QUESTION I : Lexical Semantics [10 pt]

① **[2 pt]** Define what it means for two words to be homonyms.
Define what it means for a word to be polysemic.
When a dictionary is available, indicate how one can practically make the distinction between a word that has a homonym and a word that is polysemic.
Illustrate your answer by a concrete example.

Answers:

Two words are homonyms if they are spelled and pronounced the same, but do not have the same meaning.

A word is polysemous if it corresponds to multiple related meanings within a single lexeme.

Practically, homonyms have a different etymology and correspond to distinct entries in a dictionary, while a polysemic word corresponds to a single dictionary entry associated with several definitions (e.g. meanings).

Examples: "*bat*" (the flying mammal) and "*bat*" (the wooden club) are homonyms; "*crown*" is a polysemic word that can mean, for example, "*the headgear of a king*" or "*the highest part of a tree*".

② **[5 pt]** Apply the Aristotelian "*Genus-Differentia*" principle to provide suitable representations for the following meanings of the words "*triangle*" and "*circle*":

Triangle 1: A planar figure bounded by three straight lines, called the triangle edges.

Triangle 2: An instrument consisting of a metal body to be hit by a metal bar.

Circle 1: A planar figure consisting of all points that are equally distant from a given point, called the circle center.

Circle 2: A group of persons sharing a common interest.

Answer:

```
Triangle1 -(hyponym)-> Figure
Triangle1 -(holonym)-> Edge

Triangle2 -(hyponym)-> instrument

Circle1 -(hyponym)-> Figure
Circle1 -(holonym)-> Center

Circle2 -(hyponym)-> group
```

③ **[1 pt]** What is a synset?

Answer: A synset is a set of word forms that share a common meaning

④ **[2 pt]** Use the synset approach to provide suitable definitions for the 4 word meanings given in sub-question ②.

Answer:

```
Triangle1 : {triangle, figure}
Triangle2 : {triangle, instrument}
Circle1: {circle, figure}
Circle2 : {circle, group}
```

QUESTION II : App portfolio monitoring system [27 pt]

As the lead engineer in a company that is developing smartphone applications, you have been asked to put in place an "app portfolio monitoring system", i.e. an automated tool able to search various blogs and news feeds for user comments related to apps produced by your company, and, for each of such comments, determine whether it is positive or negative, and monitor this over time.

① **[8 pt]** The first problem you are faced with is to decide how to select the various user comments that should be analyzed by your system. For this, it has been decided:

Step 1: to set up a web crawler that explores a large set of web links and systematically retrieves all the documents that contain the word "*Company_X*" (the name of your company), and any of the words "*Application 1*", or "*Application 2*", or ..., that correspond to the names of the various applications developed by your company;

Step 2: to store all the documents retrieved by the crawler into the document collection of an Information Retrieval engine; and

Step 3: for each of the applications developed by the company, to ask the development team to write a set of natural language queries that should retrieve "relevant" comments, i.e. comments related to this application.

In your opinion, does the proposed approach make sense? What are (some of) its potential drawbacks? For each of the identified drawbacks, provide a concrete illustrative example and a hint on how the approach may be improved to take the considered drawback into account.

Answers: The approach may make sense, but all steps, as well a the general procedure, need to be improved to make it truly realistic.

General improvement suggestion: run a pilot experiment for the comment acquisition pipeline and perform an intermediate validation to get some real evidence about the potential problems (and thus associated solutions) before running the pipeline at a large scale.

Step 1:
The proposed retrieval mechanism is much too simplistic, and may generate unnecessarily huge amounts of data (in addition, there may be a lot of duplicates of quasi-duplicates):

(a) the choice of the web links to be used by the crawler is crucial and should be dealt with more sophistication:
$\longrightarrow$ for example, the set of the visited sites may be first bootstrapped by a carefully selected set of highly relevant links, and then progressively expanded through more advanced mechanisms such as searching the web for documents similar to the initially retrieved ones;

(b) selecting only the documents that contain the company and application name may lead to a very poor precision and recall:

- many such documents may not correspond to real comments (advertisements, side remarks, large documents, etc.) or may not be related to the targeted company application (especially is the company and/or application name correspond to common words, or if the users do not mention the company name *and* the application name in their comment); because of this, the precision may be too poor for the document retrieval procedure to be truly exploitable

$\longrightarrow$ more sophisticated techniques should be used to increase precision, e.g. using regular expressions, using the queries planned in step 3 already in step 1, using validated comments as queries, etc.

- many comments may be missed, as users may use other words (different from the "official name") to refer to the applications (abbreviations, indirect naming –e.g. that application that ...)
  $\longrightarrow$ more sophisticated techniques should be used to increase the recall (e.g. using the queries produced by the development teams, exploiting document similarities, identifying potential synonyms, etc.).

Step 2: The precise nature of the indexing used to store the acquired documents in the collection processed by the IR engine is crucial and is not specified in the description of this step. Using the "default" pre-processing and indexing procedures will probably not be sufficient and using various techniques such as regular expressions to extract indexing terms may strongly improve the achieved performance. Language identification may also be considered, as well as splitting the document collection into several ones, one per application, e.g. by using a clustering algorithm.

Step 3: The expertise requested from the development teams responsible for the different applications is exploited too late in the pipeline, and should probably be used much earlier, typically for improving step 1 (see above). Also, in the proposed approach, queries created for a given application may apply to comments for another if both application names are present.

Furthermore, the expertise of the development teams may be very biased, as they know their applications very well. In particular, the obtained queries may use a vocabulary very different from the one used by true users. It is also questionable to ask the development teams to produce queries independently. It may be more efficient to make them collaborate, as manually producing queries is very costly.

② **[3 pt]** The next issue is to find an automated way to decide whether a selected comment is positive or negative. You decide to first test the feasibility of this task by letting some human experts perform it "by hand". Concretely, for the best-selling application developed by your company (Application 1), you extract 100 relevant comments from the available comment collection, and, for each of these comments, you ask two human experts to decide whether it is "positive" or "negative". This annotation work leads to the following experimental data:

Application 1:

Expert1 \ Expert2	Positive	Negative	Total
Positive	35	25	60
Negative	15	25	40
Total	50	50	100

Based on these data, what is your initial opinion about the feasibility of the targeted task? Justify your answer by precisely indicating how you have used the data to build your opinion (What metrics did you compute? How did you interpret their values? Etc.).

Answer: The data are used to compute the Cohen's Kappa.

The obtained value is

$$\kappa = \frac{60/100 - (60/100 * 50/100 + 40/100 * 50/100}{100/100 - (60/100 * 50/100 + 40/100 * 50/100)} = \frac{0.6 - 0.5}{1 - 0.5} = 0.2$$

This value is very low and thus indicate that the inter annotator agreement is not good enough for considering the task as feasible.

③ **[6 pt]** To improve the annotations made by the human experts, the decision has been taken to ask the annotators to tag each of the comments to annotate, either as “positive” or “negative”, if they can reach a clear-cut decision, or as “neutral”, in all the other situations.

To test these new annotation instructions, you ask the two annotators to re-annotate the 100 comments about the best-selling application (Application 1), and, furthermore, you select two additional annotators, and ask them to annotate 100 user comments related to the second best-selling application (also extracted from the available user comment collection).

The annotation work performed by the four involved annotators leads to the following experimental data:

Application 1:

Expert1\ Expert2	Positive	Neutral	Negative	Total
Positive	25	10	5	40
Neutral	0	10	10	20
Negative	5	10	25	40
Total	30	30	40	100

Application 2:

Expert3\ Expert4	Positive	Neutral	Negative	Total
Positive	68	2	0	70
Neutral	2	15	3	20
Negative	0	3	7	10
Total	70	20	10	100

Based on these new experimental data, what is your final opinion about the feasibility of the targeted task? Again, justify your answer by precisely indicating how you have used the data to build your opinion (What metrics did you compute? How did you interpret their values?...).

Note: For this sub-question, you can use that $13/33 \simeq 0.4$ and $18/23 \simeq 0.8$.

Answer: The data are used to compute the Cohen’s Kappa.

For Application 1, we get: $\kappa = \frac{13}{33} \simeq 0.4$.

For application 2, we get: $\kappa = \frac{18}{23} \simeq 0.8$.

We can conclude that:

- introducing a "neutral" tag improves the inter-annotator agreement
- the IIA can be good enough for considering the annotation task as feasible for (at least) some applications (here Application 2)
- the average IAA being 0.6 the task may be considered as feasible.

In addition, we should also check the ratio of "neutral" comments, which should not be too high for the annotations to be exploitable. With the available data we get:

- for Application 2: ratio = 10%
- for Application 2: ratio = 15%

Therefore, the approach can be considered as potentially feasible/exploitable.

④ **[2 pt]** Finally, the decision is taken to implement a first version of the monitoring system for a subset of 5 applications. Assuming that experimental data of the same nature and size as in previous sub-question (③) are available for all the applications of your company, how should the 5 applications be selected?

Answer: Choose de 5 applications with top-5 kappa scores. The selection may be made more restrictive by imposing that the kappa score must be good enough and the proportion of neutral comments not too high. Alternatively, the applications with top-5 kappa scores among those for which there is enough comments. Alternatively, if another importance metric is available (popularity for example), take the top-5 for this metric that have a good enough kappa score.

⑤ **[3 pt]** To implement the initial version of the targeted monitoring system, you decide to take the following approach:

- the available 5x100 annotated comments are stored and indexed as a specific document collection of the available Information Retrieval system;
- each new comment is allocated the same tag (positive/negative/neutral) as the most relevant document retrieved from the document collection, when the new comment is used as a query.

Provided that the used IR system is implementing a Vector Space model with the tf.idf weighting scheme, describe the various preprocessing steps that should be applied for storing/indexing the comments in the document collection.

Answer:

- tokenization
- stemming or, better, lemmatization
- filtering based on stop-word lists or, better, on occurrence frequencies, and PoS speech tags
- desequentialization (bag of words)
- computation of the tf.idf weights

⑥ **[2 pt]** You now want to evaluate the available initial version of your monitoring system. For that, you decide to build a referential ("*Golden Truth*"). Precisely describe what such a referential should consist of.

Answer: We need a referential consisting of a large, representative sample of comments, each associated with the correct labeling consisting of:

- identifier of the application the comment is related to; and
- the correct positive, negative or neutral label

⑦ **[3 pt]** How can the raw accuracy that can be computed with the available referential be expressed in terms of a metric related to the underlying Information Retrieval system?

Answer: the raw accuracy is equal to the average precision@1 of the underlying IR system.

QUESTION III : n-grams of characters **[5 pt]**

Consider the following toy corpus (18 occurrences of 7 different characters):

the tents he rents

and a 3-gram model for character sequences. We assume that there is no other character in the alphabet than the above 7.

What is the estimated value of the parameter associated with the sequence starting at the 7-th character ('*n*') of the above toy corpus:

- when maximum-likelihood estimation is used?
- when "additive smoothing" with a Dirichlet prior with parameter $(0.01, ..., 0.01)$ is used?

Provide your answers in the form of a mathematical expression only using numerical values.
Fully justify your answer.

Answers: We are looking for all 3-grams. There are 16 $(= 18 - (3-1))$ of them. The fastest way to get them is to vertically write the sentence three times, each time shifted by one character (whitespaces are here denoted by an hyphen):

```
the
he-
e-t
-te
ten
ent
nts
ts-
s-h
-he
he-
e-r
-re
ren
ent
nts
```

Then count how many times the considered 3-gram ("*nts*") is present (2 times). Thus:

$$\widehat{P}_{\text{ML}}(nts) = \frac{2}{16}$$

There are 7^3 $(= 343)$ possible trigrams with this alphabet; thus:

$$\widehat{P}_{\text{Dir.}}(nts) = \frac{2 + 0.01}{16 + 7^3 \times 0.01} \left(= \frac{2.01}{19.43}\right)$$

QUESTION IV : Tokens and tags [17 pt]

① **[5 pt]** Consider the following sentence

Mrs. ⋮ *Wang* | *’s* | *son* | *Liu* ⋮ *Deng* | *expressed* | *visiting* | *U.S.* | *Secretary* ⋮ *of* ⋮ *State* | *George* ⋮
Shultz | *China* | *’s* | *whole-hearted* | *determination* | *to* | “ | *make* | *a* | *deal* | ” | .

In order to make your tokenization representative of *linguistically meaningful* words, how would you improve the trivial tokenization which considers as a token any character sequence separated by (but not including) whitespaces?
First provide your tokenization by adding vertical bars to separate your tokens in the above sentence, and then explain how to design an automated tokenizer able to produce such a tokenization.

Answer: Tokenization is highly dependant on the application, but here “*linguistically meaningfull*” entities were required. See above tokenization as an example (whitespaces not included, simply removed between tokens, kept inside tokens).

Comment: BTW, tokenization shall not remove anything.

W.r.t. trivial whitespace tokenization, there are two “issues”:

(a) split further (see | above);

(b) maybe join (see ⋮ above).

Splitting further may be done using a bigger **set of separators** (e.g. including ’"’ (double quotes); but not ’.’ (period) otherwise we’ll have more “joining” problems; e.g. “U.S.”, “Mrs.”, ...) and **ad-hoc sequences** (stored in a lexicon or regexp; e.g. “’s” (possessive), “’m” or “n’t”.

Joining may be based on the **lexicon(s)** (e.g. “Secretary of State”, “U.S.”, “China”, ...), *but* pay attention to ambiguities (e.g. “credit card”): only join **non-ambiguous** n-grams of tokens; or based on some more advanced *regular expressions* (e.g. for “Mrs.|M.|... X...”) — “pre-NER” I would say (pure NER, however, are much more advanced than a simple tokenizer and may use higher lever information, like e.g. PoS-tagger); but this later approach will be **difficult for proper nouns** in general (e.g. how to regroup “George” and “Shultz”, but not “State” and “George”, nor “Shultz” and “China”? (this later could be lexicon-based, assuming “China” is part of the lexicon...).

Notice that “*make a deal*” shall not be joined, as this is not a single token, it is not lexicalized: could also be “*makes a deal*”, “*made a deal*”, ...

② **[1 pt]** If your above tokenization (and no other preprocessing) is used, what would be the 4-token input of a CBoW neural-network targeting output token “*son*”?

Answer: the two preceeding and two following tokens; so, with the above tokenization (including jointures): "Mrs. Wang", "’s", "Liu Deng", "expressed".

③ **[3 pt]** For each of the tokens produced for the beginning of the sentence in sub-question ① up to “*U.S.*” (included), provide the corresponding PoS tag according to the following tagset (NLTk universal tagset):

ADJ (adjective), ADP (adposition), ADV (adverb), CONJ (conjunction), DET (determiner), NOUN (noun), NUM (numeral), PRT (particle), PRON (pronoun), VERB (verb), PUNCT (punctuation marks), X (other).

Answer:

Depends on your tokenization, but typically:

Mrs./NOUN Wang/NOUN 's/PRT son/NOUN Liu/NOUN Deng/NOUN expressed/VERB visiting/VERB U.S./NOUN

ADJ is tolerated for "visiting".

④ **[3 pt]** More generally, consider the application of a standard order-2 HMM PoS tagger on a sequence of tokens $w_1, w_2, \cdots, w_n$.
Provided that $n \geq 8$, and that w_4, w_5 and w_7 are non-ambiguous:

- is the PoS tagging of w_3 possibly dependent on the one of w_6? **Answer: NO**
- is the PoS tagging of w_6 possibly dependent on the one of w_8? **Answer: YES**

For both questions, provide a detailed justification of your answer:

Justification:

Because of Markov assumption of order 2, any sequence of 2 non-ambiguous tags will lead to conditionnal independance of the tags on its left and right; this is not the case, however, for shorter sequences of non-ambiguous tags:

$$\begin{aligned}\underset{t_i,t_{i+3}}{\text{Argmax}}\, P(t_i, t_{i+1}, t_{i+2}, t_{i+3}) &= \underset{t_i,t_{i+3}}{\text{Argmax}}\, P(t_i|t_{i+1}, t_{i+2}) \cdot P(t_{i+1}, t_{i+2}) \cdot P(t_{i+3}|t_i, t_{i+1}, t_{i+2}) \\ &= \underset{t_i,t_{i+3}}{\text{Argmax}}\, P(t_i|t_{i+1}, t_{i+2}) \cdot P(t_{i+3}|t_{i+1}, t_{i+2}) \\ &= \left(\underset{t_i}{\text{Argmax}}\, P(t_i|t_{i+1}, t_{i+2}), \underset{t_{i+3}}{\text{Argmax}}\, P(t_{i+3}|t_{i+1}, t_{i+2})\right)\end{aligned}$$

Comment: Too many students still confuse parameters with overall objective and wrongly claim that tags only depend on preceeding ones (see e.g. question 9 of quiz 2 this year).

⑤ **[2 pt]** A standard way of evaluating a PoS tagger is to use a referential to compute its raw accuracy. Is this approach still valid if the evaluated PoS tagger is allowed to associate more than one tag to each token (while there still is only one tag per token in the referential)?
Justify your answer.

Answer: No, it is undefined. How to decided when there is a match, which one to choose? (Too easy if we just say it's ok if tag is present: easy to cheat by always answering all the possible tags!)

⑥ **[3 pt]** Propose some metrics (at least one) that may be suitable for evaluating a PoS tagger in the situation mentioned in the previous sub-question (⑤). Justify your answer and provide a brief suitability analysis.
In particular, what would be the score achieved by a PoS tagger that would associate to each token all the tags defined in the used tag set?
In your answer, use K to refer to the size of the tag set, if necessary.

Answers: Decent metric should take the variability of the anwser into account. And of course it shall be effectively computable.

The simple metric which consist of accepting (= $+1$) any proposition which contains the correct tag is clearly overestimating the system: it'll have a score of 1 ($= 100\%$) on the proposed system. This is not suitable.

A more decent metric would be to estimate the chance to get the right tag, e.g. by averaging $1/$size of the answer whenever the proposed answer contains the correct tag (and 0 if not). Such a metric will score $1/K$ on the proposed system.

A variant is to count how many time the first proposed tag is correct (averaged P@1). It'll score also $1/K$ on the proposed system.
This score (averaged P@1) is ever more adapted if the tagger propositions are order by decreasing probabilities (from the tagger), when available...

QUESTION V : Spam filtering [17 pt]

We consider implementing a component of a spam email classifier, using NLP techniques applied to the email body. For this, we collected a corpus. After preprocessing we have 13'482 non-empty email bodies, consisting of 1'058'741 token occurrences from 26'964 different tokens.

① **[1 pt]** What is the size of the matrix representing the corpus?

Answer: 13482×26964

② **[4 pt]** In the above corpus, 2'696 emails (20%) are considered to be spam.

How would a Naive Bayes classifier classify the following (preprocessed) email body.

ski sun money

knowing that (up to some unique constant factor not mentioned here) we have:

$P(ski\|\texttt{Spam}) = 3$,	$P(ski\|\texttt{OK}) = 4$,
$P(sun\|\texttt{Spam}) = 6$,	$P(sun\|\texttt{OK}) = 5$,
$P(money\|\texttt{Spam}) = 7$,	$P(money\|\texttt{OK}) = 2$,
$P(\texttt{Spam}\|ski) = 8$,	$P(\texttt{Ok}\|ski) = 47$,
$P(\texttt{Spam}\|sun) = 10$,	$P(\texttt{Ok}\|sun) = 37$,
$P(\texttt{Spam}\|money) = 14$,	$P(\texttt{Ok}\|money) = 18$,

Fully justify your answer.

Answer: For `Spam`: $3 \times 6 \times 7 \times 0.2 = 25.2$

For `Ok`: $4 \times 5 \times 2 \times 0.8 = 32$

It will thus be classified as `Ok`.

Comment: Don't forget priors!

③ **[4 pt]** In order to better understand your corpus, you plan to cluster it using dendrograms. To do so:

- you represent each email body by the empirical probability distribution over the tokens it contains (simply estimated by relative frequencies);
- and make use of the Hellinger distance.

What is the distance between the following two email bodies:

email 1: *ski sun money sun*

email 2: *sun ibm sun apple money sun money sun*

Express this distance as a reduced fraction with a square root; and explain the main steps of your computation.

Answer:

The non-zero components of two vectors are (order: *ski, sun, money, ibm, apple*):

email 1: $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}, 0, 0)$ email 2: $(0, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$

The Hellinger distance between the two is then:

$$\sqrt{\frac{1}{4} + \frac{2}{8}} = \frac{1}{\sqrt{2}}$$

Note: some students might have $\frac{1}{\sqrt{2}}$ times the above (i.e. $\frac{1}{2}$) as this is another definition of the Hellinger distance in the discreate case (multiplying by a global $\frac{1}{\sqrt{2}}$ factor).

④ **[3 pt]** You run the dendrogram clustering algorithm using complete linkage. At some point, it reaches a state where what remains to be clustered are the two clusters, G_1 and G_2, that have already been build so far, and two email bodies, B_1 and B_2. Here are the distances between each of them:

	B_1	B_2	G_1	G_2
B_1	0	0.7	0.6	0.2
B_2	0.7	0	0.5	0.3
G_1	0.6	0.5	0	0.4
G_2	0.2	0.3	0.4	0

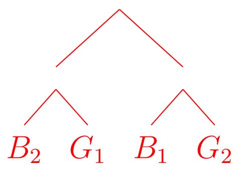
Draw the dendrogram corresponding to the final clustering (using complete linkage). Explain your steps.

Answer:

The two closest ones are B_1 and G_2 which will thus be merged in a new cluster, let's say G_3, the distances of which with the other two are (complete linkage):

- with B_2: 0.7
- with G_1: 0.6

The new closest group thus consist of B_2 and G_1, ending up in the following tree:



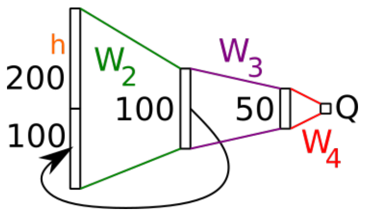
⑤ **[5 pt]** Finally, we'd like to consider a simple neural-network approach made of:

- a CBoW word embedding of size 200 with a 1-token context;
- a recurrent neural-network, fully recurrent;
- a feed-forward neural-network (a.k.a. MLP) for final classification, with:
  - input size: 100;
  - one hidden layer of size 50;
  - softmax output layer.

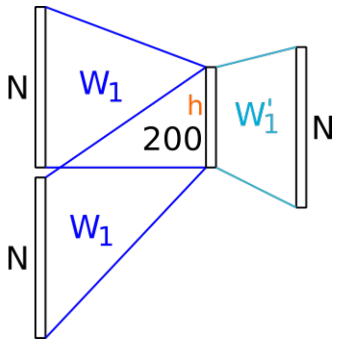
Draw the *full* architecture used for *classifying* new documents. Specify the size of each layer.

What are the parameters to be learned? Explain the size/number of each group of parameters.

Answers:



where Q the size of the output is certainly 1 (probability of one of the two classes), but $Q = 2$ is tolerated (softmax representing the two probabilities (which sums up to 1 anyway...)), and h is a word embedding which result from a CBoW learning (prior to classification usage):



Parameters are:

- W_1 of size $N \times 200$ where N is the size of the indexing vocabulary (26'964 at the beginning of the question, but this seems still a bit too high; maybe some filtering/selection shall be applied);
- (W_1' of size $200 \times N$)
- W_2 of size 300×100
- W_3 of size 100×50
- W_4 of size $50 \times Q$

QUESTION VI : Spelling-error correction [9 pt]

① **[6 pt]** What is the edit distance between "*mmaym*" and "*mummy*"? Fully justify your answer.

Answer: The first thing to do is to specify the transformation set! If considering *insertion*, *deletion*, *substitution* and *transposition*, then distance is **3** as shown by this chart:

		m	u	m	m	y
	0	1	2	3	4	5
m	1	0	1	2	3	4
m	2	1	1	1	2	3
a	3	2	2	2	2	3
y	4	3	3	3	3	2
m	5	4	4	3	3	3

If transposition is not considered, the distance is 3 as well.

If only insertion and deletion are considered, the distance is 4 (but then cell $(2, 2)$ is 2, not 1).

Comment: Providing only one possible sequence of n transformations from one string to the other only proves that the distance is smaller or equal to n, not that it is equal to n.

② **[3 pt]** If the lexicon consists only of

```
mammoth
mammy
mum
mumble
mummer
mummy
mumsy
munch
my
mycelium
nag
```

what would be the prefix string considered just after the prefix string "*mumb*", when correcting "*mmaym*" at distance 1, with the algorithm presented in the course?

Justify your answer (you may exploit of your answer to the previous sub-question).

Answer: From the above chart, we can see that the cut-off edit distance between "*mmaym*" and "*mum*" is 1, thus indeed "*mumb*" will be considered; and furthermore we can easily deduce (from the above chart) that the cut-off edit distance between "*mmaym*" and "*mumb*" is 2 (the same as between "*mmaym*" and "*mumm*"); thus this exploration branch will be over ($2 > 1$), moving to the next branch, thus considering adding the next character ('*m*', prefix of "*mummer*") to be added to "*mum*", leading to ***"mumm"*** .

QUESTION VII : SCFGs and all the rest [6 pt]

① **[4 pt]** What is the purpose of using a SCFG in NLP?
What are the main key differences between n-gram-based language models and SCFG-based parsing?

Answer:

- Purpose: to do syntactic analysis of sentences, providing most-probable syntactic parse tree for any correct sentence; it also provides a language model as the sum of the probability of all possible parse trees for a given sentence.
- They both provide a probabilistic measure on sequences of words; but n-grams compute this probability based only on the words (sequence), where the SCFG compute it using **structures** (grammar; see former point); SCFG thus requires more ressources (grammar);
- in other words, n-grams can only be used as a *probabilitic* **accepter**, while SFCGs could also be used as an **analyzer** (provide syntactic structures = parse trees);
- n-grams are a **regular** language, whereas SCFG is **context-free** (allowing "embeddings"); the computationnaly complexity of n-gram is thus lower (linear) than the one of SCFG parsing (cubic). Another simpler way to say this is that n-grams deal only with local dependencies (n words) whereas SCFGs deal with broader dependencies (whole sentences, as defined by the grammar).

② **[2 pt]** What should a lexicon contain to be usable for both PoS tagging and SCFG parsing? *Justify* your design.

Answer: It should at least contain the **surface forms** (=words=terminals) associated with their parameters, $P(w|T)$ for PoS tagging and $P(T \rightarrow w)$ for SCFG. Notice that these two are in fact exactly of the same nature (sums up to 1 over all *words*, for a given T) and could thus, if forced to be unified (e.g. estimated the same way), be the same field of the lexicon.

QUESTION VIII : SCFGs per se [15 pt]

Consider the following SCFG *excerpt*:

```
S -> NP VP          (0.7)        VP -> VP NP        (0.3)
                                 VP -> VP PNP       (0.3)
NP -> PrN           (0.1)        VP -> V            (0.1)
NP -> NP PNP        ( a )
NP -> Det N         (0.1)        PNP -> Prep NP     (0.9)
NP -> Det Adj N     (0.2)
```

where `PrN`, `N`, `Det`, `Adj`, `Prep` and `V` are pre-terminals (`PrN` refers to "proper noun").

① **[1 pt]** Provide the maximal possible value range for `a`.

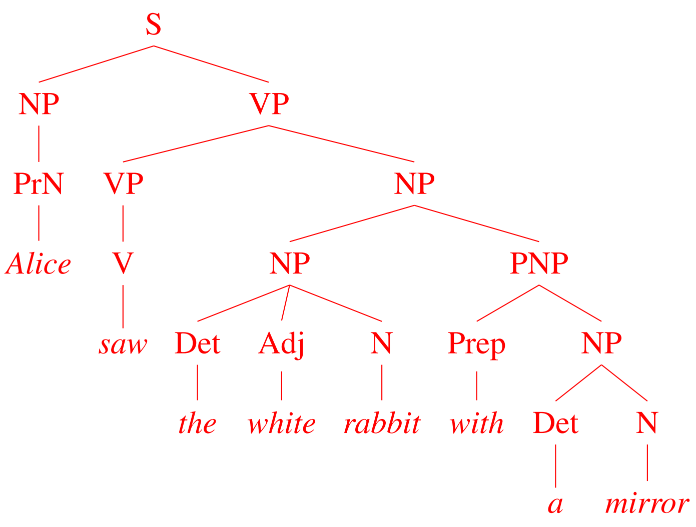
Answer: $(0; 0.6]$

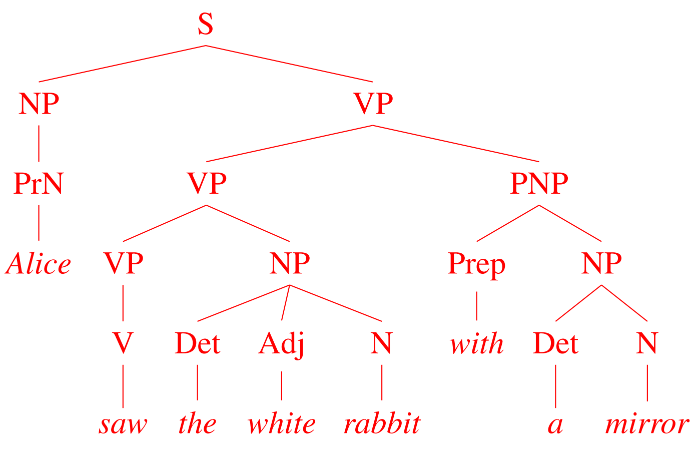
② **[3 pt]** Using the above SCFG excerpt, draw ***one*** of the parse trees for the sentence:

Alice saw the white rabbit with a mirror

(where each word is associated with the proper part-of-speech).

Answer: One of the following two trees:





③ **[3 pt]** What is the probability of the parse tree you draw? Provide the answer as a product of meaningful terms. Introduce (and explain) new notations if needed.

Answer:

first tree: $0.7 \times 0.1 \times 0.3 \times 0.1 \times a \times 0.2 \times 0.9 \times 0.1 \times L$

second tree: $0.7 \times 0.1 \times 0.3 \times 0.1 \times 0.3 \times 0.2 \times 0.9 \times 0.1 \times L$

where L is the product of the probas of the lexical rules.

④ **[8 pt]** Assume that for the following sentence:

time flies like an arrow

the output of a (unspecified) parser are the 4 objects on the top of the next page:

```
(S
  (VP
    (VP
      (VP (V time))
      (NP (N flies))
    )
    (PNP
      (Prep like)
      (NP (Det an) (N arrow))
    )
  )
)
```

```
(S
  (NP (N time))
  (VP
    (VP (V flies))
    (PNP
      (Prep like)
      (NP (Det an) (N arrow))
    )
  )
)
```

```
(S
  (NP
    (Adj time)
    (N flies)
  )
  (VP
    (VP (V like))
    (NP
      (Det an)
      (N arrow)
    )
  )
)
```

```
(S
  (VP
    (VP (V time))
    (NP
      (NP (N flies))
      (PNP
        (Prep like)
        (NP (Det an) (N arrow))
      )
    )
  )
)
```

How would the corresponding information be represented in the chart of a CYK analyzer? Answer by filling the chart below (not drawing the inner pointers):

S (4 pointers) VP (2 pointers)				
	NP VP			
		VP PNP		
NP VP			NP	
VP NP Adj V N	VP NP V N	VP Prep V	Det	N
time	flies	like	an	arrow

Pay attention **NOT** to have an exponential version of CYK (multiple S in top cell).