Speaker recognition using keywords and key hypernyms

20130690 Sumin Han

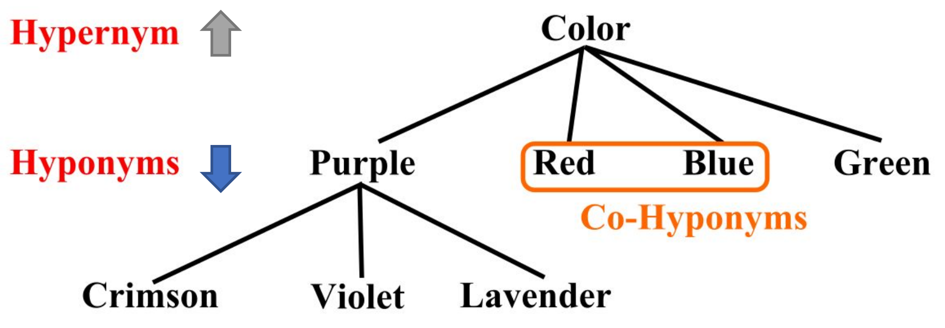
Introduction

Question

- Can we recognize the speaker in a spoken corpus using **keywords** and **hypernyms**?

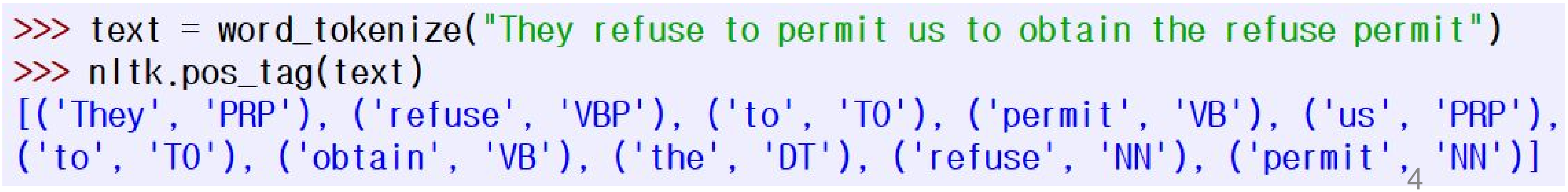
???: I love my job at the Krusty Krab.
I like Jelly Fishing and Bubble Blowing.
I've never been late for work!

KEYWORDS VS KEY HYPERNYM



- KEYWORDS
  - Comparing the word **frequency**.
  - *Noun, Pronoun, Verb, Adjective, Adverb*
- KEY HYPERNYM
  - Abstract and conceptual words, key content.
  - Not about the frequency but **the diversity of hyponyms**.
  - We have to know synonym sets (e.g. Is, Are, Was, Were ➔ Be.v)
  - *Noun (Normally Verbs don't have enough hypernyms to construct a network)*

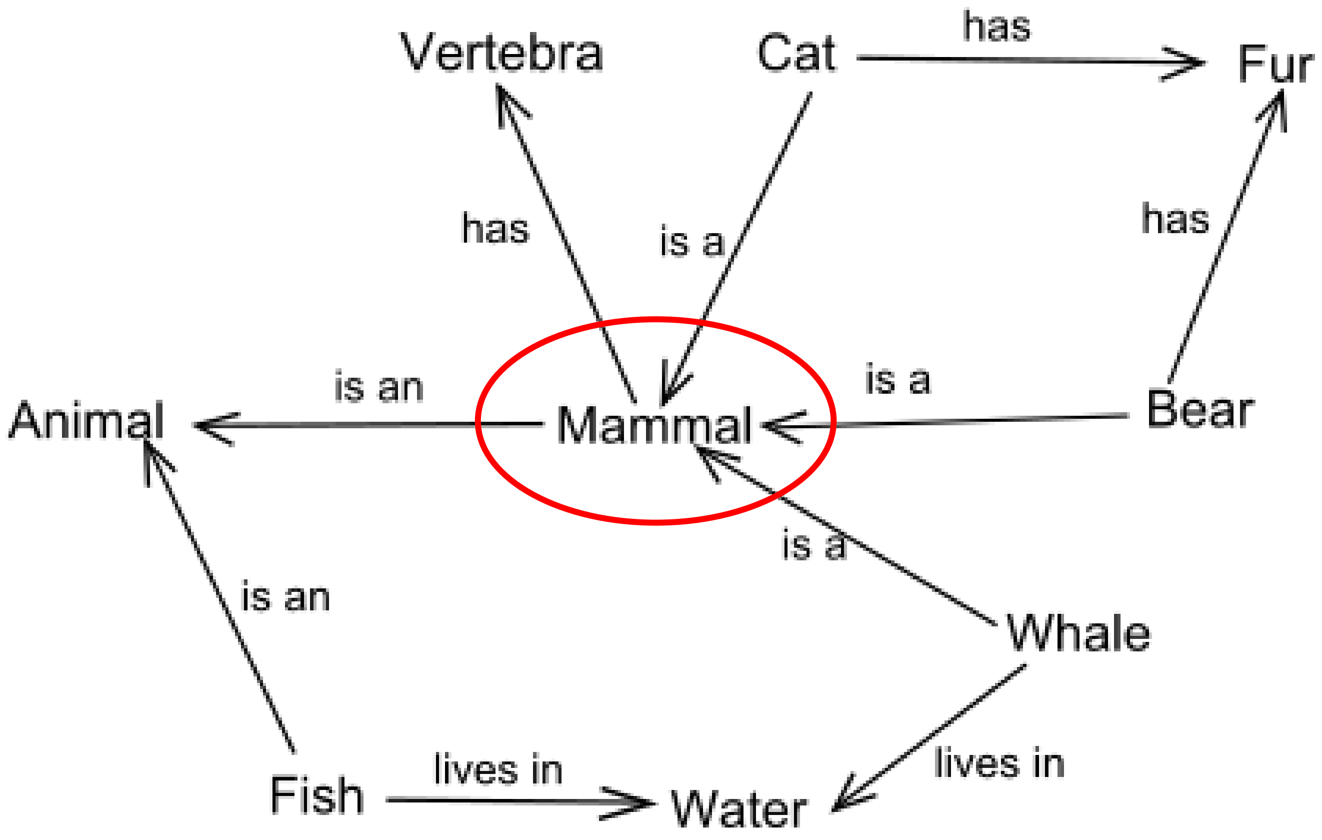
Used NLTK Tagger and TAGANT for POS Tagging



```
>>> text = word_tokenize("They refuse to permit us to obtain the refuse permit")
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'),
('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
```

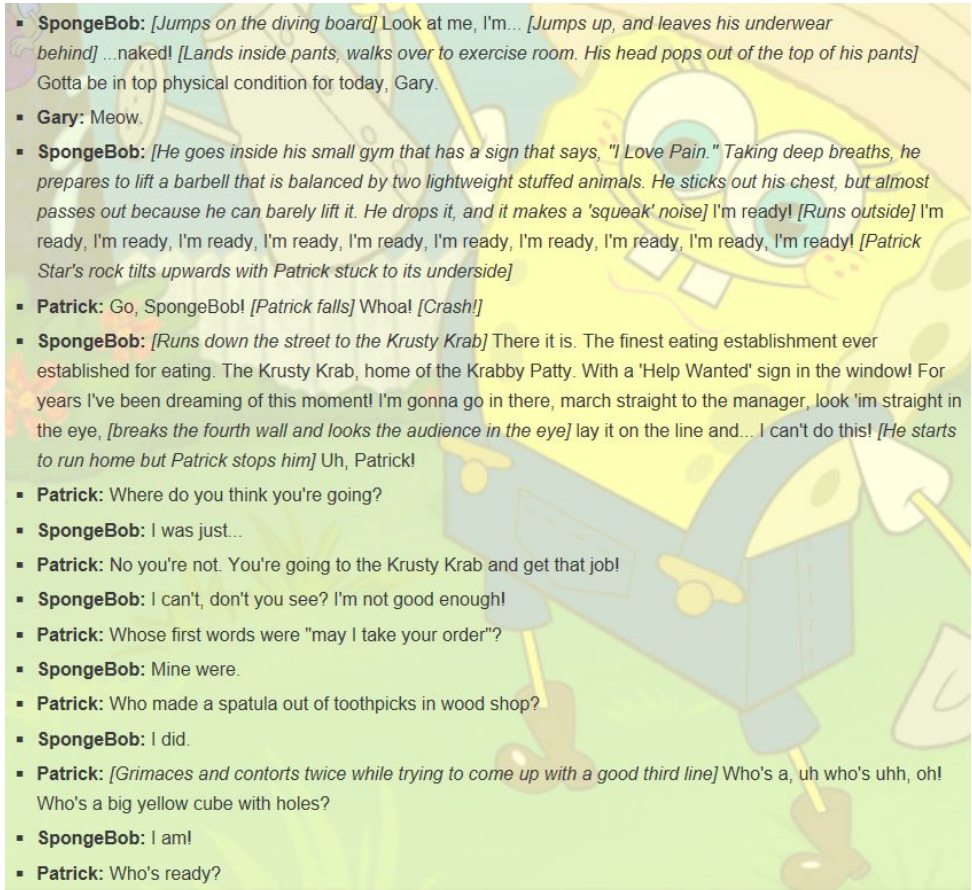
Characteristics of Key Hypernym

- Likely to have many linked hyponyms.
- Using different words in the similar category will increase the keyness.

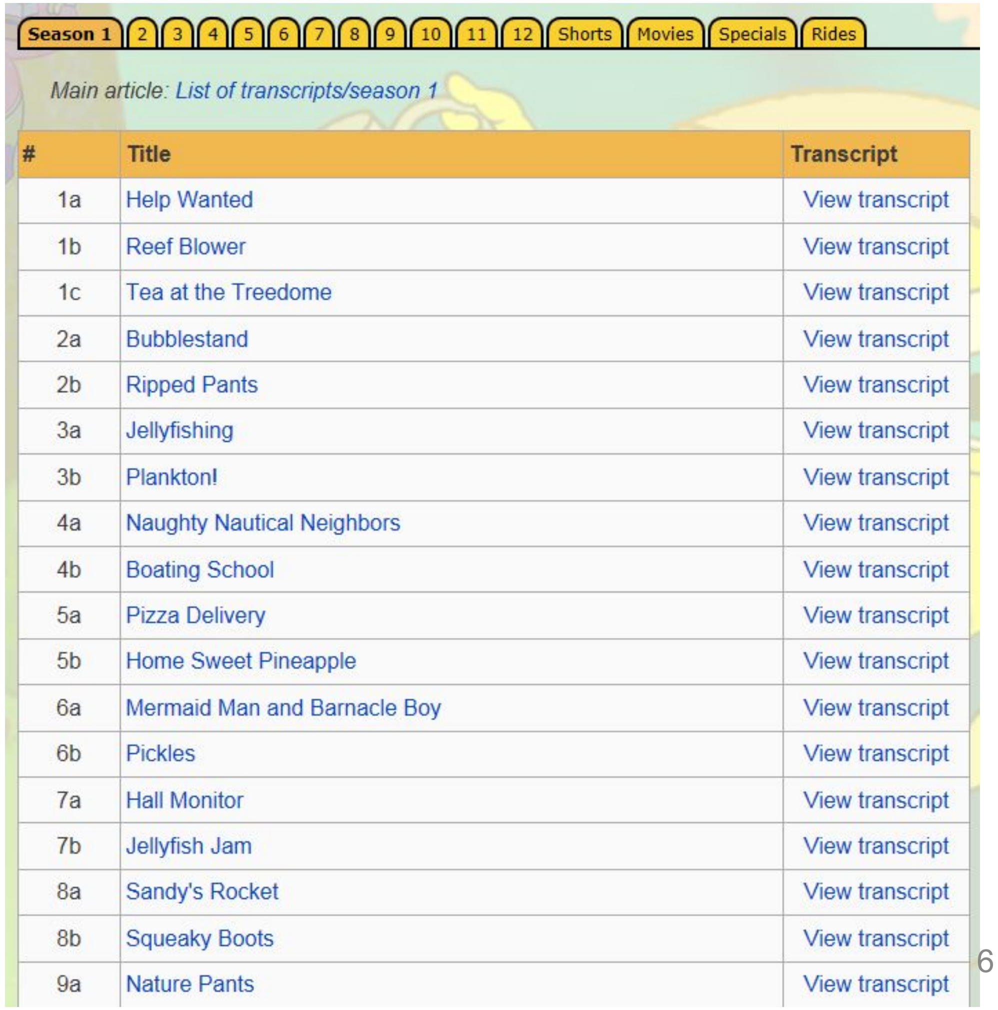


Spoken Corpus

- http://spongebob.wikia.com/wiki/List_of_transcripts
- Transcripts of “SpongeBob”



- **SpongeBob:** *[Jumps on the diving board]* Look at me, I'm... *[Jumps up, and leaves his underwear behind]* ...naked! *[Lands inside pants, walks over to exercise room. His head pops out of the top of his pants]* Gotta be in top physical condition for today, Gary.
- **Gary:** Meow.
- **SpongeBob:** *[He goes inside his small gym that has a sign that says, "I Love Pain." Taking deep breaths, he prepares to lift a barbell that is balanced by two lightweight stuffed animals. He sticks out his chest, but almost passes out because he can barely lift it. He drops it, and it makes a 'squeak' noise]* I'm ready! *[Runs outside]* I'm ready, I'm ready, I'm ready, I'm ready, I'm ready, I'm ready, I'm ready, I'm ready, I'm ready, I'm ready! *[Patrick Star's rock tilts upwards with Patrick stuck to its underside]*
- **Patrick:** Go, SpongeBob! *[Patrick falls]* Whoa! *[Crash!]*
- **SpongeBob:** *[Runs down the street to the Krusty Krab]* There it is. The finest eating establishment ever established for eating. The Krusty Krab, home of the Krabby Patty. With a 'Help Wanted' sign in the window! For years I've been dreaming of this moment! I'm gonna go in there, march straight to the manager, look 'im straight in the eye, *[breaks the fourth wall and looks the audience in the eye]* lay it on the line and... I can't do this! *[He starts to run home but Patrick stops him]* Uh, Patrick!
- **Patrick:** Where do you think you're going?
- **SpongeBob:** I was just...
- **Patrick:** No you're not. You're going to the Krusty Krab and get that job!
- **SpongeBob:** I can't, don't you see? I'm not good enough!
- **Patrick:** Whose first words were "may I take your order"?
- **SpongeBob:** Mine were.
- **Patrick:** Who made a spatula out of toothpicks in wood shop?
- **SpongeBob:** I did.
- **Patrick:** *[Grimaces and contorts twice while trying to come up with a good third line]* Who's a, uh who's uhh, oh! Who's a big yellow cube with holes?
- **SpongeBob:** I am!
- **Patrick:** Who's ready?

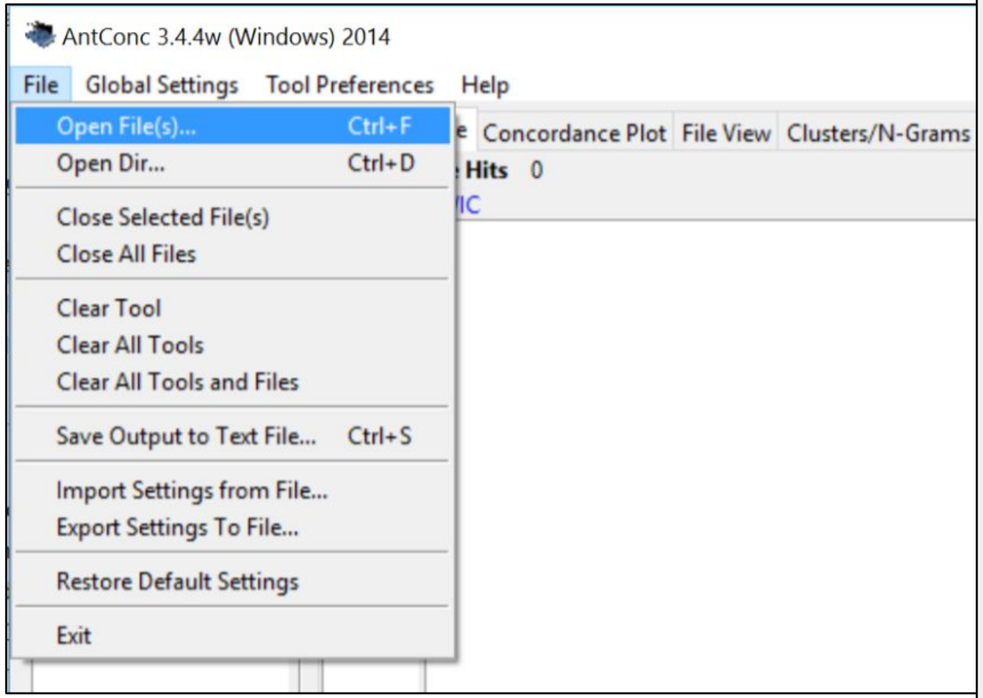


Main article: List of transcripts/season 1

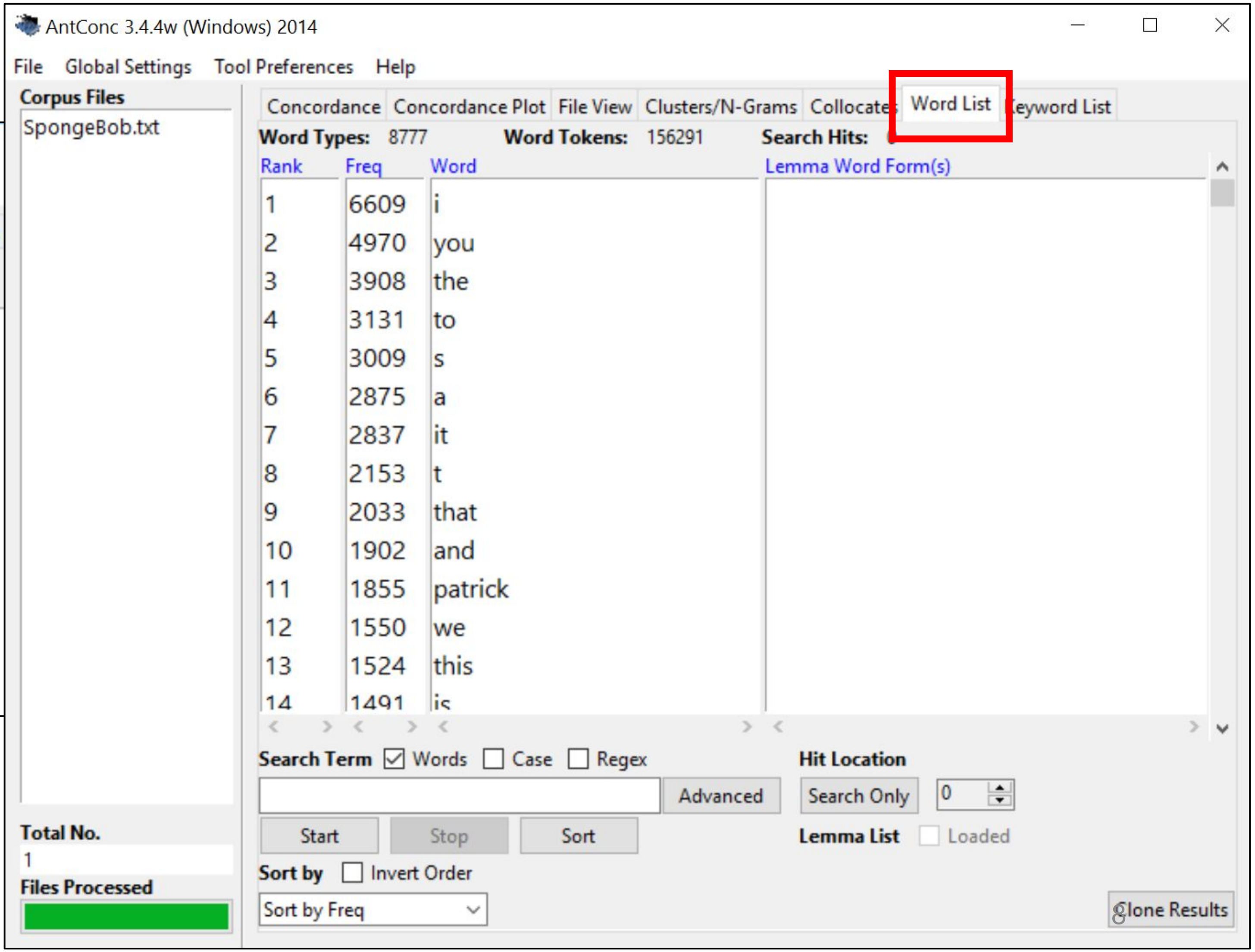
#	Title	Transcript
1a	Help Wanted	View transcript
1b	Reef Blower	View transcript
1c	Tea at the Treedome	View transcript
2a	Bubblestand	View transcript
2b	Ripped Pants	View transcript
3a	Jellyfishing	View transcript
3b	Plankton!	View transcript
4a	Naughty Nautical Neighbors	View transcript
4b	Boating School	View transcript
5a	Pizza Delivery	View transcript
5b	Home Sweet Pineapple	View transcript
6a	Mermaid Man and Barnacle Boy	View transcript
6b	Pickles	View transcript
7a	Hall Monitor	View transcript
7b	Jellyfish Jam	View transcript
8a	Sandy's Rocket	View transcript
8b	Squeaky Boots	View transcript
9a	Nature Pants	View transcript

Keyword Approach

METHOD – KEYWORDS



Hits 0



Corpus Files: SpongeBob.txt

Word Types: 8777 Word Tokens: 156291 Search Hits:

Rank	Freq	Word
1	6609	i
2	4970	you
3	3908	the
4	3131	to
5	3009	s
6	2875	a
7	2837	it
8	2153	t
9	2033	that
10	1902	and
11	1855	patrick
12	1550	we
13	1524	this
14	1491	is

Lemma Word Form(s)

Search Term ☑ Words ☐ Case ☐ Regex

Hit Location 0

Lemma List ☐ Loaded

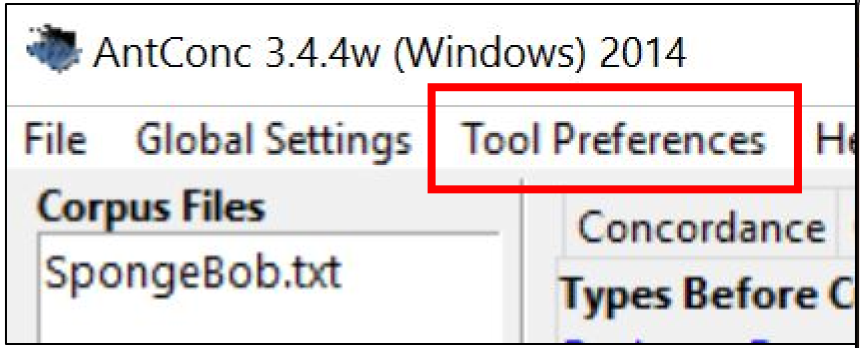
Sort by ☐ Invert Order

Sort by Freq

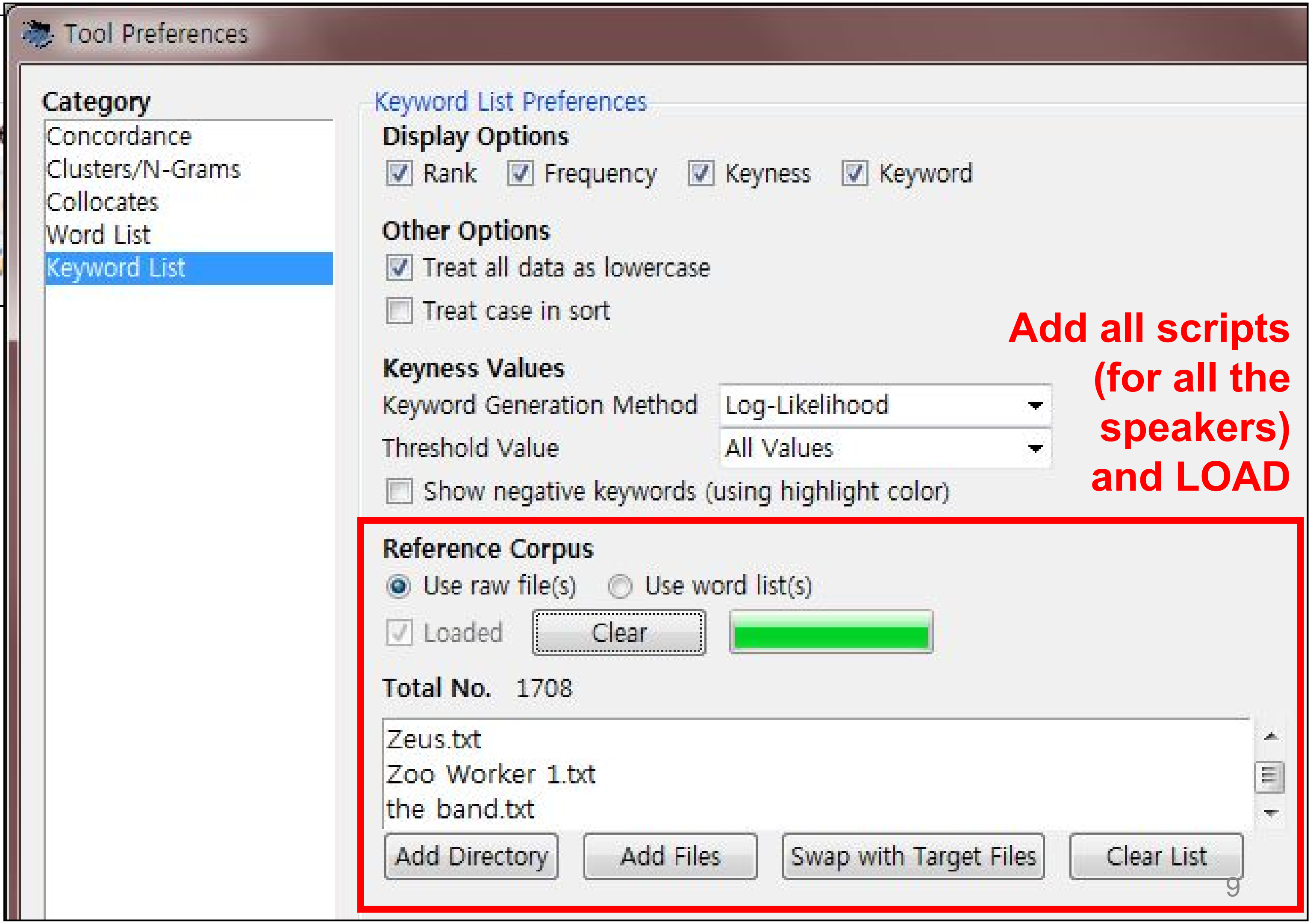
Total No. 1

Files Processed

METHOD – KEYWORDS (cont.)



SpongeBob.txt



Display Options

☑ Rank ☑ Frequency ☑ Keyness ☑ Keyword

Other Options

☑ Treat all data as lowercase

☐ Treat case in sort

Keyness Values

Keyword Generation Method: Log-Likelihood

Threshold Value: All Values

☐ Show negative keywords (using highlight color)

Reference Corpus

● Use raw file(s) ○ Use word list(s)

☑ Loaded

Total No. 1708

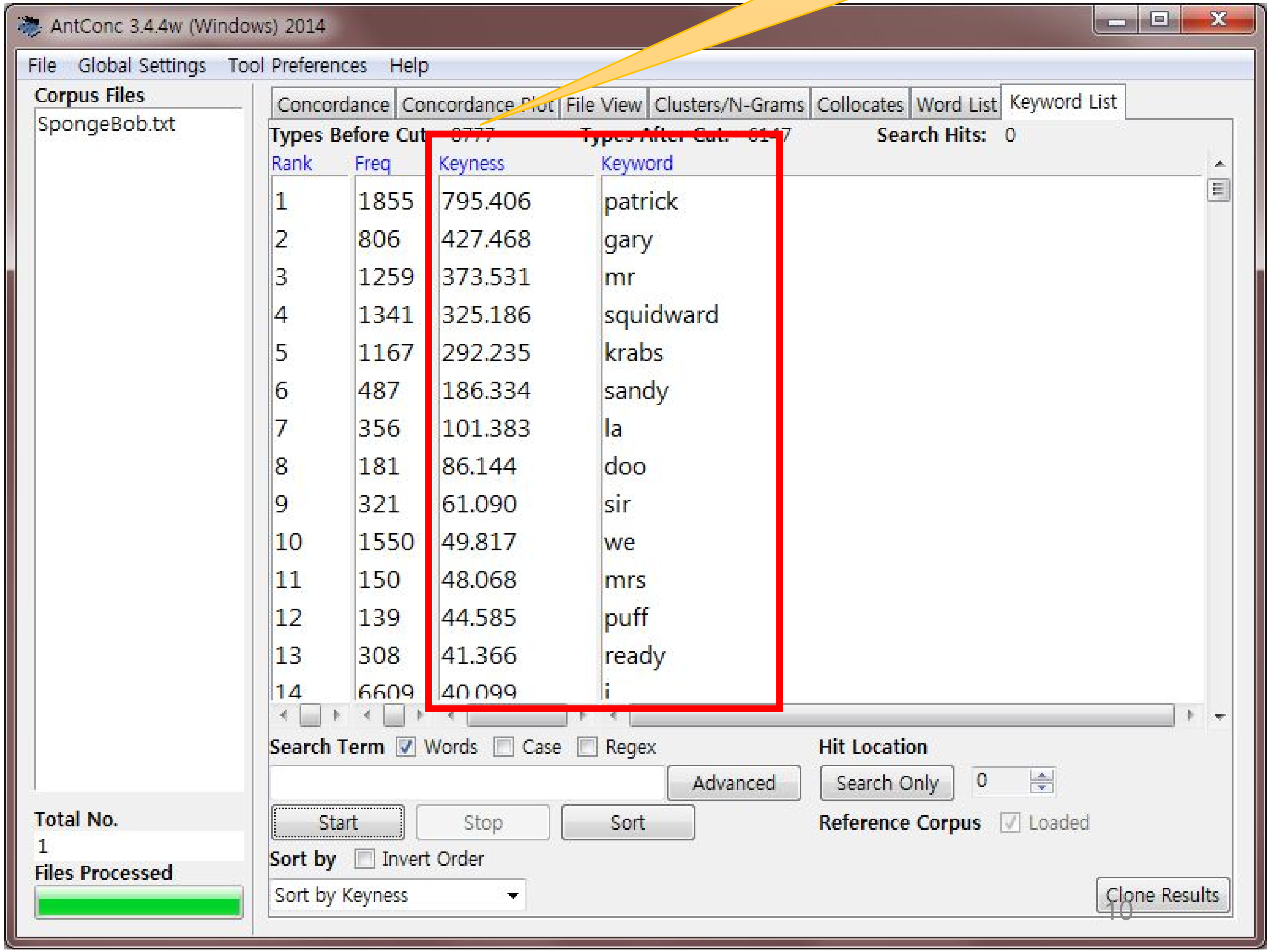
Zeus.txt
Zoo Worker 1.txt
the band.txt

Add all scripts (for all the speakers) and LOAD

METHOD – KEYWORDS (Cont.)

- Sort by **Keyness**:
  - What is **keyness**?

What is Keyness?



Types Before Cut: 8777 Types After Cut: 8147 Search Hits: 0

Rank	Freq	Keyness	Keyword
1	1855	795.406	patrick
2	806	427.468	gary
3	1259	373.531	mr
4	1341	325.186	squidward
5	1167	292.235	krabs
6	487	186.334	sandy
7	356	101.383	la
8	181	86.144	doo
9	321	61.090	sir
10	1550	49.817	we
11	150	48.068	mrs
12	139	44.585	puff
13	308	41.366	ready
14	6609	40.099	i

Total No.
1
Files Processed

Search Term ☑ Words □ Case □ Regex

Hit Location: Search Only 0

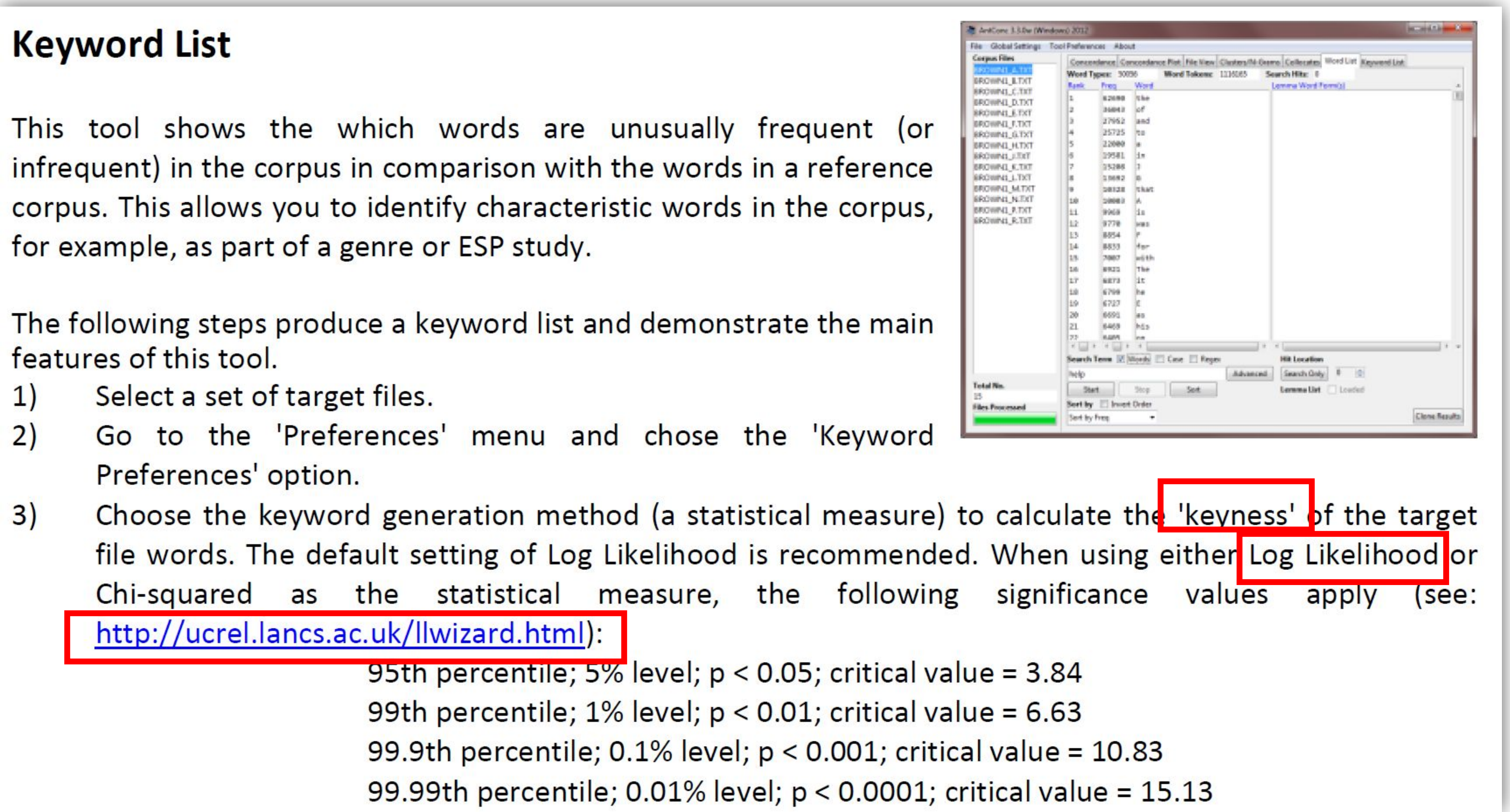
Reference Corpus ☑ Loaded

Sort by □ Invert Order

Sort by Keyness

METHOD – KEYWORDS (Cont.)

Help >> View Readme Help File



Keyword List

This tool shows the which words are unusually frequent (or infrequent) in the corpus in comparison with the words in a reference corpus. This allows you to identify characteristic words in the corpus, for example, as part of a genre or ESP study.

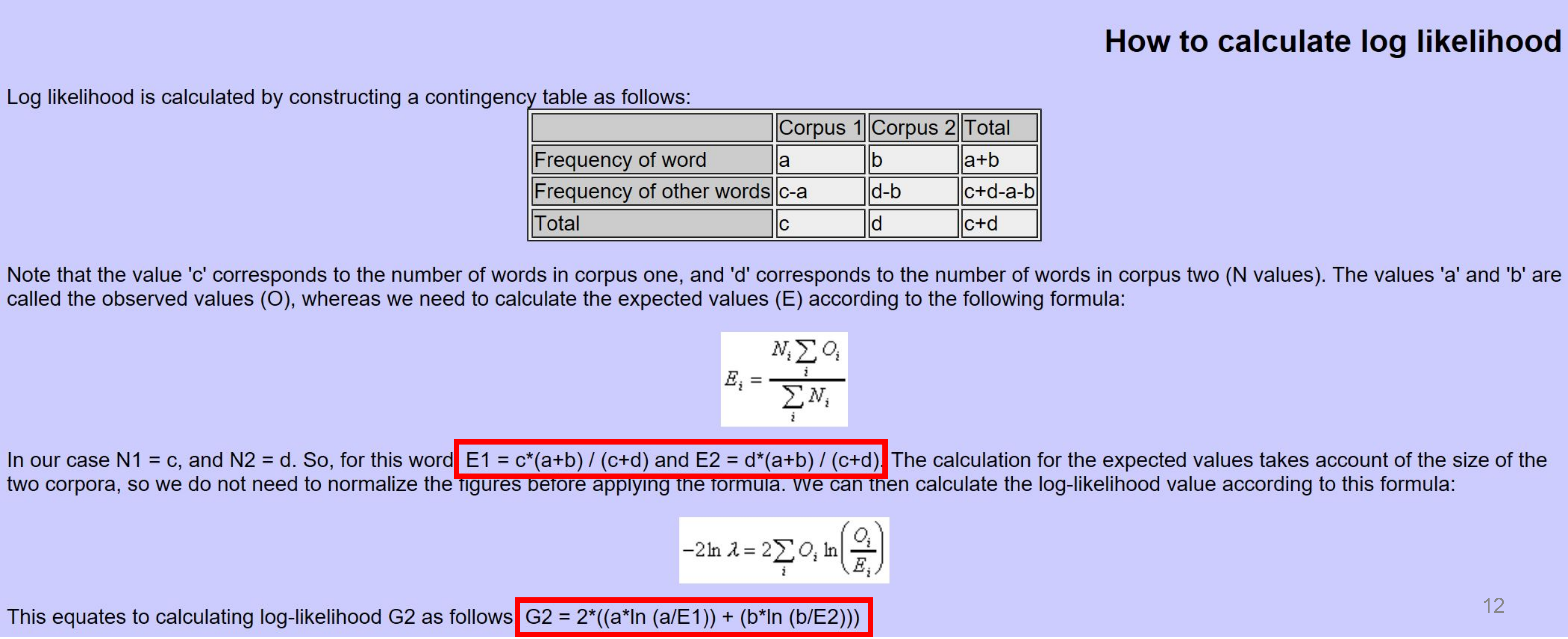
The following steps produce a keyword list and demonstrate the main features of this tool.

1) Select a set of target files.
2) Go to the 'Preferences' menu and chose the 'Keyword Preferences' option.
3) Choose the keyword generation method (a statistical measure) to calculate the 'keyness' of the target file words. The default setting of Log Likelihood is recommended. When using either Log Likelihood or Chi-squared as the statistical measure, the following significance values apply (see: http://ucrel.lancs.ac.uk/llwizard.html):

   95th percentile; 5% level; p < 0.05; critical value = 3.84
   99th percentile; 1% level; p < 0.01; critical value = 6.63
   99.9th percentile; 0.1% level; p < 0.001; critical value = 10.83
   99.99th percentile; 0.01% level; p < 0.0001; critical value = 15.13

METHOD – KEYWORDS (Cont.)

- http://ucrel.lancs.ac.uk/llwizard.html



How to calculate log likelihood

Log likelihood is calculated by constructing a contingency table as follows:

	Corpus 1	Corpus 2	Total
Frequency of word	a	b	a+b
Frequency of other words	c-a	d-b	c+d-a-b
Total	c	d	c+d

Note that the value 'c' corresponds to the number of words in corpus one, and 'd' corresponds to the number of words in corpus two (N values). The values 'a' and 'b' are called the observed values (O), whereas we need to calculate the expected values (E) according to the following formula:

$$E_i = \frac{N_i \sum_i O_i}{\sum_i N_i}$$

In our case N1 = c, and N2 = d. So, for this word, E1 = c*(a+b) / (c+d) and E2 = d*(a+b) / (c+d). The calculation for the expected values takes account of the size of the two corpora, so we do not need to normalize the figures before applying the formula. We can then calculate the log-likelihood value according to this formula:

$$-2\ln\lambda = 2\sum_i O_i \ln\left(\frac{O_i}{E_i}\right)$$

This equates to calculating log-likelihood G2 as follows: G2 = 2*((a*ln (a/E1)) + (b*ln (b/E2)))

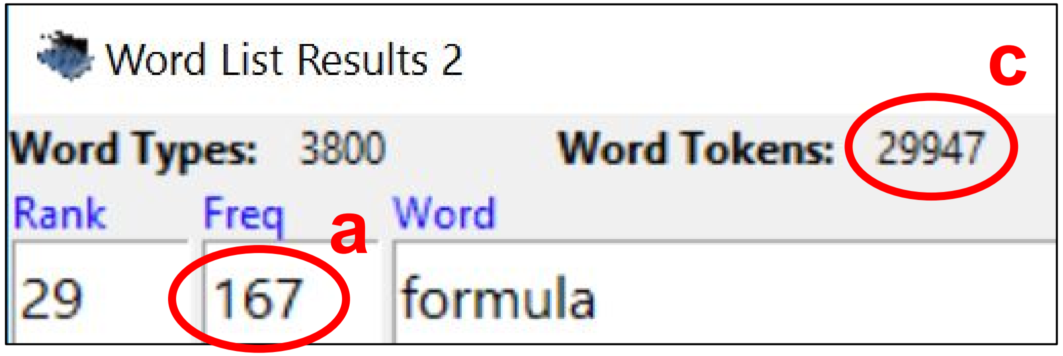
METHOD – KEYWORDS (Cont.)

[token]	Corpus 1	Corpus 2
Frequency of Word	a	b
Corpus size	c	d

Summary:

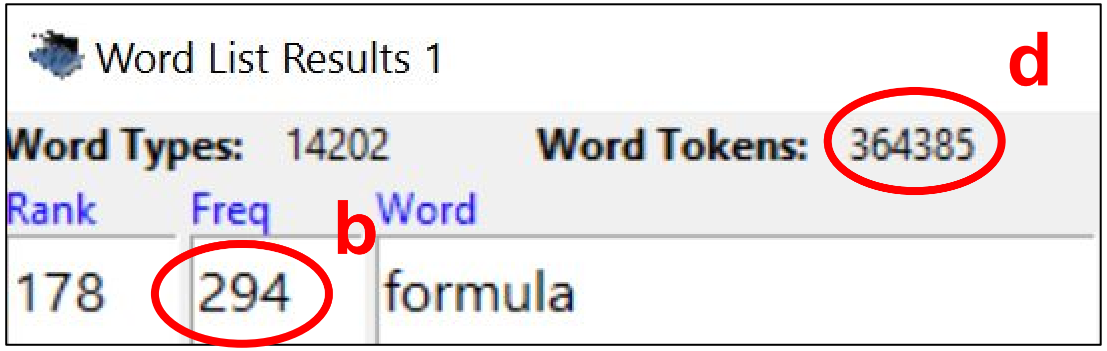
- E1 = c*(a+b) / (c+d)
- E2 = d*(a+b) / (c+d)
- **Keyness** = 2*((a*ln (a/E1)) + (b*ln (b/E2)))

METHOD – KEYWORDS (Cont.)



Word Types: 3800 Word Tokens: 29947

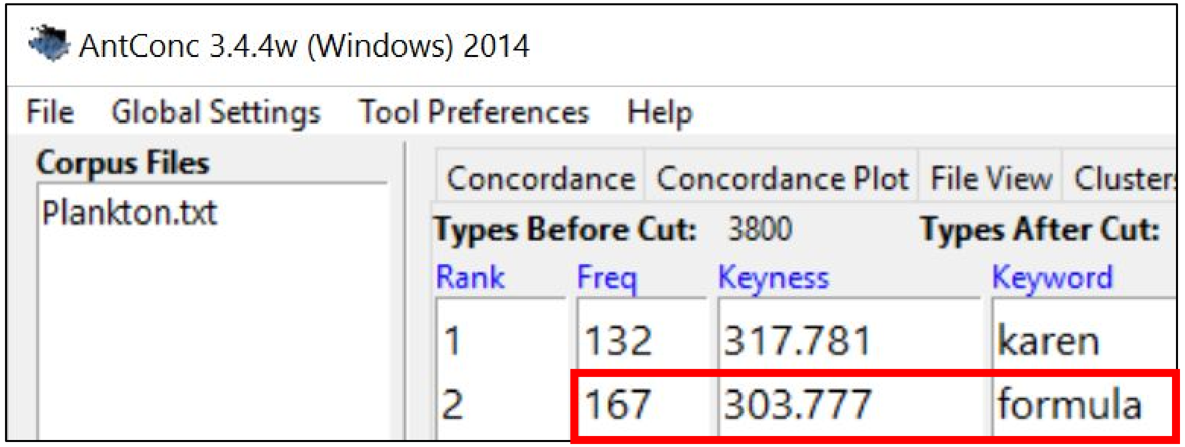
Rank	Freq	Word
29	167	formula



Word Types: 14202 Word Tokens: 364385

Rank	Freq	Word
178	294	formula

- Verification

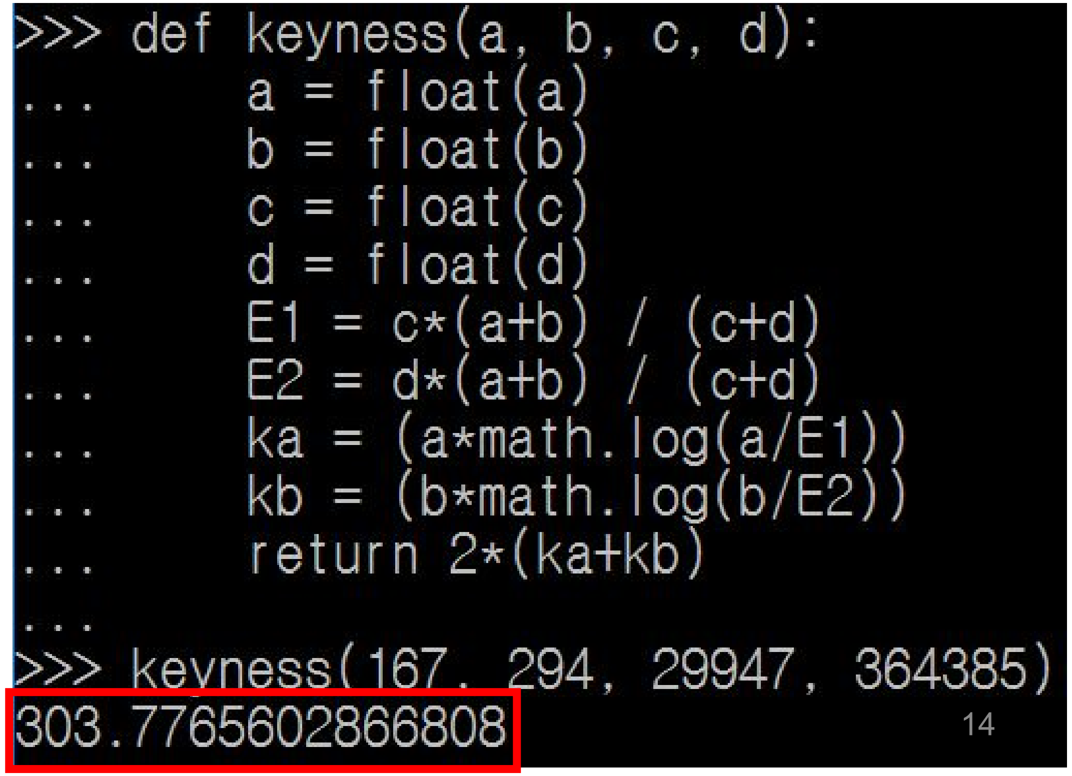


Corpus Files

Plankton.txt

Types Before Cut: 3800 Types After Cut:

Rank	Freq	Keyness	Keyword
1	132	317.781	karen
2	167	303.777	formula



```
>>> def keyness(a, b, c, d):
...     a = float(a)
...     b = float(b)
...     c = float(c)
...     d = float(d)
...     E1 = c*(a+b) / (c+d)
...     E2 = d*(a+b) / (c+d)
...     ka = (a*math.log(a/E1))
...     kb = (b*math.log(b/E2))
...     return 2*(ka+kb)
...
>>> keyness(167, 294, 29947, 364385)
303.7765602866808
```

Not showing:
Pronouns(Patrick, Gary, Mr. Krabs), Interjection(oh, wow)

Keyword Result

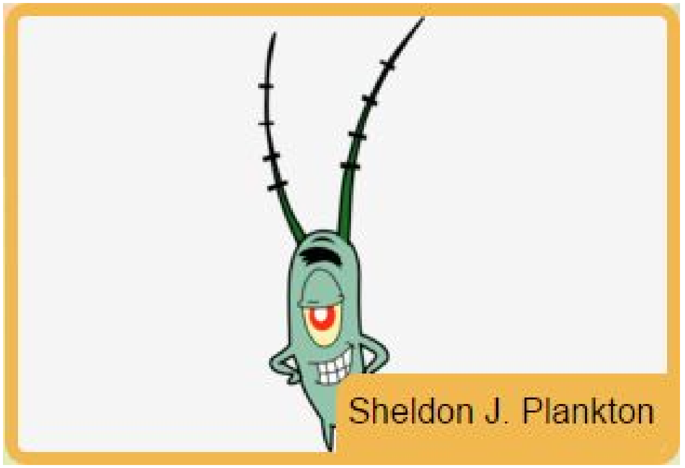
SpongeBob SquarePants (character)

Rank	Frequency	Keyness	Word
13	**308**	41.366	**ready**
18	281	27.577	**sorry**
19	210	25.872	guess
20	197	24.558	**best**
21	178	23.572	**friend**
22	**406**	20.901	**okay**
25	102	18.692	**jellyfish**
26	138	17.694	**worry**
34	63	13.799	**spatula**
36	155	12.487	**buddy**

Keyword Result (Cont.)

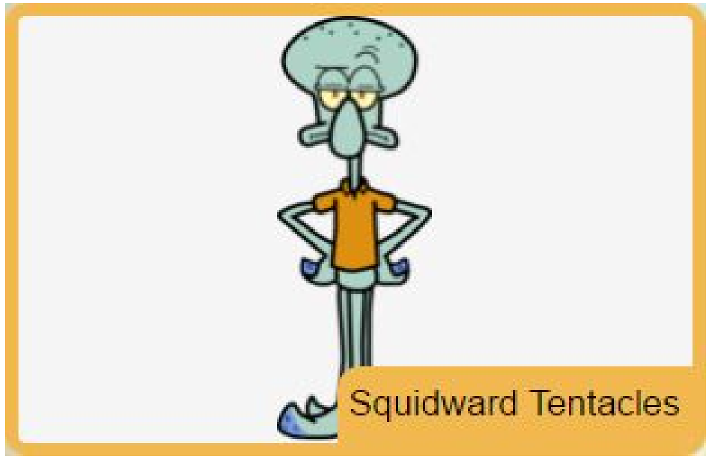
Rank	Frequency	Keyness	Word
1	**354**	467.163	**money**
2	**408**	338.615	**boy**
3	**1027**	226.042	**me**
4	90	149.547	**lad**(젊은이)
5	105	115.565	**customers**
6	162	84.869	**ya**
7	78	68.870	**boys**
10	138	42.642	**patties**
11	44	39.369	**dollar**
13	80	37.533	**free**

Keyword Result (Cont.)


Sheldon J. Plankton

Rank	Frequency	Keyness	Word
2	**167**	341.692	**formula**
3	98	172.930	**chum**
4	102	130.935	**secret**
5	50	82.089	**bucket**
7	26	59.169	**wife**
8	47	56.949	mine
9	36	55.701	**plan**
10	122	53.999	**patty**
11	33	52.922	**recipe**
12	34	51.706	**steal**

Keyword Result (Cont.)

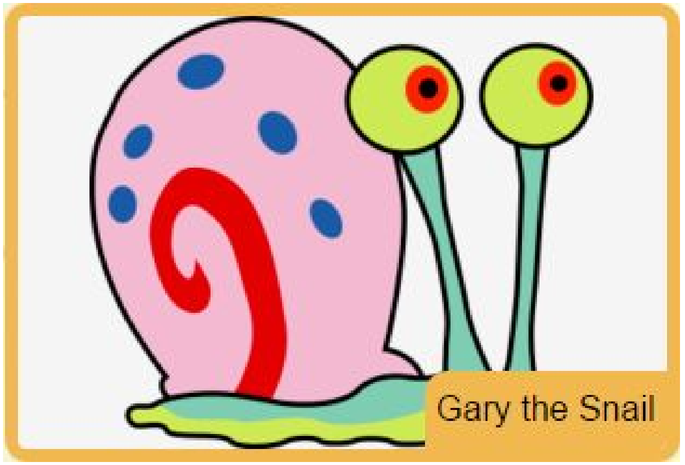


Rank	Frequency	Keyness	Word
2	**611**	94.085	**no**
3	174	75.149	**two**
5	**748**	57.677	**what**
7	45	45.993	**art**
8	37	43.834	**clarinet**
9	23	39.927	**morons**
10	26	37.625	**squilliam**
12	42	35.634	**whatever**
13	43	31.021	**stupid**
14	30	30.996	**quiet**

Keyword Result (Cont.)

Rank	Frequency	Keyness	Word
6	20	74.906	**critter**
7	28	56.301	**karate**
8	27	56.189	**air**
9	17	56.160	**rodeo**
10	13	51.327	**critters**
14	16	45.009	**nuts**
15	12	43.866	**experiment**
16	14	36.561	**science**
18	8	31.115	**tarnation**
20	11	29.212	**helmet**

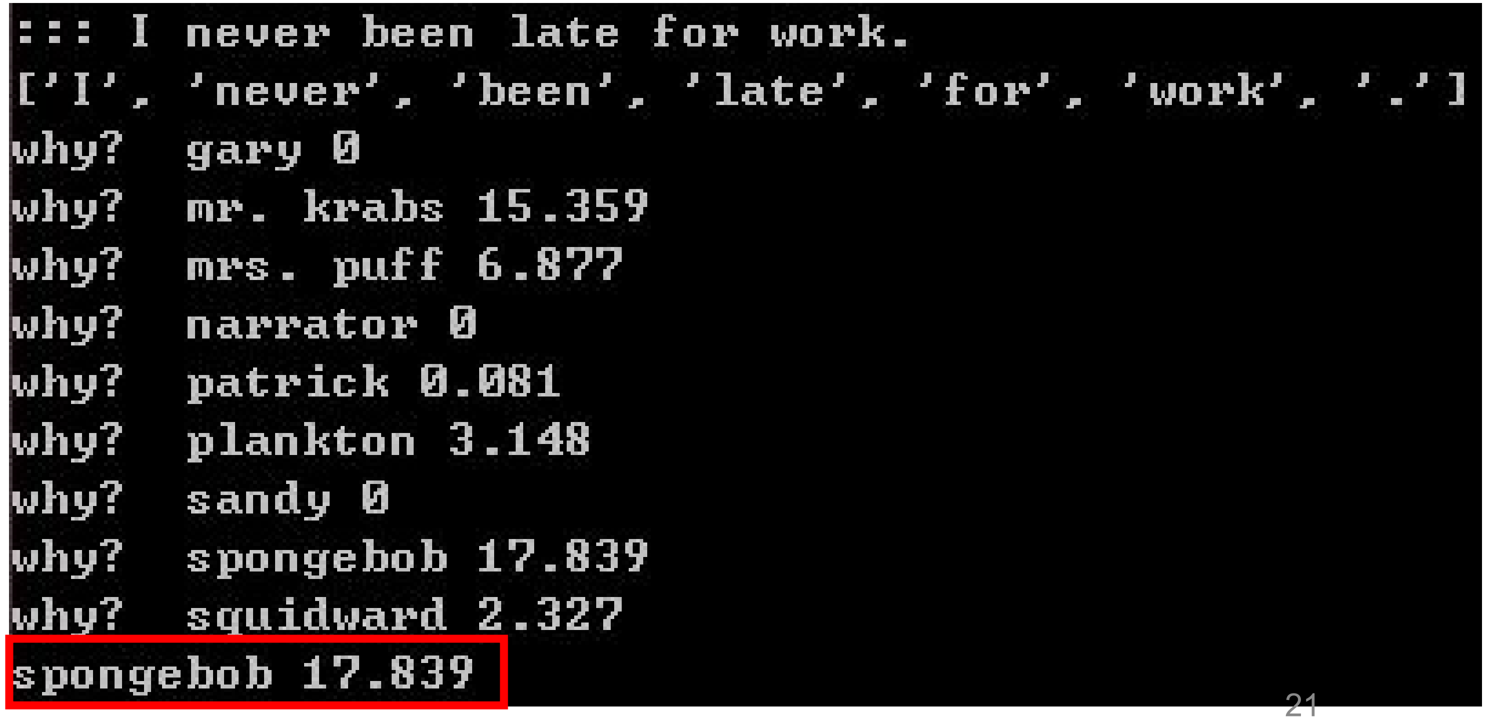
Keyword Result (Cont.)



Rank	Frequency	Keyness	Word
1	**521**	5156.911	**meow**
2	4	40.943	**mooowww**
3	3	30.707	**reow**
4	3	29.467	**mah**
5	3	28.446	**meoooow**
6	2	20.471	**meowow**
7	2	20.471	**moooowww**
8	2	20.471	**mrlooooow**
…			

METHOD – KEYWORDS (Cont.)

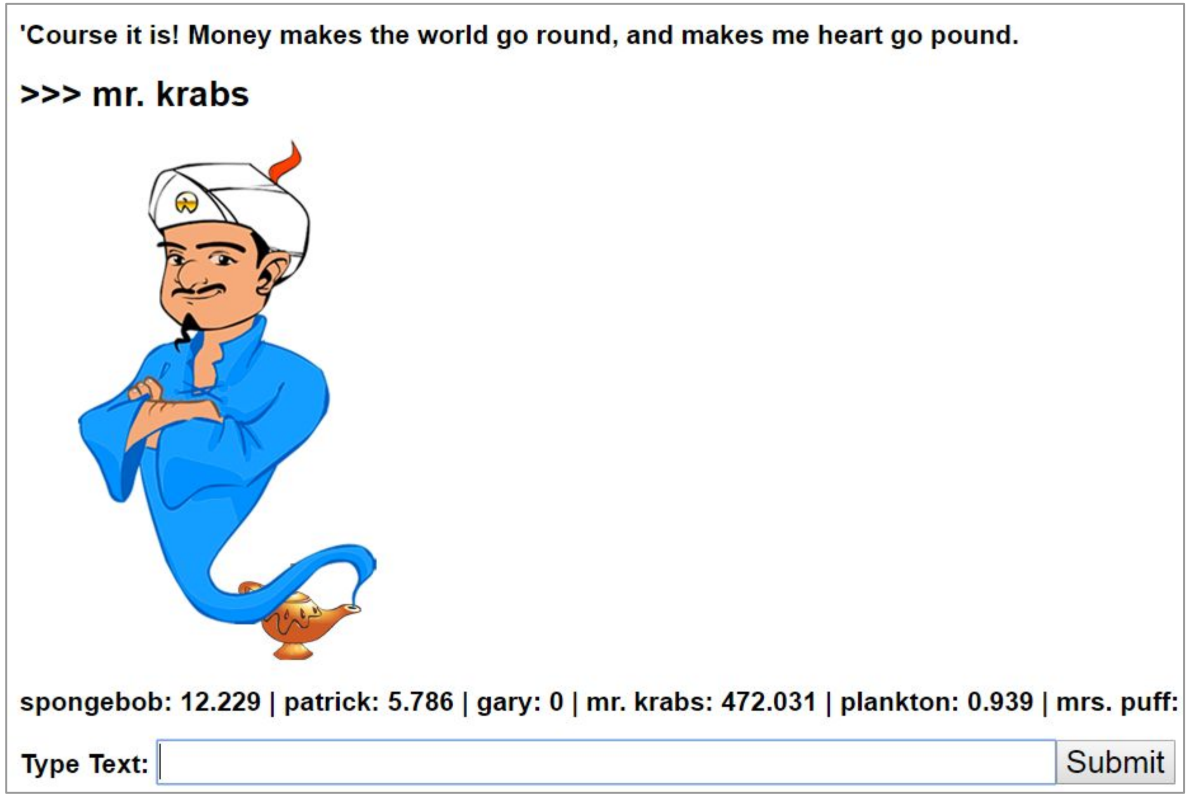
- Keyness shows the importance of the word in the text, based on the impact in the full text.
- So, I used this metric to construct the speaker recognition system.
    - Score for each token.
    - Sum up and find result.



```
::: I never been late for work.
['I', 'never', 'been', 'late', 'for', 'work', '.']
why?  gary 0
why?  mr. krabs 15.359
why?  mrs. puff 6.877
why?  narrator 0
why?  patrick 0.081
why?  plankton 3.148
why?  sandy 0
why?  spongebob 17.839
why?  squidward 2.327
spongebob 17.839
```

LIVE DEMO

- Gets score when the word hits.
- Ignore **Names** (e.g. SpongeBob, Patrick, Krusty Krabs ...)
- Ignore **Stop words** (e.g. i, me, my, myself, you, yourself …)
- **Run!**



'Course it is! Money makes the world go round, and makes me heart go pound.

>>> mr. krabs

spongebob: 12.229 | patrick: 5.786 | gary: 0 | mr. krabs: 472.031 | plankton: 0.939 | mrs. puff:

LIVE DEMO

Test sets:

Plankton: Why couldn't I see it before? The way to get the Krabby Patty formula was so obvious! Spend an inordinate amount of time training several dozen sea bears to take over your restaurant and force you to give it up! can turn them from their central purpose!

SpongeBob: Yoo-hoo! Who wants their tummies tickled?

Plankton: No... My weapons! Ouch!

SpongeBob: Sea bears aren't weapons, Plankton. They're furry buckets of love. See? And what do sea bears love more than tummy tickles? Jellyfish honey!

SpongeBob: Come and get it!

Plankton: No! Come back!

Mr. Krabs: Why do you keep doing this, Plankton?

Plankton: Heh-heh-heh...

Mr. Krabs: When you mess with me business, ya mess with me money!

Plankton: Er, money's not everything, you know.

Mr. Krabs: 'Course it is! Money makes the world go round, and makes me heart go pound.

Higher Value: Strong Identity

Testing Result

Character	Correct	Total	Percentage
SpongeBob	3150	13348	23.60 %
Gary	**415**	**420**	**98.81 %**
Mr. Krabs	**2168**	**4812**	**45.05 %**
Patrick	**2421**	**5206**	**46.50 %**
Plankton	**848**	**2164**	**39.19 %**
Sandy	**517**	**1403**	**36.85 %**
Squidward	**1416**	**4796**	**29.52 %**

However, POS keyword approach failed

- **Tagging** the words, **categorizing** into Nouns, Pronouns, Verbs, Adverbs, Adjective, and analyzing **keyness** was not effective.
- Tagger problem
  - Lots of scripts were difficult to tag for the program.
    - MAN OVERBOARD! Climb, Mr. Squidward! Climb!
    - N N N NP NP N
    - Backing up! Backing up! Ba-a-a-a-a-a-a-a-a-a-a-a-ack-i-i-i-i-ng up!
    - VVG RP VVG RP NP RB
  - Both NLTK tagger and TAGANT was not working correctly.
    - e.g. SpongeBob was categorized into noun, verb, adverb…
    - **Imperative (명령형) sentences are all categorized into noun…**

Hypernym Approach

Hypernym in NLTK

1. Tokenize the sentence and **pos tag**. (possible tagger problem)
    a. words = nltk.pos_tag(nltk.word_tokenize(line))
2. Consider the word which is ‘NN’ tag: Noun.
3. Find synsets (synnonym sets).
    a. syns = wn.synsets(w, pos=wn.NOUN)
    b. e.g. is, was, am, are → “be.v.01, be.v.03 ...”
4. Choose the first synset.
    a. ws = syns[0]
5. Follow up the hypernym path.
    a. ws.hypernym_paths()[0] ← sometimes many paths are available
6. Create the network.

Hypernym in NLTK

1. Testing: I never been late for work.

2.
   ```
   >>> nltk.pos_tag(nltk.word_tokenize('I never been late for work.'))
   [('I', 'PRP'), ('never', 'RB'), ('been', 'VBN'), ('late', 'JJ'), ('for', 'IN'), ('work', 'NN'), ('.', '.')]
   ```

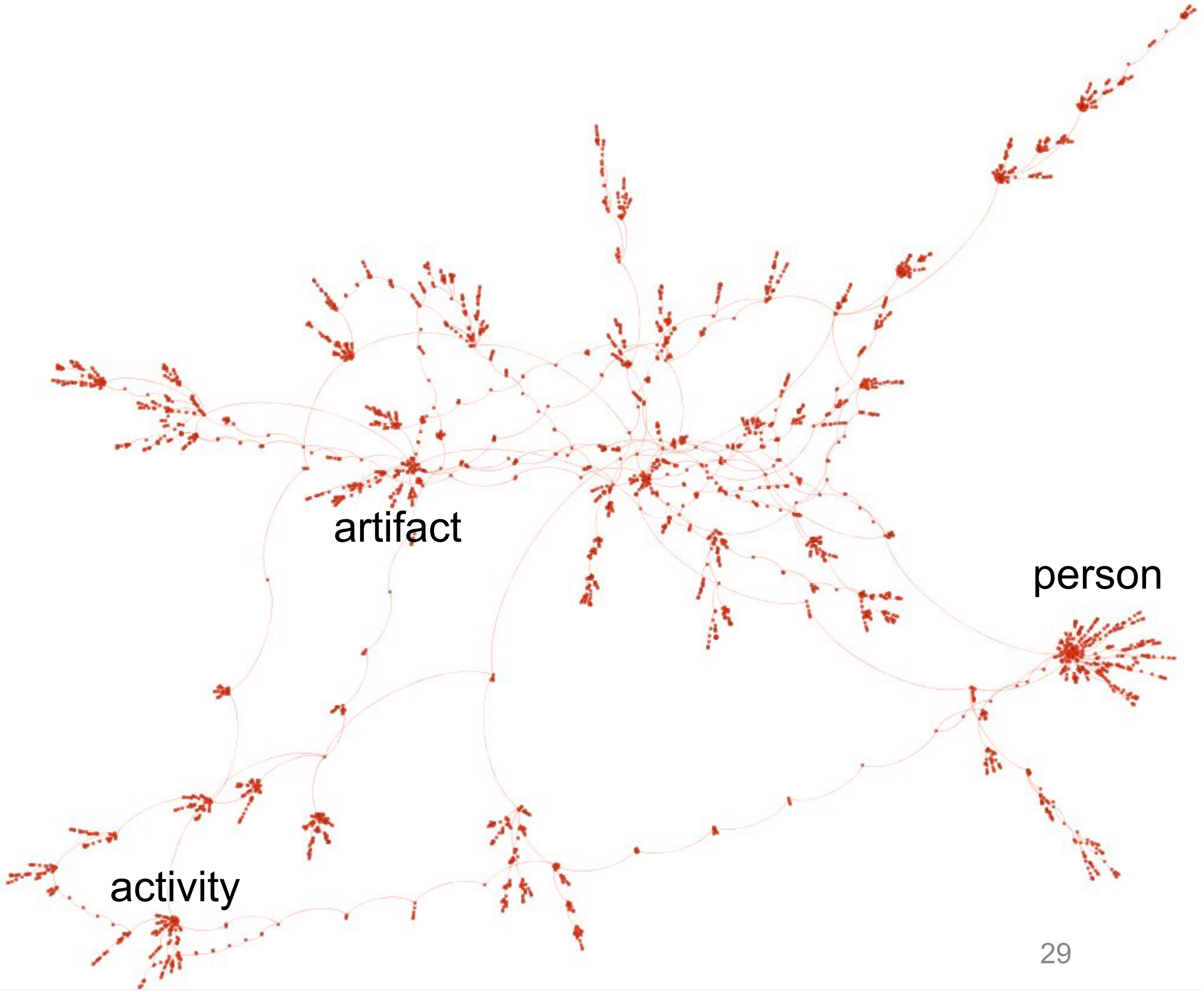
3.
   ```
   >>> wn.synsets('work', pos=wn.NOUN)
   [Synset('work.n.01'), Synset('work.n.02'), Synset('employment.n.02'), Synset('study.n.02'), Synset('work.n.05'), Synset('workplace.n.01'), Synset('oeuvre.n.01')]
   ```

4.
   ```
   >>> wn.synsets('work', pos=wn.NOUN)[0].hypernym_paths()[0]
   [Synset('entity.n.01'), Synset('abstraction.n.06'), Synset('psychological_feature.n.01'), Synset('event.n.01'), Synset('act.n.02'), Synset('activity.n.01'), Synset('work.n.01')]
   ```
5. Using the hypernym path, draw the network.

HYPERNYM Approach (Nouns)

- #FILE:SpongeBob
- #Vertices 6170
- #Arcs 6369
- #MaxPath 18
- #LeafN 4757

Rank	Count	Synset
01	369	person.n.01
02	142	activity.n.01
03	142	device.n.01
04	126	state.n.02
05	126	artifact.n.01
06	118	quality.n.01
07	111	happening.n.01
08	105	time_period.n.01
09	104	condition.n.01
10	96	material.n.01

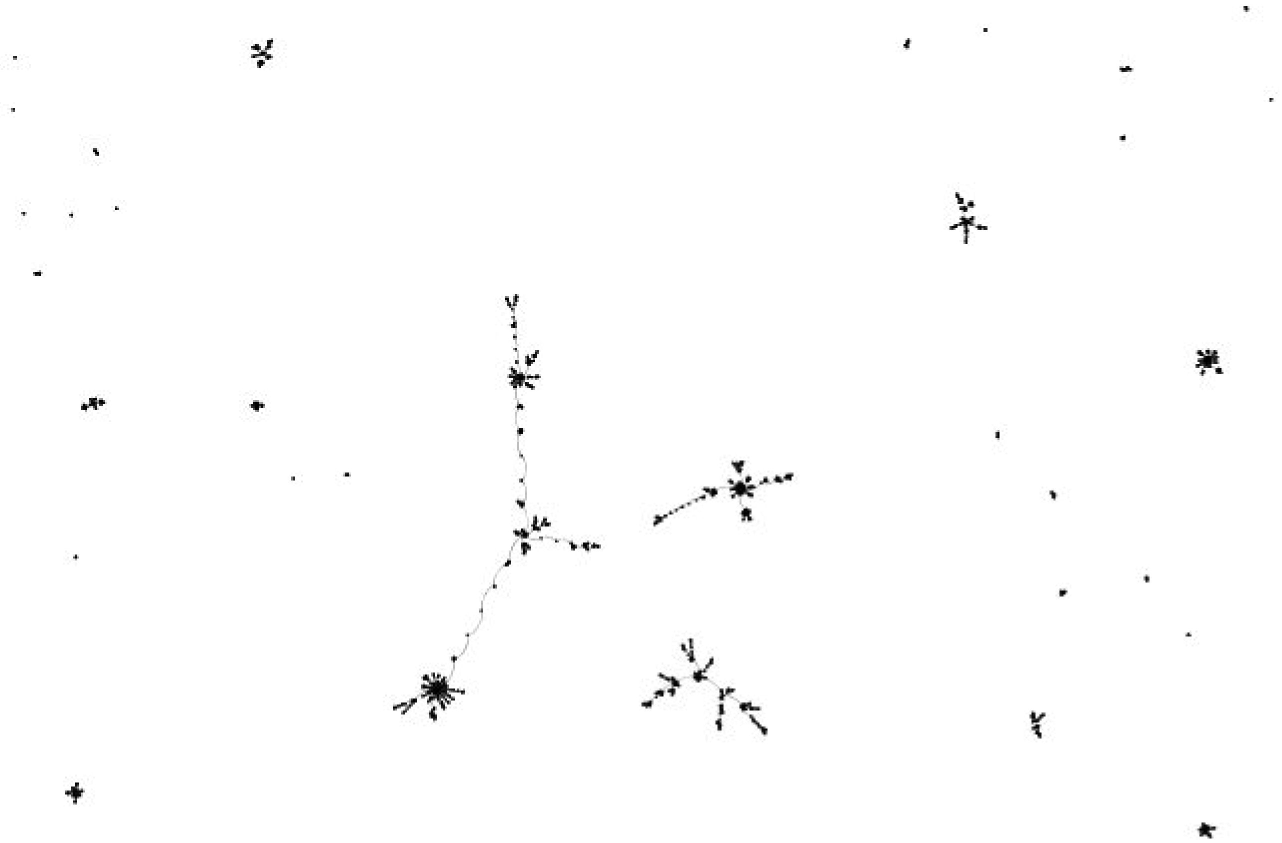


HYPERNYM Approach (Verbs)

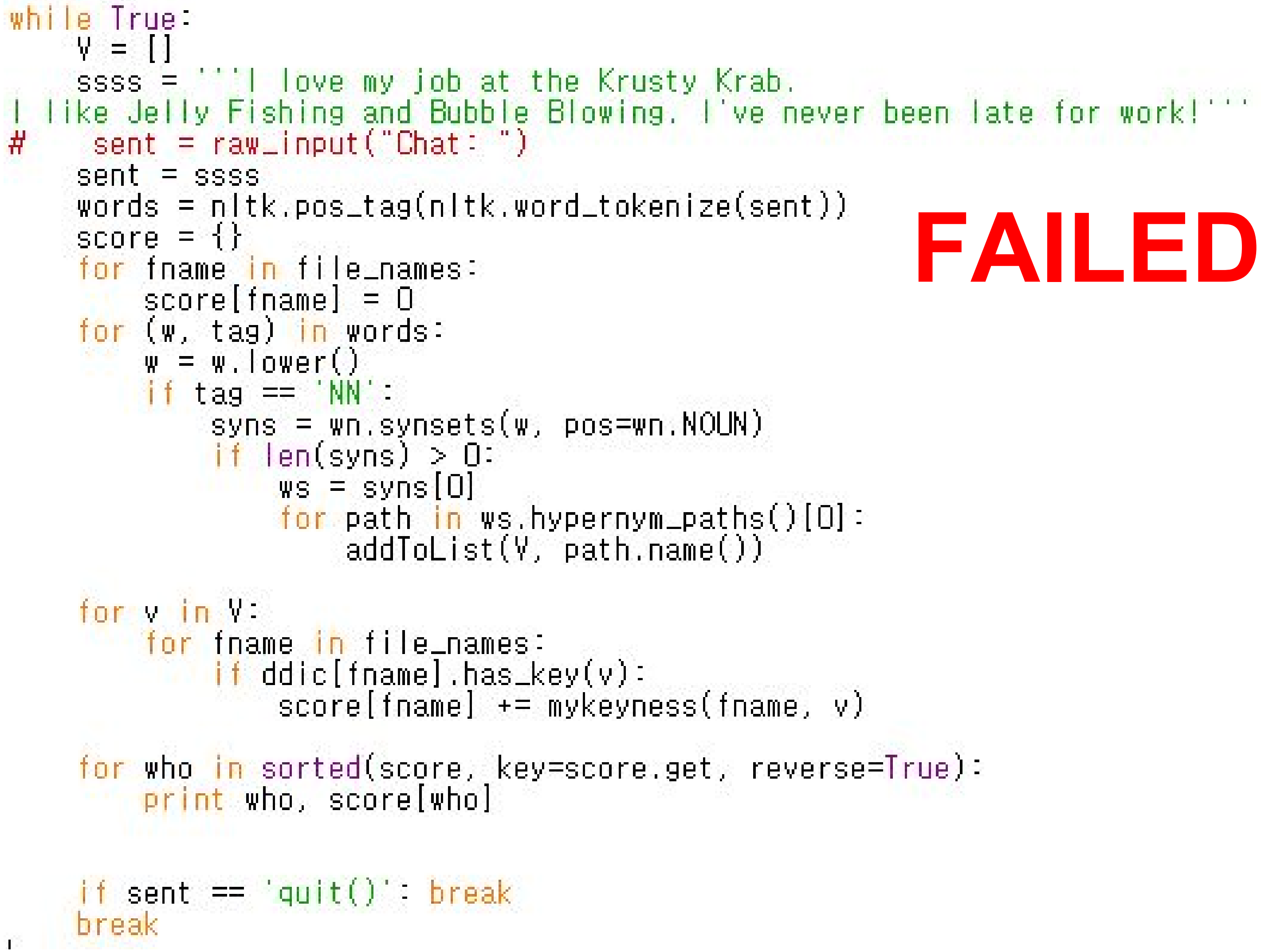
Verbs don’t cluster that much.

- #FILE:SpongeBob
- #Vertices 2130
- #Arcs 1916
- #MaxPath 12
- #LeafN 3504

Rank	Count	Synset
01	205	change.v.01
02	173	travel.v.01
03	122	change.v.02
04	101	be.v.01
05	98	move.v.02
06	77	move.v.03
07	67	act.v.01
08	60	make.v.03
09	55	inform.v.01
10	53	communicate.v.02



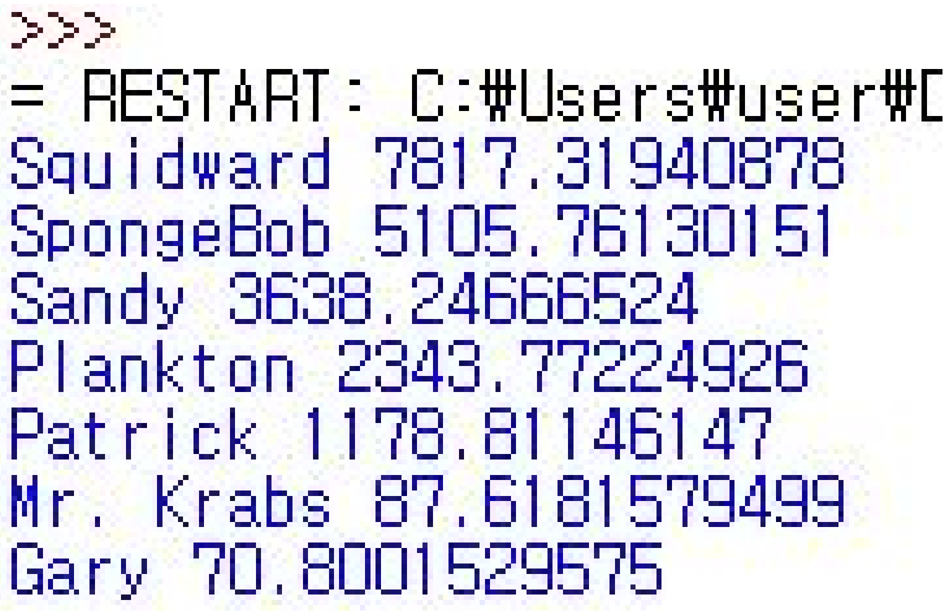
Keyness? (Nouns)



```
while True:
    V = []
    ssss = '''I love my job at the Krusty Krab.
I like Jelly Fishing and Bubble Blowing. I've never been late for work!'''
#    sent = raw_input("Chat: ")
    sent = ssss
    words = nltk.pos_tag(nltk.word_tokenize(sent))
    score = {}
    for fname in file_names:
        score[fname] = 0
    for (w, tag) in words:
        w = w.lower()
        if tag == 'NN':
            syns = wn.synsets(w, pos=wn.NOUN)
            if len(syns) > 0:
                ws = syns[0]
                for path in ws.hypernym_paths()[0]:
                    addToList(V, path.name())

    for v in V:
        for fname in file_names:
            if ddic[fname].has_key(v):
                score[fname] += mykeyness(fname, v)

    for who in sorted(score, key=score.get, reverse=True):
        print who, score[who]


    if sent == 'quit()': break
    break
```

FAILED

I tested many other sentences, but they didn't show the speaker well.



```
>>>
= RESTART: C:₩Users₩user₩D
Squidward 7817.31940878
SpongeBob 5105.76130151
Sandy 3638.24666524
Plankton 2343.77224926
Patrick 1178.81146147
Mr. Krabs 87.6181579499
Gary 70.8001529575
```

Failures

- Tagger Problem
  - Not working correctly, especially for imperative form.
- Synset and Hypernym Path problem
  - In fact, we should choose the right synset and hypernym path manually.
- Hypernym Keyness Speaker Detector
  - Difficult to test by each sentence. (Sentence has too few word to make an analizable hypernym network)
  - Difficult to find out the meaning from the graph.

Comparison with my previous project.

- I used Brown Corpus into those categories:
- Then tested with the real news articles (Chosun english news, Reuters)

- A. PRESS: Reportage (*44 texts*)
- B. PRESS: Editorial (*27 texts*)
- C. PRESS: Reviews (*17 texts*)
- D. RELIGION (*17 texts*)
- E. SKILL AND HOBBIES (*36 texts*)
- F. POPULAR LORE (*48 texts*)
- G. BELLES-LETTRES - Biography, Memoirs, etc. (*75 texts*)
- H. MISCELLANEOUS: US Government & House Organs (*30 texts*)
- J. LEARNED (*80 texts*)
- K. FICTION: General (*29 texts*)
- L. FICTION: Mystery and Detective Fiction (*24 texts*)
- M. FICTION: Science (*6 texts*)
- N. FICTION: Adventure and Western (*29 texts*)
- P. FICTION: Romance and Love Story (*29 texts*)
- R. HUMOR (*9 texts*)

Conclusion

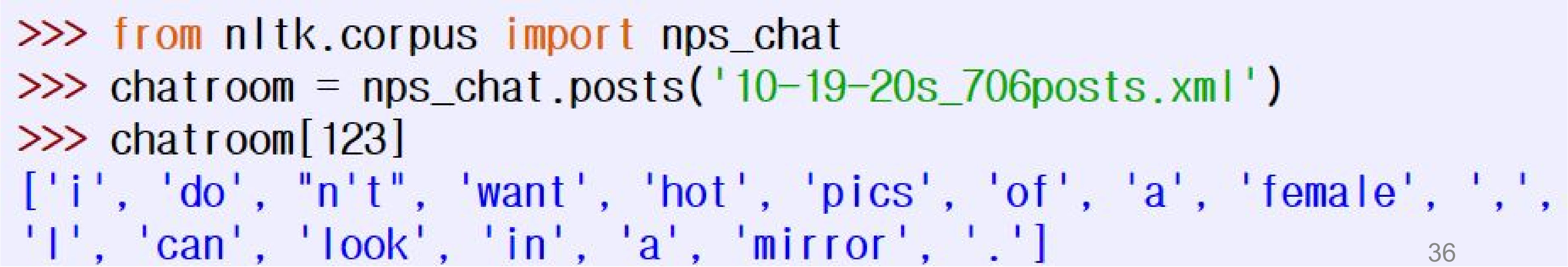
Conclusion

- Successfully built speaker recognition system.
  - Keywords were detected very well.
- For the spoken corpus, it is better to calculate the **keyness** of the word directly.
  - POS Tagging and categorization approach is inefficient.
- POS Tagging fails for spoken text in many case.
- Hypernym approach **might be** better for **written corpus.**
  - We don’t use various words when we are speaking.
  - Mostly, the size of the spoken data is small to make network.
  - <u>But it might be effective for written data</u>, since we try to avoid to use same word frequently, which makes easy to recognize the key hypernym.

Future research

- NLTK also provides the chat corpus (NPS):
  - originally collected by the Naval Postgraduate School
  - The corpus contains over 10,000 posts, anonymized by replacing usernames with generic names of the form "UserNNN", and manually edited to remove any other identifying information.
  - The corpus is organized into 15 files, where each file contains several hundred posts collected on a given date, for an **age-specific chatroom (teens, 20s, 30s, 40s, plus a generic adults chatroom)**

Age Detection?



```
>>> from nltk.corpus import nps_chat
>>> chatroom = nps_chat.posts('10-19-20s_706posts.xml')
>>> chatroom[123]
['i', 'do', "n't", 'want', 'hot', 'pics', 'of', 'a', 'female', ',',
'I', 'can', 'look', 'in', 'a', 'mirror', '.']
```

Thank you!