

COMP 6721 Applied Artificial Intelligence (Summer 2023)

Assignment #10: Introduction to NLP

Solutions

Question 1 Assume that we are working with the Shloutan language. If you don't know Shloutan, don't worry; it is a simple language made of only 5 words: loola nikee aloka bibi vo.

You want to build a word language model for Shloutan. The training corpus that you use is the following:

“Loola nikee. Aloka bibi vo. Vo bibi loola. Loola nikee bibi vo. Vo. Vo. Aloka bibi loola. Loola aloka aloka. Loola loola. Nikee nikee nikee. Bibi vo. Bibi vo. Vo Vo. Nikee loola.”

You can ignore case distinctions and sentence boundaries when answering the following questions.

(a) What is the value of $P(\text{vo} \mid \text{bibi})$?

$$P(\text{vo} \mid \text{bibi}) = \frac{\text{freq}(\text{bibi vo})}{\sum_{w_i} \text{freq}(\text{bibi } w_i)} = \frac{4}{6}$$

(b) What is the value of $P(\text{bibi vo})$?

$$P(\text{bibi vo}) = \frac{\text{freq}(\text{bibi vo})}{\sum_{w_j} \sum_{w_i} \text{freq}(w_j w_i)} = \frac{4}{32}$$

- (c) Build a bigram language model based on this training corpus. Show the frequencies and the probabilities for each bigram.

Frequencies

	loola	nikee	aloka	bibi	vo
loola	3	3	1	0	0
nikee	1	2	1	2	0
aloka	1	0	1	2	0
bibi	2	0	0	0	4
vo	0	1	1	2	5

Conditional Probabilities

	loola	nikee	aloka	bibi	vo
loola	0.43	0.43	0.14	0	0
nikee	0.17	0.33	0.17	0.33	0
aloka	0.25	0	0.25	0.50	0
bibi	0.33	0	0	0	0.67
vo	0	0.11	0.11	0.22	0.56

- (d) Smooth your bigram language model using “add 0.5”. Show the frequencies and the probabilities for each bigram.

Frequencies

	loola	nikee	aloka	bibi	vo
loola	3.5	3.5	1.5	0.5	0.5
nikee	1.5	2.5	1.5	2.5	0.5
aloka	1.5	0.5	1.5	2.5	0.5
bibi	2.5	0.5	0.5	0.5	4.5
vo	0.5	1.5	1.5	2.5	5.5

Conditional Probabilities

	loola	nikee	aloka	bibi	vo
loola	0.37	0.37	0.16	0.05	0.05
nikee	0.18	0.29	0.18	0.29	0.06
aloka	0.23	0.08	0.23	0.38	0.08
bibi	0.29	0.06	0.06	0.06	0.53
vo	0.04	0.13	0.13	0.22	0.48

- (e) Using each language model from parts (c) and (d), which of the following two sentences is more probable. Show all your work.

sentence 1: Aloka vo nikee aloka.

Bigrams: (aloka vo), (vo nikee), (nikee aloka)

Model (c): $0 \times 0.11 \times 0.17 = 0$

Model (d): $0.08 \times 0.13 \times 0.18 \approx 0.0019$

sentence 2: Vo nikee nikee aloka.

Bigrams: (vo nikee), (nikee nikee), (nikee aloka)

Model (c): $0.11 \times 0.33 \times 0.17 \approx 0.0062$

Model (d): $0.13 \times 0.29 \times 0.18 \approx 0.0068$

Sentence 2 is more probable using either model.

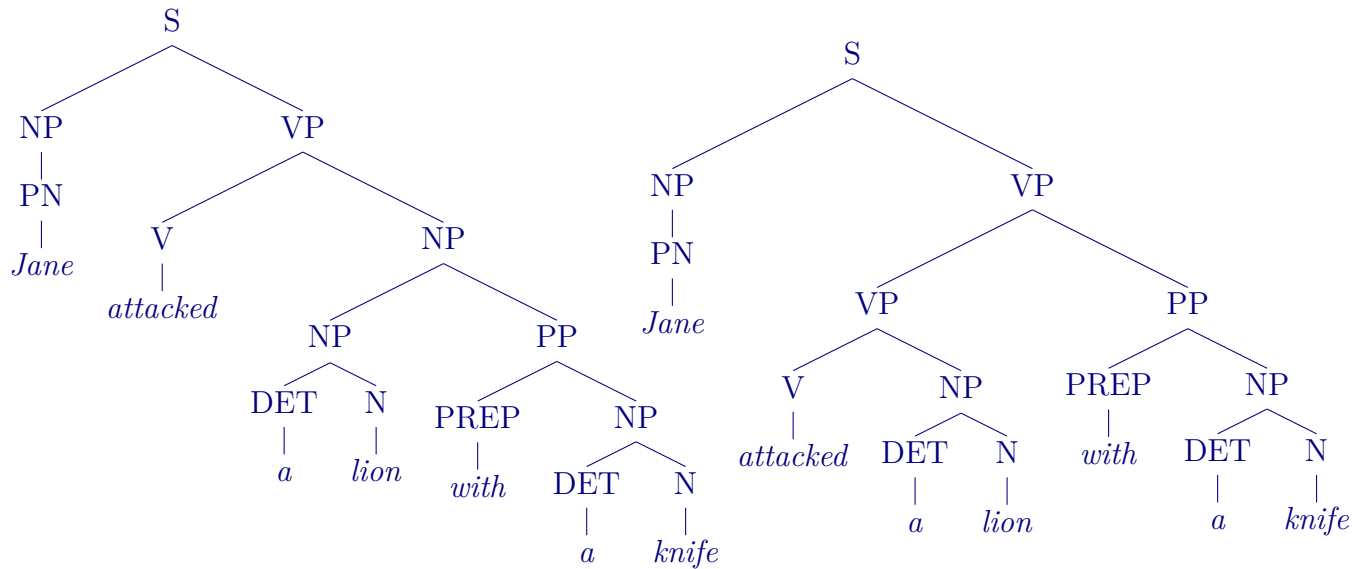
Question 2 Consider the following context-free grammar:

$S \rightarrow NP VP$	$N \rightarrow \text{lion}$
$VP \rightarrow V$	$N \rightarrow \text{knife}$
$VP \rightarrow V NP$	$N \rightarrow \text{zoo}$
$VP \rightarrow VP PP$	$V \rightarrow \text{attacked}$
$NP \rightarrow \text{DET } N$	$PN \rightarrow \text{Jane}$
$NP \rightarrow PN$	$\text{DET} \rightarrow \text{the}$
$NP \rightarrow NP PP$	$\text{DET} \rightarrow \text{a}$
$PP \rightarrow \text{PREP } NP$	$\text{PREP} \rightarrow \text{with}$
	$\text{PREP} \rightarrow \text{in}$

(a) Generate all possible parse trees for the following sentence:

i *Jane attacked a lion with a knife*

There are two possible parse trees:



Question 3 Consider the following sentences as training set for disambiguating the sense of the word *light*.

Sentence	Sense
There is a ray of light even if it is dim.	Sense1
This faint ray of light is barely enough to see.	Sense1
It needs to be light in weight to be portable.	Sense2
I can see dim red light coming from that room.	Sense1
This laptop is light enough to carry around.	Sense2
I am not hurt since I was hit by something as light as a candle.	Sense2
There is a faint hint of light at the end of the street.	Sense1
But the room was bright and light from the candle was warm as it fell on the surroundings.	Sense1
My bright red overcoat is light but fairly warm.	Sense2
This red candle stand is light enough to carry it to the room.	Sense2

Further consider this list of stop words:

a, and, are, as, at, be, for, from, in, is, it, if, of, that, this, the, to, was

Using a Naive Bayes approach with:

- a context window of ± 3 words,
- a vocabulary with the size of 31,
- smoothing with the value of 0.2, and
- stop-word removal

Calculate the scores of each possible sense and find the most probable sense of the word *light* in the following sentence:

In her dark red room, that dim light from the candle was warm enough.

The sentence with the word to disambiguate-

~~In~~ her dark red room, ~~that~~ dim **light** ~~from the~~ candle ~~was~~ warm enough.

Note: The stop words have been depicted as strike outs, since they will not be considered in predicting the sense. The words in the box are the words that are a part of the sentence and lie within the context window. The probabilities of these words for each sense in the training set will be considered.

Each of the words in the box above is also highlighted (boxed) in each of the training sentences below, when it lies within the context window of the word we are trying to disambiguate:

~~There is a ray of~~ **light** ~~even if it is~~ dim.

~~This faint ray of~~ **light** ~~is barely~~ enough ~~to see~~.

~~It needs to be~~ **light** ~~in weight to be~~ portable.

I can see dim red **light** coming from that room.

This laptop is **light** enough to carry around.

I am not hurt since I was hit by something as **light** as a candle.

There is a faint hint of **light** at the end of the street.

But the room was bright and **light** from the candle was warm as it fell on
the surroundings.

My bright red overcoat is **light** but fairly warm.

This red candle stand is **light** enough to carry it to the room.

$$P(\text{Sense1}) = 5/10 = 1/2 = 0.5$$

$$P(\text{Sense2}) = 5/10 = 1/2 = 0.5$$

$$P(\text{red}|\text{Sense1}) = \frac{1 + 0.2}{25 + (31 \times 0.2)} = \frac{1.2}{31.2} = 0.038461538$$

$$P(\text{room}|\text{Sense1}) = \frac{2 + 0.2}{25 + (31 \times 0.2)} = \frac{2.2}{31.2} = 0.070512821$$

$$P(\text{dim}|\text{Sense1}) = \frac{2 + 0.2}{25 + (31 \times 0.2)} = \frac{2.2}{31.2} = 0.070512821$$

$$P(\text{candle}|\text{Sense1}) = \frac{1 + 0.2}{25 + (31 \times 0.2)} = \frac{1.2}{31.2} = 0.038461538$$

$$P(\text{warm}|\text{Sense1}) = \frac{1 + 0.2}{25 + (31 \times 0.2)} = \frac{1.2}{31.2} = 0.038461538$$

$$P(\text{enough}|\text{Sense1}) = \frac{1 + 0.2}{25 + (31 \times 0.2)} = \frac{1.2}{31.2} = 0.038461538$$

$$P(\text{red}|\text{Sense2}) = \frac{2 + 0.2}{23 + (31 \times 0.2)} = \frac{2.2}{29.2} = 0.075342466$$

$$P(\text{room}|\text{Sense2}) = \frac{1 + 0.2}{23 + (31 \times 0.2)} = \frac{1.2}{29.2} = 0.04109589$$

$$P(\text{dim}|\text{Sense2}) = \frac{0 + 0.2}{23 + (31 \times 0.2)} = \frac{0.2}{29.2} = 0.006849315$$

$$P(\text{candle}|\text{Sense2}) = \frac{2 + 0.2}{23 + (31 \times 0.2)} = \frac{2.2}{29.2} = 0.075342466$$

$$P(\text{warm}|\text{Sense2}) = \frac{1 + 0.2}{23 + (31 \times 0.2)} = \frac{1.2}{29.2} = 0.04109589$$

$$P(\text{enough}|\text{Sense2}) = \frac{2 + 0.2}{23 + (31 \times 0.2)} = \frac{2.2}{29.2} = 0.075342466$$

$$\text{score}(\text{Sense1})$$

$$= \log(\text{Sense1}) + \log(P(\text{red}|\text{Sense1})) + \log(P(\text{room}|\text{Sense1})) + \log(P(\text{dim}|\text{Sense1})) +$$

$$\log(P(\text{candle}|\text{Sense1})) + \log(P(\text{warm}|\text{Sense1})) + \log(P(\text{enough}|\text{Sense1}))$$

$$= \log(0.5) + \log(0.038461538) + \log(0.070512821) + \log(0.070512821) + \log(0.038461538) +$$

$$\log(0.038461538) + \log(0.038461538)$$

$$= -8.264387214$$

$$\begin{aligned}
& \text{score}(\text{Sense2}) \\
&= \log(\text{Sense2}) + \log(P(\text{red}|\text{Sense2})) + \log(P(\text{room}|\text{Sense2})) + \log(P(\text{dim}|\text{Sense2})) + \\
&\log(P(\text{candle}|\text{Sense2})) + \log(P(\text{warm}|\text{Sense2})) + \log(P(\text{enough}|\text{Sense2})) \\
&= \log(0.5) + \log(0.075342466) + \log(0.006849315) + \log(0.04109589) + \log(0.075342466) + \\
&\log(0.04109589) + \log(0.075342466) \\
&= -8.606666574
\end{aligned}$$

Since $\text{score}(\text{Sense1}) > \text{score}(\text{Sense2})$, **Sense1** is more probable.

Question 4 Preprocessing text is a very crucial step for any NLP related task. There are many frameworks (GATE¹, UIMA²) and libraries (CoreNLP³, NLTK⁴, spaCy⁵) available out there to facilitate text analysis.

Today we will be looking into **spaCy** to get you started with the basic steps.

Installing spaCy. Depending on your environment, use *one* of the following:
pip: Before installing **spaCy** lets first make sure your **pip** and **setuptools** are up-to-date and install them:

```
pip install -U pip setuptools wheel
pip install -U spacy
```

conda:

```
conda install -c conda-forge spacy
```

After the installation, execute the following command to verify that you have installed **spaCy** version 3 or upper.

```
python -m spacy info
```

Downloading language models. Not all languages are preprocessed in the same way. Depending on the language, tokenization, sentence splitting, POS tagging, etc. varies. To facilitate analyzing different languages and genres, **spaCy** provides different pre-trained language pipelines to work with.⁶ For starters, we will be working with the basic English model. We can download the model with the following command:

```
python -m spacy download en_core_web_sm
```

Let's start coding! Use the following to import the **spaCy** library and load the language model:

```
import spacy
nlp = spacy.load("en_core_web_sm")
```

Now let's try to preprocess the text in your worksheet and check if the POS tags we assigned are what **spaCy** returns as well:

```
doc = nlp("I prefer a direct flight to Chicago.")
```

¹<https://gate.ac.uk/>

²<http://uima.apache.org/>

³<https://stanfordnlp.github.io/CoreNLP/index.html>

⁴<https://www.nltk.org/book/>

⁵<https://spacy.io/usage/spacy-101>

⁶<https://spacy.io/usage/models>

Depending on the language model and sentence provided as input, **spaCy** outputs an object **doc**, in our case with a variety of annotations encoded within. Linguistic annotations are available as token attributes. Let's try to print some basic attributes:

```
for token in doc:
    print(token.text, token.lemma_, token.tag_, token.dep_)
```

- (a) Now try to process the following paragraph to see how **spaCy** performs sentence splitting:

“Superman was born on the planet Krypton and was given the name Kal-El at birth. As a baby, his parents sent him to Earth in a small spaceship moments before Krypton was destroyed in a natural cataclysm. His ship landed in the American countryside, near the fictional town of Smallville. He was found and adopted by farmers Jonathan and Martha Kent, who named him Clark Kent. Clark developed various superhuman abilities, such as incredible strength and impervious skin. His adoptive parents advised him to use his abilities for the benefit of humanity, and he decided to fight crime as a vigilante. To protect his privacy, he changes into a colorful costume and uses the alias "Superman" when fighting crime. Clark Kent resides in the fictional American city of Metropolis, where he works as a journalist for the Daily Planet. Superman's supporting characters include his love interest and fellow journalist Lois Lane, Daily Planet photographer Jimmy Olsen and editor-in-chief Perry White. His classic foe is Lex Luthor, who is either a mad scientist or a ruthless businessman, depending on the story.”

Start by processing the text:

```
doc = nlp(text)
```

This performs *sentence splitting* as part of the analysis pipeline. Now add code to print all the resulting sentences.

```
sentence_spans = list(doc.sents)
for sentences in sentence_spans:
    print(sentences)
```

- (b) In addition to the constituent parse trees discussed in the lecture, dependency parsing⁷ is another formalism in the family of English grammars. In dependency parse trees, two tokens are connected by a single arc, where the label of the arc is the dependency relation. The start node of the arc

⁷See <https://web.stanford.edu/~jurafsky/slp3/14.pdf> and <https://spacy.io/usage/linguistic-features#dependency-parse> for more details on dependency parse trees

is known as the *Governor* (*head* in **spaCy**) and the token at which the arc points at is known as the *Dependant* (*child* in **spaCy**). A *Dependant/Child* can only have one *Governor/Head* but the vice versa is not true.

Now, try to use **spaCy**'s “displaCy visualizer”⁸ to visualize the dependency parse tree for the sentence given in your worksheet.

```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("I prefer a direct flight to Chicago.")
displacy.serve(doc, style="dep")
```

You should be able to visualize the graph by clicking the link from the run command window or by opening your browser at <http://0.0.0.0:5000>

As depicted in the graph, the token which has no arrows points pointed at (or the token from which the arcs starts) is known as the *root*, which is the main action in a sentence. Then, **nsubj** (i.e., *noun subject*) refers to the main subject of the sentence, which is “I” in our case and **dobj** (i.e., *direct object*) refers to the object that we are talking about in a sentence; here it is about “flight”. The tokens “a” and “direct” are *determiners* (**det**) and *adjective modifiers* (**amod**), respectively; hence, for the token “flight” you see the arc starting from here.

Through a dependency graph, you can see how tokens are connected with one another and you can extract the main subject, object and the action given a sentence, no matter how long it is. This is an important aspect in many language related tasks, for example to summarize a text.

⁸See <https://spacy.io/usage/visualizers>