CS918: LECTURE 3

Introduction to Language Models

Arkaitz Zubiaga, 10th October, 2018

LECTURE 3: CONTENTS

- Statistical language models.
  - N-grams.
  - Estimating probabilities of n-grams.
- Evaluation and perplexity.

N-GRAMS

- **N-gram:** sequence of n words.

- e.g. I want to go to the cinema.
  - 2-grams (bigrams): I want, want to, to go, go to, to the,…
  - 3-grams (trigrams): I want to, want to go, to go to,…
  - 4-grams: I want to go, want to go to, to go to the,…
  - …

STATISTICAL LANGUAGE MODELLING

WARWICK

- **Statistical language model:** probability distribution over sequence of words.
- *** I want to *** → high probability, common sequence in English
- *** want I to *** → low probability or zero

STATISTICAL LANGUAGE MODELLING

WARWICK

- How are these probability distributions useful?
  - **Machine translation:** P("please, call me") > P("please, call I")
  - **Spelling correction:**
    "its 5pm now" → correct to "it's 5pm now", higher P
  - **Speech recognition:**
    P("I saw a van") >> P("eyes awe of an")
  - **Smart compose:**
    You're typing "see you", the system suggests the next word: "tomorrow"

STATISTICAL LANGUAGE MODELLING

WARWICK

- Probability of sequence of words (W):

  $$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

- Also, we often look at probability of upcoming word:

  $$P(w_4 | w_1, w_2, w_3)$$

- Both of the above are known as **language models**.
  - i.e. how likely is a sequence of words?

HOW DO WE COMPUTE P(W)?

- How likely is the following sequence?

  P(I, found, two, pounds, in, the, library)

- We can rely on the **Chain Rule of Probability.**

THE CHAIN RULE OF PROBABILITY

WARWICK

- Definition of the rule:

$$\mathrm{P}(A, B) = \mathrm{P}(B \mid A)\mathrm{P}(A)$$

- More variables:

$$\mathrm{P}(A_1, A_2, A_3, A_4) = \mathrm{P}(A_4 \mid A_3, A_2, A_1) \cdot \mathrm{P}(A_3 \mid A_2, A_1) \cdot \mathrm{P}(A_2 \mid A_1) \cdot \mathrm{P}(A_1)$$

- Generalisation of the rule:

$$\mathrm{P}(A_1, \ldots, A_n) = \mathrm{P}(A_n | A_{n-1}, \ldots, A_1) \cdot \mathrm{P}(A_{n-1}, \ldots, A_1)$$

THE CHAIN RULE OF PROBABILITY

WARWICK

- Definition of the rule:

$$P(A, B) = P(B \mid A)P(A)$$

-

Let's say we have the following sentences to learn our language models:

see what I found
you found a penny
it has been found
the book you found
you came yesterday

What is the probability of "you found"?

P(you, found) = P(found | you) * P(you)

COMPUTING P(W) USING THE CHAIN RULE

- P(I, found, two, pounds, in, the, library) =

  P(I) *
  P(found | I) *
  P(two | I found) *
  P(pounds | I found two) *
  P(in | I found two pounds) *
  P(the | I found two pounds in) *
  P(library | I found two pounds in the)

AND HOW DO WE COMPUTE PROBABILITIES?

- Diving the number of occurrences?

  P(library | I found two pounds in the) =

$$\frac{\text{count(I found two pounds in the library)}}{\text{count(I found two pounds in the)}}$$

- **Problem:** there are so many different sequences, we won't observe enough instances in our data!

MARKOV ASSUMPTION

WARWICK

- **Approximate** the probability **by simplifying** it:

  P(library | I found two pounds in the) ≈ P(library | the)

  (first-order Markov assumption)

- Or:

  P(library | I found two pounds in the) ≈ P(library | in the)

  (2nd-order Markov assumption)

- It's much **more likely** that we'll observe "the library" or "in the library" in our training data.

MARKOV ASSUMPTION

WARWICK

- **Which we can generalise as:**

  $P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-k}, w_{i-k+1}, \ldots, w_{i-1})$

  i.e., we will only look at the **last *k* words**.

  (k^{th}-order Markov assumption)

COMPUTING PROBABILITIES

- Let's say we have the following sentences to learn our language models:

  see what I found
  you found a penny
  it has been found
  the book you found
  you came yesterday

What is the probability of "you found"?

With the 1st-order Markov assumption:

$$P(\text{you, found}) = P(\text{found} \mid \text{you})$$

COMPUTING PROBABILITIES

- Let's say we have the following sentences to learn our language models:

  see what I found
  you found a penny
  it has been found
  the book **you found**
  you came yesterday

What is the probability of "you found"?

With the 1st-order Markov assumption:

P(you, found) = P(found | you)

= count(you found) / count(you) = 2/3

LANGUAGE MODELS

- We can go with bigrams, trigrams, 4-grams,…

- Note: the longer the length:
  - The **more detailed** our language model.
    i.e. long sequences will capture more grammar than short sequences.

  - But **the more sparse** our counts.
    i.e. many observations only seen once.

- **Note:** it's not always perfect, e.g.:

  My request for using the department's supercomputer to run experiments next month is approved.

  "month is approved" → unlikely
  "request … approved" → more likely, not captured by n-grams

- **N-grams** are however often a **good solution**.

ESTIMATING N-GRAM PROBABILITIES

ESTIMATING BIGRAM PROBABILITIES

- **Maximum Likelihood Estimate (MLE):**

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

e.g.

$$P(the \mid in) = \frac{count("in\ the")}{count("in")}$$

AN EXAMPLE OF MLE COMPUTATION

- For the following sentences:

  <s> Yes, I am watching the TV right now </s>

  <s> The laptop I gave you is very good </s>

  <s> I am very happy to be here </s>

AN EXAMPLE OF MLE COMPUTATION

WARWICK

- For the following sentences:

  <s> Yes, **I am** watching the TV right now </s>

  <s> The laptop **I** gave you is very good </s>

  <s> **I am** very happy to be here </s>

- P(am | I) = count("**I am**") / count("**I**") = 2 / 3 = 0.667

AN EXAMPLE OF MLE COMPUTATION

WARWICK

- For the following sentences:

  <s> Yes, I am watching the TV right now </s>
  <s> The laptop I gave you is very good </s>
  <s> I am very happy to be here </s>

- P(I | <s>) = count("<s> I") / count("<s>") = 1 / 3 = 0.333

AN EXAMPLE OF MLE COMPUTATION

- For the following sentences:

  <s> Yes, I am watching the TV **right now** </s>
  <s> The laptop I gave you is very good </s>
  <s> I am very happy to be here </s>

- P(now | right) = count("**right now**") / count("**right**") = 1 / 1 = 1

Note: we have a very small corpus, so our system is learning that "right" is ALWAYS followed by "now"

EXAMPLE OF BIGRAM COUNTS

WARWICK

- Sample of bigrams counted in a corpus of 9,222 sentences.

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

“i” is followed by another “i” on 5 occasions

“to” is followed by “eat” on 686 occasions

INFERRING BIGRAM PROBABILITIES

- We also have the counts of words:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- To infer probabilities of bigrams, we'll divide bigram counts by word counts.

  (remember: $P(w_1, w_2) = count(w_1\ w_2) / count(w_1)$)

WARWICK

INFERRING BIGRAM PROBABILITIES

- Having the following two:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- We infer the following probabilities:

5 / 2533

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

BIGRAM ESTIMATE OF SENTENCE PROBABILITY

- Now we want to estimate the probability of a given sentences:

  I want to eat chinese food.

- P(I want to eat chinese food) =

  P(want | I) **[0.33]**

  * P(to | want) **[0.66]**

  * P(eat | to) **[0.28]**

  * P(chinese | eat) **[0.021]**

  * P(food | chinese) **[0.52]**

BIGRAM ESTIMATE OF SENTENCE PROBABILITY

WARWICK

- Now we want to estimate the probability of a given sentences:

  I want to eat chinese food.

- P(I want to eat chinese food) =

  P(want | I) **[0.33]**
  * P(to | want) **[0.66]**
  * P(eat | to) **[0.28]**

  * P(chinese | eat) **[0.021]**

  * P(food | chinese) **[0.52]**

  = **0.000665945**

PRACTICAL SOLUTION

WARWICK

- 0.000665945 is a very low probability.
  - We may end up with a **floating point underflow**!
- **Solution:**
  - Use **log space** instead.
  - **Sum** instead of multiplication.

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

PRACTICAL SOLUTION

WARWICK

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

I want to eat chinese food.

log(0.33) + log(0.66) + log(0.28) + log(0.021) + log(0.52) = **-3.1766**

want I eat food to lunch.

log(0.0022) + log(0.0036) + log(0.0027) + log(0.014) + log(0.0025) = **−12.1258**

Log scale is leading to **negative values, but** we just want them to be **comparable**.

i.e. in English, sentence 1 much more likely than sentence 2.

COLLECTIONS OF LANGUAGE MODELS

- Collection of Google N-grams:

```
File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens:    1,024,908,267,229
Number of sentences:    95,119,665,584
Number of unigrams:         13,588,391
Number of bigrams:         314,843,401
Number of trigrams:        977,069,902
Number of fourgrams:     1,313,818,354
Number of fivegrams:     1,176,470,663
```

COLLECTIONS OF LANGUAGE MODELS

- Collection of Google N-grams:

```
ceramics collection , 144
ceramics collection . 247
ceramics collection </S> 120
ceramics collection and 43
ceramics collection at 52
ceramics collection is 68
ceramics collection of 76
ceramics collection | 59
ceramics collections , 66
ceramics collections . 60
ceramics combined with 46
ceramics come from 69
ceramics comes from 660
```

COLLECTIONS OF LANGUAGE MODELS

WARWICK

- **Google N-grams** is the **best collection** of language models today.

- **Caveat:** it has a cost, **$150**.

  https://catalog.ldc.upenn.edu/LDC2006T13

COLLECTIONS: FREE ALTERNATIVE

WARWICK

- **Google Books N-grams:**

1-grams 0 1 2 3 4 5 6 7 8 9 a b c d e f g h i j k l m n o other p pos punctuation q r s t u v w x y z

2-grams 0 1 2 3 4 5 6 7 8 9 _ADJ_ _ADP_ _ADV_ _CONJ_ _DET_ _NOUN_ _NUM_ _PRON_ _PRT_ _VERB_ a_ aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay az b_ ba bb bc bd be bf bg bh bi bj bk bl bm bn bo bp bq br bs bt bu bv bw bx by bz c_ ca cb cc cd ce cf cg ch ci cj ck cl cm cn co cp cq cr cs ct cu cv cw cx cy cz d_ da db dc dd de df dg dh di dj dk dl dm dn do dp dq dr ds dt du dv dw dx dy dz e_ ea eb ec ed ee ef eg eh ei ej ek el em en eo ep eq er es et eu ev ew ex ey ez f_ fa fb fc fd fe ff fg fh fi fj fk fl fm fn fo fp fq fr fs ft fu fv fw fx fy fz g_ ga gb gc gd ge gf gg gh gi gj gk gl gm gn go gp gq gr gs gt gu gv gw gx gy gz h_ ha hb hc hd he hf hg hh hi hj hk hl hm hn ho hp hq hr hs ht hu hv hw hx hy hz i_ ia ib ic id ie if ig ih ii ij ik il im in io ip iq ir is it iu iv iw ix iy iz j_ ja jb jc jd je jf jg jh ji jj jk jl jm jn jo jp jq jr js jt ju jv jw jx jy jz k_ ka kb kc kd ke kf kg kh ki kj kk kl km kn ko kp kq kr ks kt ku kv kw kx ky kz l_ la lb lc ld le lf lg lh li lj lk ll lm ln lo lp lq lr ls lt lu lv lw lx ly lz m_ ma mb mc md me mf mg mh mi mj mk ml mm mn mo mp mq mr ms mt mu mv mw mx my mz n_ na nb nc nd ne nf ng nh ni nj nk nl nm nn no np nq nr ns nt nu nv nw nx ny nz o_ oa ob oc od oe of og oh oi oj ok ol om on oo op oq or os ot other ou ov ow ox oy oz p_ pa pb pc pd pe pf pg ph pi pj pk pl pm pn po pp pq pr ps pt pu punctuation pv pw px py pz q_ qa qb qc qd qe qf qg qh qi qj qk ql qm qn qo qp qq qr qs qt qu qv qw qx qy qz r_ ra rb rc rd re rf rg rh ri rj rk rl rm rn ro rp rq rr rs rt ru rv rw rx ry rz s_ sa sb sc sd se sf sg sh si sj sk sl sm sn so sp sq sr ss st su sv sw sx sy sz t_ ta tb tc td te tf tg th ti tj tk tl tm tn to tp tq tr ts tt tu tv tw tx ty tz u_ ua ub uc ud ue uf ug uh ui uj uk ul um un uo up uq ur us ut uu uv uw ux uy uz v_ va vb vc vd ve vf vg vh vi vj vk vl vm vn vo vp vq vr vs vt vu vv vw vx vy vz w_ wa wb wc wd we wf wg wh wi wj wk wl wm wn wo wp wq wr ws wt wu wv ww wx wy wz x_ xa xb xc xd xe xf xg xh xi xj xk xl xm xn xo xp xq xr xs xt xu xv xw xx xy xz y_ ya yb yc yd ye yf yg yh yi yj yk yl ym yn yo yp yq yr ys yt yu yv yw yx yy yz z_ za zb zc zd ze zf zg zh zi zj zk zl zm zn zo zp zq zr zs zt zu zv zw zx zy zz

3-grams 0 1 2 3 4 5 6 7 8 9 _ADJ_ _ADP_ _ADV_ _CONJ_ _DET_ _NOUN_ _NUM_ _PRON_ _PRT_ _VERB_ a_ aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay az b_ ba bb bc bd be bf bg bh bi bj bk bl bm bn bo bp bq br bs bt bu bv bw bx by bz c_ ca cb cc cd ce cf cg ch ci cj ck cl cm cn co cp cq cr cs ct cu cv cw cx cy cz d_ da db dc dd de df dg dh di dj dk dl dm dn do dp dq dr ds dt du dv dw dx dy dz e_ ea eb ec ed ee ef eg eh ei ej ek el em en eo ep eq er es et eu ev ew ex ey ez f_ fa fb fc fd fe ff fg fh fi fj fk fl fm fn fo fp fq fr fs ft fu fv fw fx fy fz g_ ga gb gc gd ge gf gg gh gi gj gk gl gm gn go gp gq gr gs gt gu gv gw gx gy gz h_ ha hb hc hd he hf hg hh hi hj hk hl hm hn ho hp hq hr hs ht hu hv hw hx hy hz i_ ia ib ic id ie if ig ih ii ij ik il im in io ip iq ir is it iu iv iw ix iy iz j_ ja jb jc jd je jf jg jh ji jj jk jl jm jn jo jp jq jr js jt ju jv jw jx jy jz k_ ka kb kc kd ke kf kg kh ki kj kk kl km kn ko kp kq kr ks kt ku kv kw kx ky kz l_ la lb lc ld le lf lg lh li lj lk ll lm ln lo lp lq lr ls lt lu lv lw lx ly lz m_ ma mb mc md me mf mg mh mi mj mk ml mm mn mo mp mq mr ms mt mu mv mw mx my mz n_ na nb nc nd ne nf ng nh ni nj nk nl nm nn no np nq nr ns nt nu nv nw nx ny nz o_ oa ob oc od oe of og oh oi oj ok ol om on oo op oq or os ot other ou ov ow ox oy oz p_ pa pb pc pd pe pf pg ph pi pj pk pl pm pn po pp pq pr ps pt pu punctuation pv pw px py pz q_ qa qb qc qd qe qf qg qh qi qj qk ql qm qn qo qp qq qr qs qt qu qv qw qx qy qz r_ ra rb rc rd re rf rg rh ri rj rk rl rm rn ro rp rq rr rs rt ru rv rw rx ry rz s_ sa sb sc sd se sf sg sh si sj sk sl sm sn so sp sq sr ss st su sv sw sx sy sz t_ ta tb tc td te tf tg th ti tj tk tl tm tn to tp tq tr ts tt tu tv tw tx ty tz u_ ua ub uc ud ue uf ug uh ui uj uk ul um un uo up uq ur us ut uu uv uw ux uy uz v_ va vb vc vd ve vf vg vh vi vj vk vl vm vn vo vp vq vr vs vt vu vv vw vx vy vz w_ wa wb wc wd we wf wg wh wi wj wk wl wm wn wo wp wq wr ws wt wu wv ww wx wy wz x_ xa xb xc xd xe xf xg xh xi xj xk xl xm xn xo xp xq xr xs xt xu xv xw xx xy xz y_ ya yb yc yd ye yf yg yh yi yj yk yl ym yn yo yp yq yr ys yt yu yv yw yx yy yz z_ za zb zc zd ze zf zg zh zi zj zk zl zm zn zo zp zq zr zs zt zu zv zw zx zy zz

4-grams 0 1 2 3 4 5 6 7 8 9 _ADJ_ _ADP_ _ADV_ _CONJ_ _DET_ _NOUN_ _NUM_ _PRON_ _PRT_ _VERB_ a_ aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay az b_ ba bb bc bd be

http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

EVALUATION AND PERPLEXITY

EVALUATION OF OUR MODEL

- We want to evaluate whether our language model is good.
  i.e. does our language model prefer good sentences to bad ones?

- i.e. does it assign higher probability:
    - to “real” or “frequent” sentences **(e.g. I want to)**
    - than “ungrammatical” or “rarely observed” sentences? **(e.g. want I to)**

EVALUATION OF OUR MODEL

- **Evaluation:**
  - Is our language model good in giving high probabilities to sentences in our corpus?

- Usually done in a comparative way:
  - Train language model 1 (LM1) from corpus 1.
  - Train language model 2 (LM2) from corpus 2.
  - For sentences in corpus 3, which of LM1 and LM2 is giving me higher probabilities?

- We need an **evaluation metric** to determine which of LM1 or LM2 is best.

EVALUATION APPROACHES

WARWICK

- Two different evaluation approaches:
  - **Extrinsic or in-vivo** evaluation.
    i.e. test externally on another task.
  - **Intrinsic or in-vitro** evaluation.
    i.e. evaluate the models directly.

EXTRINSIC EVALUATION

WARWICK

- Test LM1 and LM2 in some NLP task (sentiment analysis, MT, spell corrector, etc.).

- **Which of them leads to better performance?**

  i.e. what it the accuracy of my sentiment analysis tool by using LM1 and LM2?

DISADVANTAGE OF EXTRINSIC EVALUATION

- **Extrinsic** evaluation is ideal but **can take very long** to run.

- Alternatively **intrinsic evaluation**.
    - Consists in computing **perplexity**.
    - Rough **approximation**.
        - Only **valid if tested in similarly looking data**.

          i.e. don’t train a LM from news articles, and test it on social media.

INTRINSIC EVALUATION: PERPLEXITY

WARWICK

- **Perplexity:**

  given a language model, on average:
  how difficult is it to **predict the next word**?

  e.g. I always order pizza with cheese and _____ → ???

INTRINSIC EVALUATION: PERPLEXITY

- **The Shannon Game:**
  - How well can we predict the next word?

pizza with cheese and ____

mushrooms 0.1
pepperoni 0.1
jalapeños 0.01
….
biscuits 0.000001

- **A good model:**
  the one that gives higher probability to the actual next word.

- If the actual sentence is "pizza with cheese and biscuits", my model is quite bad.
- If the actual sentence is "pizza with cheese and mushrooms", my model is better.

INTRINSIC EVALUATION: PERPLEXITY

- To compute the **perplexity,**
  we will use an unseen test set (held out corpus).

  (different from the training data used to build the model)

- While we want to maximise P(sentence)
  we aim to get a low perplexity, it's inverse to probability.

INTRINSIC EVALUATION: PERPLEXITY

- Perplexity of a sequence of N words W, PP(W):

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

INTRINSIC EVALUATION: PERPLEXITY

WARWICK

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

$\xrightarrow{\text{Chain rule}}$

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

$\big\downarrow$ Bigrams

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

PERPLEXITY AS A BRANCHING FACTOR

- Suppose we have a **sentence consisting of random digits [0-9]**.
  - All digits have the same probability: 1/10.

- What is the perplexity?

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= (\frac{1}{10}^{N})^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

INTERPRETING PERPLEXITY

- **Lower perplexity** (i.e. higher probability), **better model**!

- e.g. WSJ corpus: 38 million words for training, 1.5 million for testing.

	Unigram	Bigram	Trigram
Perplexity	962	170	109

- **Remember:** language models will work best on similarly looking data.

- If we train on social media data, that may not work for novels.
    - OMG I'm ROTFL! → may be frequent in SM, unlikely in novels!

- **Limitation:** We're assuming that we have computed all probabilities for n-grams in the test data.
  - Is this the reality though?

- I may train with the following n-grams:
  I found a penny.
  I found a fiver.
  I found a book.

- If my test data has "I found a tenner", is the probability of this sentence really 0?
  just because I haven't seen it in my training data?

- There are different approaches for **dealing with zeros** and to enable **generalisation of trained models**.
  - In the next lecture.

ASSOCIATED READING

- Jurafsky, Daniel, and James H. Martin. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 3rd edition. **Chapters 3.1-3.2**.

- Bird Steven, Ewan Klein, and Edward Loper. Natural Language Processing with Python. O'Reilly Media, Inc., 2009. **Chapter 2, Section 2**.