Introduction to Natural Language Processing

NOTES FROM THE FALL 2015 SESSION ON COURSERA

DRAGOMIR RADEV, UNIVERSITY OF MICHIGAN

Contents

Useful Advice for the Course

Introduction

The first iteration of the “Introduction to Natural Language Processing” course from the University of Michigan was offered in October-December 2015 on Coursera.

Coursera switched to a new, on-demand platform in 2016 and has decided to delete the old forums from 2015. With help from Yilin Guo, Nick Serra, and David Walker, we now have this document that includes a lot of useful information from these old forums.

The versions of the course that will start on July 4, 2016 and August 1, 2016 are identical with the version from Fall 2015.

A new version of the course (that includes Deep Learning) will be available in early 2017.

In the meantime, I hope that this document will be useful to you.

Drago

June 29, 2016

Index of course concepts (by Franz Calvo)

https://franzcalvo.files.wordpress.com/2015/11/index_of_nlp_mooc_at_umich.pdf

Student Goals

I have come across students who have very different goals:

- learn about the wide range of problems that NLP addresses
- build applications using open source code such as NLTK and the many other tools mentioned elsewhere
- understand the mathematics behind the existing algorithms
- design new algorithms
- write academic papers, etc.

Collins's class on Coursera covers item (3) in more detail than my class.

Both the 2012 Manning and Jurafsky class on Coursera and my class cover (1) in detail.

I also spend some time on (2) and (3) but not on (4) or (5), which are more research-oriented.

Students interested in (4) or (5) are encouraged to pursue a PhD and/or work for a company that does R&D work.

Python

Installing Python environments for this class

Suggestions for Setting up Python and NLTK on Windows

https://github.com/jevad/NLTKinstallForNlpClass

How to install a specific older Python library

```
  pip uninstall scikit-learn
  pip install scikit-learn==0.15.2
```

or

```
  pip uninstall sklearn
  pip install scikit-learn==0.15.2
```

NLTK

Natural Language Processing with Python

http://www.nltk.org/book/

Where can I download the slides?

The slides from the 2015 and 2016 sessions are publicly available here:
http://web.eecs.umich.edu/~radev/coursera-slides/

NACLO and IOL Puzzles

The main site for NACLO is http://www.nacloweb.org/

The IOL site is http://www.ioling.org

Recommended NACLO puzzles by course week. The URLs for the solutions are formed by adding “S” before “.pdf”, e.g., N2007-A.pdf

Week 1

We are all molistic in a way

http://www.nacloweb.org/resources/problems/2007/N2007-A.pdf

Letters for Cuzco

http://www.nacloweb.org/resources/problems/2009/N2009-C.pdf

Week 2

Gelda’s House of Gelbelgarg

http://www.nacloweb.org/resources/problems/2010/A.pdf

Orwellspeak

http://www.nacloweb.org/resources/problems/2009/N2009-M.pdf

Week 3

Nok-nok!

http://www.nacloweb.org/resources/problems/2009/N2009-B.pdf

Pooh’s encyclopedia

http://www.nacloweb.org/resources/problems/2007/N2007-B.pdf

The lost tram

http://www.nacloweb.org/resources/problems/2007/N2007-F.pdf

Week 4

Twodee

http://www.nacloweb.org/resources/problems/2013/N2013-H.pdf

One, Two, Tree

http://www.nacloweb.org/resources/problems/2012/N2012-R.pdf

Grammar Rules!

http://www.nacloweb.org/resources/problems/2013/N2013-F.pdf

Week 5

CCG

http://www.nacloweb.org/resources/problems/2014/N2014-O.pdf

Combining Categories in Tok Pisin

http://www.nacloweb.org/resources/problems/2014/N2014-P.pdf

Dogs and cats on trees

http://www.nacloweb.org/resources/problems/2010/I.pdf

Week 6

Use the Force

http://www.nacloweb.org/resources/problems/2015/N2015-E.pdf

Springing up baby

http://www.nacloweb.org/resources/problems/2008/N2008-B.pdf

Week 7

A fox among the h

http://nacloweb.org/resources/problems/2012/N2012-K.pdf

Fakepapershelfmaker

http://nacloweb.org/resources/problems/2008/N2008-F.pdf

Word Salad

http://nacloweb.org/resources/problems/2011/M.pdf

Week 8

Yesbot

http://www.nacloweb.org/resources/problems/2013/N2013-L.pdf

Levenshtein’s Fine Signs

http://www.nacloweb.org/resources/problems/2014/N2014-C.pdf

Week 9

Summer Eyes

http://www.nacloweb.org/resources/problems/2009/N2009-E.pdf

Week 10

Hypo-Hmong-driac

http://www.nacloweb.org/resources/problems/2009/N2009-J.pdf

Week 11

Zoink!

http://www.nacloweb.org/resources/problems/2015/N2015-G.pdf

The Little Engine that Could… Read

http://www.nacloweb.org/resources/problems/2012/N2012-O.pdf

Week 12

Interstellar First Contact

http://www.nacloweb.org/resources/problems/2012/N2012-C.pdf

Running on MT

http://www.nacloweb.org/resources/problems/2011/A.pdf

Grice’s Grifter Gadgets

http://www.nacloweb.org/resources/problems/2013/N2013-Q.pdf

Bonus Puzzles

The Old Man the Boats

http://nacloweb.org/resources/problems/2015/N2015-P.pdf

All in the Family

http://www.naclo.cs.cmu.edu/problems2012/N2012-D.pdf

International Linguistics Olympiad

http://www.ioling.org/

2016 International Linguistics Olympiad

http://iol14.plo-in.org/index

Course Errata (from 2015 and 2016)

Week 1

QUESTION: Week 1 Course goals differ slightly in Lecture 1 vs Lecture 4

ANSWER:

Week 1, Lecture 1, slide 8, Goals of this Class:

Understand that NLP is hard

Understand key problems in NLP

Learn about the methods to solve those problems

Understand the limitations of these methods

Week 1, Lecture 4, slide 3, Three Major Goals:

Learn the basic principles and theoretical issues underlying NLP.

Learn techniques and tools used to develop practical robust systems that can understand text and communicate with users in one or more languages.

Gain insight into some open research problems in natural language.

Indeed, there is overlap between the two. But I think it is helpful to have 100% clarity on the goals.

Sorry if this comes across as nit-picking :)

ANSWER (DRAGO):

Thanks for the question.

MOOCs have a very diverse audience and people come here with different goals in mind. For example, many students just get the lecture notes and try the quizzes but don't do any of the programming assignments. Others focus on the programming part.

A fuller list of course goals was posted in the Announcement for Week One. Here it is again:

The goals of this class are (1) understanding the need for NLP, (2) understanding why language processing is very hard, (3) learning about the key tasks in NLP, (4) learning about the methods used to solve these tasks, (5) obtaining the linguistic and mathematical background behind these methods, (6) understanding the premises and limitations of these methods, (7) learning about the availability of external resources, such as programming libraries, tutorials, tools, and text corpora, and (8) getting experience building specific NLP components.

QUESTION: 01.05 - Quiz 1 incorrect answer

ANSWER:

in the first quiz "time flies like an arrow" the grader accepts "0 possible interpretations" as the correct answer. This would be incorrect and the instructor immediately provides possible interpretations on the next slide. so the correct answer should be "2 or more"

QUESTION: 01.05 - Why is NLP hard? Yehoshua Bar-Hillel

In the captions in 2:40 says

"this is an example from Yehorshwa Bahilow from close to a 100 years ago." but it is "Yehoshua Bar-Hillel" and is from the year 1960 not 1915 (100 years ago).

https://en.wikipedia.org/wiki/Yehoshua_Bar-Hillel

QUESTION: Quiz 1 - misleading spelling error in quiz 1

ANSWER:

There's is misspelling error in quiz 1 where the word entrepreneur is spelled enterpreneur. This may induce an error when replying to this question.

Week 2

QUESTION: 02.05 - Slide 14 - D(1,j-1)+1 should be D(i,j-1)+1

On the slide 14 in 02.05 it looks like the pseudocode has a typo:

D(1,j-1)+1 should be D(i,j-1)+1

QUESTION: 02.05 - Edit Distance between 'trend' and 'strength'

In the 02.05 video, the professor said that the minimum edit distance between 'trend' and 'strength' is 4. That is correct, I suppose.

However, he proposes the following for "strength" to be converted to "trend":

Drop the "s" ("trength")

Substitute the "g" with "d" (trendth)

Inserting a "t" and an "h" (trendthth) ???

Not sure if I'm wrong here, but I suspect the instructor intended to say "drop the 't' and 'h'". Nonetheless I was pretty confused.

ANSWER:

Yes, I am certain he meant to say "drop the 't' and 'h'".

He was trying to convert "strength" to "trend", so there is no way he could have meant to insert the t and h.

I actually found another typo in that section.

Slide deck: 02.05 - Spelling Similarity: Edit Distance

Page: Recurrence Relation (page 14)

See the section on the right "Recursive Dependencies". The middle element in the min() is:

D(1,j-1)+1

It should be:

D(i,j-1)+1

Week 3

QUESTION: 03.02 - Probabilities of ungulate and deer

In the 3 - 2 - 03.02 - Thesaurus-based Word Similarity Methods - [11 slides] (07-52) video, the professor says:

And if we want to look at the probability of those, 72

00:04:34,870 --> 00:04:37,890

it should be pretty obvious that in any given corpus,

73

00:04:37,890 --> 00:04:42,330

the probability of ungulate will be larger than the probability of deer, because

74

00:04:42,330 --> 00:04:47,400

ungulates are more common than deer, with deer being a special case of ungulates.

Here I don't assume so. Because in my assumption ungulate is more of an academic word, while deer can be used in more informal settings. Thus probability of deer can be higher than ungulate in some corpora, or some sections of a corpus.

I am only speaking from my assumption. I hope someone can explain this. Thanks!

ANSWER:

I am going to guess that the probability of a word (or word sense?) is the relative frequency of it and all of its hyponyms in a certain corpus. So ungulate has a higher probability not because it is more frequent but because it is a hypernym for lots of other words. But can anyone confirm this? I don't think it has been defined in the lectures.

ANSWER:

I believe you have described it perfectly, based Philip Resnik's paper that defines this method: Using Information

Content to Evaluate Semantic Similarity in a Taxonomy.

http://arxiv.org/PS_cache/cmp-lg/pdf/9511/9511007v1.pdf

ANSWER:

I was still very confused by the explanation and I read the article (well, partially). I like it that the author has introduced an intermediate concept: the information content. More general concepts have lower information content, then more specific ones. And yes, the information content is defined recursively as -log the sum of frequencies of all subsumed notions. I find this intermediate step with information content very useful for understanding.

ANSWER:

Maybe this makes sense... The numerator in the Lin formula is log(P(LCS(v,w))). Since P is in the range 0 to 1, then log(P) is in the range -infinity to 0. The larger is P(LCS) the smaller the absolute value of log(P(LCS)), making Sim(v,w) smaller. (The numerator and denominator are both negative so the division result will always be positive.)

Basically this is saying that the higher you have to go to find an LCS then the lower log(P(UCS)) and the lower the similarity.

If v and w are the same word (as similar as you can get) then LCS(v,w) = v = w. Plugging in any value for P(v) (0 < P(v) <1) gives Sim(v,w) = 1

If the are the words have no similarity then the only UCS is "any word" and then P(UCS) is 1 (or so I assume). Then log(P(UCS) is 0 so Sim(v,w) = 0 no matter what are the values P(v) and P(w).

It took me a while to think this through. In this course a lot of tricky concepts pop up without much explanation. Several hours work got me nowhere on the reductions of dimensions topic; I could generally follow the math but had little idea what was being done with it.

ANSWER:

Yes, your considerations are exactly to the point. We take -log in order to take, as you say, the absolute value. And the division by the sum of information contents of two words separately is just a normalisation.

I believe, that all of this is much based on the information theory. This is why it looks tricky at the first glance. I don't know much about information theory, but taking a log to assess information content rings a bell.

I did not get to reductions of dimensions yet, so can't comment, but it can be based on some linear algebra theory ;-)

ANSWER:

Well, I have heard now the dimension reduction lecture. This is a well-known machine learning method (which probably came from statistics, like factor analysis). I have started doing machine learning on coursera, done until now about 3 weeks of it, and it was explained in more details there. It was applied for an image simplification there, where you could still recognise the image after half of the eigenvalues were thrown away. As far as I understood, this method is used either for information compressing or for finding hidden reasons behind the data. Anyway it is a much broader topic then it was presented here, may be you could google for SVD to learn more.

QUESTION: 03.03 - problem with the discussion of the cosine similarity in the vector space model.

@1:43: “similarity between d1 and q is proportional to the angle alpha…” – this is not correct. In fact, the smaller the angle, the larger the similarity => it is not a proportionality relation.

The following (@2:05) discussion of the cosine approximation “This is cos(alpha) vs cos(theta), in this quadrant they are related in the same direction. So if the angle is smaller, that means that the cosine is also smaller” – sounds misleading to me. In fact, the two vectors are the closer, the smaller is the angle between them. In terms of the cosine, however, the similarity means the closeness of the cosine to 1 (as cos(alpha) = 1 – (alpha)^2/2 for small alpha). So this part should read: “The similarity of two vectors d and q is determined by they scalar product, which is equal to |d||q|cos(alpha). When alpha is close to zero, cos(alpha) => 1, and the scalar product (and therefore the similarity) is maximized. The larger alpha, the smaller the cosine, and therefore the similarity”.

QUESTION: 03.03 - Prep Phrase Att 3/3 11:50 Other methods- on course site, where can these papers be found

ANSWER:

The Video mentions that papers referred on this slide are available on course site. I am unable to locate this additional material.

QUESTION: 03.03 - Wrong answer in 2nd quiz, lecture 3.03

The second quiz within lecture 3.03 has two parts. The correct answer for the second part is "0.6". But "0.6" was shown as incorrect and "30" was given a the correct answer.

ANSWER:

I had the same issue. The similarity is limited to 1 so "3" is not possible at all.

ANSWER:

The same thing happened to me. Yet, the next slide after the quiz shows that 0.6 is indeed correct. TAs, could you fix this confusion?

QUESTION: 03.04 - Dimensionality reduction slide

left plot, axes show height and weight (not shown on the slide). What stays on both axes at the right plot?

ANSWER:

@12:21: I believe, the diagonal elements of Sigma are the square roots of the eigenvalues of ATA, not the eigenvalues themselves.

@12:33: Third bullet should be “If A has 5 rows and 3 columns...” – otherwise the dimensions of U and V are wrong.

@13:33: D2={T1,T2}, while @14:31 the second column of the A-matrix is {1,1,0,0,1,0,0,0,0} (has one extra D5 component). Same for the normalized matrix.

@15:02: “V is 7x9 matrix” misspoken; should be “V is 7x7 matrix”. Furthermore, according to its definition as eigenvalues of ATA, Sigma-matrix should also be 7x7. Would be nice either to clarify that the last two zero rows were added so that the matrices U=(9x9), Sigma_new=(9x7) and V^T=(7x7) are mutually multipliable, or to define Sigma-matrix as a rectangular matrix.

QUESTION: 03.04 - wrong dimension of matrix A

Slide 12 reports that A has 5 columns and 3 rows, but it should be 5 rows and 3 columns.

ANSWER:

True. One more thing: on slide 13 D2={T1,T2}, while in slide 15 the second column of the A-matrix is

{1,1,0,0,1,0,0,0,0}. Which D2 was actually used in the numerics shown on the next slides?

ANSWER:

In fact, there is one more inaccuracy in this lecture.

In the DT-example, with A = (9x7)-matrix, the dimensions of V and S are 7x7 -- as these matrices are composed from eigenvalues and eigenstates of A^T*A. The lecture wrongly says that they are 7x9 matrices. Looks like after S-matrix was calculated, two additional zero rows were inconspicuously added to it (see slide 17) -- to make the matrices U=(9x9), S_new=(9x7) and V^T=(7x7) multipliable (otherwise the dimensions misfit and the multiplication is impossible). I don't know how and why it works, however, this is the only way I can figure it out, given A = U*S*V^T and the definitions of S,U,V from the slide 12. Please correct me if you have other interpretations...

ANSWER:

We're in the same interpretation. The 7x9 is misspoken, the V matrix in the slide is 7x7.

"..two additional zero rows were inconspicuously added to it (see slide 17)... ".

Hermiticity or symmetry is not required for singular values. Zero-row x any vector = zero terms in reconstruction. Furthermore, when we perform lower n-rank approximation, it's reduced to n x n. So, it's supercalifragilisticexpialidocious.

ANSWER:

Sorry, one more thing. At slide 12 it should stay "The components of S are the *square roots of the* eigenvalues of ATA".

QUESTION: 03.04 - dimensionality reduction -- general sense of the terms used in the lecture

A term is a word in this context. In principle, it could also be a phrase, e.g., "New York".

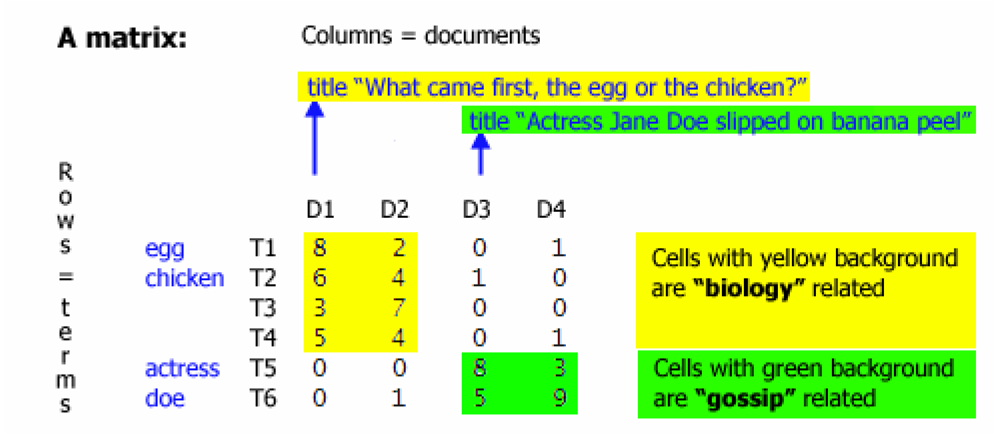
A concept would be something like "art", "medicine", "jazz". Many words are related (with different probabilities) to the same "concept" (non-exclusively).

Consider,
A = m x n matrix, i.e. m terms x n documents,
Other examples,

(1) some people use the opposite: m documents x n terms (NLP, similarity),

(2) m movies x n users (e.g. a Netflix recommender system),

(3) m items x n customers (e.g. an Amazon recommender system), etc.



A matrix:

Columns = documents

title "What came first, the egg or the chicken?"

title "Actress Jane Doe slipped on banana peel"

Rows = terms

		D1	D2	D3	D4
egg	T1	8	2	0	1
chicken	T2	6	4	1	0
	T3	3	7	0	0
	T4	5	4	0	1
actress	T5	0	0	8	3
doe	T6	0	1	5	9

Cells with yellow background are **"biology"** related

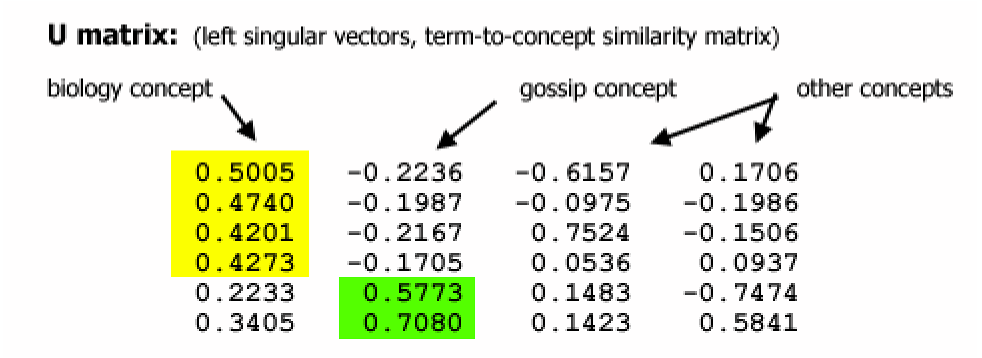
Cells with green background are **"gossip"** related

Each matrix's cell represents frequency of term T in respective document D.

- 1st document D1's title is "*What came first, the egg or the chicken?*",
- 2nd document D2 has the same "concept" with 1st document,
- 3rd document D3's title is "*Actress Jane Doe slipped on banana peel.*",
- 4th document D4 has the same "concept" with 3rd document,

- 1st row T1 represents "*egg*" term,
- 2nd row T2 represents "*chicken*" term,
- 5th row T5 represents "*actress*" term,
- 6th row T6 represents "*doe*" term,

(we pretend that Jane Doe is a very famous actress)
By applying SVD to matrix A, we get decomposed matrices, U, S, V.
U = m x r matrix, left singular vectors, i.e. m terms x r concepts, r is rank,
this is "term-to-concept" similarity matrix.

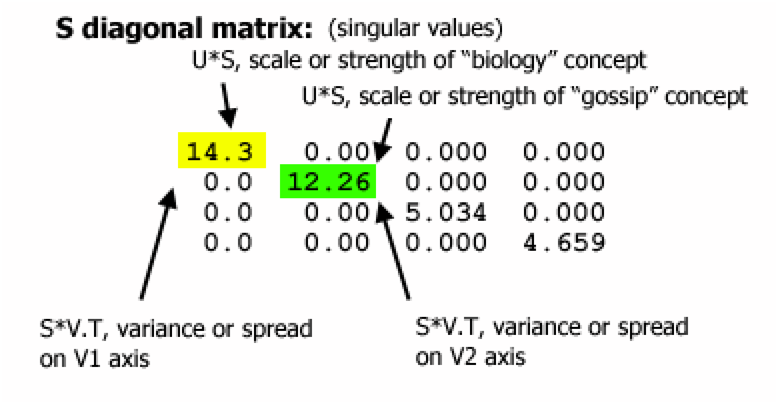


U matrix: (left singular vectors, term-to-concept similarity matrix)

$$\begin{array}{rrrr}
0.5005 & -0.2236 & -0.6157 & 0.1706 \\
0.4740 & -0.1987 & -0.0975 & -0.1986 \\
0.4201 & -0.2167 & 0.7524 & -0.1506 \\
0.4273 & -0.1705 & 0.0536 & 0.0937 \\
0.2233 & 0.5773 & 0.1483 & -0.7474 \\
0.3405 & 0.7080 & 0.1423 & 0.5841
\end{array}$$

biology concept
gossip concept
other concepts

S = r x r diagonal matrix, singular values, r is rank, i.e. the scale (or strength) of each concept, or the variance (spread) on the concept axes.

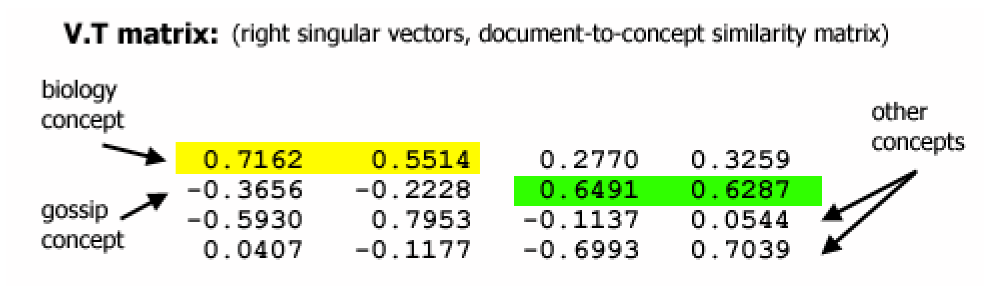


S diagonal matrix: (singular values)

$$\begin{array}{rrrr}
14.3 & 0.00 & 0.000 & 0.000 \\
0.0 & 12.26 & 0.000 & 0.000 \\
0.0 & 0.00 & 5.034 & 0.000 \\
0.0 & 0.00 & 0.000 & 4.659
\end{array}$$

U*S, scale or strength of "biology" concept
U*S, scale or strength of "gossip" concept
S*V.T, variance or spread on V1 axis
S*V.T, variance or spread on V2 axis

V = n x r matrix, right singular vectors, i.e. n documents x r concepts, r is rank,
this is "document-to-concept" similarity matrix.

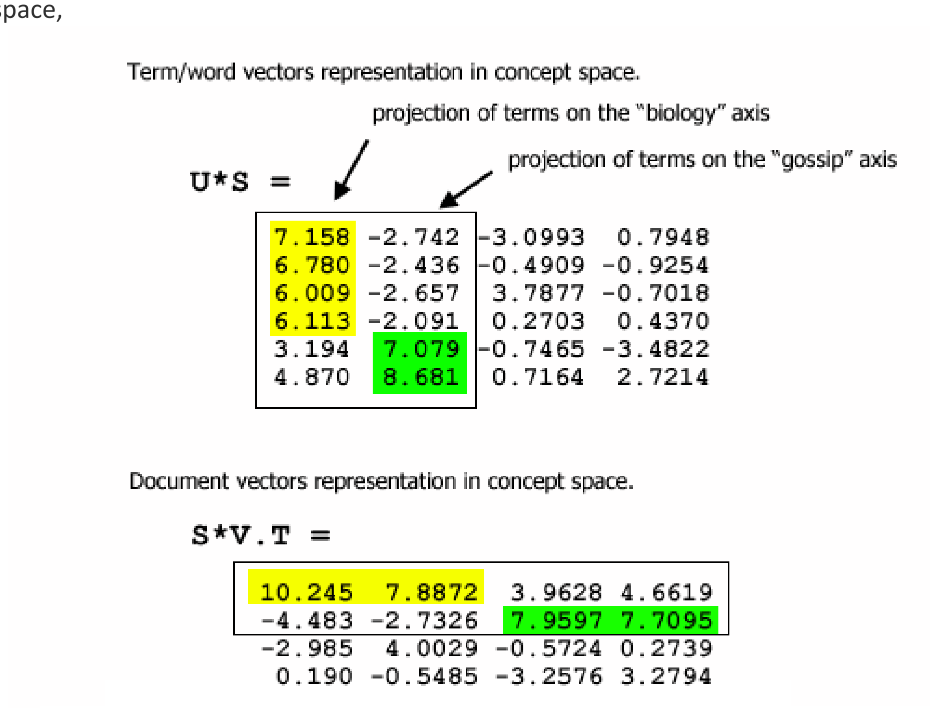


V.T matrix: (right singular vectors, document-to-concept similarity matrix)

$$\begin{array}{l|rrrr|l}
\text{biology concept} & 0.7162 & 0.5514 & 0.2770 & 0.3259 & \\
\text{gossip concept} & -0.3656 & -0.2228 & 0.6491 & 0.6287 & \\
 & -0.5930 & 0.7953 & -0.1137 & 0.0544 & \text{other concepts} \\
 & 0.0407 & -0.1177 & -0.6993 & 0.7039 & \text{other concepts}
\end{array}$$

(.T is transpose),

The concept space,



Term/word vectors representation in concept space.

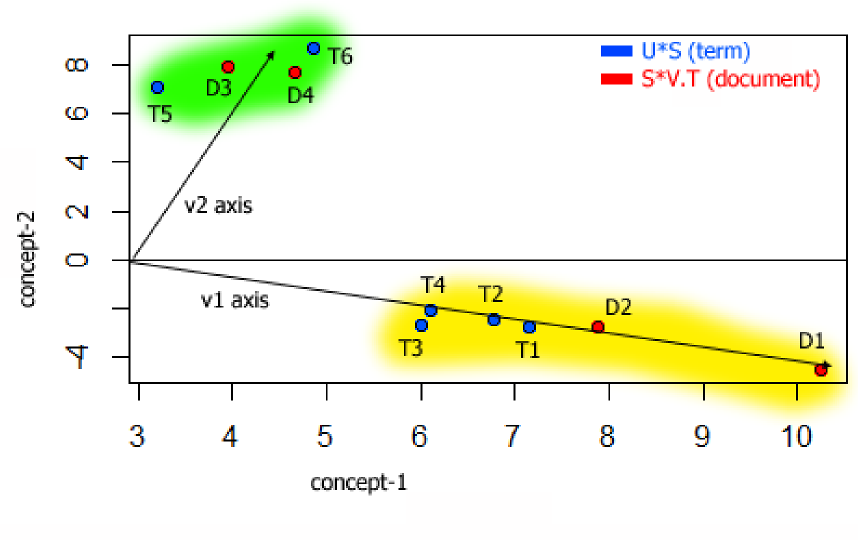
projection of terms on the "biology" axis

projection of terms on the "gossip" axis

$$U*S = \begin{array}{|rr|rr}
7.158 & -2.742 & -3.0993 & 0.7948 \\
6.780 & -2.436 & -0.4909 & -0.9254 \\
6.009 & -2.657 & 3.7877 & -0.7018 \\
6.113 & -2.091 & 0.2703 & 0.4370 \\
3.194 & 7.079 & -0.7465 & -3.4822 \\
4.870 & 8.681 & 0.7164 & 2.7214
\end{array}$$

Document vectors representation in concept space.

$$S*V.T = \begin{array}{rrrr}
10.245 & 7.8872 & 3.9628 & 4.6619 \\
-4.483 & -2.7326 & 7.9597 & 7.7095 \\
-2.985 & 4.0029 & -0.5724 & 0.2739 \\
0.190 & -0.5485 & -3.2576 & 3.2794
\end{array}$$



In the plot, similarity is calculated from "angle" between vectors. The closer, the smaller the vectors' angle, the higher the cosine similarity (dot product). E.g.

- T5 (actress) is closer to D3 (with frequency of 8 in A matrix).
- T6 (doe) is closer to D4 (with frequency of 9 in A matrix).
- T1 (egg) is closer to D1 (with frequency of 8 in A matrix), although the angle difference between T1-D1 vector and T1-D2 vector is very small.
- T3 is inaccurate, because it's closer to D1. Remember this is rank-2 approximation (plot in 2D), error is expected. More accurate plot should be in 4D, I don't know how to plot in 4D space. (--- is this understanding correct?)

Now, we want to know what concept is "chicken" T2-term in 4-documents with frequencies of [7, 5, 0, 2] respectively.

Multiply the vector [7, 5, 0, 2] and V vector (document-to-concept similarity).

---Result is [**8.422** -2.416 -0.06612 1.105], 8.422 in first column represents "biology" concept.

Now, we want to know what concept is a document contains ["chicken" T2-term, T3-term, and "actress" T5-term], with frequencies of [0, 6, 4, 0, 3, 0] respectively.
Multiply the vector [0, 6, 4, 0, 3, 0] and U vector (term-to-concept similarity).

---Result is [**5.194** -0.3267 1.98 -4.037], 5.194 in first column represents "biology" concept (concept 1). High frequencies of T2 and T3 belongs to biology concept.

Now, we want to know what concept is a document contains ["chicken" T2-term, T3-term, and "actress" T5-term], with frequencies of [0, 1, 2, 0, 7, 0] respectively.

Multiply the vector [0, 1, 2, 0, 7, 0] and U vector (term-to-concept similarity).

---Result is [3.101 **3.987** 0.221 -6.48], 3.987 in second column represents it's closer to "gossip" concept (concept 2) than "biology" (3.101). High frequency of T5 (actress) compared to lower frequencies of T2 and T3.

QUESTION: 03.07 - Reference to ATT demo in lecture 3.7 not found

The webpage with the ATT speech demo referred to in lecture 3.7 appears to exist no longer. Does it have a new address?

ANSWER:

Many other softwares,

http://www.howtogeek.com/125305/the-best-text-to-speech-tts-software-programs-and-online-tools

Windows has built-in Microsoft Narrator and Windows Speech Recognition. Dictation & Text to Speech in Mac OS X.

Week 4

QUESTION: 04.01 Parsing Exercises at the end of lecture 4.01

Are solutions available anywhere? I would like to check my answers if possible.

Week 5

QUESTION: 05.04 - Error in Section 05.04 Prepositional Phrase Attachment 3/3 "Collins and Brooks" slide

On Page 14 of the PDF file nlpintro_co1_05.04_DR_Edit.pdf of slides for

Section 05.04 Prepositional Phrase Attachment 3/3

(titled "Collins and Brooks") it says:

If the denominator of the next formula is 0, then use the classification for the 4-tuple

Surely that ought to be:

If the denominator of the next formula > 0, then use the classification for the 4-tuple

QUESTION: 05.04 - Collins and Brooks algorithm

The last step of the Collins and Brooks algorithm is "else label as default". The audio recommends to mark it as "high, i.e. 0", while the slide suggests to mark it as "low, i.e. 0". Previously, "default" was defined "low, i.e. 1", however we learned at some point that there is a "better default" = "high, i.e. 0 -- after we have looked at of and to". As far as I understand, the Collins and Brooks algorithm does not use the of/to-rule as Algorithm 2a does -- so which default should one use for the last step?

QUESTION: 05.04 - problem with last line of the slide

ANSWER:

@10:24. Audio: "...in the final step, we go to the default, which is to label everything else as high attachment, or zero". The slide suggests to mark it as “low”, but still gives 0 as a numeric value (though “low” corresponds to 1). To me, as we don’t use of/to-rule here, the default should be “low = 1”. I assume a typo in the last line of the slide (1 should stay instead of 0), while “high attachment, or zero” should be replaced with “low attachment, or one” in the audio.

QUESTION: 05.05 - print of Jurafsky and Martin contains error

On slide 21 of Statistical Parsing there seems to be an error in the table "Lexicon" of Jurafsky and Martin. The second row (Noun) adds up to 1.1 instead of 1.0.

ANSWER:

You are right.

The table is the same as on page 471 of my Jurafsky and Martin "Speech and Language Processing" 2nd edition.

It is wrong there too.

QUESTION: 05.08 - Week 5 Video 8 question on CFG

Could someone clarify this fact for me:

a^nb^n can be captured by a CFG

Link https://en.wikipedia.org/wiki/Context-free_language

However, in the video, Professor Radev mentions that this cannot be captured by a CFG.

ANSWER (DRAGO):

a^nb^n can be captured by a CFG. You push the first n symbols in the stack, then you pop one of them from the stack for each of the remaining n symbols in the input string.

What a CFG cannot capture is a^nb^nc^n. Once you've captured the a^nb^n part, you have lost track of the value of n.

a^nb^nc^nd^n is another language that cannot be captured by a CFG.

QUESTION: Quiz 5: Question 2

Consider the sample ATIS grammar shown on slide 21 of Unit 05.05.

What is the probability of "flights to Houston" to be parsed as NP?

Although the answers make the meaning clear, the probability to be parsed as NP in this case is 1 since there is no other possible parse. What the answers show is the probability to generate "flights to Houston" from an NP

ANSWER:

Yes I think you are right.

"parsed as NP" means we do not need to consider it, or P(NP) = 1.

Week 7

QUESTION: 07.03 - Week 7 - video 07.03

On page 14, in slide 07.03; the two last lines in the calculation says:

+ .2 x .5 x .4 x .2

+ .2 x .5 x .6 x .1

Are the two rightmost values correct? Is not 'z' generated on A = .1, and 'z' generated on B = .2? Should it not be:

+ .2 x .5 x .4 x .1

+ .2 x .5 x .6 x .2

ANSWER:

Great catch, but actually I believe he merely transposed the last two lines.

P(yz) = P(yz | AA) + P(yz | AB) + P(yz | BA) + P(yz | BB)

P(yz | AA) = P(A | A) x P(y | A) x P(A | A) x P(z | A) = .8 x .2 x .8 x .1

P(yz | AB) = P(A | A) x P(y | A) x P(B | A) x P(z | B) = .8 x .2 x .2 x .2

P(yz | BA) = P(B | A) x P(y | B) x P(A | B) x P(z | A) = .2 x .5 x .6 x .1

P(yz | BB) = P(B | A) x P(y | B) x P(B | B) x P(z | B) = .2 x .5 x .4 x .2

where

Transition:

P(A | A) = 0.8

P(A | B) = 0.6

P(B | A) = 0.2

P(B | B) = 0.4

Emission:

P(x | A) = 0.7, P(y | A) = 0.2, P(z | A) = 0.1

P(x | B) = 0.3, P(y | B) = 0.5, P(z | B) = 0.2

Week 11

QUESTION: 11.05 - Lecture 11.5: 3+4 = 12

add(3,4) = (?x,?y(x+y))(3)(4)= (?y 3+y)(4) = 3+4 = 12

Nice logic in the end :)

ANSWER:

Oh, add(3,4)=(?x,?y(x+y))(3)(4)=?y(3+y)(4)=3+4=7!

Notes

QUESTION: What is the impact of "human accuracy = 88%"?

Regarding PP attachment, a human accuracy of 88% seems low.

I'm wondering what is the impact of such a relatively low accuracy for language comprehension in the real world:

1. May we still understand a sentence reasonably well even when we incorrectly attach a PP as high or low?
   For example, regarding the tuple:
   "bring attention to problem" [1]
   I don't think comprehension would be much compromised if we attach the PP as "low".
2. Do humans have other mechanisms for language comprehension that would compensate for a wrongly attached PP?

I would really appreciate your ideas.

[1] Video 05.02, slide 12

QUESTION: Definition of binarized grammar

Does binarized grammar refer to one having 2 and ONLY 2 non-terminals on the right hand side? Or one having a MAXIMUM of 2 non-terminals? In the latter case, unary grammar, that is, having only 1 non-terminal does not count as binarized grammar, as it seems in the videos.

Am I right?

ANSWER (DRAGO):

CKY works when the grammar is in Chomsky Normal Form (CNF).

CNF requires all rules to be in the format:

```
A -> B C
A -> b
```

where A B C are non-terminals and b is a terminal.

The following types of rules are not allowed:

```
A -> B

A -> B C D

A -> b C D

A -> b C

A -> epsilon
```

QUESTION: Why standardize the matrix columns before SVD

ANSWER:

I was also wondering the same thing,

http://stats.stackexchange.com/questions/69157/why-do-we-need-to-normalize-data-before-analysis

http://stats.stackexchange.com/questions/12200/normalizing-variables-for-svd-pca

QUESTION: Tree plot of parsed sentences

ANSWER:

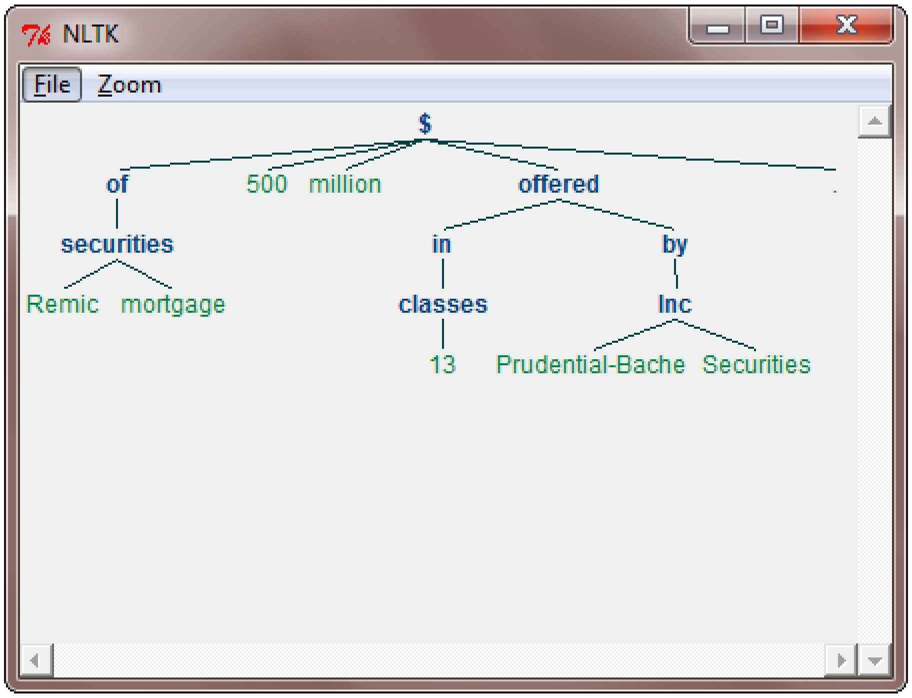
Want to have fun with tree plot of the parsed sentences?
If so, in your modified "test.py" of Assignment-1, insert this statements after the "tp.parse()",

```
        # tree of 4th sentence,
        # change "4" to 0, 1, 2.., 5, 6,.., 12,..
        # or do it in a loop,
        tree = parsed[4].tree()
        print tree
        tree.draw()
```

E.g. with English model, 4th sentence of "en-universal-test.conll",

```
1     $     _     .     $     _     0     ROOT     _     _
2     500     _     NUM     CD     _     1     num     _     _
3     million     _     NUM     CD     _     1     num     _     _
4     of     _     ADP     IN     _     1     adpmod     _     _
5     Remic     _     NOUN     NNP     _     7     compmod     _     _
6     mortgage     _     NOUN     NN     _     7     compmod     _     _
7     securities     _     NOUN     NNS     _     4     adpobj     _     _
8     offered     _     VERB     VBN     _     1     partmod     _     _
9     in     _     ADP     IN     _     8     adpmod     _     _
10     13     _     NUM     CD     _     11     num     _     _
11     classes     _     NOUN     NNS     _     9     adpobj     _     _
12     by     _     ADP     IN     _     8     adpmod     _     _
13     Prudential-Bache     _     NOUN     NNP     _     15     compmod     _     _
14     Securities     _     NOUN     NNP     _     15     compmod     _     _
15     Inc     _     NOUN     NNP     _     12     adpobj     _     _
16     .     _     .     .     _     1     p     _     _
```

Plot:



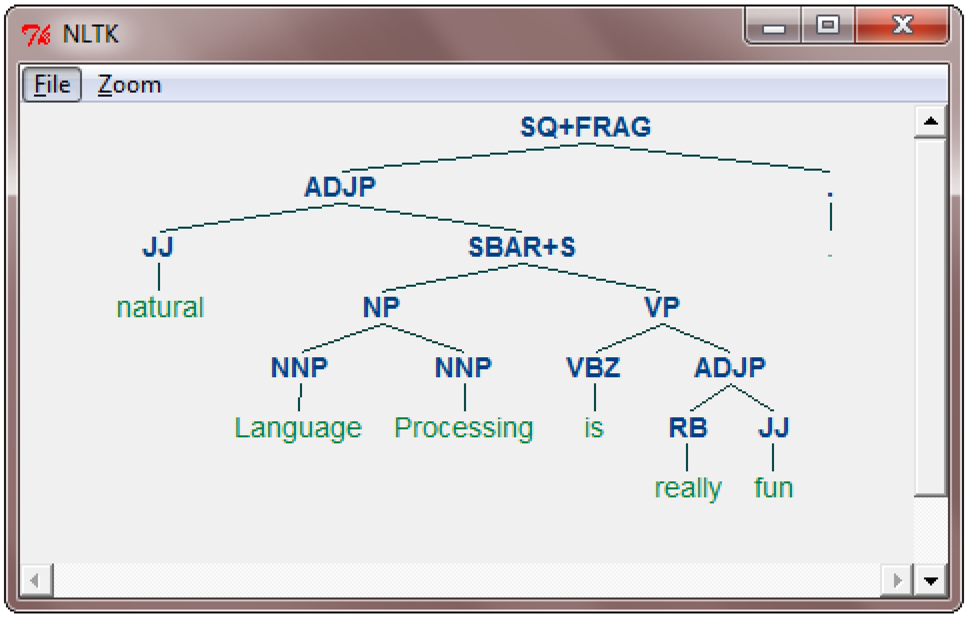
Print:

```
($
(of (securities Remic mortgage))
500
million
(offered (in (classes 13)) (by (Inc Prudential-Bache Securities)))
.)
```

Or, by using PyStatParser, by Emil Mont, simple Python statistical (CKY) parser together with scripts for learning the PCFG from the QuestionBank and Penn treebanks.

```
from stat_parser import Parser, display_tree
parser = Parser()

tree = parser.parse("Natural Language Processing is really fun.")
display_tree(tree)
print tree
```

Plot:



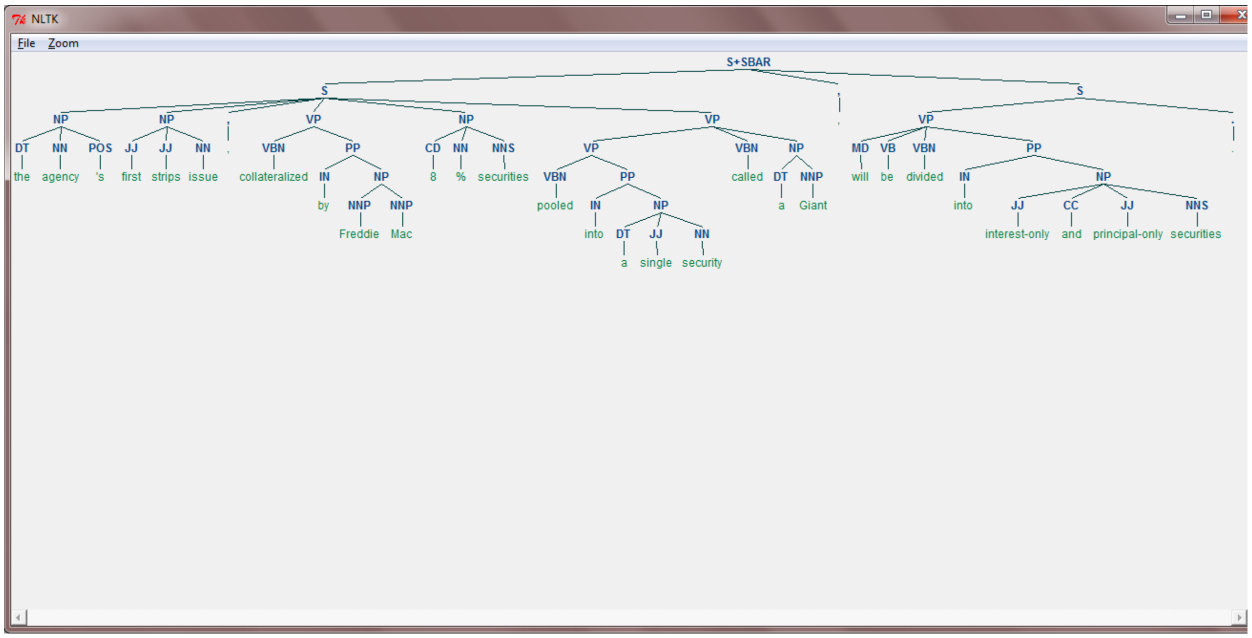
Print:

```
(SQ+FRAG
  (ADJP
    (JJ natural)
    (SBAR+S
      (NP (NNP Language) (NNP Processing))
      (VP (VBZ is) (ADJP (RB really) (JJ fun)))))
  (. .))
```

Another example,

```
sent1 = "The agency's first strips issue, collateralized by Freddie Mac 8%
securities pooled into a single security called a Giant, will be divided
into interest-only and principal-only securities."

tree = parser.parse(sent1)
display_tree(tree)
print tree
```

Plot:



Print:

```
(S+SBAR
  (S
    (NP (DT the) (NN agency) (POS 's))
    (NP (JJ first) (JJ strips) (NN issue))
    (, ,)
    (VP
      (VBN collateralized)
      (PP (IN by) (NP (NNP Freddie) (NNP Mac))))
    (NP (CD 8) (NN %) (NNS securities))
    (VP
      (VP
        (VBN pooled)
        (PP (IN into) (NP (DT a) (JJ single) (NN security))))
      (VBN called)
      (NP (DT a) (NNP Giant))))
  (, ,)
  (S
    (VP
      (MD will)
      (VB be)
      (VBN divided)
      (PP
        (IN into)
        (NP
          (JJ interest-only)
          (CC and)
          (JJ principal-only)
          (NNS securities))))
     (. .)))
```

QUESTION: Stuck in feats extraction part

Sorry, but I am very confused. I changed this in the test module:

```
tp = TransitionParser.load('featureextractor.py')
```

and I completed some code transition.py according to the video lectures. I made sure all the modules have all the imports they need and every time I run test.py I get the message: This is not a very good feature extractor. I made the part of the code that shows this message to comments with # and now when I run test.py it keeps going on and on for ages, using up the full capacity of my CPU.

How am I supposed to evaluate my features if there is no result?

ANSWER:

`TransitionParser.load()` should only be called with the name of a file that you created with `TransitionParser.save()`. These will be `swedish.model, english.model`, and `danish.model`.

What save and load actually do are use Python's pickle module to persist or reconstitute an in-memory instance of the TransitionParser class to/from disk. However, while you are just trying to get your code working, there's no need to save or load these files.

Usually if test.py runs for more than a couple of minutes, certainly if it goes for many minutes, then there is probably a bug in your implementation of the functions in transition.py. Things to look out for are: correctly checking for the preconditions of the operations you implement, correctly moving items between stack and buffer, and returning correct values, namely -1 if preconditions are not met, and anything else if they are.

QUESTION: SVD code with Python and R

The lecture 3.4 uses Matlab when describing SVD. This is code in Python and R.

Python, with Numpy and Scipy,

The example is the same as in lecture 3.4, there is 6 documents with 9 terms,

D1: T6, T9
D2: T1, T2
D3: T2, T5, T8
D4: T1, T4, T6, T8, T9
D5: T1, T7

D6: T3, T7
D7: T1, T3

The matrix, in Numpy array,

```
import numpy as np
# matrix, columns represent documents, rows represent terms,
A0 = np.array([
    [0, 1, 0, 1, 1, 0, 1],
    [0, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 1],
    [0, 0, 0, 1, 0, 0, 0],
    [0, 1, 1, 0, 0, 0, 0],
    [1, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 1, 1, 0],
    [0, 0, 1, 1, 0, 0, 0],
    [1, 0, 0, 1, 0, 0, 0]], dtype=float)


# normalize

from sklearn.preprocessing import normalize


A = normalize(A0, axis=0, norm='l2')


np.set_printoptions(precision=4, suppress=True)
print A
```

Output:

```
[[ 0.     0.5774 0.     0.4472 0.7071 0.     0.7071 ]
 [ 0.     0.5774 0.5774 0.     0.     0.     0.     ]
 [ 0.     0.     0.     0.     0.     0.7071 0.7071 ]
 [ 0.     0.     0.     0.4472 0.     0.     0.     ]
 [ 0.     0.5774 0.5774 0.     0.     0.     0.     ]
 [ 0.7071 0.     0.     0.4472 0.     0.     0.     ]
 [ 0.     0.     0.     0.     0.7071 0.7071 0.     ]
 [ 0.     0.     0.5774 0.4472 0.     0.     0.     ]
 [ 0.7071 0.     0.     0.4472 0.     0.     0.     ]]
```

(1) SVD with numpy.linalg.svd, in this example we reduce matrix to rank-4 approximation,

```
from numpy.linalg import svd

    U, s, V = svd(A)
    # diagonal matrix
    S = np.zeros((U.shape[0], V.shape[0]), dtype=float)
    S[:V.shape[0], :V.shape[0]] = np.diag(s)


    # dimension
    print U.shape, V.shape, S.shape


    print "U"
    print U
    print "S"
    print S
    print "V"
    print V.T


    # reconstruction
    A1 = np.dot(U, np.dot(S, V))
    print "reconstruction"
    print A1

  # is close?

  print np.allclose(A, A1)
```

Output,

```
Shape:

(9, 9) (7, 7) (9, 7)


U

[[-0.6977 -0.0931  0.0175 -0.6951 -0.      0.0157  0.1441 -0.      0.    ]
 [-0.2619  0.2966  0.4681  0.1969 -0.     -0.2468 -0.157  -0.6356  0.3099]
 [-0.3527 -0.4491 -0.1017  0.4013  0.7071 -0.0066 -0.0493 -0.      0.    ]
 [-0.1121  0.141  -0.1478 -0.0733 -0.      0.4842 -0.8402 -0.     -0.    ]
 [-0.2619  0.2966  0.4681  0.1969  0.     -0.2468 -0.157   0.6356 -0.3099]
 [-0.1874  0.3747 -0.5049  0.127  -0.     -0.2287  0.0338 -0.3099 -0.6356]
 [-0.3527 -0.4491 -0.1017  0.4013 -0.7071 -0.0066 -0.0493 -0.      0.    ]
 [-0.2104  0.3337  0.0954  0.282   0.      0.734   0.4657  0.      0.    ]
 [-0.1874  0.3747 -0.5049  0.127   0.     -0.2287  0.0338  0.3099  0.6356]]


S

[[ 1.5777  0.      0.      0.      0.      0.      0.    ]
 [ 0.      1.2664  0.      0.      0.      0.      0.    ]
 [ 0.      0.      1.189   0.      0.      0.      0.    ]
 [ 0.      0.      0.      0.7962  0.      0.      0.    ]
 [ 0.      0.      0.      0.      0.7071  0.      0.    ]
 [ 0.      0.      0.      0.      0.      0.5664  0.    ]
 [ 0.      0.      0.      0.      0.      0.      0.1968]
 [ 0.      0.      0.      0.      0.      0.      0.    ]
 [ 0.      0.      0.      0.      0.      0.      0.    ]
]


V

[[-0.168   0.4184 -0.6005  0.2256  0.     -0.571   0.2432]
 [-0.4471  0.228   0.4631 -0.2185 -0.     -0.4872 -0.4986]
 [-0.2687  0.4226  0.5009  0.49    0.      0.2451  0.4451]
 [-0.3954  0.3994 -0.3929 -0.1305  0.      0.6132 -0.3697]
 [-0.4708 -0.3028 -0.0501 -0.2609 -0.7071  0.0113  0.3405]
 [-0.3162 -0.5015 -0.121   0.7128  0.     -0.0166 -0.3542]
 [-0.4708 -0.3028 -0.0501 -0.2609  0.7071  0.0113  0.3405]
]


reconstruction

[[ 0.      0.5774  0.      0.4472  0.7071 -0.      0.7071]
 [-0.      0.5774  0.5774  0.      0.     -0.      0.    ]
 [ 0.      0.      0.     -0.      0.      0.7071  0.7071]
 [ 0.      0.      0.      0.4472  0.      0.     -0.    ]
 [ 0.      0.5774  0.5774  0.      0.      0.      0.    ]
 [ 0.7071  0.      0.      0.4472  0.      0.      0.    ]
 [ 0.      0.      0.     -0.      0.7071  0.7071  0.    ]
 [ 0.      0.      0.5774  0.4472  0.     -0.      0.    ]
 [ 0.7071 -0.      0.      0.4472  0.      0.      0.    ]
]


True
```

Reconstruction matrix is the same as original matrix.

Now, rank-4 approximation,

```
# now, reduce to DIM, rank-4 approximation

  DIM = 4

    print "REDUCE to", DIM
    U1 = U[:, :DIM]
    S1 = S[:DIM, :DIM]
    V1 = V.T[:, :DIM].T


    print "U1"
```

```
      print U1
      print "S1"
      print S1
      print "V1"
      print V1.T


      # reconstruction on matrices reduction
      A2 = np.dot(U1, np.dot(S1, V1))
      print "reduced reconstruction:"
      print A2
```

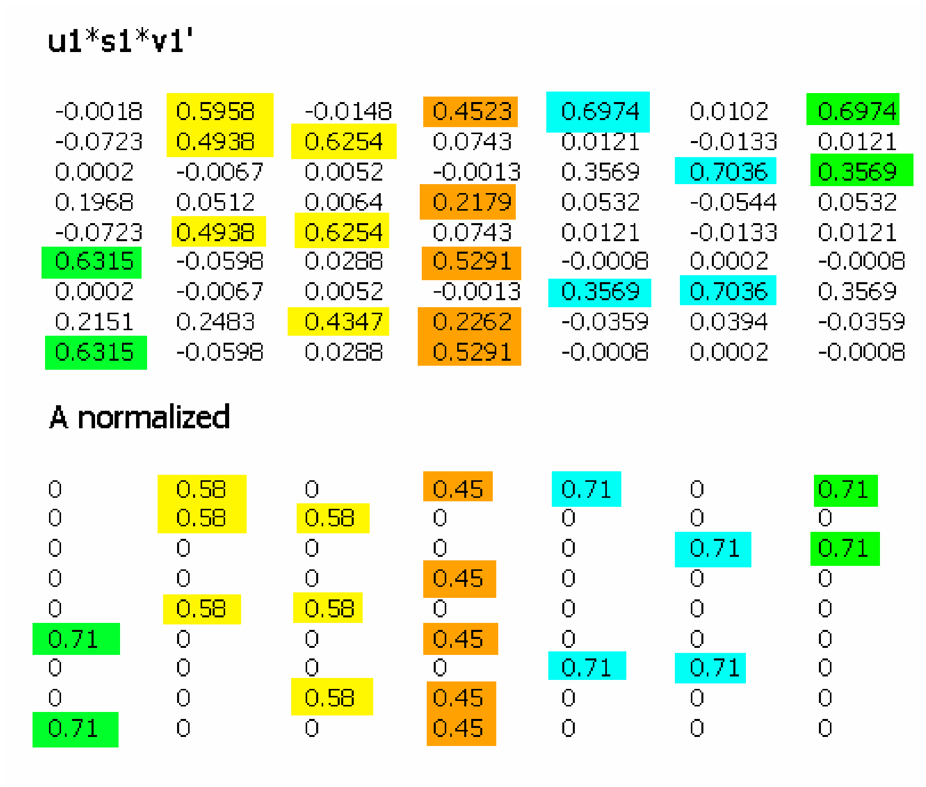
Output: (U1 is now 9x4, V1 is now 7x4, S1 is now 4x4)

REDUCE to 4

```
  U1

  [[-0.6977 -0.0931  0.0175 -0.6951]
   [-0.2619  0.2966  0.4681  0.1969]
   [-0.3527 -0.4491 -0.1017  0.4013]
   [-0.1121  0.141  -0.1478 -0.0733]
   [-0.2619  0.2966  0.4681  0.1969]
   [-0.1874  0.3747 -0.5049  0.127 ]
   [-0.3527 -0.4491 -0.1017  0.4013]
   [-0.2104  0.3337  0.0954  0.282 ]
   [-0.1874  0.3747 -0.5049  0.127 ]]

  S1
  [[ 1.5777  0.      0.      0.    ]
   [ 0.      1.2664  0.      0.    ]
   [ 0.      0.      1.189   0.    ]
   [ 0.      0.      0.      0.7962]]


  V1

  [[-0.168   0.4184 -0.6005  0.2256]
   [-0.4471  0.228   0.4631 -0.2185]
   [-0.2687  0.4226  0.5009  0.49  ]
   [-0.3954  0.3994 -0.3929 -0.1305]
   [-0.4708 -0.3028 -0.0501 -0.2609]
   [-0.3162 -0.5015 -0.121   0.7128]
   [-0.4708 -0.3028 -0.0501 -0.2609]]


  reduced reconstruction:
  [[-0.0018  0.5958 -0.0148  0.4523  0.6974  0.0102  0.6974]
   [-0.0723  0.4938  0.6254  0.0743  0.0121 -0.0133  0.0121]
   [ 0.0002 -0.0067  0.0052 -0.0013  0.3569  0.7036  0.3569]
   [ 0.1968  0.0512  0.0064  0.2179  0.0532 -0.054   0.0532]
   [-0.0723  0.4938  0.6254  0.0743  0.0121 -0.0133  0.0121]
```

```
[ 0.6315 -0.0598  0.0288  0.5291 -0.0008  0.0002 -0.0008]
[ 0.0002 -0.0067  0.0052 -0.0013  0.3569  0.7036  0.3569]
[ 0.2151  0.2483  0.4347  0.2262 -0.0359  0.0394 -0.0359]
[ 0.6315 -0.0598  0.0288  0.5291 -0.0008  0.0002 -0.0008]
]
```

Compare the reduced (rank-4 approximation) reconstruction matrix with the original A matrix, (the result can be different a bit because of different precision),



u1*s1*v1'

-0.0018	0.5958	-0.0148	0.4523	0.6974	0.0102	0.6974
-0.0723	0.4938	0.6254	0.0743	0.0121	-0.0133	0.0121
0.0002	-0.0067	0.0052	-0.0013	0.3569	0.7036	0.3569
0.1968	0.0512	0.0064	0.2179	0.0532	-0.0544	0.0532
-0.0723	0.4938	0.6254	0.0743	0.0121	-0.0133	0.0121
0.6315	-0.0598	0.0288	0.5291	-0.0008	0.0002	-0.0008
0.0002	-0.0067	0.0052	-0.0013	0.3569	0.7036	0.3569
0.2151	0.2483	0.4347	0.2262	-0.0359	0.0394	-0.0359
0.6315	-0.0598	0.0288	0.5291	-0.0008	0.0002	-0.0008

A normalized

0	0.58	0	0.45	0.71	0	0.71
0	0.58	0.58	0	0	0	0
0	0	0	0	0	0.71	0.71
0	0	0	0.45	0	0	0
0	0.58	0.58	0	0	0	0
0.71	0	0	0.45	0	0	0
0	0	0	0	0.71	0.71	0
0	0	0.58	0.45	0	0	0
0.71	0	0	0.45	0	0	0

(2) SVD with `scipy.sparse.linalg.svds`, just the same as previous example, in this example we reduce matrix A to rank-4 approximation (dimensionality reduction),

```
from scipy.sparse.linalg import svds
 u, s, vt = svds(A, 4)
 print "u"
 print u
 print "s4"
 print np.diag(s)
 print "vt"
 print vt


 print "u * s4 * vt"
 print np.dot(u, np.dot(np.diag(s), vt))
```

Output:

```
u

[[-0.6951  0.0175 -0.0931  0.6977]
 [ 0.1969  0.4681  0.2966  0.2619]
 [ 0.4013 -0.1017 -0.4491  0.3527]
 [-0.0733 -0.1478  0.141   0.1121]
 [ 0.1969  0.4681  0.2966  0.2619]
 [ 0.127  -0.5049  0.3747  0.1874]
 [ 0.4013 -0.1017 -0.4491  0.3527]
 [ 0.282   0.0954  0.3337  0.2104]
 [ 0.127  -0.5049  0.3747  0.1874]]


s4

[[ 0.7962  0.      0.      0.    ]
 [ 0.      1.189   0.      0.    ]
 [ 0.      0.      1.2664  0.    ]
 [ 0.      0.      0.      1.5777]]


vt

[[ 0.2256 -0.2185  0.49   -0.1305 -0.2609  0.7128 -0.2609]
 [-0.6005  0.4631  0.5009 -0.3929 -0.0501 -0.121  -0.0501]
 [ 0.4184  0.228   0.4226  0.3994 -0.3028 -0.5015 -0.3028]
 [ 0.168   0.4471  0.2687  0.3954  0.4708  0.3162  0.4708]]


u * s4 * vt

[[-0.0018  0.5958 -0.0148  0.4523  0.6974  0.0102  0.6974]
 [-0.0723  0.4938  0.6254  0.0743  0.0121 -0.0133  0.0121]
 [ 0.0002 -0.0067  0.0052 -0.0013  0.3569  0.7036  0.3569]
 [ 0.1968  0.0512  0.0064  0.2179  0.0532 -0.054   0.0532]
 [-0.0723  0.4938  0.6254  0.0743  0.0121 -0.0133  0.0121]
 [ 0.6315 -0.0598  0.0288  0.5291 -0.0008  0.0002 -0.0008]
 [ 0.0002 -0.0067  0.0052 -0.0013  0.3569  0.7036  0.3569]
 [ 0.2151  0.2483  0.4347  0.2262 -0.0359  0.0394 -0.0359]
 [ 0.6315 -0.0598  0.0288  0.5291 -0.0008  0.0002 -0.0008]]
```

This Scipy's reconstruction matrix of rank-4 approximation is the same as previous Numpy's reconstruction result.

Conclusion, if we have a large Documents-Terms matrix, then we reduce its dimensionality with SVD, we are able to reproduce the approximation of original matrix with smaller size of decomposed matrices.

(3) With R, almost the same,

```
options(digits=4)


A0 = c(
    0, 1, 0, 1, 1, 0, 1,
    0, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 1, 1,
    0, 0, 0, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 0, 0,
    1, 0, 0, 1, 0, 0, 0,
    0, 0, 0, 0, 1, 1, 0,
    0, 0, 1, 1, 0, 0, 0,
    1, 0, 0, 1, 0, 0, 0)

  A = matrix(A0, nrow=9, ncol=7, byrow=T)


  # l2 norm by matrixcolumn
  A = apply(A, 2, FUN=function(x) {return(x/sqrt(sum(x^2)))})


  # SVD
  s = svd(A)


  # left singular vectors,
  s$u
```

output:

```
          [,1]     [,2]     [,3]     [,4]        [,5]      [,6]     [,7]
   [1,] -0.6977 -0.09314  0.01754 -0.69510 -6.103e-17  0.015667  0.14406
   [2,] -0.2619  0.29659  0.46807  0.19690  1.949e-17 -0.246804 -0.15701
   [3,] -0.3527 -0.44910 -0.10170  0.40133  7.071e-01 -0.006643 -0.04929
   [4,] -0.1121  0.14104 -0.14778 -0.07332 -1.013e-16  0.484193 -0.84017
   [5,] -0.2619  0.29659  0.46807  0.19690  1.610e-17 -0.246804 -0.15701
   [6,] -0.1874  0.37468 -0.50492  0.12700  6.641e-33 -0.228665  0.03385
   [7,] -0.3527 -0.44910 -0.10170  0.40133 -7.071e-01 -0.006643 -0.04929
   [8,] -0.2104  0.33370  0.09544  0.28200  1.623e-16  0.734046  0.46573
   [9,] -0.1874  0.37468 -0.50492  0.12700  6.641e-33 -0.228665  0.03385
```

```
  # right singular vectors

  s$v
```

output:

```
         [,1]    [,2]     [,3]    [,4]        [,5]     [,6]    [,7]
  [1,] -0.1680  0.4184 -0.60054  0.2256  0.000e+00 -0.57097  0.2432
  [2,] -0.4471  0.2280  0.46307 -0.2185 -1.497e-16 -0.48721 -0.4986
  [3,] -0.2687  0.4226  0.50089  0.4900  1.482e-16  0.24510  0.4451
  [4,] -0.3954  0.3994 -0.39291 -0.1305  0.000e+00  0.61320 -0.3697
  [5,] -0.4708 -0.3028 -0.05005 -0.2609 -7.071e-01  0.01127  0.3405
  [6,] -0.3162 -0.5015 -0.12097  0.7128  2.081e-16 -0.01659 -0.3542
  [7,] -0.4708 -0.3028 -0.05005 -0.2609  7.071e-01  0.01127  0.3405
```

```
  # singular values, to diagonal matrix,

  diag(s$d)
```

output:

```
        [,1]  [,2]  [,3]   [,4]   [,5]   [,6]   [,7]
  [1,] 1.578 0.000 0.000 0.0000 0.0000 0.0000 0.0000
  [2,] 0.000 1.266 0.000 0.0000 0.0000 0.0000 0.0000
  [3,] 0.000 0.000 1.189 0.0000 0.0000 0.0000 0.0000
  [4,] 0.000 0.000 0.000 0.7962 0.0000 0.0000 0.0000
  [5,] 0.000 0.000 0.000 0.0000 0.7071 0.0000 0.0000
  [6,] 0.000 0.000 0.000 0.0000 0.0000 0.5664 0.0000
  [7,] 0.000 0.000 0.000 0.0000 0.0000 0.0000 0.1968
```

```
  # helper function
  reduce = function(s, d) {
    u = as.matrix(s$u[, 1:d])
    v = as.matrix(s$v[, 1:d])
    # Create the diagonal matrix dm,
    dm = as.matrix(diag(s$d)[1:d, 1:d])
    return(list(u=u, v=v, d=dm))
  }


  # reduced to rank-4
  reduced_s = reduce(s, 4)
  reduced_s$u
```

output:

```
           [,1]     [,2]     [,3]     [,4]
   [1,] -0.6977 -0.09314  0.01754 -0.69510
   [2,] -0.2619  0.29659  0.46807  0.19690
   [3,] -0.3527 -0.44910 -0.10170  0.40133
   [4,] -0.1121  0.14104 -0.14778 -0.07332
   [5,] -0.2619  0.29659  0.46807  0.19690
   [6,] -0.1874  0.37468 -0.50492  0.12700
   [7,] -0.3527 -0.44910 -0.10170  0.40133
   [8,] -0.2104  0.33370  0.09544  0.28200
   [9,] -0.1874  0.37468 -0.50492  0.12700
```

```
  reduced_s$v

  output:
          [,1]    [,2]     [,3]    [,4]
   [1,] -0.1680  0.4184 -0.60054  0.2256
   [2,] -0.4471  0.2280  0.46307 -0.2185
   [3,] -0.2687  0.4226  0.50089  0.4900
   [4,] -0.3954  0.3994 -0.39291 -0.1305
   [5,] -0.4708 -0.3028 -0.05005 -0.2609
   [6,] -0.3162 -0.5015 -0.12097  0.7128
   [7,] -0.4708 -0.3028 -0.05005 -0.2609
```

```
reduced_s$d
```

output:

```
      [,1]  [,2]  [,3]   [,4]
[1,] 1.578 0.000 0.000 0.0000
[2,] 0.000 1.266 0.000 0.0000
[3,] 0.000 0.000 1.189 0.0000
[4,] 0.000 0.000 0.000 0.7962
```

```
# reconstruction,
recon_A = reduced_s$u %*% reduced_s$A %*% t(reduced_s$v)
recon_A
```

output:

```
           [,1]     [,2]      [,3]      [,4]      [,5]       [,6]      [,7]
[1,] -0.0018294  0.59581 -0.014794  0.452253  0.697353  0.0101897  0.697353
[2,] -0.0722961  0.49384  0.625364  0.074291  0.012096 -0.0132637  0.012096
[3,]  0.0002113 -0.00667  0.005240 -0.001279  0.356899  0.7036082  0.356899
[4,]  0.1967947  0.05117  0.006379  0.217931  0.053211 -0.0540198  0.053211
[5,] -0.0722961  0.49384  0.625364  0.074291  0.012096 -0.0132637  0.012096
[6,]  0.6315410 -0.05978  0.028778  0.529090 -0.000809  0.0002113 -0.000809
[7,]  0.0002113 -0.00667  0.005240 -0.001279  0.356899  0.7036082  0.356899
[8,]  0.2150831  0.24825  0.434657  0.226166 -0.035893  0.0393623 -0.035893
[9,]  0.6315410 -0.05978  0.028778  0.529090 -0.000809  0.0002113 -0.000809
```

Again, the reconstruction matrix is similar to previous calculations in Python.

Why the result is a bit different between Numpy, Scipy, R, Matlab/Octave?

http://stackoverflow.com/questions/5935893/any-reason-why-octave-r-numpy-and-lapack-yield-different-...

http://stackoverflow.com/questions/26589448/numpy-seems-to-produce-incorrect-eigenvectors

Another example, http://glowingpython.blogspot.com/2011/05/latent-semantic-analysis-with-term.html

External Links

Python

Python Tutorial and Reference Books

Official Python Docs

https://docs.python.org/2/

Learning Python

http://www.worldcat.org/title/learning-python/oclc/863949628?referer=br&ht=edition

The Python Standard Library by Example

http://www.worldcat.org/title/python-standard-library-by-example/oclc/682896798%26referer=brief_results

Python in a Nutshell

http://www.worldcat.org/title/python-in-a-nutshell/oclc/754143508?referer=br&ht=edition

Python Cookbook

http://www.worldcat.org/title/python-cookbook/oclc/826659391%26referer=brief_results

The Quick Python Book

http://www.worldcat.org/title/quick-python-book/oclc/866849765

Think Python

http://www.greenteapress.com/thinkpython/

Python Phrasebook

http://www.worldcat.org/title/python-phrasebook-essential-code-and-commands/oclc/150388343%26referer=brief_results

Python in Practice

http://www.worldcat.org/title/python-phrasebook-essential-code-and-commands/oclc/150388343%26referer=brief_results

Expert Python Programming

http://www.worldcat.org/title/expert-python-programming-learn-best-practices-to-designing-coding-and-distributing-your-python-software/oclc/401988536%26referer=brief_results

Online Python Classes

Programming for Everybody (Getting Started with Python)

https://www.coursera.org/learn/python

Learn to Program: The Fundamentals

https://www.coursera.org/course/programming1

Codecademy Learn Python

https://www.codecademy.com/learn/python

Subreddit dedicated to learning python

https://www.reddit.com/r/learnpython

Corpus Linguistics

Mark Davies, a corpus linguist at Brigham Young University.

Web-based search interface to a list of corpora, including the British National Corpus (bnc) and the Corpus of Contemporary American English (coca).

This interface allows searching for words, lemmas, POS tags, and combinations of those with a limited support for regular expressions.

In order to use the search interface, one should register, which is free. Without registration, the system limits the number of queries.

No interfacing API.

http://corpus.byu.edu/

Deep Learning for NLP

Deep Learning for NLP Links

http://web.eecs.umich.edu/~radev/dlnlp/list.txt

Richard Socher's class

http://cs224d.stanford.edu/syllabus.html

Deep learning libraries

Theano (Python)

http://deeplearning.net/software/theano/

cafe - image classification (PyCaffe interface)

http://caffe.berkeleyvision.org/

torch - machine learning framework (Lua/C)

http://www.torch.ch/

Open courseware for mathematical prerequisites, machine learning in general, and deep learning

Deep Learning for Text Mining From Scratch

http://textminingonline.com/deep-learning-for-text-mining-from-scratch

More deep learning links

http://deeplearning.net/software_links

Edit Distance

http://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance/

http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Edit/

http://www.let.rug.nl/~kleiweg/lev/

Coreference Tools

JavaRAP

http://aye.comp.nus.edu.sg/~qiu/NLPTools/JavaRAP.html

Bart

http://www.bart-coref.org/

GuiTAR

http://cswww.essex.ac.uk/Research/nle/GuiTAR/

ARKref

http://www.ark.cs.cmu.edu/ARKref/

Definite Coreference

http://nlp.stanford.edu/software/dcoref.shtml

Information Extraction

OpenIE extracts binary relations as 3-tuples of natural language text

OpenIE project page

http://openie.allenai.org/

OpenIE source code

https://github.com/knowitall/openie

Paralex maps open-domain factoid questions to executable queries over a tuple store

Paralex home page

http://knowitall.cs.washington.edu/paralex/

Paraphrase-Driven Learning for Open QA

http://turing.cs.washington.edu/papers/acl-2013-fader.pdf

Latent Semantic Analysis

http://glowingpython.blogspot.com/2011/05/latent-semantic-analysis-with-term.html

Linguistics courses

Coursera - Miracles of Human Language

https://www.coursera.org/course/humanlanguage

Machine translation tutorials

Philipp Koehn

http://mt-class.org/jhu/slides/lecture-ibm-model1.pdf

Chris Manning

http://web.stanford.edu/class/cs224n/handouts/cs224n-lecture3-MT-6up.pdf

Chris Callison-Burch

https://www.cs.jhu.edu/~jason/465/PowerPoint/lect32a-mt-word-based-models.pdf

Michael Collins

http://www.cs.columbia.edu/~mcollins/ibm12.pdf

More Pointers

Quora includes a lot of relevant questions and answers about NLP and related topics.

http://www.quora.com/

Morphological Analysis of German

https://code.google.com/p/morphisto

http://www.aclweb.org/aclwiki/index.php?title=Resources_for_German

https://en.wikipedia.org/wiki/Germanic_languages

http://stackoverflow.com/questions/680907/is-there-a-free-library-for-morphological-analysis-of-the-german-language

Natural Language Generation

nlgserv

A server that accepts JSON representations of sentences and generates English sentences. A wrapper on simplenlg.

https://pypi.python.org/pypi/nlgserv/

nlgserv PyPi installation page

https://github.com/mnestis/nlgserv

nlgserv source code

https://github.com/mnestis/nlgserv

simplenlg source code (Java)

https://github.com/simplenlg/simplenlg

pypolibox

database-to-text generation using openccg

pypolibox PyPi installation page

https://pypi.python.org/pypi/pypolibox

pypolibox source code

https://github.com/arne-cl/pypolibox

OpenCCG source code (Java)

https://github.com/OpenCCG/openccg

Definitions of some NLG terms, links to NLG systems, and a long list of related papers.

meta-guide: realizers in natural language processing

http://meta-guide.com/dialog-systems/nlg/realizers-in-natural-language-processing/

NLG paper by Iyyer, Boyd-Graber, Daumé III

Generating Sentences from Semantic Vector Space Representations

http://www.cs.umd.edu/~miyyer/pubs/2014_nips_generation.pdf

Functional Unification Formalism NLG

Note: this appears to be a Google Summer of Code project that has not been updated since 2008

Implementing FUF in NLTK

http://www.ling.helsinki.fi/kit/2010s/clt310gen/docs/ImplementingFUFinNLTK.pdf

FUF in NLTK source code

https://github.com/nltk/nltk_contrib/tree/master/nltk_contrib/fuf

Notation guides

Parts of Speech

Penn Treebank parts of speech notations guide:

Full guide

https://catalog.ldc.upenn.edu/docs/LDC99T42/tagguid1.pdf

Summary

http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html

Constituents

Penn Treebank constituents notations guide:

https://catalog.ldc.upenn.edu/docs/LDC99T42/prsguid1.pdf

Dependences

Stanford parser dependencies notations guide:

http://nlp.stanford.edu/software/dependencies_manual.pdf

Summary of various linguistic notations:

http://web.mit.edu/6.863/www/PennTreebankTags.html

Parsers

Simple Python statistical (CKY) parser together with scripts for learning the PCFG from the QuestionBank and Penn treebanks:

https://github.com/emilmont/pyStatParser

Earley parser:

http://www.inf.ed.ac.uk/teaching/courses/inf2a/slides/2012_inf2a_L19_slides.pdf

http://courses.washington.edu/ling571/ling571_fall_2010/slides/parsing_earley.pdf

http://www.cse.unt.edu/~tarau/teaching/NLP/Earley%20parser.pdf

https://www.cs.unm.edu/~luger/ai-final2/CH9_Dynamic%20Programming%20and%20the%20Earley%20Parser.pdf

Polysemous words

http://norvig.com/ticip.html

Porter's Stemming Algorithm

https://www.eecis.udel.edu/~trnka/CISC889-11S/lectures/dan-porters.pdf

Singular Value Decomposition (SVD)

Decoding Dimensionality Reduction, PCA and SVD

http://bigdata-madesimple.com/decoding-dimensionality-reduction-pca-and-svd/

Relationship between SVD and PCA. How to use SVD to perform PCA?

http://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca

Why Octave, R, Numpy, and LAPACK yield different SVD results on the same matrix

http://stackoverflow.com/questions/5935893/any-reason-why-octave-r-numpy-and-lapack-yield-different-svd-results-on-the-sa

"Normalizing" variables for SVD / PCA

http://stats.stackexchange.com/questions/12200/normalizing-variables-for-svd-pca

Numpy seems to produce incorrect eigenvectors

http://stackoverflow.com/questions/26589448/numpy-seems-to-produce-incorrect-eigenvectors

Why we need to normalize data before analysis

http://stats.stackexchange.com/questions/69157/why-do-we-need-to-normalize-data-before-analysis

Understand Complex Datasets, Data Mining with Matrix Decompositions

https://books.google.co.id/books?id=9SzLDi8jUnAC

dimensionality reduction -- general sense of the terms used in the lecture

https://class.coursera.org/nlpintro-001/forum/thread?thread_id=104#post-410

Low-rank approximation of a matrix

https://inst.eecs.berkeley.edu/%7Eee127a/book/login/l_svd_low_rank.html

Singular Value Decomposition of General Matrices https://inst.eecs.berkeley.edu/%7Eee127a/book/login/l_svd_main.html

Text to Speech (TTS)

How-To Geek

The Best Text to Speech Software and Online Tools

http://www.howtogeek.com/125305/the-best-text-to-speech-tts-software-programs-and-online-tools/

Wizzard Speech LLC

AT&T Natural Voices Text to Speech SDK

http://www.wizzardsoftware.com/text-to-speech-sdk.php

Tokenization Comparisons

http://text-processing.com/demo/tokenize/

Topic modeling and text similarity

https://radimrehurek.com/gensim/