noun, Singular	nom commun singulier	*cat*
NNS	Noun, Plural	nom commun pluriel	*cats*
PRP$	Possessive Pronoun	pronom possessif	*my*, *one's*
RB	Adverb	adverbe	*never*, *quickly*
VBD	Verb, Past Tense	verbe au passé	*ate*
VBN	Verb, Past Participle	participe passé	*eaten*
VBZ	Verb, Present 3P Sing	verbe au présent, 3e pers. sing.	*eats*
WP$	Possessive wh-	pronom relatif (poss.)	*whose*

① What kind of model (of PoS tagger) is it? What assumption(s) does it rely on?

② What are its parameters? Give examples and the appropriate name for each.

We use the following (part of) lexicon:

adult	JJ	has	VBZ
adult	NN	just	RB
daughter	NN	my	PRP$
developed	VBD	programs	NNS
developed	VBN	programs	VBZ
first	JJ	tooth	NN
first	RB	whose	WP$

and consider the following sentence:

my daughter whose first adult tooth has just developed programs

③ With this lexicon, how many different PoS taggings does this sentence have? Justify your answer.

④ What (formal) parameters make the difference in the choice of these different PoS taggings (for the above model)?

Give the explicit mathematical formulas of these parts that are different.

⑤ Assume that the following tagging is produced:

my/PRP$ daughter/NN whose/WP$ first/JJ adult/JJ tooth/NN has/VBZ just/RB developed/VBN programs/NNS

How is it possible? Give an explanation using the former formulas.

Solutions

① This is an HMM of order 1 (Well, the picture is actualy a part of a Markov chain. The "hidden" part will be provide by the emission probabilities, i.e. the lexicon).

HMM relies on two asumptions (see course): limited lexical contionning ($P(w_i|...C_i...) = P(w_i|C_i)$) and limited scope for syntactic dependencies ($P(C_i|C_1...C_{i-1}) = P(C_i|C_{i-k}...C_{i-1})$).

② Its parameters are:

1. initial probabilities: $P_I(\text{tag})$
2. transition probabilities: $P(\text{tag}_i|\text{tag}_{i-1}...\text{tag}_{i-k})$
3. emission probabilities: $P(\text{word}|\text{tag})$

Exemples: initial: $P_I(\text{JJ})$, transition: $P(\text{JJ}|\text{NN})$, emission: $P(\text{adult}|\text{JJ})$.

③

my	PRP$	
daughter	NN	
whose	WP$	
first	JJ	RB
adult	JJ	NN
tooth	NN	
has	VBZ	
just	RB	
developed	VBN	VBD
programs	NNS	VBZ

2x2x2x2=16 possible taggings

Examples:

```
my/PRP\$ daughter/NN whose/WP\$ first/JJ adult/JJ tooth/NN has/VBZ
just/RB developed/VBN programs/VBZ

my/PRP\$ daughter/NN whose/WP\$ first/JJ adult/JJ tooth/NN has/VBZ
just/RB developed/VBN programs/NN
```

④ Differences are due to two subproducts:

On one hand:

$$P(X|\text{WP\$}) \cdot P(\text{first}|X) \cdot P(Y|X) \cdot P(\text{adult}|Y) \cdot P(\text{NN}|Y)$$

for X either "JJ" or "RB" and Y either "JJ" of "NN", and on the other hand:

$$P(X|\text{RB}) \cdot P(\text{developed}|X) \cdot P(Y|X) \cdot P(\text{programs}|Y)$$

for X either "VBD" or "VBN" and Y either "NNS" of "VBZ",

NOTICE:

1. do not forget emision probabilities
2. do not forget the right hand part of each tag, e.g for "adult", not only $P(NN|RB)$ (for instance), but also $P(NN|NN)$ for the transition to "tooth".

⑤ It is possible simply by the fact that the product

$$P(\text{JJ}|\text{WP\$}) \cdot P(\text{first}|\text{JJ}) \cdot P(\text{JJ}|\text{JJ}) \cdot P(\text{adult}|\text{JJ}) \cdot P(\text{NN}|\text{JJ}) \cdot P(\text{VBN}|\text{RB}) \cdot P(\text{developed}|\text{VBN}) \cdot P(\text{NNS}|\text{VBN})$$
$$\cdot P(\text{programs}|\text{NNS})$$

is bigger than any other of the products for the same part, which is possible (e.g. each term bigger than any corresponding other, or even one much bigger than all the other products, etc.)

Exercise V.3

① What is the problem addressed by a Part-of-Speech (PoS) tagger?
Why isn't it trivial? What are the two main difficulties?

② Assume that you have to quickly search for the existence of given {word , part-of-speech} pairs within the set of all the English words associated with their part(s)-of-speech. Which data structure(s) would you use if memory is an issue?

③ Assume that the texts to be tagged contain unknown words, which are either capitalized words, or spelling errors, or simply general common words not seen during the learning. Almost all capitalized words correspond to proper nouns, and most of the spelling-errors correspond to

words already in the lexicon (only a few of the spelling errors correspond to words not seen during the learning).

How would you handle such a situation in a concrete NLP application (that uses a PoS tagger)? Explicit your solution(s).

④ Assume that the texts to be tagged contain 1.5% of unknown words and that the performance of the tagger to be used is 98% on known words.

What will be its typical overall performance in the following two situations:

(a) all unknown words are systematically wrongly tagged?

(b) using the solution you proposed in ③ is used in a situation where 80% of the unknown words are capitalized among which 98% are proper nouns, 15% are general common words not seen during learning, and 5% are spelling-errors, among which 1% corresponds to correct words which were not in the learning set?

Provide both a calculation (a complete formula but not necessarily the final numerical result) and an explanation.

Solutions

① The problem addressed by a PoS tagger is to assign part-of-speech tags (i.e. grammatical roles) to words within a given context (sentence, text).

This task is not trivial because of lexical ambiguity (words can have multiple grammatical roles, e.g. can/N can/V) and out-of-vocabulary forms (i.e. unknown words).

Lexical ambiguity is not trivial to handle because it leads to an exponential number of possible solution w.r.t. the sentence length.

Unknow words are not trivial because we have to decide how to cope with them, which often involves high level linguistic features (and compromise to be made). This is the role of the "guesser".

② Finite-State Transducers seems really appropriate for this task under the memory consumption constraint since they are the optimal representation of paired-regular languages.

Another possible solution, however, could be to build the FSA of words, use it to map words to numbers and then associate a table of list of PoS tags (maybe also represented in the form of numbers through another FSA).

It is not clear how the overheads of each implementation will compare one to another in a real implementation.
(The second propostion being actually one possible implementation for the corresponding FST).

③ The first idea is to tag capitalized words as proper nouns.

Then we'd like to cope with spelling errors as much as possible. This is hard in a completely autonomous manner because there might be several solution of a real spelling errors, but also there might be some possible correction for unknown words which correspond to correct words

but unseen during the learning. The idea is thus to use a low threshold for the spelling error correction and keep all possible tags for all possible solutions in case of ambiguity, leting then the tagger to disambiguate the tag.

For the rest, the guesser corresponding to the tagger used should be used anyway.

④ (a) is simple : 1.5% is for sure wrongly tagged. For the rest (100%-1.5%), only 98% are correctly tagged. So the overall score is $0.985 \times 0.98 \simeq 0.96$.

(b) this is less obvious: still we have 0.985×0.98, but on the remaining 1.5% we cannot be sure:

- regading capitalized words, we can expect to have 98% correct, thus: $0.015 \times 0.8 \times 0.98$
- but for the rest, this really depends on the perfomances/ambiguities in the spelling error correction and on the performance of the guesser.

This could be summarized as:

1.5% unknown	80% capitalizes	98% proper nouns: **OK**
		2% **WRONG**
	15% unseen: **??**	
	5% spell. err.	99% corrected: 98%?? **OK**
		1%: **??**
98.5% known	98%: **OK**	
	2% **WRONG**	

Exercise V.4

① Consider an HMM Part-of-Speech tagger, the tagset of which contains, among others:

DET, N, V, ADV and ADJ,

and some of the parameters of which are:

$$P_1(\text{a}|\texttt{DET}) = 0.1, \quad P_1(\text{accurately}|\texttt{ADV}) = 0.1, \quad P_1(\text{computer}|\texttt{N}) = 0.1,$$
$$P_1(\text{process}|\texttt{N}) = 0.095, \quad P_1(\text{process}|\texttt{V}) = 0.005,$$
$$P_1(\text{programs}|\texttt{N}) = 0.080, \quad P_1(\text{programs}|\texttt{V}) = 0.020,$$

$P_2(\texttt{Y}|\texttt{X})$: (for instance $P_2(\texttt{N}|\texttt{DET}) = 0.55$)

		Y →				
		DET	N	V	ADJ	ADV
X ↓	DET	0	0.55	0	0.02	0.03
	N	0.01	0.10	0.08	0.01	0.02
	V	0.16	0.11	0.06	0.08	0.08
	ADJ	0.01	0.65	0	0.05	0
	ADV	0.08	0.02	0.09	0.04	0.04

and:
$P_3(\texttt{DET}) = 0.20, \quad P_3(\texttt{N}) = 0.06, \quad P_3(\texttt{V}) = 0.08, \quad P_3(\texttt{ADV}) = 0.07, \quad P_3(\texttt{ADJ}) = 0.02.$

(a) How are the propabilities P_1, P_2 and P_3 usually called?

P_1: emission

P_2: transition

P_3: initialization

(b) What are all the possible taggings of the sentence

```
a computer process programs accurately
```

```
a   computer process programs accurately
DET N        V       V        ADV
             N       N
```

which leads to 4 solutions.

(c) What would be the output of the HMM PoS tagger on the above sentence? Fully justify your answer.

x	y	x|N	process|x	y|x	programs|y	ADV|y
N	N	10	95	10	80	2
V	N	8	5	11	80	2
N	V	10	95	8	20	8
V	V	8	5	6	20	8

Noticing that $80 \cdot 2 = 20 \cdot 8$, only the first three enter the game, among which the first is clerarly the best.

The output will thus be

```
a   computer process programs accurately
DET N        N       N        ADV
```

7 Text Classification

Exercise VII.1

In an automated email router of a company, we want to make the distinction between three kind of emails: technical (about computers), financial, and the rest ("irrelevant"). For this we plan to use a Naive Bayes approach.

① What is the main assumption made by Naive Bayes classifiers? Why is it "Naive"?

We will consider the following three messages:

> The Dow industrials tumbled 120.54 to 10924.74, hurt by GM's sales forecast and two economic reports. Oil rose to $71.92.

from `www.wsj.com/`

> BitTorrent Inc. is boosting its network capacity as it prepares to become a centralized hub for legal video content. In May, BitTorrent announced a deal with Warner Brothers to distribute its TV and movie content via the BT platform. It has now lined up IP transit for streaming videos at a few gigabits per second

from `slashdot.org/`

> Intel will sell its XScale PXAxxx applications processor and 3G baseband processor businesses to Marvell for $600 million, plus existing liabilities. The deal could make Marvell the top supplier of 3G and later smartphone processors, and enable Intel to focus on its core x86 and wireless LAN chipset businesses, the companies say.

from `www.linuxdevices.com/`

② What pre-processing steps (before actually using the Naive Bayes Classifier) do you consider applying to the input text?

③ For the first text, give an example of the corresponding output of the pre-processor.

continues on back ☞

Suppose we have collected the following statistics[2] about the word frequencies within the corresponding classes, where "0.00…" stands for some very small value:

	technical	financial	irrelevant
$<number>	0.01	0.07	0.05
Dow	0.00…	0.08	0.00…
GM	0.00…	0.03	0.00…
IP	0.03	0.00…	0.00…
Intel	0.02	0.02	0.00…
business	0.01	0.07	0.04
capacity	0.01	0.00…	0.00…
chipset	0.04	0.01	0.00…
company	0.01	0.04	0.05

	technical	financial	irrelevant
deal	0.01	0.02	0.00…
forecast	0.00…	0.03	0.01
gigabit	0.03	0.00…	0.00…
hub	0.06	0.00…	0.01
network	0.04	0.01	0.00…
processor	0.07	0.01	0.00…
smartphone	0.04	0.04	0.01
wireless	0.02	0.01	0.00…

④ In a typical NLP architecture, where/how would you store this information? Explicit your answer, e.g. provide an illustrative example.

⑤ For each of the above three texts, in what category will it be classified, knowing that on average 50% of the emails happen to be technical, 40% to be financial and 10% to be of no interest.

You can assume that all the missing information is irrelevant (i.e. do not impact the results).

Provide a full explanation of all the steps and computations that lead to your results.

We now want to specifically focus on the processing of compounds such as "network capacity" in the second text.

⑥ How are the compounds handled by a Naive Bayes classifier if no specific pre-processing of compounds is used?

⑦ What changes if the compounds are handled by the NL pre-processor?

Discuss this situation (NL pre-processing handling compounds) with respect to the Naive Bayes main assumption.

⑧ Outline how you would build a pre-processor for compound words.

Solutions

Q2.1 The main assumption is that features/attributes contributing to the likelihood are independant, conditionnaly to classes:

$$P(f_1 \ldots f_n|C) = \prod_i P(f_i|C)$$

[2]Note that this is only partial information, statistics about other words not presented here have also been collected.

This is in practice definitely a strong assumption. This is the reason why is is called “Naive”

Q2.2 In text classification, preprocessing is really crutial in order to allow a "good" representation mainly through proper lexical variability reduction.

Usual NLP steps for reducing lexical variability include: tokenization (removal of punctuation), PoS tagging, lemmatization and suppression of grammatical ("meaningless") words (stopword, some PoS tags, low frequencies).

If lemmatization is not possible, stemming could be consisered either.

We could also have a more evolved tokenizer including Entity Recognition (e.g. based on regular patterns) or even Name Entity Recognition for Proper Nouns.

Q2.3 Lemmatized and with number entity recognition, this could lead to:

```
Dow industrial tumble <number> <number> hurt GM sale
forecast economic report oil rise \$<number>
```

If a multi-set representation is even included in the preprocessing (this was not expected as an answer), the output could even be:

```
(\$<number>,1) (<number>,2) (Dow,1) (GM,1) (economic,1) (forecast,1)
(hurt,1) (industrial,1) (oil,1) (report,1) (rise,1) (sale,1) (tumble,1)
```

Q2.4 This is a more difficult question than it seems because it actually depends on the representation choosen for the lexicon. If this representation allows to have several numeric fields associated to lexical entries, then definitly it should be stored there.

Otherwise some external (I mean out of the lexicon) array would be build, the role of the lexicon then being to provide a mapping between lexical entries and indexes in these arrays.

The choice of the implementation also highly depends on the size of the vocabulary to be stored (and also on the timing specifications for this tasks: realtime, off-line, ...)

Anyway this is typically a lexical layer level resource.

Example for the case where a associative memory (whatever it’s implementation) is available:

capacity $\rightarrow$ 123454 by the lexicon then an array such that `ARRAY[1][123454]=0.01`

It should be noticed that these probability arrays are very likely to be very sparse. Thus sparse matrix representations of these would be worth using here.

Q2.5 What make the discrimination between the class are the $P(\text{word}|textclass)$ and the priors $P(C)$. Ideed, the Naive Bayes classifier uses (see lectures):

$$\text{Argmax}\, P(C|w_1 \ldots w_n) = \text{Argmax}\, P(C) \prod_i P(w_i|C)$$

As stated out in the question, assuming that all the rest is irrelevant, the first text will have

	technical	financial	irrelevant
Dow	0.00…	0.08	0.00…
GM	0.00…	0.03	0.00…
forecast	0.00…	0.03	0.01
$<number>	0.01	0.07	0.05

the maximal product of which is clearly for the second class: “financial”.

For the second text we have:

	technical	financial	irrelevant
network	0.04	0.01	0.00…
capacity	0.01	0.00…	0.00…
hub	0.06	0.00…	0.01
deal	0.01	0.02	0.00…
gigabit	0.03	0.00…	0.00…
IP	0.03	0.00…	0.00…

the maximal product of which is clearly for the first class: “technical.

For the third text:

	technical	financial	irrelevant
Intel	0.02	0.02	0.00…
processor	0.07	0.01	0.00…
processor	0.07	0.01	0.00…
business	0.01	0.07	0.04
$<number>	0.01	0.07	0.05
deal	0.01	0.02	0.00…
smartphone	0.04	0.04	0.01
processor	0.07	0.01	0.00…
Intel	0.02	0.02	0.00…
wireless	0.02	0.01	0.00…
chipset	0.04	0.01	0.00…
business	0.01	0.07	0.04
company	0.01	0.04	0.05

which could be organized:

	technical	financial	irrelevant
Intel	0.02	0.02	0.00…
Intel	0.02	0.02	0.00…
smartphone	0.04	0.04	0.01
processor	0.07	0.01	0.00…
$<number>	0.01	0.07	0.05
processor	0.07	0.01	0.00…
business	0.01	0.07	0.04
processor	0.07	0.01	0.00…
business	0.01	0.07	0.04
deal	0.01	0.02	0.00…
wireless	0.02	0.01	0.00…
company	0.01	0.04	0.05
chipset	0.04	0.01	0.00…

showing that the $\prod_i P(w_i|C)$ part is the same for the first two classes (and much smaller for "irrelevant")

Thus the prior $P(C)$ will make the decision and this last text is classified as "technical".

Q2.6 compounds are simply ignored as such by the Naive Bayes and are, due to the "Naive" independance assumption, handled as separated tokens.

Q2.7 If the preprocessor is able to recognized coumponds as such they will thus be included as such in the set of features and would thus be handled as such. This is actually a way (preprocessor) to increase the independance between "features" of the Naive Bayes, these features no loger corresponding to single tokens only.

Mathematically: if the coumpound is w_1w_2 as a feature in itself after prepocessing we now have a $p(w_1w_2|C)$ appearing in the parameters, which is no longer be assumed to be $p(w_1|C)p(w_2|C)$

Q2.8 coumpond preprocessor is a wide topic in itself (lexical acquisition), but as in many NLP domains two main ways could be considered, which whould definitly be exploited in complement one of the other: the statistical way and the linguistic/human knowledge way.

The most naive linguistic approach could be to add by hand coumponds to the lexicon.

For the statistical, simply extract all correlated pairs or biger tuples of word, using e.g. mutual information, chi-square or whatever measure of correlation. This could be enhanced using human knowledge by selecting which PoS tags could enter this correlation game (e.g. lookig for NN NN, NN of NN etc..) but also by filtering out manually automatically extracted lists of candidates.

Exercise VII.2

You are responsible for a project aiming at providing on-line recommendations to the customers of a on-line book selling company.

The general idea behind this recommendation system is to cluster books according to both customers and content similarities, so as to propose books similar to the books already bought by a given customer. The core of the recommendation system is a clustering algorithm aiming at regrouping books likely to be appreciate by the same person. This clustering should not only be achieved based on the purchase history of customers, but should also be refined by the content of the books themselves. It's that latter aspect we want to address in this exam question.

① Briefly explain how books could be clustered according to similar content. Give the main steps and ideas.

"*similar content*": meaning .vs. surface content or even structural content

main steps:

- preprocessing: keep *semantically* meaningful elements, remove less semantically important lexical variability

  Usual NLP steps for reducing lexical variability include: tokenization (removal of punctuation), PoS tagging, lemmatization and suppression of grammatical ("meaningless") words (stopwords, some well chosen PoS tags).

  If lemmatization is not possible, stemming could be consisered either.

  We could also have a more evolved tokenizer including Name Entity Recognition (e.g. based on regular patterns).
- counting: frequencies, IDF, ...
- indexing / Bag of words representation: from word sequences to vectors
- compute (dis)similarities btw representations
- (choose and) use classification method.

② The chosen clustering algorithm is the dendrogram. What other algorithms could you propose for the same task? Briefly review advantages and disadvantages of each of them (including dendrograms). Which one would you recommend for the targeted task?

We are in the *unsupervised* case. A possible baseline altenative are the K-means.

drawbacks: what K should be use for K-mean? converges only to a local min, what linkage to use for dendrograms

advantages: planar representation for dendrograms (could be complemented with minimal spanning tree), K-means are incremental: can choose to stop if too long (monitor intra-class variance, however)

Maybe the best to do should be to try both (and even more) and evaluated them, if possible, in real context...

③ Consider the following six "documents" (toy example):

d_1 "*Because cows are not sorted as they return from the fields to their home pen, cow flows are improved.*"

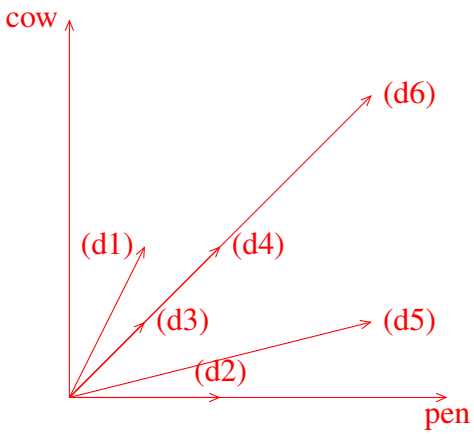
...

d_6 *"What pen for what cow? A red pen for a red cow, a black pen for a black cow, a brown pen for a brown cow, ... Understand?"*

and suppose (toy example) that they are indexed only by the two words: *pen* and *cow*.

(a) Draw their vector representations.

```
(d1) at (1,2)
(d2) at (2,0)
(d3) at (1,1)
(d4) at (2,2)
(d5) at (4,1)
(d6) at (4,4)
```

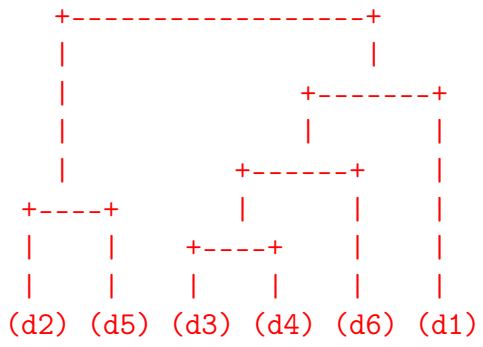


(b) Give the definition of the cosine similarity. What vector's feature(s) is it sensible to?
see lectures. Only sensible to vector angle/direction, not length, i.e. to word relative proportions, not to absolute counts.

(c) What is the result of the dendrogram clustering algorithm on those six documents, using the cosine similarity and single linkage?
Explain all the steps.

Hint: $5/\sqrt{34} < 3/\sqrt{10} < 4/\sqrt{17}$.

Solution : Notice that there is absolutely no need to compute every pair of similarities!!! Only 3 of them are useful, cf drawing (a); and even the drawing alone might be sufficient (no computation at all)!



Notice that, of course, more similar means bigger cosine!

In case you really want to do some computation, here are some values: $D(2,5) = 4/\sqrt{17}$, $D(3,5) = 5/\sqrt{34}$, and $D(1,3) = 3/\sqrt{10}$.

8 Information Retrieval

Exercise VIII.1

① Describe the main principles of the standard vector space model for semantics.

② Consider the following document:

D = "`the exports from Switzerland to the USA are increasing in 2006`"

Propose a possible indexing set for this document. Justify your answer.

③ What is the similarity between the above document D and

D' = "`Swiss exports have increase this year`"

Justify your answer.

④ Briefly describe the important limitation(s) of the standard vector space approach.

Explain how more sophisticated techniques such as the *Distributional Semantics* can be used to circumvent this/these limitation(s).

⑤ Give some concrete examples of NLP applications that might benefit from the semantic vectorial representations.

⑥ Using the standard vector space model, does the indexing set you considered in question Q2 allow to discriminate between D and this other document:

D" = "`the exports from the USA to Switzerland are increasing in 2006`"

If yes: how? If not, why?

⑦ Would a parser be available, how could it be used to provide a (partial) solution to the problem?

Solutions

① The standard approach to vector semantics can be decomposed into two mains steps:

- the indexing (or desequalization) phase: during this phase, the documents for which a vectorial semantic representation needs to be produced, are processed with linguistic tools in order to identify the indexing features (words, stems, lemmas, ...) they will be associated with.

  This phase results in the association with each of the documents of a set of indexing features. Notice that, for the rest of the processing, on the sets of indexing features will be considered. The rest of the documents will be ignored. Notice also that the sets of indexing features are sets!... and that therefore any notion of word order is lost after the indexing phase.

  For example, if we consider the toy document collection consisting of the two following documents:

```
D1 = "the results of the experiments on transgenic plants
will be issued soon."

D2 = "as soon as the experiments will be over, the laboratory
will close."
```

A possible output of the indexing phase for these documents might be:

```
D1 --> {result, experiment, transgenic, plant, issue}

D2 --> {experiment, over, laboratory, close}
```

but it is important to notice that the order of the word lemmas in the indexing sets is in fact meaningless, and D1 and D2 might be equivalently indexed by:

```
D1 --> {experiment, issue, plant, result, transgenic}

D2 --> {close, experiment, laboratory, over}
```

where the indexing features have been arbitrarily put in alphabetic order.

- The second step of the vector semantics modeling is the representation phase.

  During this phase, each of the indexing features that have been identified is associated with one of the dimensions of a (usually highly dimensional) vector space and a method must be designed to transform the indexing sets associated with the documents into vectors.

  A possible approach is to use binary vectors in which the 0/1 coordinates simply indicated whether the corresponding indexing feature is or is not associated with a given document.

  A more sophisticated approach consists in using the occurrence statistics of the indexing features in the documents to derive less brittle importance scores for each of the indexing features appearing in a document. A simple version of this approach if to use the (usually normalized) occurrence frequency of a feature in a document as a measure of the importance of this feature for the document. For example, a feature appearing in a document 3 times more frequently than another will be considered as three times more important for that document.

  The importance scores can then be used as coordinates for the vectors representing the topical content of the documents.

  Once each of the documents can be represented in the indexing feature vector space, the remaining problem is to define a similarity in this vector space in order to be able to evaluate the semantic proximities between the documents.

  The standard approach is to use the cosine similarity, defined as:

  if V1 is the vector representing document D1 and V2 is the vector representing document D2,

  the semantic proximity between D1 and D2 is simply defined as:

$$sim(D_1, D_2) = cos(V_1, V_2) = \frac{V_1 \cdot V_2}{\|V1\|\|V2\|},$$

  where $X \cdot Y$ denotes the dot-product between vector X and vector Y, and $\|X\| = \sqrt{X \cdot X}$ represents the norm (i.e. the length) of vector X.

Notice that this simple similarity might be further sophisticated in order to take into account varying importance for the various dimensions of the vector space.

A possible approach is to use a weighted dot-product of the form:

for $V1 = (v_{11}, v_{12}, ..., v_{1n})$

and $V2 = (v_{21}, v_{22}, ..., v_{2n})$

$V_1 \cdot V2 = \sum_{i=1}^{n} a_i v_{1i} v_{2i}$, where the a_i are some (usually positive) coefficients.

A standard approach for the weighting of the vector space dimensions is to use the "inverse document frequency" (i.e. fact any function f() decreasing with the document frequency of an indexing feature, i.e. the inverse of the number of documents containing the given indexing feature).

For example, if we take: ai = idf(i)2 = log(1/DF(i))2, where DF(i) is the document frequency of the indexing feature associated with the i-th dimension of the vector space, we get:

sim(D1,D2) = cos(V1', V2'), where Vi' = (tf(i,k).idf(k)), where tf(i,k) is the measure of importance of the k-th indexing feature for the i-th document and idf(k) is a measure of importance of the k-th dimension of the vector space.

This approach corresponds to the standard "tf.idf" weighting scheme.

② The simplest solution to produce the indexing set associated with a document is to use a stemmer associated with stop lists allowing to ignore specific non content bearing terms. In this case, the indexing set associated with D might be:

```
I(D) = {2006, export, increas, Switzerland, USA}.
```

A more sophisticated approach would consist in using a lemmatizer in which case, the indexing set might be:

```
I(D) = {2006_NUM, export_Noun, increase_Verb, Switzerland_ProperNoun,
USA_ProperNoun}.
```

③ The answer to this question depends on the indexing set considered.

One solution could be:

```
I(D) = {2006, export, increas, Switzerland, USA}.
```

```
I(D') = {export, increas, Swiss, year}.
```

Then several similarity measures could be considered, e.g. Dice, Jacard, cosine.

For cosine: the dot product is 2 (`export` and `increas`) and the norms are $\sqrt{5}$ and $\sqrt{4}$, thus:

$$\cos(D, D') = \frac{1}{\sqrt{5}}$$

④ One of the important limitations of the standard vector space approach is that the use of the cosine similarity imposes that the dimensions of the vector space are orthogonal and that therefore the indexing features associated with the dimensions have, by construction, a null similarity.

This is in fact a problem as it is extremely difficult to guarantee that the indexing features associated with the dimension are indeed semantically fully uncorrelated. For example, it is sufficient that one (or more) document(s) contain(s) the two words "car" and "vehicle" to imply that sim("car", "vehicle") = 0 which should be interpreted as the (not very convincing) fact that "car" and "vehicle" has nothing in common.

A possible (partial) solution for this problem is to use more sophisticated representation techniques such the Distributional Semantics (DS).

In DS, the semantic content of an indexing feature does not only rely on its occurrences in the document collection, but also on the co-occurrences of this indexing feature with other indexing features appearing the the same documents. In fact, the vectorial representations use in DS are a mixture of the standard occurrence vectors (as they are used in the traditional vector space model) with the co-occurrence vectors characterizing the indexing features appearing in the documents. Thus, even is the similarity between the occurrence vectors of "car" and "vehicle" have, by definition, a zero similarity, in DS, the vectors representing the documents are of the form:

V(D) = a*OccV(D) + (1-a)*CoocV(D)

and therefore, is "car" and "vehicle" share some co-occurrences (i.e. appear in documents together with some identical words), their similarity will not be zero anymore.

⑤ Any NLP application that requires the assessment of the semantic proximity between textual entities (text, segments, words, ...) might benefit from the semantic vectorial representation. Information retrieval is of course one of the prototypical applications illustrating the potentiality of the VS techniques. However, many other applications can be considered:

- automated summarization: the document to summarize is split into passages; each of the passages is represented in a vector space and the passage(s) that is the "most central" in the set of vector thus produced are taken as good candidates for the summary to generate;
- semantic desambiguisation: when polysemic words (such as "pen" which can be a place to put cow or a writing instrument) are a problem –for example in machine translation– vectorial representations can be generated for the different possible meanings of a word (for example from machine readable disctionalires) and used to desambiguate the occurrences of an ambiguous word in documents;
- automated routing of messages to users: each user is represented by the vector representing the semantic content of the messages s/he has received so far, and any new incoming message is routed only to those users the representative vector of which is enough similar to the vector representing the content of the incoming message;
- text categorization or clustering
- ...

⑥ No. The indexing sets associated with D and D' would be exactly the same and would therefore not allow to discriminate between these two documents (which nevertheless do not mean the same!...).

⑦ If a parser would be available, grammatical roles might be automatically associated with the reduced indexing features. For example, specific grammatical roles could be associated with prepositional nominal phrases such as "to the USA" or "from Switzerland" which could then be represented as "`to_USA`" and "`from_Switzerland`".

In this case, the indexing sets associated with D and D' would be:

`I(D) = {2006, export, from_Switzerland, increase, to_USA}`

and

`I(D) = {2006, export, from_USA, increase, to_Switzerland}`

and would allow D and D' to be discriminated.

Exercise VIII.2

① What is the cosine similarity?

② Consider the following two documents:

```
D1: Dog eat dog. Eat cat too!
D2: Eat home, it's raining cats and dogs.
```

What would be their cosine similarity in a typical information retrieval setup? Explain all the steps.

③ In a standard cosine-based tf-idf information retrieval system, can a query retrieve a document that does not contain any of the query words? Justify your answer.

④ Other measures than the cosine are possible. For instance, Jaccard similarity computes the ratio between the number of words in common and the total number of occurring words (i.e. "intersection over union").

(a) Can this measure be used on the same document representations as the one used for the cosine similarity can be? Justify your answer.

(b) Using the boolean representation for documents, find an example of three distinct documents d_1, d_2 and d_3 such that d_1 is closest to d_2 using cosine similarity, whereas d_2 and d_3 have the same similarity with respect to d_1 using Jaccard similarity.
Conclude on the comparison between these two similarity measures.

⑤ An information retrieval system with high precision and low recall can be useful for:

(a) Retrieving all relevant documents from a database of legal cases.

(b) Retrieving some interesting documents for a given a topic from the web.

(c) Retrieving a large set of interesting documents for a given a topic from the web.

(d) Checking the existence of a document in a very large document collection.

Choose all possible useful situations in the above list (maybe several). Justify your answer; in particular, define the notions of precision and recall.

Solutions

① It's a possible measure used for document semantic content similarity. It operated on a vector representation of the document "meaning" and is computed as

$$cos(d,q) = \frac{d \cdot q}{||d||\,||q||} = \frac{d \cdot q}{\sqrt{(d \cdot d)(q \cdot q)}}$$

② first a part-of-speech tagger might be applied in order to both filter out some stop words (and make the distinction between can/V, to be removed, and can/N, to be kept), and prepare for lemmatization, i.e. normalization of the surface forms of the "full words".

On this example, typically (maybe "raining" or "rain/V"):

```
D1: dog eat dog eat cat
D2: eat home rain cat dog
```

Then a vectorial representation is build, typically a words frequency (tf) vector. In this example (corresponding to words dog, cat, eat, rain, home):

```
D1: (2, 2, 1, 0, 0)
D2: (1, 1, 1, 1, 1)
```

Then the above cosine formula is used, leading to $\frac{5}{\sqrt{45}} = \frac{\sqrt{5}}{3}$.

③ Well... in principe NO... unless the system is urged to answer something anyway.

The numerator of the cosine will be 0 for every document (this query is orthogonal to all documents). However, depending on how the norm of the query is computed, this should also lead to a 0. Thus the cosine is undefined (0 over 0).

And it's up to the system engineering details to decide what to do in such a case.

④ (a) yes. it is indeed easy to compute intersection and union from the tf vector, for instance simply degrade it into a binary vector.

(b) This appeared to be a difficult question. Several missed the "boolean representation" constraint and none was able to properly find the example.

First notice that on a boolean representation, the dot product is the same as the (cardinal of the) intersection. Thus cosine and Jaccard have the same numerator.

Futhermore, the L2 norm correspond to the document length in the boolean case (binary vector).

Thus in the boolean case cosine reduces to $\frac{|d \cap q|}{|d|\,|q|}$ (where Jaccard is $\frac{|d \cap q|}{|d \cup q|}$).

Notice also that $|d \cup q| = |d| + |q| - |d \cap q|$.

The fact that the two Jaccard are the same but the cosine are different implies that $|d_1 \cap d_2|$ and $|d_1 \cap d_3|$ have to differ (easy proof).

For instance, let's take the simplest case: $|d_1 \cap d_2| = 1$ and $|d_1 \cap d_3| = 2$.

This implies that $|d_1 \cup d_3| = 2\,|d_1 \cap d_2|$. Since they cannot be 1 and 2, neither 2 and 4 (easy to see that the cosine are then equal), let us try with 3 and 6, for which several examples can be found, for instance

```
D1: 1 1 1 0 0 0
D2: 1 0 0 0 0 0
D3: 1 0 1 1 1 1
```

In this case the two Jaccard are both equal to 1/3 and the two cosines are $1/\sqrt{3}$ and $2/\sqrt{15}$.

⑤ (b) + see lectures.

Exercise VIII.3

① Official NLP evaluations (especially for task such as Information Retrieval or Information Extraction) are often carried out in the form of "evaluation campaigns".

Precisely describe the various steps of such an evaluation campaign.

For each of the steps, clearly indicate the main goals.

Solution: The first step is to define the *control task*. Then you have to gather a significant amount of *data*. Third you have to anotate some *reference test data* (the "*golden truth*"), typicaly by some *human experts*. Then you can run your system on a typical *test set* (different from both *learning* and *tuning* (a.k.a *validation*) set). You thus produce some *quantitative scores* describing the results, which you can *publish*, *analyse* (confidence) and *discuss*.

② In an IR evaluation campaign, the following "referential" ("golden truth") has been produced by a set of human judges:

```
q1: d01 d02 d03 d04
q2: d05 d06
q3: d07 d08 d09 d10 d11
q4: d12 d13 d14 d15
```

where the list of document references `dj` associated with a query reference `qi` defines the set of documents considered to be relevant for the query by the human judges.

Is such a referential easy to produce?

Indicate the various problems that might arise when one tries to produce it.

(a) task ambiguity (meaning of the text, of the question, is the solution unique?)

(b) subjectivity (see inter-anotator agreement)

(c) size matters (too small $\Longrightarrow$ too biaised)

(d) exhaustivity

(a) and (b) , resp. (c) and (d) are related ; the later being a consequence of the former.

③ Consider two Information Retrieval systems S_1 and S_2 that produced the following outputs for the 4 reference queries `q1`, `q2`, `q3`, `q4`:

```
S1:                                              | referential:
 q1: d01 d02 d03 d04 dXX dXX dXX dXX             |  q1: d01 d02 d03 d04
 q2: d06 dXX dXX dXX dXX                         |  q2: d05 d06
 q3: dXX d07 d09 d11 dXX dXX dXX dXX dXX         |  q3: d07 d08 d09 d10 d11
 q4: d12 dXX dXX d14 d15 dXX dXX dXX dXX         |  q4: d12 d13 d14 d15

S2::                                             | referential:
 q1: dXX dXX dXX dXX d04                         |  q1: d01 d02 d03 d04
 q2: dXX dXX d05 d06                             |  q2: d05 d06
 q3: dXX dXX d07 d08 d09                         |  q3: d07 d08 d09 d10 d11
 q4: dXX d13 dXX d15                             |  q4: d12 d13 d14 d15
```

where dXX refer to document references that do not appear in the referential. To make the answer easier, we copied the referential on the right.

For each of the two systems, compute the mean Precision and Recall measures (provide the results as fractions). Explain all the steps of your computation.

```
S1:
q1: P=4/8   R=4/4      q2: P=1/5   R=1/2
q3: P=3/9   R=3/5      q4: P=3/9   R=3/4

mean P(S1) = 41/120    mean R(S1) = 57/80

S2:
q1: P=1/5   R=1/4      q2: P=2/4   R=2/2
q3: P=3/5   R=3/5      q4: P=2/4   R=2/4

mean P(S1) = 9/20     mean R(S1) = 47/80
```

④ Explain how it is possible to compute Precision at different Recalls.

Force the system to ouput a given number of documents (increasing) so as to increase recall (ultimatly to recall max. when we ask the system to decidedfor all the available documents whether they are pertinent or not)

⑤ How is it possible to compute the average Precision/Recall curves? Explain in detail the various steps of the computation.

As it would be too tedious to compute the average Precision/Recall curves by hand, plot, on a Precision/Recall graph, the Precision and Recall values obtained in subquestion ③ for each of the two systems and for each of the 4 queries.

Based on the resulting curves, what is your relative evaluation of the two systems?

(1) Use average precision : for each relevant document compute the average precision for all relevant docs below rank Rk (see formula in course).

Then we can have difference precisions for different recall and plot these values.

(3) Ideal is in the top right corner.

S1 has better recall, S2 better precision. In general S2 performs slightly better.

⑥ The Precision/Recall based evaluation of the IR systems S1 and S2 above does not explicitly take into account the order in which the documents have been retrieved by the systems. For this purpose, another metric can be used: the Precision at k (`P@k`), which corresponds to the fraction of truly relevant documents among the top k documents retrieved by a system.

Compute the average `P@k` values for `k` between 1 and 5 for the IR systems S1 and S2 above. What additional insight do these values provide in addition to the Precision/Recall curves?

Based on these results, what is your relative evaluation of the two systems? How does it compare to ③?

	k	1	2	3	4	5
S1	q1	1	1	1	1	4/5
	q2	1	1/2	1/3	1/4	1/5
	q3	0	1/2	2/3	3/4	3/5
	q4	1	1/2	1/3	1/2	3/5
	P	3/4	5/8	1/12	10/16	11/20
S2	q1	0	0	0	0	1/5
	q2	0	0	1/3	1/2	2/5
	q3	0	0	1/3	1/2	3/5
	q4	0	1/2	1/3	1/2	2/5
	P	0	1/8	3/12	6/16	8/20

(2) Since higherly-ranked documents should have more relevance, we can see if a system can produce relevant results quicly in the first retrieved documents.

(3) S1 is better than S2 since it has a lot of relevant docs in the top results. This give a completely different view wrt former evaluation. S1 is better for web-like, S2 maybe for law-like (see Q8).

⑦ It is often desirable to be able to express the performance of an NLP system in the form of a single number, which is not the case when the Precision/Recall framework is used.

Indicate what scores can be used to convert Precision/Recall measures into a unique number. For each score, give the corresponding formula.

F score :

$$\frac{(b^2+1)\cdot P\cdot R}{b^2\cdot P+R}$$

When $b^2 > 1$ emphasizes P otherwise emphasies R.

Accuracy: ratio of correct results provided by the system (wrt total number of results from the system)

Error = 1-Accuracy

⑧ Give well chosen examples of applications that illustrate:

- a situation where more importance should be given to Precision;
- a situation where more importance should be given to Recall.

More importance should be given to precision in Web-like search applications because a few relevant document out of huge amount of possibly pertinent document is enough (large enough).

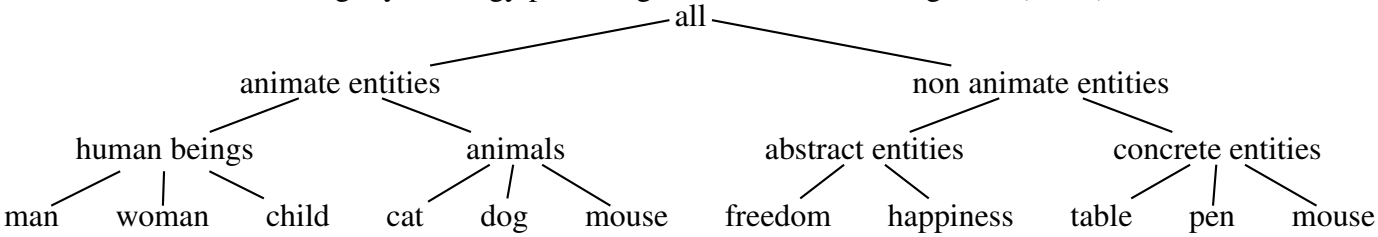
Recall shall be prefered in legal or medical-like search where exhaustivity (of correct documents) is important.

9 Lexical Semantics

Exercise IX.1

The objective of this question is to illustrate the use of a lexical semantics resource to compute lexical cohesion.

Consider the following toy ontology providing a semantic structuring for a (small) set of nouns:



① What is the semantic relation that has been used to build the ontology?

Cite another semantic relations that could also be useful for building lexical semantics resources.

For this semantic relation, give a short definition and a concrete example.

"is a" relation: *hyperonymy*

other: *meronymy* ("part of")

② The word "*mouse*" appears at two different places in the toy ontology. What does this mean?

What specific problems does it raise when the ontology is used?

How could such problems be solved? (just provide a sketch of explanation.)

semantic ambiguity (two different meanings, polysemy, homonymy(homography)).

Word Sense Disambiguation (WSD) through the information from the context (e.g. coehsion).

③ Consider the following short text:

```
Cats are fighting dogs. There are plenty of pens on the table.
```

What pre-processing should be performed on this text to make it suitable for the use of the available ontology?

identify surface forms (tokenization), maybe stopwords filtering, PoS tagging for nouns, lemmatization (or stemming).

④ We want to use lexical cohesion to decide whether the provided text consists of one single topical segment corresponding to both sentences, or of two distinct topical segments, each corresponding to one of the sentences.

Let's define the lexical cohesion of any set of words (in canonical form) as the average lexical distance between all pairs of words present in the set[3]. The lexical distance between any two

[3]Here, is actually the *lack of* cohesion that we measure: since it's a distance, the lower the more cohesion and the bigger the less cohesion.

words is be defined as the length of a shortest path between the two words in the available ontology.

For example, "*freedom*" and "*happiness*" are at distance 2 (length, i.e. number of links, of the path: happiness $\longrightarrow$ abstract entities $\longrightarrow$ freedom), while "*freedom*" and "*dog*" are at distance 6 (length of the path: freedom $\longrightarrow$ abstract entities $\longrightarrow$ non animate entities $\longrightarrow$ all $\longrightarrow$ animate entities $\longrightarrow$ animals $\longrightarrow$ dog)

Compute the lexical distance between all the pairs of words present in the above text and in the provided ontology (there are 6 such pairs).

```
D(cat,dog) = 2     D(cat,pen)=6     D(cat,table)=6
                   D(dog,pen)=6     D(dog,table)=6
                                    D(pen,table)=2
```

⑤ Compute the lexical cohesion of each of the two sentences, and then the lexical cohesion of the whole text.

Based on the obtained values, what decision should be taken as far as the segmentation of the text into topical segments is concerned?

```
D(S1) = 2    D(S2) = 2
D(S1,S2) = 1/6 ( 2+6+6+6+6+2) = 14/3
```

We here have a distance that we want to minimize, as the lower the distance, the more lexically coherent is the text (closer words in the ontology). We thus here decide to segment the text into two topical segments (one sentence each).

⑥ Give some examples of NLP tasks for which lexical cohesion might be useful. Explain why.

IR: find document based on lexical cohesion wrt query words.

automatic summarization: check coherence of extracted sentences

semantic disamguation of possibles choices in spelling error correction (e.g. bag or bug for "bxg")

WSD

Machine translation (semantic filter)