speech_tools for ROS

John Parker
Intelligent Robotics III
Spring 2015

speech_tools

- A set of ROS nodes that accomplish the following:
  - 1) **speech_recognition.py** – uses pocketsphinx speech recognizer with the cmu07a.dict dictionary and a probabilistic language model en-us.lm, which is the best general language model for that dictionary
  - 2) **speech_recognition_nb.py** – calls pocketsphinx using jeeves.dict and jeeves.lm, custom files that the user generates to match only the words found in the Naive Bayes and Keyword Extraction training data.
  - 3) **speech_synthesis.py** – uses the Festival speech synthesis system to 'speak' all messages sent to the ROS topic to which it subscribes
  - 4) **speech_integrated.py** – combines the Naive Bayes text classification method, and the A.L.I.C.E chatbot system to make a robust speech system that can use flexible AIML scripts for conversation and can also respond to commands with high accuracy using the Naive Bayes classification system.

speech_tools

Software Requrements:

The speech_recognition.py and speech_recognition_nb.py nodes require that you have installed on your machine pocketsphinx from CMU. You can install either from a debian repository, or from source, but the pocketsphinx binary must be in your bash execution path for these ROS nodes to work correctly (the binary will be placed in your bash execution path automatically if you install from debian repo or follow all source install instructions from CMU)

The speech_synthesis.py node requires that you have installed the Festival speech recognition system. This can be obtained from the Ubuntu debain repositories.

speech_tools

System Startup:

If you want to use the entire system, start all four ROS nodes in no particular order.

If you only want to use the speech recognition modules, then startup speech_recognition.py if you want to use the standard 60,000+ word dictionary from CMU, or startup speech_recognition_nb.py if you would like to use your own custom language dictionary and language model (read further slides for instructions). Before you start the node(s), make sure that the audio system settings on your machine are configured to receive input from your microphone channel. After you have started the node, you can listen on the ROS topic that the node publishes to in order to read the output of the pocketsphinx speech recognizer.

If you only want to use the speech synthesizer, then startup speech_synthesis.py. If you publish messages on the ROS topic that this node subscirbes to, the text string in those messages will be synthesized into speech over your machine's audio system by the Festival speech synthesis program.

speech_integrated.py – How it Works

- Combines the A.L.I.C.E. and Naive Bayes speech methods
- ***If*** an A.L.I.C.E. string was matched, print the A.L.I.C.E response
- ***Else*** print the Naive Bayes classification, which always gives a match

- NOTE: to make sure the system works, make sure there is no 'catch all' phrase using wildcards in any of the A.L.I.C.E. scripts, as this will prevent the Naive Bayes classification system from functioning

Making changes to the A.L.I.C.E. scripts

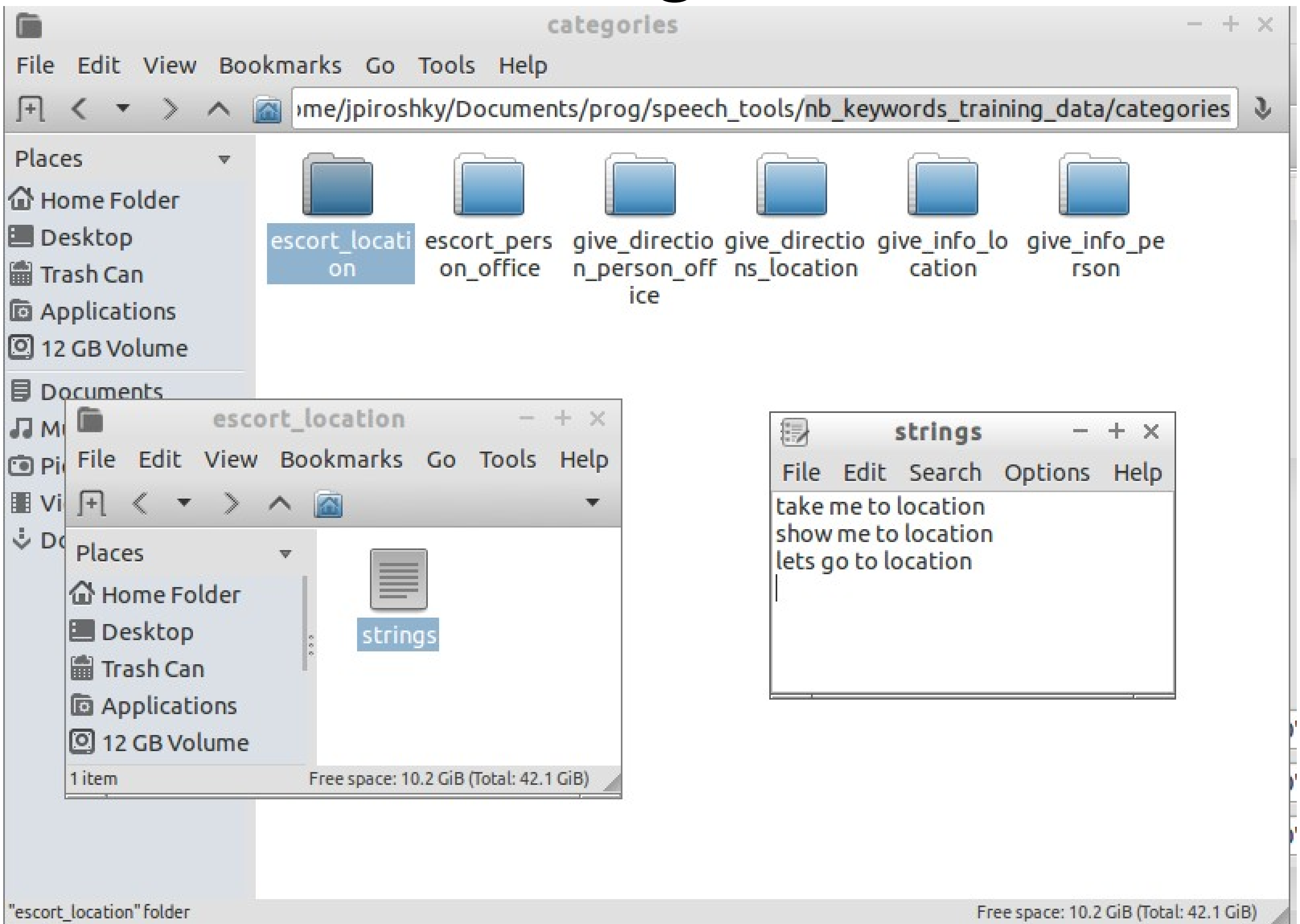
- **To change the A.L.I.C.E. scripts:**
  - All AIML scripts to be used with A.L.I.C.E. are found in the folder ./jeevespersonality/
  - You can edit the scripts, or if you add a new one, be sure to add it to ./jeevespersonality-startup.xml
  - All changes are effective as soon as the speech_integrated.py ROS node is restarted
  - For information on the AIML standard, reference:
    - AIML tutorial: http://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html
    - AIML reference manual: http://www.alicebot.org/documentation/aiml-reference.html

Making changes to the Naive Bayes training data

- All training data is found in ./nb_keywords_training_data/
- The structure is as follows:
  - ./nb_keywords_training_data/categories/

  contains a set of directories whose names are the text classes to be recognized by the Naive Bayes classifier system. Each directory contains a single text document that contains all the training data for that class.
- To make changes, simply add or remove sentences from the text document in a particular class.
- To add a new class, simply make a new directory whose name is the class, and create a text document in the directory that includes the sentences that comprise the training data for that class.

Making changes to the Naive Bayes training data

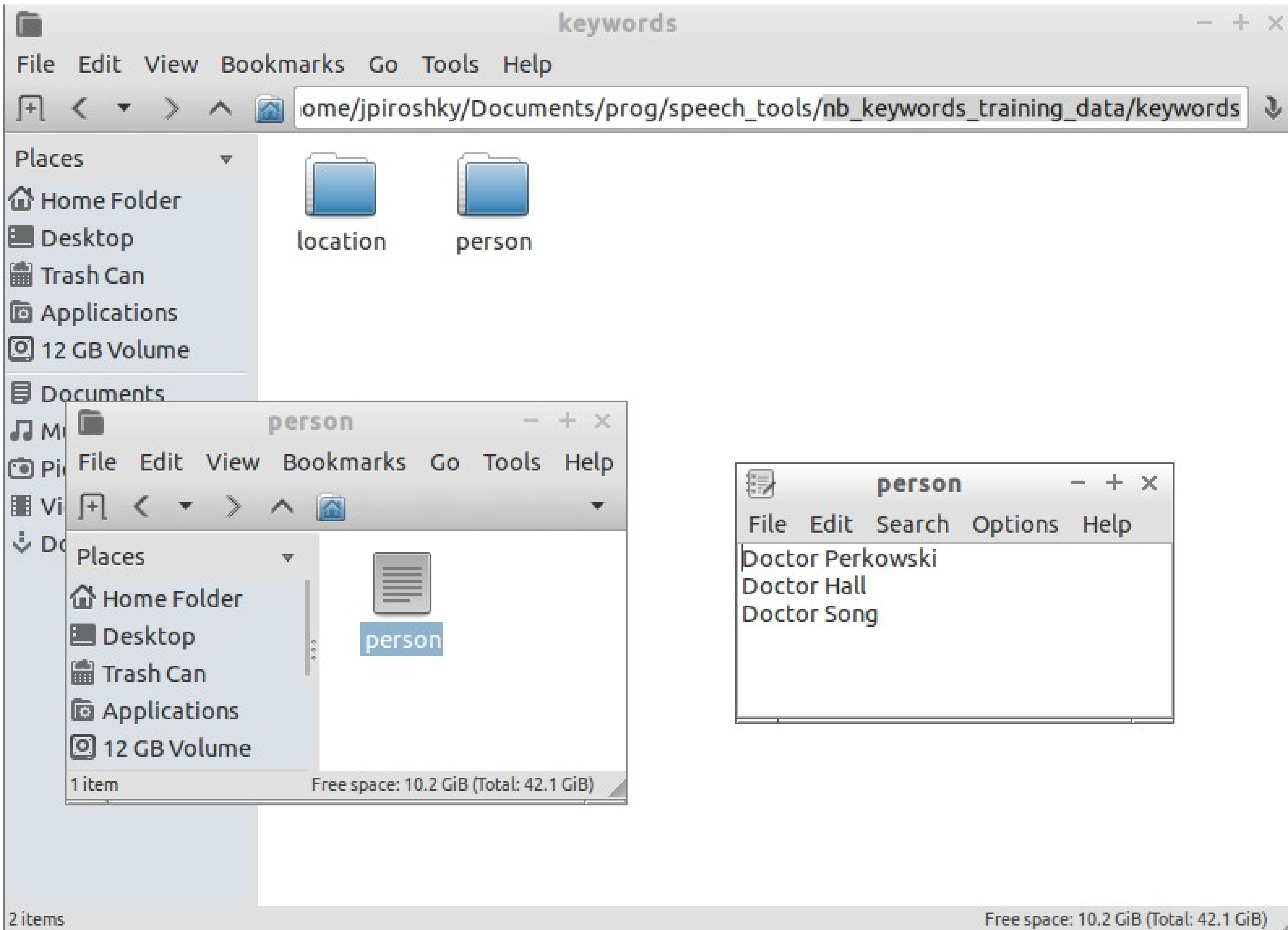


Making changes to the Keywords training data

- All training data is found in ./nb_keywords_training_data/
- The structure is as follows:
  - ./nb_keywords_training_data/keywords/

  contains a set of directories whose names are the keyword classes to be recognized by the Keyword extraction system. Each directory contains a single text document that contains all the training data for that class.
- To make changes, simply add or remove keywords from the text document in a particular class.
- To add a new class, simply make a new directory whose name is the class, and create a text document in the directory that includes the keywords that comprise the training data for that class.

Making changes to the Keywords training data



Updating the jeeves.dic and jeeves.lm file

- After making changes to the Naive Bayes and Keyword training data, the speech recognition dictionary and language model need to be updated to reflect the changes
- Run the script **./nb_keyword_training_data/grab_training_data.py**

to generate a file called **for_lmtool.txt** containing all the sentences and keywords found in the training data.

- Go to http://speech.cs.cmu.edu/tools/lmtool-new.html and use the online form to choose the file **for_lmtool.txt**, and then click **'COMPILE KNOWLEDGE BASE'**
- Download and save the .dic and .lm files as **jeeves.dic** and **jeeves.lm** in the **/speech_tools/** folder

Naive Bayes Text Classification and CMU Sphinx Dictionaries and Language Models

John Parker
Portland State University
Winter 2015

Naive Bayes Text Classification

What we will learn today:

- A machine learning technique that will take a text, and classify it according to categories that we specify.
- In essence, text classification is just a way of giving weighted scores based on prior evidence (training data), and we use Bayes Theorem instead of other ML techniques.
- For every input text, our algorithm will output a score for each category, indicating how likely it is that the text belongs to that category. The best score is our best guess.
- We will use Bayes Theorem with certain assumptions to make our score calculations very simple (naïve).

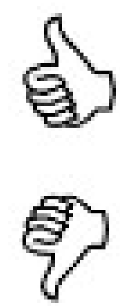
Assumptions

- Our first assumption is that every text is just a **'Bag of Words'**
  - This means in our Bag of Words model, only the frequency (how often) a word occurs in a text is important
  - Not important: word order, how near or far apart different words are from each other
- Our second assumption is **Conditional Independence**
  - This means that we treat the appearance of each word in our text like a statistically independent event
  - i.e. our score is **NOT** based on words x,y,z appearing together in our text, only that x appears, y appears, z..

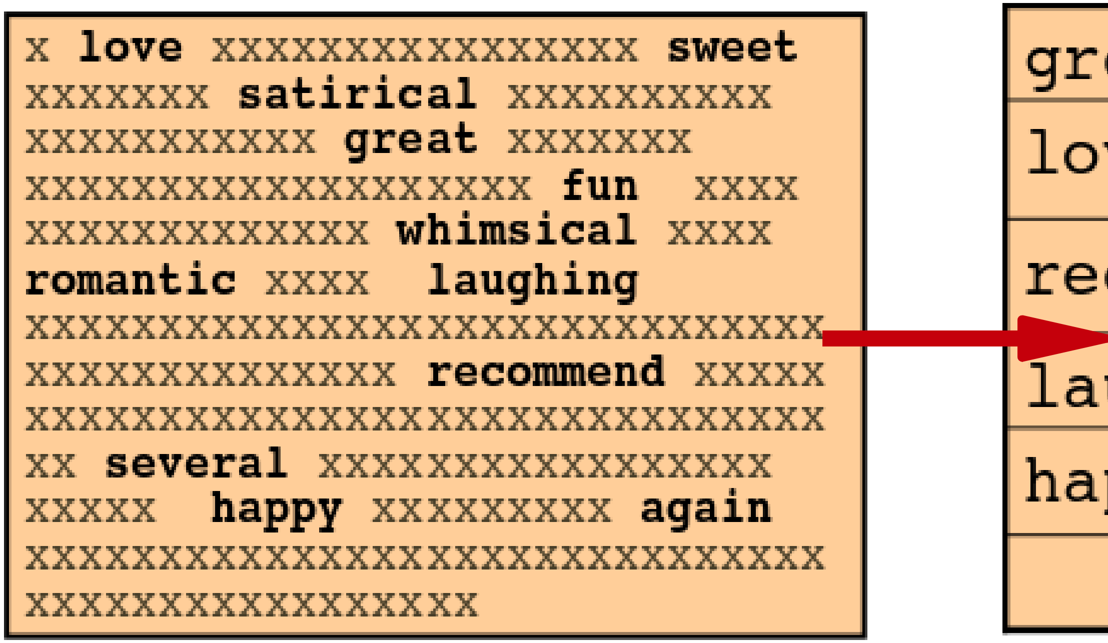
'Bag of Words' Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

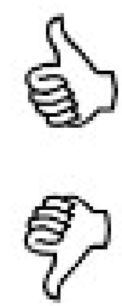
I **love** this movie! It's **sweet**, but with **satirical** humor. The dialogue is **great** and the adventure scenes are **fun…** It manages to be **whimsical** and **romantic** while **laughing** at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I've seen it **several** times, and I'm always **happy** to see it **again** whenever I have a friend who hasn't seen it yet.



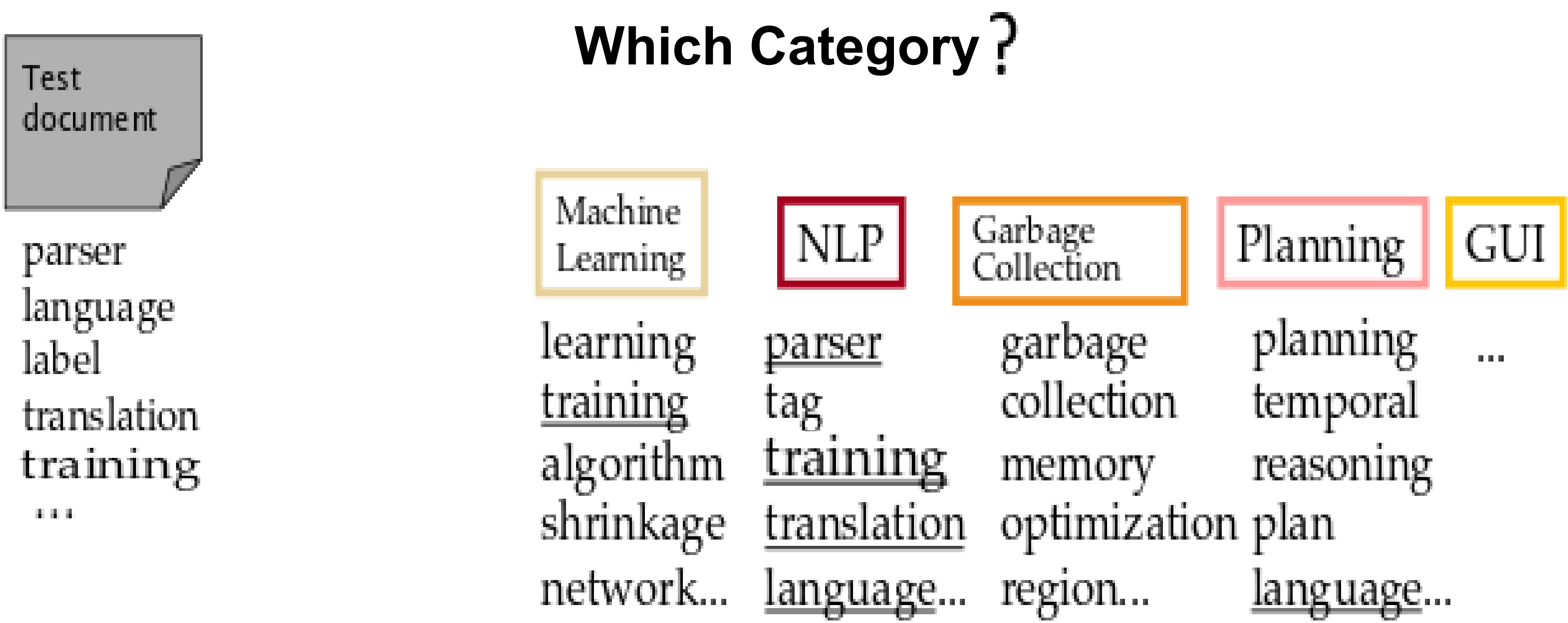
'Bag of Words' Representation



great	2
love	2
recommend	1
laugh	1
happy	1
...	...



'Bag of Words' Representation



Classification is a Score based on Weights

Dan Jurafsky

Bayes' Rule Applied to Documents and Classes

- For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Classification is a Score based on Weights

Dan Jurafsky

Bayes' Rule Applied to Documents and Classes

- For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Score

Weight s

Classification is a Score based on Weights

Dan Jurafsky

Bayes’ Rule Applied to Documents and Classes

- For a document *d* and a class *c*

probability that word set d is found in a given class c(i)

base probability of class c(i) relative to the other classes

$$P(c \mid d) = \frac{P(d \mid c)\,P(c)}{P(d)}$$

base probability of words in set d regardless of class

Score

**Weight
s**

Naïve Bayes Classifier (I)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

Naïve Bayes Classifier (II)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

Multinomial Naïve Bayes Independence Assumptions

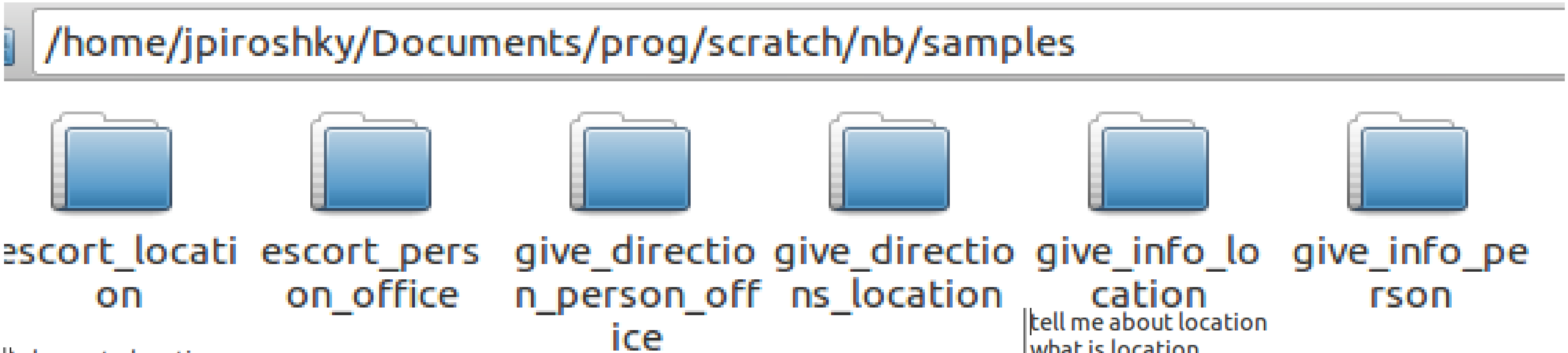
$$P(x_1, x_2, \ldots, x_n \mid c)$$

- **Bag of Words assumption**: Assume position doesn't matter
- **Conditional Independence**: Assume the feature probabilities $P(x_i \mid c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Probability of word x_n given class c

i.e. probability we draw word x_n out of our bag of words for class c!



take me to location
show me to where location is
lets go to location

take me to person office
show me to where person office is
lets go to person office

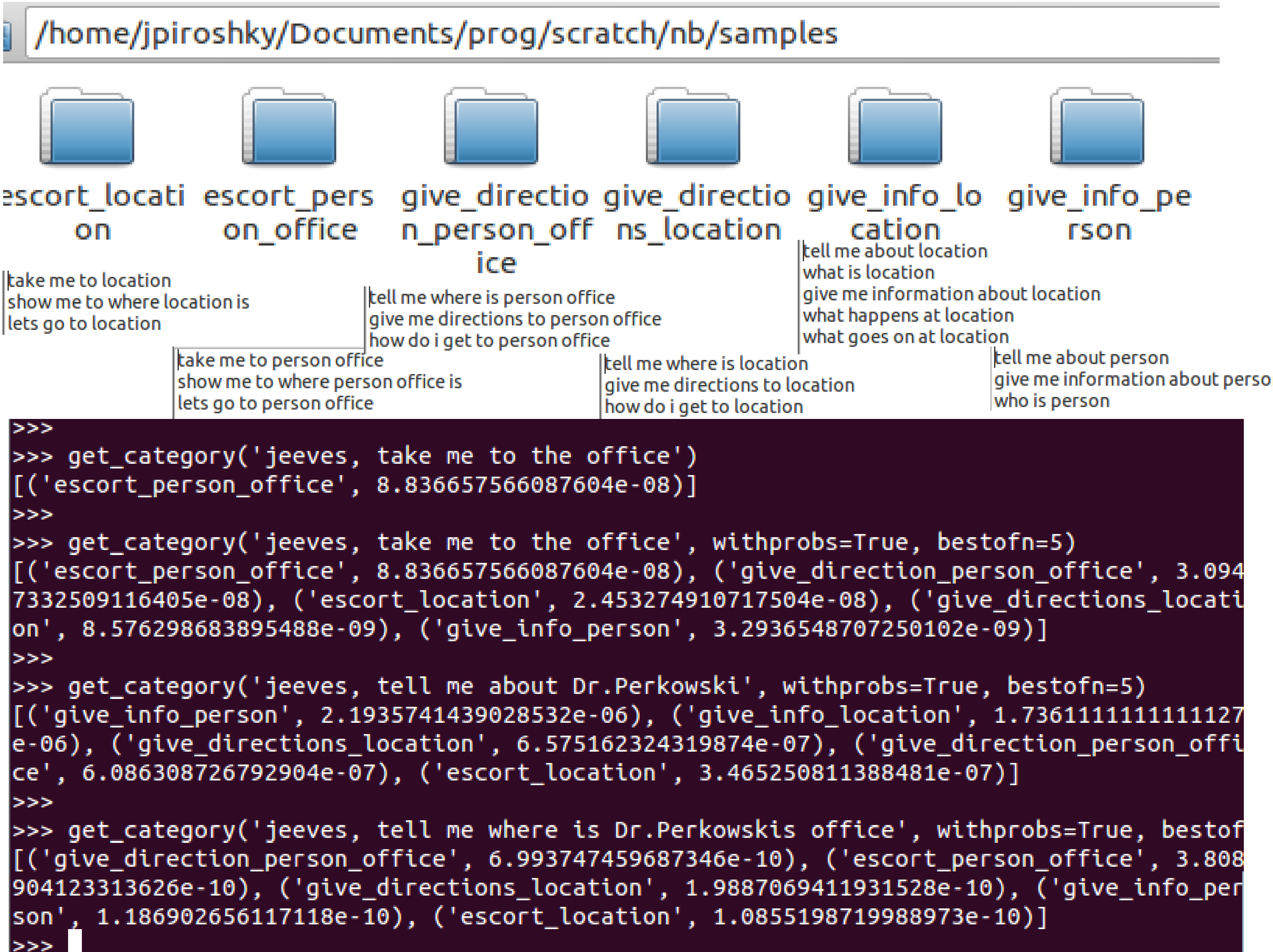
tell me where is person office
give me directions to person office
how do i get to person office

tell me where is location
give me directions to location
how do i get to location

tell me about location
what is location
give me information about location
what happens at location
what goes on at location

tell me about person
give me information about person
who is person

```
>>> execfile('nbeditlp.py')
>>> make_probabilities()
>>> p_word_given_category
{'give_info_person': {'office': 0.009009009009009009, 'give': 0.018018018018018018, 'is': 0.018018018018018018, 'at': 0.009009009009009009, 'g
o': 0.009009009009009009, 'happens': 0.009009009009009009, 'information': 0.018018018018018018, 'what': 0.009009009009009009, 'how': 0.0090090
09009009009, 'show': 0.009009009009009009, 'to': 0.009009009009009009, 'lets': 0.009009009009009009, 'take': 0.009009009009009009, 'goes': 0.0
09009009009009009, 'tell': 0.018018018018018018, 'location': 0.009009009009009009, 'do': 0.009009009009009009, 'get': 0.009009009009009009, 'w
ho': 0.018018018018018018, 'directions': 0.009009009009009009, 'me': 0.02702702702702703, 'on': 0.009009009009009009, 'about': 0.0270270270270
2703, 'i': 0.009009009009009009, 'person': 0.036036036036036036, 'where': 0.009009009009009009}, 'give_info_location': {'office': 0.0083333333
33333333, 'give': 0.016666666666666666, 'is': 0.016666666666666666, 'at': 0.025, 'go': 0.008333333333333333, 'happens': 0.016666666666666666,
 'information': 0.016666666666666666, 'what': 0.03333333333333333, 'how': 0.008333333333333333, 'show': 0.008333333333333333, 'to': 0.00833333
3333333333, 'lets': 0.008333333333333333, 'take': 0.008333333333333333, 'goes': 0.016666666666666666, 'tell': 0.016666666666666666, 'location':
 0.05, 'do': 0.008333333333333333, 'get': 0.008333333333333333, 'who': 0.008333333333333333, 'directions': 0.008333333333333333, 'me': 0.025,
 'on': 0.016666666666666666, 'about': 0.025, 'i': 0.008333333333333333, 'person': 0.008333333333333333, 'where': 0.008333333333333333}, 'give_d
irections_location': {'office': 0.008695652173913044, 'give': 0.017391304347826087, 'is': 0.017391304347826087, 'at': 0.008695652173913044, 'g
o': 0.008695652173913044, 'happens': 0.008695652173913044, 'information': 0.008695652173913044, 'what': 0.008695652173913044, 'how': 0.0173913
04347826087, 'show': 0.008695652173913044, 'to': 0.02608695652173913, 'lets': 0.008695652173913044, 'take': 0.008695652173913044, 'goes': 0.00
8695652173913044, 'tell': 0.017391304347826087, 'location': 0.034782608695652174, 'do': 0.017391304347826087, 'get': 0.017391304347826087, 'wh
o': 0.008695652173913044, 'directions': 0.017391304347826087, 'me': 0.02608695652173913, 'on': 0.008695652173913044, 'about': 0.00869565217391
3044, 'i': 0.017391304347826087, 'person': 0.008695652173913044, 'where': 0.017391304347826087}, 'escort_person_office': {'office': 0.03448275
8620689655, 'give': 0.008620689655172414, 'is': 0.017241379310344827, 'at': 0.008620689655172414, 'go': 0.017241379310344827, 'happens': 0.008
620689655172414, 'information': 0.008620689655172414, 'what': 0.008620689655172414, 'how': 0.008620689655172414, 'show': 0.017241379310344827,
 'to': 0.034482758620689655, 'lets': 0.017241379310344827, 'take': 0.017241379310344827, 'goes': 0.008620689655172414, 'tell': 0.0086206896551
72414, 'location': 0.008620689655172414, 'do': 0.008620689655172414, 'get': 0.008620689655172414, 'who': 0.008620689655172414, 'directions': 0
.008620689655172414, 'me': 0.02586206896551724, 'on': 0.008620689655172414, 'about': 0.008620689655172414, 'i': 0.008620689655172414, 'person'
: 0.034482758620689655, 'where': 0.017241379310344827}, 'escort_location': {'office': 0.008849557522123894, 'give': 0.008849557522123894, 'is'
: 0.017699115044247787, 'at': 0.008849557522123894, 'go': 0.017699115044247787, 'happens': 0.008849557522123894, 'information': 0.00884955752
123894, 'what': 0.008849557522123894, 'how': 0.008849557522123894, 'show': 0.017699115044247787, 'to': 0.035398230088495575, 'lets': 0.0176991
```



/home/jpiroshky/Documents/prog/scratch/nb/samples
escort_location
escort_person_office
give_direction_person_office
give_directions_location
give_info_location
give_info_person
take me to location
show me to where location is
lets go to location
take me to person office
show me to where person office is
lets go to person office
tell me where is person office
give me directions to person office
how do i get to person office
tell me where is location
give me directions to location
how do i get to location
tell me about location
what is location
give me information about location
what happens at location
what goes on at location
tell me about person
give me information about person
who is person
>>>
>>> get_category('jeeves, take me to the office')
[('escort_person_office', 8.836657566087604e-08)]
>>>
>>> get_category('jeeves, take me to the office', withprobs=True, bestofn=5)
[('escort_person_office', 8.836657566087604e-08), ('give_direction_person_office', 3.094
7332509116405e-08), ('escort_location', 2.453274910717504e-08), ('give_directions_locati
on', 8.576298683895488e-09), ('give_info_person', 3.293654870725010 2e-09)]
>>>
>>> get_category('jeeves, tell me about Dr.Perkowski', withprobs=True, bestofn=5)
[('give_info_person', 2.193574143902853 2e-06), ('give_info_location', 1.736111111111127
e-06), ('give_directions_location', 6.575162324319874e-07), ('give_direction_person_offi
ce', 6.086308726792904e-07), ('escort_location', 3.465250811388481e-07)]
>>>
>>> get_category('jeeves, tell me where is Dr.Perkowskis office', withprobs=True, bestof
[('give_direction_person_office', 6.993747459687346e-10), ('escort_person_office', 3.808
904123313626e-10), ('give_directions_location', 1.988706941193152 8e-10), ('give_info_per
son', 1.186902656117118e-10), ('escort_location', 1.085519871998897 3e-10)]
>>>

NB.py

```
# nb.py - naive bayes classifier in form P(category|words) = P(words|category) * P(category)
# with Laplace smoothing - P(word|category) = wordfrequency + a / #classvocabulary + (a * #allvocab),
# where a is a smoothing factor (=1 for Laplace smoothing)
import re
import math
import os
import string
from operator import itemgetter
from itertools import islice

def sanitize(text):
    table = string.maketrans("","")
    text = text.translate(table, string.punctuation)
    text = text.lower()
    return text

# function to create a dictionary of word counts from a text
def make_word_count(text):
    word_count = {}
    for word in re.split("\W+", sanitize(text)):
#       if len(word) <= 3:
#           continue
        word_count[word] = word_count.get(word, 0.0) + 1.0
    return word_count

```

NB.py

```
vocab = {}
p_word_given_category = {}
p_category = {}

# get categories from directories in ./samples/
def make_probabilities(directory = './samples/'):
    category_word_counts = {}
    for category in os.listdir(directory):
        category_word_counts[category] = {}
        p_word_given_category[category] = {}
        p_category[category] = 0.0
        # make word count of documents in ./samples/*category_directory*
        for doc in os.listdir(directory + category):
            counts = make_word_count(open(directory + category + '/' + doc).read())
            p_category[category] += 1
            # enter words and counts into the vocab and category_word_counts
            for word, count in counts.items():
                if word not in vocab:
                    vocab[word] = 0.0
                if word not in category_word_counts[category]:
                    category_word_counts[category][word] = 0.0
                vocab[word] += count
                category_word_counts[category][word] += count
    #calculate P(word|category) for each word in vocab into each category
    for word, count in vocab.iteritems():
        for category in category_word_counts:
            p_word_given_category[category][word] = (category_word_counts[category].get(word, 0.0) + 1) / (sum(category_word_counts[
    # finish calculating p_category
    numsdoc = sum(p_category.values())
    for category in p_category:
        p_category[category] = p_category[category] / numsdoc
```

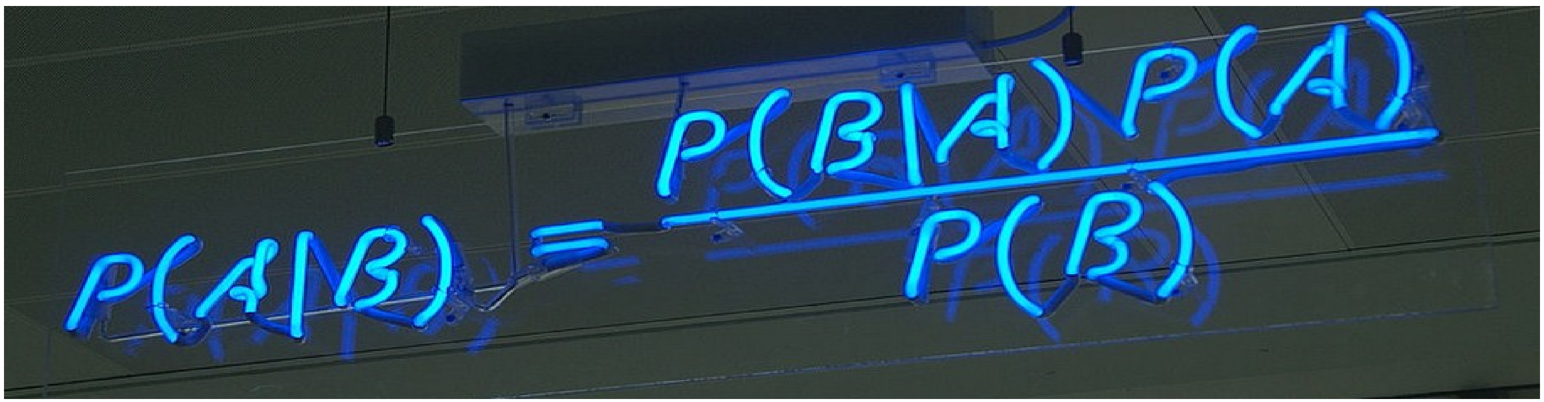
```
(sum(category_word_counts[category].values()) + sum(vocab.values()))
```

NB.py

```python

# return 'maximum likelihood' category given text
def get_category(s, withprobs=False, bestofn=1):
    # initialize dictionaries
    p_category_given_words = {}
    for category in p_word_given_category:
        p_category_given_words[category] = 0.0

    # for every word in our test sentence, make P(word|category) for each category,
    # compound them and then add P(category) to give P(category|words in test sentence)
    for word, count in make_word_count(s).items():
        # skip word if not in vocab
        if not word in vocab:
            continue
        for category in p_word_given_category:
            # add (log) probabilities to our running count, giving probability given *category*
            if p_word_given_category[category].get(word, 0.0) > 0.0:
                # note: x * log(z) == log(z^x), i.e. a word repeated in the sample counts as several features
                p_category_given_words[category] += count * math.log(p_word_given_category[category][word])
    # add p_category and then convert probabilities from log
    for category in p_category_given_words:
        p_category_given_words[category] = math.exp(p_category_given_words[category]+math.log(p_category[category]))
    #return results
    if withprobs == False:
        return list(islice(sorted(p_category_given_words, key=itemgetter(1), reverse=True), bestofn))
    else:
        return list(islice(sorted(p_category_given_words.items(), key=itemgetter(1), reverse=True), bestofn))

```

What is Bayes Theorem and Conditional Probability

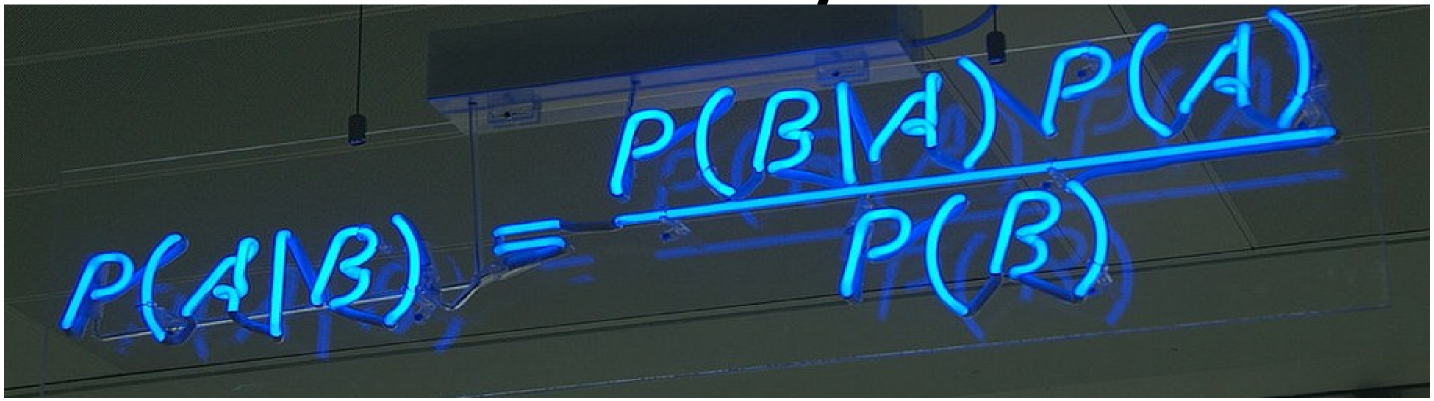
Bayes Theorem:



Conditional Probability:

- P(A|B): read "probability of A given B", i.e. what is the probability of A given we have condition B
- Example: someone is much more likely to cough if they have a cold, so the conditional probability P(cough|cold) is much higher than just P(cough)

Illustration of Bayes' Theorem



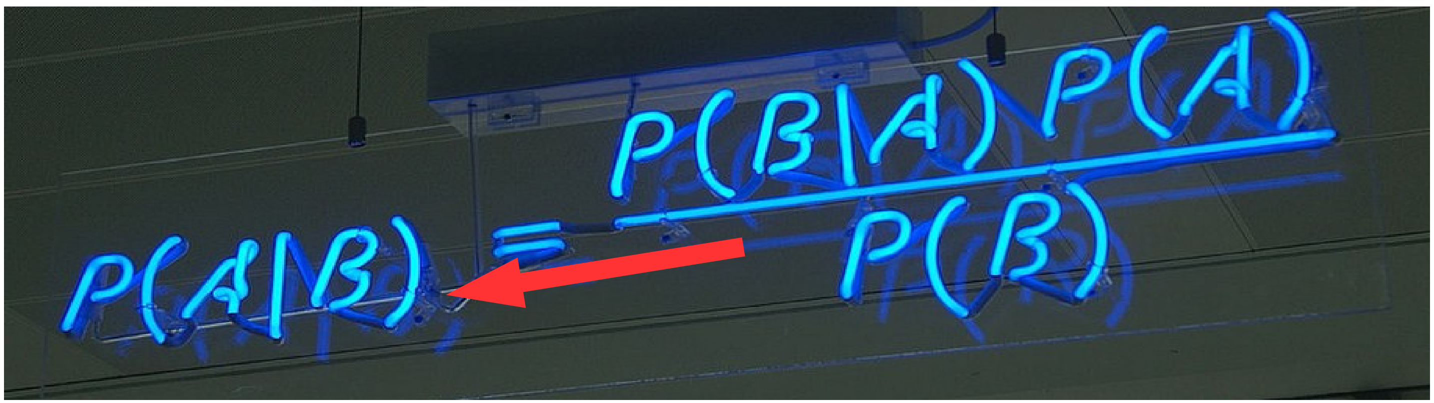
What is the probability that a student will be absent next week, given they are supposed to present next week?

- Let P(A) = probability student will be absent (.1)
- Let P(B) = probability student will be presenting (.2)
- So P(A|B) is probability of absent|'supposed to present'
- And since it is known (by experience) that a student is almost always supposed to present on the day they are absent, we can say P(B|A) is about (.9) ← probability of 'supposed to present'|absent

Illustration of Bayes' Theorem

presenting|absent (.9)

absent (.1)



(.45)
absent|presenting !

(.2)
presenting

What does this mean?

- Not surprising: If you present twice as often as you are absent, then missing every other presentation day is reasonable, right?
- Dr. Perkowski really is right, because missing almost every other day to present means a 90% chance that he will ask you to present given that you are going to be absent next week! ^_^

Why Text Classification?

Dan Jurafsky

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

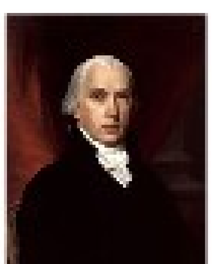
Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

Why Text Classification?

Dan Jurafsky

Who wrote which Federalist papers?

- 1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.
- Authorship of 12 of the letters in dispute
- 1963: solved by Mosteller and Wallace using Bayesian methods



James Madison

Alexander Hamilton

Why Text Classification?

Dan Jurafsky

Male or female author?

1. By 1925 present-day Vietnam was divided into three parts under French colonial rule. The southern region embracing Saigon and the Mekong delta was the colony of Cochin-China; the central area with its imperial capital at Hue was the protectorate of Annam…
2. Clara never failed to be astonished by the extraordinary felicity of her own name. She found it hard to trust herself to the mercy of fate, which had managed over the years to convert her greatest shame into one of her greatest assets…

S. Argamon, M. Koppel, J. Fine, A. R. Shimoni, 2003. "Gender, Genre, and Writing Style in Formal Written Texts," Text, volume 23, number 3, pp. 321–346

Why Text Classification?

Dan Jurafsky

Positive or negative movie review?

- unbelievably disappointing
- Full of zany characters and richly applied satire, and some great plot twists
- this is the greatest screwball comedy ever filmed
- It was pathetic. The worst part about it was the boxing scenes.

Why Text Classification?

Dan Jurafsky

What is the subject of this article?

MEDLINE Article

?

MeSH Subject Category Hierarchy

- Antogonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- ...

Why Text Classification?

Could use any kind of classifier:

Dan Jurafsky

Classification Methods: Supervised Machine Learning

- Any kind of classifier
  - Naïve Bayes
  - Logistic regression
  - Support-vector machines
  - k-Nearest Neighbors
  - ...

CMU Sphinx

What to know about Sphinx:

- Probabilistic speech recognizer
- Uses 3 basic concepts
  1) Features – extracts 'phones' from audio sample as features making up 'subwords' → 'subword' features (syllables) form words → words are features in a grammar
  2) Model – a combination of a function and a learned values for the function, and the function returns the most probable next feature given the current feature
  3) Hidden Markov Model Assumption – assumes that speech behaves like a sequence of states where several next states are possible, and their probabilities (scores) depend on the current state

CMU Sphinx

Sphinx uses models at 3 different levels:

1) Acoustic Model – models the likelihood of the next senone (phone)
2) Phonetic Dictionary – maps words to phones; usually contains a few different 'pronunciations' for each word
3) Language Model – models the likely next words given the current word state: **necessary to restrict search space – better restriction, better accuracy**

CMU Sphinx

Uses a lattice (nodes + links) to model match possibilities

Other concepts used

A **Lattice** is a directed graph that represents variants of the recognition. Often, getting the best match is not practical; in that case, lattices are good intermediate formats to represent the recognition result.

N-best lists of variants are like lattices, though their representations are not as dense as the lattice ones.

Word confusion networks (sausages) are lattices where the strict order of nodes is taken from lattice edges.

Speech database - a set of typical recordings from the task database. If we develop dialog system it might be dialogs recorded from users. For dictation system it might be reading recordings. Speech databases are used to train, tune and test the decoding systems.

Text databases - sample texts collected for language model training and so on. Usually, databases of texts are collected in sample text form. The issue with collection is to put present documents (PDFs, web pages, scans) into spoken text form. That is, you need to remove tags and headings, to expand numbers to their spoken form, and to expand abbreviations.

CMU Sphinx

How to evaluate speech recognition:

When speech recognition is being developed, the most complex issue is to make search precise (consider as many variants to match as possible) and to make it fast enough to not run for ages. There are also issues with making the model match the speech since models aren't perfect.

Usually the system is tested on a test database that is meant to represent the target task correctly.

The following characteristics are used:

Word error rate. Let we have original text and recognition text of length of N words. From them the I words were inserted D words were deleted and S words were substituted Word error rate is

WER = (I + D + S) / N

WER is usually measured in percent.

Accuracy. It is almost the same thing as word error rate, but it doesn't count insertions.

Accuracy = (N - D - S) / N

Accuracy is actually a worse measure for most tasks, since insertions are also important in final results. But for some tasks, accuracy is a reasonable measure of the decoder performance.

Speed. Suppose the audio file was 2 hours and the decoding took 6 hours. Then speed is counted as 3xRT.

ROC curves. When we talk about detection tasks, there are false alarms and hits/misses; ROC curves are used. A curve is a graphic that describes the number of false alarms vs number of hits, and tries to find optimal point where the number of false alarms is small and number of hits matches 100%.

There are other properties that aren't often taken into account, but still important for many practical applications. Your first task should be to build such a measure and systematically apply it during the system development. Your second task is to collect the test database and test how does your application perform.

CMU Sphinx

```
READY....
Listening...
Stopped listening, please wait...
INFO: cmn_prior.c(121): cmn_prior_update: from < 25.37 -1.15  2.16 -0.59 -0.92 -
0.98 -0.36  0.36  0.16 -0.08 -0.05 -0.46  0.04 >
INFO: cmn_prior.c(139): cmn_prior_update: to   < 25.53 -1.18  2.09 -0.62 -1.05 -
0.97 -0.29  0.41  0.15 -0.01 -0.11 -0.49  0.08 >
INFO: ngram_search_fwdtree.c(1549):      5069 words recognized (27/fr)
INFO: ngram_search_fwdtree.c(1551):    618559 senones evaluated (3326/fr)
INFO: ngram_search_fwdtree.c(1553):    941302 channels searched (5060/fr), 83109 
1st, 174110 last
INFO: ngram_search_fwdtree.c(1557):     11055 words for which last channels evalu
ated (59/fr)
INFO: ngram_search_fwdtree.c(1560):     86237 candidate words for entering last p
hone (463/fr)
INFO: ngram_search_fwdtree.c(1562): fwdtree 0.76 CPU 0.406 xRT
INFO: ngram_search_fwdtree.c(1565): fwdtree 1.90 wall 1.021 xRT
INFO: ngram_search_fwdflat.c(302): Utterance vocabulary contains 226 words
INFO: ngram_search_fwdflat.c(937):      2840 words recognized (15/fr)
INFO: ngram_search_fwdflat.c(939):    225672 senones evaluated (1213/fr)
INFO: ngram_search_fwdflat.c(941):    347676 channels searched (1869/fr)
INFO: ngram_search_fwdflat.c(943):     18246 words searched (98/fr)
INFO: ngram_search_fwdflat.c(945):     14018 word transitions (75/fr)
INFO: ngram_search_fwdflat.c(948): fwdflat 0.16 CPU 0.084 xRT
INFO: ngram_search_fwdflat.c(951): fwdflat 0.17 wall 0.090 xRT
INFO: ngram_search.c(1266): lattice start node <s>.0 end node </s>.174
INFO: ngram_search.c(1294): Eliminated 0 nodes before end node
INFO: ngram_search.c(1399): Lattice has 447 nodes, 1651 links
INFO: ps_lattice.c(1365): Normalizer P(O) = alpha(</s>:174:184) = -1254741
INFO: ps_lattice.c(1403): Joint P(O,S) = -1366674 P(S|O) = -111933
INFO: ngram_search.c(888): bestpath 0.01 CPU 0.006 xRT
INFO: ngram_search.c(891): bestpath 0.02 wall 0.009 xRT
000000013: what did you the for lunch today
```

Running pocketsphinx

Run from command line

```
jpiroshky@Omoikanefuji:~$ pocketsphinx_continuous
INFO: cmd_ln.c(691): Parsing command line:
pocketsphinx_continuous

Current configuration:
[NAME]          [DEFLT]         [VALUE]
-adcdev
-agc            none            none
-agcthresh      2.0             2.000000e+00
```

Running pocketsphinx

Default acoustic model: hub4wsj_sc_8k

```
INFO: acmod.c(246): Parsed model-specific feature parameters from /usr/local/sha
re/pocketsphinx/model/hmm/en_US/hub4wsj_sc_8k/feat.params
INFO: feat.c(713): Initializing feature stream to type: '1s_c_d_dd', ceplen=13,
CMN='current', VARNORM='no', AGC='none'
INFO: cmn.c(142): mean[0]= 12.00, mean[1..12]= 0.0
INFO: acmod.c(167): Using subvector specification 0-12/13-25/26-38
INFO: mdef.c(517): Reading model definition: /usr/local/share/pocketsphinx/model
/hmm/en_US/hub4wsj_sc_8k/mdef
INFO: mdef.c(528): Found byte-order mark BMDF, assuming this is a binary mdef fi
le
INFO: bin_mdef.c(336): Reading binary model definition: /usr/local/share/pockets
phinx/model/hmm/en_US/hub4wsj_sc_8k/mdef
INFO: bin_mdef.c(513): 50 CI-phone, 143047 CD-phone, 3 emitstate/phone, 150 CI-s
en, 5150 Sen, 27135 Sen-Seq
INFO: tmat.c(205): Reading HMM transition probability matrices: /usr/local/share
/pocketsphinx/model/hmm/en_US/hub4wsj_sc_8k/transition_matrices
INFO: acmod.c(121): Attempting to use SCHMM computation module
INFO: ms_gauden.c(198): Reading mixture gaussian parameter: /usr/local/share/poc
ketsphinx/model/hmm/en_US/hub4wsj_sc_8k/means
INFO: ms_gauden.c(292): 1 codebook, 3 feature, size:
INFO: ms_gauden.c(294):  256x13
INFO: ms_gauden.c(294):  256x13
INFO: ms_gauden.c(294):  256x13
```

Running pocketsphinx

Default dictionary: cmu07a.dic <-133,000+ ent.

```
INFO: dict.c(317): Allocating 137543 * 20 bytes (2686 KiB) for word entries
INFO: dict.c(332): Reading main dictionary: /usr/local/share/pocketsphinx/model/
lm/en_US/cmu07a.dic
INFO: dict.c(211): Allocated 1010 KiB for strings, 1664 KiB for phones
INFO: dict.c(335): 133436 words read
INFO: dict.c(341): Reading filler dictionary: /usr/local/share/pocketsphinx/mode
l/hmm/en_US/hub4wsj_sc_8k/noisedict
INFO: dict.c(211): Allocated 0 KiB for strings, 0 KiB for phones
INFO: dict.c(344): 11 words read
INFO: dict2pid.c(396): Building PID tables for dictionary
INFO: dict2pid.c(404): Allocating 50^3 * 2 bytes (244 KiB) for word-initial trip
hones
INFO: dict2pid.c(131): Allocated 30200 bytes (29 KiB) for word-final triphones
INFO: dict2pid.c(195): Allocated 30200 bytes (29 KiB) for single-phone word trip
hones
INFO: ngram_model_arpa.c(77): No \data\ mark in LM file
INFO: ngram_model_dmp.c(142): Will use memory-mapped I/O for LM file
INFO: ngram_model_dmp.c(196): ngrams 1=5001, 2=436879, 3=418286
INFO: ngram_model_dmp.c(242):     5001 = LM.unigrams(+trailer) read
INFO: ngram_model_dmp.c(288):   436879 = LM.bigrams(+trailer) read
INFO: ngram_model_dmp.c(314):   418286 = LM.trigrams read
INFO: ngram_model_dmp.c(339):    37293 = LM.prob2 entries read
INFO: ngram_model_dmp.c(359):    14370 = LM.bo_wt2 entries read
INFO: ngram_model_dmp.c(379):    36094 = LM.prob3 entries read
INFO: ngram_model_dmp.c(407):      854 = LM.tseg_base entries read
INFO: ngram_model_dmp.c(463):     5001 = ascii word strings read
INFO: ngram_search_fwdtree.c(99): 788 unique initial diphones
INFO: ngram_search_fwdtree.c(147): 0 root, 0 non-root channels, 60 single-phone
words
```

Running pocketsphinx

Default language model: NONE!

```
INFO: dict2pid.c(404): Allocating 50^3 * 2 bytes (244 KiB) for word-initial trip
hones
INFO: dict2pid.c(131): Allocated 30200 bytes (29 KiB) for word-final triphones
INFO: dict2pid.c(195): Allocated 30200 bytes (29 KiB) for single-phone word trip
hones
INFO: ngram_model_arpa.c(77): No \data\ mark in LM file
INFO: ngram_model_dmp.c(142): Will use memory-mapped I/O for LM file
INFO: ngram_model_dmp.c(196): ngrams 1=5001, 2=436879, 3=418286
INFO: ngram_model_dmp.c(242):     5001 = LM.unigrams(+trailer) read
INFO: ngram_model_dmp.c(288):   436879 = LM.bigrams(+trailer) read
INFO: ngram_model_dmp.c(314):   418286 = LM.trigrams read
INFO: ngram_model_dmp.c(339):    37293 = LM.prob2 entries read
INFO: ngram_model_dmp.c(359):    14370 = LM.bo_wt2 entries read
INFO: ngram_model_dmp.c(379):    36094 = LM.prob3 entries read
INFO: ngram_model_dmp.c(407):      854 = LM.tseg_base entries read
INFO: ngram_model_dmp.c(463):     5001 = ascii word strings read
INFO: ngram_search_fwdtree.c(99): 788 unique initial diphones
INFO: ngram_search_fwdtree.c(147): 0 root, 0 non-root channels, 60 single-phone
words
INFO: ngram_search_fwdtree.c(186): Creating search tree
INFO: ngram_search_fwdtree.c(191): before: 0 root, 0 non-root channels, 60 singl
e-phone words
INFO: ngram_search_fwdtree.c(326): after: max nonroot chan increased to 13428
INFO: ngram_search_fwdtree.c(338): after: 457 root, 13300 non-root channels, 26
single-phone words
INFO: ngram_search_fwdflat.c(156): fwdflat: min_ef_width = 4, max_sf_win = 25
INFO: continuous.c(371): pocketsphinx_continuous COMPILED ON: Jan  9 2015, AT: 0
1:00:40

READY....
```

Dictionary + Language Model example: turtle.dic + turtle.lm

<turtle.dic>

File Edit Search Options Help

```
a               AH
a(2)            EY
and             AE N T
and(2)          AH N T
are             AA R
are(2)          ER
around          ER AW N
around(2)       ER AW N T
backward        B AE K W ER T
backwards       B AE K W ER D Z
bye             B AY
centimeter      S EH N T AH M IY T ER
centimeters     S EH N T AH M IY T ER Z
chase           CH EY S
color           K AH L ER
color(2)        K AO L ER
degrees         D IH G R IY Z
display         D IH S P L EY
do              D UW
doing           D UW IH NG
eight           EY T
eighteen        EY T IY N
eighty          EY T IY
eleven          AH L EH V AH N
eleven(2)       IY L EH V AH N
exit            EH G Z AH T
exit(2)         EH K S AH T
explore         IH K S P L AO R
fifteen         F IH F T IY N
fifty           F IH F T IY
find            F AY N T
finish          F IH N IH SH
five            F AY V
forty           F AO R T IY
```

```
Language model created by QuickLM on Wed Aug  4 14:21:56 EDT 1999
Carnegie Mellon University (c) 1996

This model based on a corpus of  sentences and  words
The (fixed) discount mass is 0.5

\data\
ngram 1=91
ngram 2=212
ngram 3=177

\1-grams:
-0.9129 </s> -0.3010
-0.9129 <s> -0.2144
-2.6031 A -0.2999
-2.6031 AND -0.2989
-2.3021 ARE -0.2978
-2.6031 AROUND -0.2444
-2.3021 BACKWARD -0.2338
-2.9042 BACKWARDS -0.2444
-2.6031 BYE -0.2432
-2.9042 CENTIMETER -0.2444
-2.9042 CENTIMETERS -0.2444
-2.9042 CHASE -0.2989
...
\2-grams:
-2.2923 <s> AND -0.1761
-1.9912 <s> ARE -0.0969
-2.2923 <s> CENTIMETER 0.0000
-2.2923 <s> CENTIMETERS 0.0000
-2.2923 <s> CHASE 0.0000
-1.9912 <s> DISPLAY 0.0000
-2.2923 <s> DO 0.0000
-2.2923 <s> EIGHT 0.0000
-2.2923 <s> EIGHTEEN 0.0000
-2.2923 <s> EIGHTY -0.2218
-2.2923 <s> ELEVEN 0.0000
...
-0.3010 WANDER AROUND 0.0000
-0.3010 WHAT ARE -0.0969
-0.3010 WINDOW </s> -0.3010
-1.0000 YOU ARE -0.2430
-1.0000 YOU DOING 0.0000
-1.0000 YOU LISTENING 0.0000
-1.0000 YOU READY 0.0000
-1.0000 YOU UNDERSTAND 0.0000
...
\3-grams:
-0.3010 <s> AND THEN
-0.3010 <s> ARE YOU
-0.3010 <s> CENTIMETER </s>
-0.3010 <s> CENTIMETERS </s>
-0.3010 <s> CHASE THE
-0.3010 <s> DISPLAY A
...
```

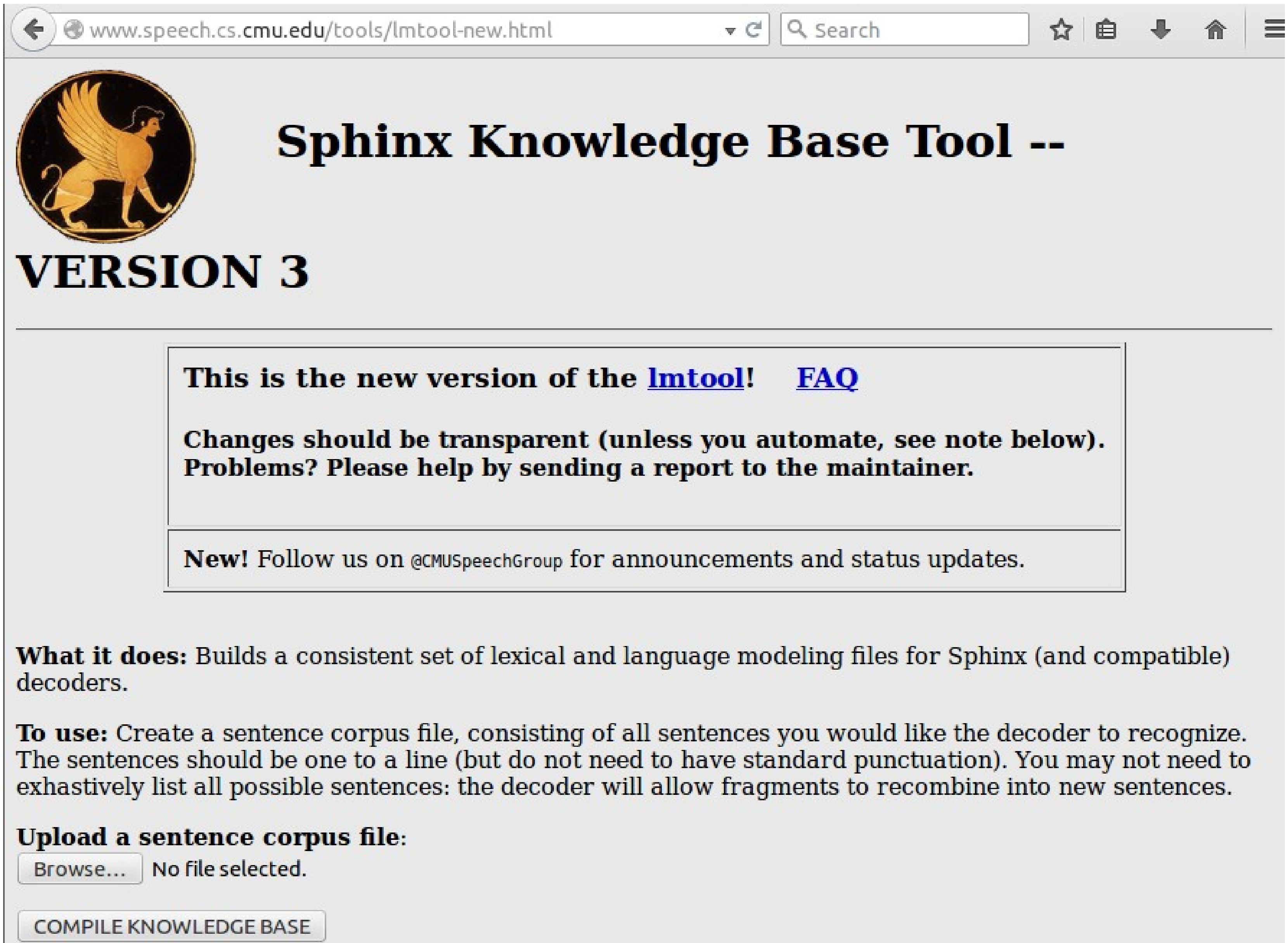
Jeevesphrase with cmu07a.dic and no language model

sphinxoutput.txt

File Edit Search Options Help

```
READY....
Listening...
Stopped listening, please wait...
000000000: it at all
READY....
Listening...
Stopped listening, please wait...
000000001: it's me to the office
READY....
Listening...
Stopped listening, please wait...
000000002: take me to the office
READY....
Listening...
Stopped listening, please wait...
000000003: take me to the office
READY....
Listening...
Stopped listening, please wait...
000000004: tell me about person
READY....
Listening...
Stopped listening, please wait...
000000005: tell me about person
READY....
Listening...
Stopped listening, please wait...
000000006: tell me about person
READY....
Listening...
Stopped listening, please wait...
000000007: tony that person
READY....
```

Creating a Dictionary + Language Model



Creating a Dictionary + Language Model

www.speech.cs.cmu.edu/tools/product/1423854870_31864/

Search

Sphinx knowledge base generator [lmtool.3a]

Your Sphinx knowledge base compilation has been successfully processed!

The base name for this set is **8589**. TAR8589.tgz is the compressed version.
Note that this set of files is internally consistent and is best used together.

IMPORTANT: Please download these files as soon as possible; they will be deleted in approximately a half hour.

```
SESSION 1423854870_31864
[_INFO_] Found corpus: 20 sentences, 99 unique words
[_INFO_] Found 0 words in extras  (0)
[_INFO_] Language model completed  (0)
[_INFO_] Pronounce completed  (0)
[_STAT_] Elapsed time: 0.029 sec
```

Please include these messages in bug reports.

Name	Size	Description
8589.dic	552	*Pronunciation Dictionary*
8589.lm	3.8K	*Language Model*
8589.log_pronounce	345	*Log File*
8589.sent	705	*Corpus (processed)*
8589.vocab	136	*Word List*
TAR8589.tgz	2.0K	**COMPRESSED TARBALL**

Apache/2.2.22 (Ubuntu) Server at www.speech.cs.cmu.edu Port 80

Creating a Dictionary + Language Model

jeevesphrase.txt

File Edit Search Options Help

```
TAKE ME TO LOCATION
SHOW ME TO WHERE LOCATION IS
LETS GO TO LOCATION
TAKE ME TO PERSON OFFICE
SHOW ME TO WHERE PERSON OFFICE IS
LETS GO TO PERSON OFFICE
TELL ME WHERE IS PERSON OFFICE
GIVE ME DIRECTIONS TO PERSON OFFICE
HOW DO I GET TO PERSON OFFICE
TELL ME WHERE IS LOCATION
GIVE ME DIRECTIONS TO LOCATION
HOW DO I GET TO LOCATION
TELL ME ABOUT LOCATION
WHAT IS LOCATION
GIVE ME INFORMATION ABOUT LOCATION
WHAT HAPPENS AT LOCATION
WHAT GOES ON AT LOCATION
TELL ME ABOUT PERSON
GIVE ME INFORMATION ABOUT PERSON
WHO IS PERSON
```

8589.dic

File Edit Search Options Help

```
ABOUT	AH B AW T
AT	AE T
DIRECTIONS	D ER EH K SH AH N Z
DIRECTIONS(2)	D IY R EH K SH IH N Z
DIRECTIONS(3)	D AY R EH K SH IH N Z
DIRECTIONS(4)	D IH R EH K SH IH N Z
DO	D UW
GET	G EH T
GET(2)	G IH T
GIVE	G IH V
GO	G OW
GOES	G OW Z
HAPPENS	HH AE P AH N Z
HOW	HH AW
I	AY
INFORMATION	IH N F AO R M EY SH AH N
IS	IH Z
LETS	L EH T S
LOCATION	L OW K EY SH AH N
ME	M IY
OFFICE	AO F AH S
ON	AA N
ON(2)	AO N
PERSON	P ER S AH N
SHOW	SH OW
TAKE	T EY K
TELL	T EH L
TO	T UW
TO(2)	T IH
TO(3)	T AH
WHAT	W AH T
WHAT(2)	HH W AH T
WHERE	W EH R
WHERE(2)	HH W EH R
WHO	HH UW
```

8589.lm

File Edit Search Options Help

```
-0.6990 TO PERSON -0.1249
-1.0000 TO WHERE -0.1761
-0.7782 WHAT GOES 0.0000
-0.7782 WHAT HAPPENS 0.0000
-0.7782 WHAT IS -0.2218
-0.6021 WHERE IS -0.1249
-0.9031 WHERE LOCATION -0.2808
-0.9031 WHERE PERSON -0.1249
-0.3010 WHO IS -0.2218

\3-grams:
-0.3010 <s> GIVE ME
-0.3010 <s> HOW DO
-0.3010 <s> LETS GO
-0.3010 <s> SHOW ME
-0.3010 <s> TAKE ME
-0.3010 <s> TELL ME
-0.7782 <s> WHAT GOES
-0.7782 <s> WHAT HAPPENS
-0.7782 <s> WHAT IS
-0.3010 <s> WHO IS
-0.3010 ABOUT LOCATION </s>
-0.3010 ABOUT PERSON </s>
-0.3010 AT LOCATION </s>
-0.6021 DIRECTIONS TO LOCATION
-0.6021 DIRECTIONS TO PERSON
-0.3010 DO I GET
-0.6021 GET TO LOCATION
-0.6021 GET TO PERSON
-0.6021 GIVE ME DIRECTIONS
-0.6021 GIVE ME INFORMATION
-0.6021 GO TO LOCATION
-0.6021 GO TO PERSON
-0.3010 GOES ON AT
-0.3010 HAPPENS AT LOCATION
-0.3010 HOW DO I
-0.3010 I GET TO
-0.6021 INFORMATION ABOUT LOCATION
-0.6021 INFORMATION ABOUT PERSON
-0.3010 IS LOCATION </s>
-0.6021 IS PERSON </s>
-0.6021 IS PERSON OFFICE
```

jeevesprhase with custom dictionary + language model

```
jpiroshky@Omoikanefuji:~/jeevesphrase$ pocketsphinx_continuous -dict 8589.di
c -lm 8589.lm
```

```
triphones
INFO: dict2pid.c(131): Allocated 30200 bytes (29 KiB) for word-final triphon
es
INFO: dict2pid.c(195): Allocated 30200 bytes (29 KiB) for single-phone word
triphones
INFO: ngram_model_arpa.c(477): ngrams 1=28, 2=50, 3=61
INFO: ngram_model_arpa.c(135): Reading unigrams
INFO: ngram_model_arpa.c(516):       28 = #unigrams created
INFO: ngram_model_arpa.c(195): Reading bigrams
INFO: ngram_model_arpa.c(533):       50 = #bigrams created
INFO: ngram_model_arpa.c(534):       15 = #prob2 entries
INFO: ngram_model_arpa.c(542):        9 = #bo_wt2 entries
INFO: ngram_model_arpa.c(292): Reading trigrams
INFO: ngram_model_arpa.c(555):       61 = #trigrams created
INFO: ngram_model_arpa.c(556):        7 = #prob3 entries
INFO: ngram_search_fwdtree.c(99): 31 unique initial diphones
INFO: ngram_search_fwdtree.c(147): 0 root, 0 non-root channels, 13 single-ph
one words
INFO: ngram_search_fwdtree.c(186): Creating search tree
INFO: ngram_search_fwdtree.c(191): before: 0 root, 0 non-root channels, 13 s
ingle-phone words
INFO: ngram_search_fwdtree.c(326): after: max nonroot chan increased to 193
INFO: ngram_search_fwdtree.c(338): after: 31 root, 65 non-root channels, 12
single-phone words
INFO: ngram_search_fwdflat.c(156): fwdflat: min_ef_width = 4, max_sf_win = 2
5
INFO: continuous.c(371): pocketsphinx_continuous COMPILED ON: Jan  9 2015, A
T: 01:00:40

READY....
```

jeevesprhase with custom dictionary + language model

```
triphones
INFO: dict2pid.c(131): Allocated 30200 bytes (29 KiB) for word-final triphon
es
INFO: dict2pid.c(195): Allocated 30200 bytes (29 KiB) for single-phone word
triphones
INFO: ngram_model_arpa.c(477): ngrams 1=28, 2=50, 3=61
INFO: ngram_model_arpa.c(135): Reading unigrams
INFO: ngram_model_arpa.c(516):       28 = #unigrams created
INFO: ngram_model_arpa.c(195): Reading bigrams
INFO: ngram_model_arpa.c(533):       50 = #bigrams created
INFO: ngram_model_arpa.c(534):       15 = #prob2 entries
INFO: ngram_model_arpa.c(542):        9 = #bo_wt2 entries
INFO: ngram_model_arpa.c(292): Reading trigrams
INFO: ngram_model_arpa.c(555):       61 = #trigrams created
INFO: ngram_model_arpa.c(556):        7 = #prob3 entries
INFO: ngram_search_fwdtree.c(99): 31 unique initial diphones
INFO: ngram_search_fwdtree.c(147): 0 root, 0 non-root channels, 13 single-ph
one words
INFO: ngram_search_fwdtree.c(186): Creating search tree
INFO: ngram_search_fwdtree.c(191): before: 0 root, 0 non-root channels, 13 s
ingle-phone words
INFO: ngram_search_fwdtree.c(326): after: max nonroot chan increased to 193
INFO: ngram_search_fwdtree.c(338): after: 31 root, 65 non-root channels, 12
single-phone words
INFO: ngram_search_fwdflat.c(156): fwdflat: min_ef_width = 4, max_sf_win = 2
5
INFO: continuous.c(371): pocketsphinx_continuous COMPILED ON: Jan  9 2015, A
T: 01:00:40

READY....
```

Jeevesprhase with custom dictionary + language model

```
3  0.05  0.29  0.54  0.50 -0.05 -0.07 -0.66  0.02 >
INFO: cmn_prior.c(139): cmn_prior_update: to   < 21.36 -1.31  1.66 -0.46  0.0
9  0.12  0.21  0.55  0.47 -0.02 -0.05 -0.63 -0.01 >
INFO: ngram_search_fwdtree.c(1549):     1602 words recognized (11/fr)
INFO: ngram_search_fwdtree.c(1551):    56062 senones evaluated (376/fr)
INFO: ngram_search_fwdtree.c(1553):    33668 channels searched (225/fr), 4495
 1st, 24425 last
INFO: ngram_search_fwdtree.c(1557):     2582 words for which last channels ev
aluated (17/fr)
INFO: ngram_search_fwdtree.c(1560):     2293 candidate words for entering las
t phone (15/fr)
INFO: ngram_search_fwdtree.c(1562): fwdtree 0.06 CPU 0.043 xRT
INFO: ngram_search_fwdtree.c(1565): fwdtree 1.51 wall 1.011 xRT
INFO: ngram_search_fwdflat.c(302): Utterance vocabulary contains 20 words
INFO: ngram_search_fwdflat.c(937):      575 words recognized (4/fr)
INFO: ngram_search_fwdflat.c(939):    45926 senones evaluated (308/fr)
INFO: ngram_search_fwdflat.c(941):    33655 channels searched (225/fr)
INFO: ngram_search_fwdflat.c(943):     1717 words searched (11/fr)
INFO: ngram_search_fwdflat.c(945):     1173 word transitions (7/fr)
INFO: ngram_search_fwdflat.c(948): fwdflat 0.04 CPU 0.024 xRT
INFO: ngram_search_fwdflat.c(951): fwdflat 0.06 wall 0.039 xRT
INFO: ngram_search.c(1266): lattice start node <s>.0 end node </s>.118
INFO: ngram_search.c(1294): Eliminated 0 nodes before end node
INFO: ngram_search.c(1399): Lattice has 73 nodes, 57 links
INFO: ps_lattice.c(1365): Normalizer P(O) = alpha(</s>:118:147) = -912235
INFO: ps_lattice.c(1403): Joint P(O,S) = -912884 P(S|O) = -649
INFO: ngram_search.c(888): bestpath 0.00 CPU 0.000 xRT
INFO: ngram_search.c(891): bestpath 0.01 wall 0.004 xRT
000000004: TAKE ME TO OFFICE
READY....
```