In [1]:
```python
import numpy as np
from __future__ import division
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/adilrhoulam/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[1]: True

In [3]:
```python
#Inspired from the contex2vec evaluation function explore_context2vec
import numpy
import six
import sys
import traceback
import re

from chainer import cuda
from context2vec.common.context_models import Toks
from context2vec.common.model_reader import ModelReader

class ParseException(Exception):
    def __init__(self, str):
        super(ParseException, self).__init__(str)

target_exp = re.compile('\[.*\]')

def parse_input(line):
    sent = line.strip().split()
    target_pos = None
    for i, word in enumerate(sent):
        if target_exp.match(word) != None:
            target_pos = i
            if word == '[]':
                word = None
            else:
                word = word[1:-1]
            sent[i] = word
    return sent, target_pos


def mult_sim(w, target_v, context_v):
    target_similarity = w.dot(target_v)
    target_similarity[target_similarity<0] = 0.0
    context_similarity = w.dot(context_v)
    context_similarity[context_similarity<0] = 0.0
    return (target_similarity * context_similarity)


n_result = 10  # number of search result to show
gpu = -1 # todo: make this work with gpu

if gpu >= 0:
    cuda.check_cuda_available()
    cuda.get_device(gpu).use()
xp = cuda.cupy if gpu >= 0 else numpy

model_reader = ModelReader("/Users/adilrhoulam/Downloads/context2vec.ukwac.model.\
package/context2vec.ukwac.model.params")
w = model_reader.w
word2index = model_reader.word2index
index2word = model_reader.index2word
model = model_reader.model

def find_most_similar(line,n_result):
        sent, target_pos = parse_input(line)
        if target_pos == None:
            raise ParseException("Can't find the target position.")

        if sent[target_pos] == None:
            target_v = None
        elif sent[target_pos] not in word2index:
            raise ParseException("Target word is out of vocabulary.")
        else:
            target_v = w[word2index[sent[target_pos]]]
        if len(sent) > 1:
            context_v = model.context2vec(sent, target_pos)
            context_v = context_v / xp.sqrt((context_v * context_v).sum())
        else:
            context_v = None

        if target_v is not None and context_v is not None:
            similarity = mult_sim(w, target_v, context_v)
        else:
            if target_v is not None:
                v = target_v
            elif context_v is not None:
                v = context_v
            else:
                raise ParseException("Can't find a target nor context.")
            similarity = (w.dot(v)+1.0)/2 # Cosine similarity can be negative, mapping similari
ty to [0,1]

        count = 0
        resultat = []
        for i in (-similarity).argsort():
            if numpy.isnan(similarity[i]):
                continue
            resultat.append([index2word[i],similarity[i]])
            #print('{0}: {1}'.format(index2word[i], similarity[i]))
            count += 1
            if count == n_result:
                break
        return resultat
```

```
/anaconda2/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the
second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it wil
l be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

```
Reading config file: /Users/adilrhoulam/Downloads/context2vec.ukwac.model.package/context2ve
c.ukwac.model.params
Config:  {'config_path': '/Users/adilrhoulam/Downloads/context2vec.ukwac.model.package/', 'm
odel_file': 'context2vec.ukwac.model', 'deep': 'yes', 'drop_ratio': '0.0', 'words_file': 'co
ntext2vec.ukwac.words.targets', 'unit': '300'}
```

In [10]:
```python
import re
from collections import Counter

words = re.findall(r'\w+', open('/Users/adilrhoulam/Downloads/words.txt').read().lower())

Dict = Counter(words)

def known(words):
    return set(w for w in words if w in Dict)

def candidates(word):
    return (known([word]) or known(edits1(word)) or known(edits2(word)) or [word])

def P(word, N=sum(Dict.values())):
    return Dict[word] / N


def edits1(word):
    "All edits that are one edit away from `word`."
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)]
    deletes    = [L + R[1:]               for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in letters]
    inserts    = [L + c + R               for L, R in splits for c in letters]
    return set(deletes + transposes + replaces + inserts)

def edits2(word):
    "All edits that are two edits away from `word`."
    return (e2 for e1 in edits1(word) for e2 in edits1(e1))

def edit_distance(s1,s2):
    m = np.zeros((len(s1),len(s2)),dtype=int)
    m[:,0] = np.arange(len(s1))
    m[0,:] = np.arange(len(s2))
    for i in range(1,len(s1)):
        for j in range(1,len(s2)):
            if s1[i]==s2[j]:
                m[i,j] = min(m[i-1,j]+1,m[i,j-1]+1,m[i-1,j-1])
            else:
                m[i,j] = min(m[i-1,j]+1,m[i,j-1]+1,m[i-1,j-1]+1)
    return m[len(s1)-1,len(s2)-1]
    #return m[len(s1)-1,len(s2)-1]

def correction_context(word,candidates):
    edit1 = []
    edit2 = []
    for i in range(len(candidates)):
        dist = edit_distance(word,candidates[i][0])
        if dist<=1:
            edit1.append(candidates[i][0])
        if dist == 2:
            edit2.append(candidates[i][0])
    if len(edit1)==0:
        if len(edit2)==0:
            return correction_spelling(word)
        return edit2[0]
    return edit1[0]

def correction_spelling(word):
    return max(candidates(word), key=P)
```

In [11]:
```python
path = '/Users/adilrhoulam/Downloads/CorpusBataclan_en.1M.raw.txt'
def parsefile(path,samples):
    with open(path) as f:
        lines = f.readlines()
    sentences = []
    for i in range(samples):
        sentences.append(" ".join(filter(lambda x: x[0]!='@' and x[0]!='#' and \
                                x[:5]!='https' and x[0:2] != 'RT' , lines[i].split())))
    for i in range(len(sentences)):
        sentences[i] = ''.join([x.lower() for x in sentences[i] if ord(x) < 128])

    tokens_1 = []
    for i in range(len(sentences)):

        tokens_1.append(nltk.word_tokenize(sentences[i]))

    nouns_1 = [[] for i in range(len(tokens_1))]
    nouns_2 = [[] for i in range(len(tokens_1))]
    for i in range(len(tokens_1)):
        for j in range(len(tokens_1[i])):
            nouns_1[i].append("".join([x.lower() for x in tokens_1[i][j] if x.isalpha()]))
            nouns_2[i].append("".join([x.lower() for x in tokens_1[i][j] if x.isalpha()]))
    return [tokens_1,nouns_1,nouns_2]

tokens_1,nouns_1,nouns_2 = parsefile(path,50000)
```

In [12]:
```python
def text_to_correct(nouns_1,nouns_2,tokens_1):
    corrected = nouns_2
    for i in range(len(nouns_1)):
        if i%100==0:
            print('i=',i)
        for j in range(len(nouns_1[i])):
            if nouns_1[i][j] in Dict and len(nouns_1[i][j])>1:
                corrected[i][j] = nouns_1[i][j]
                continue
            elif len(nouns_1[i][j])==0:
                corrected[i][j]= tokens_1[i][j]
            else:
                word = nouns_1[i][j]
                line = ''
                for p in range(0,j):
                    if p==0:
                        line = line + nouns_1[i][p]
                    else:
                        line = line + ' ' + nouns_1[i][p]
                if len(line)==0:
                    line+='[]'
                else:
                    line = line+' '+'[]'
                for p in range(j+1,len(nouns_1[i])):
                    line=line+' '+nouns_1[i][p]
                if "".join(line.split())=='[]':
                    corrected[i][j] = correction_spelling(word)
                else:
                    candidates_ = find_most_similar(line,10)
                    corrected[i][j] = correction_context(word,candidates_)
    return corrected

corrected = text_to_correct(nouns_1,nouns_2)
```

```
('i=', 0)
('i=', 100)
('i=', 200)
('i=', 300)
('i=', 400)
('i=', 500)
('i=', 600)
('i=', 700)
('i=', 800)
('i=', 900)
('i=', 1000)
('i=', 1100)
('i=', 1200)
('i=', 1300)
('i=', 1400)
('i=', 1500)
('i=', 1600)
('i=', 1700)
('i=', 1800)
('i=', 1900)
('i=', 2000)
('i=', 2100)
('i=', 2200)
('i=', 2300)
('i=', 2400)
('i=', 2500)
('i=', 2600)
('i=', 2700)
('i=', 2800)
('i=', 2900)
('i=', 3000)
('i=', 3100)
('i=', 3200)
('i=', 3300)
('i=', 3400)
('i=', 3500)
('i=', 3600)
('i=', 3700)
('i=', 3800)
('i=', 3900)
('i=', 4000)
('i=', 4100)
('i=', 4200)
('i=', 4300)
('i=', 4400)
('i=', 4500)
('i=', 4600)
('i=', 4700)
('i=', 4800)
('i=', 4900)
('i=', 5000)
('i=', 5100)
('i=', 5200)
('i=', 5300)
('i=', 5400)
('i=', 5500)
('i=', 5600)
('i=', 5700)
('i=', 5800)
('i=', 5900)
('i=', 6000)
('i=', 6100)
('i=', 6200)
('i=', 6300)
('i=', 6400)
('i=', 6500)
('i=', 6600)
('i=', 6700)
('i=', 6800)
('i=', 6900)
('i=', 7000)
('i=', 7100)
('i=', 7200)
('i=', 7300)
('i=', 7400)
('i=', 7500)
('i=', 7600)
('i=', 7700)
('i=', 7800)
('i=', 7900)
('i=', 8000)
('i=', 8100)
('i=', 8200)
('i=', 8300)
('i=', 8400)
('i=', 8500)
('i=', 8600)
('i=', 8700)
('i=', 8800)
('i=', 8900)
('i=', 9000)
('i=', 9100)
('i=', 9200)
('i=', 9300)
('i=', 9400)
('i=', 9500)
('i=', 9600)
('i=', 9700)
('i=', 9800)
('i=', 9900)
('i=', 10000)
('i=', 10100)
('i=', 10200)
('i=', 10300)
('i=', 10400)
('i=', 10500)
('i=', 10600)
('i=', 10700)
('i=', 10800)
('i=', 10900)
('i=', 11000)
('i=', 11100)
('i=', 11200)
('i=', 11300)
('i=', 11400)
('i=', 11500)
('i=', 11600)
('i=', 11700)
('i=', 11800)
('i=', 11900)
('i=', 12000)
('i=', 12100)
('i=', 12200)
('i=', 12300)
('i=', 12400)
('i=', 12500)
('i=', 12600)
('i=', 12700)
('i=', 12800)
('i=', 12900)
('i=', 13000)
('i=', 13100)
('i=', 13200)
('i=', 13300)
('i=', 13400)
('i=', 13500)
('i=', 13600)
('i=', 13700)
('i=', 13800)
('i=', 13900)
('i=', 14000)
('i=', 14100)
('i=', 14200)
('i=', 14300)
('i=', 14400)
('i=', 14500)
('i=', 14600)
('i=', 14700)
('i=', 14800)
('i=', 14900)
('i=', 15000)
('i=', 15100)
('i=', 15200)
('i=', 15300)
('i=', 15400)
('i=', 15500)
('i=', 15600)
('i=', 15700)
('i=', 15800)
('i=', 15900)
('i=', 16000)
('i=', 16100)
('i=', 16200)
('i=', 16300)
('i=', 16400)
('i=', 16500)
('i=', 16600)
('i=', 16700)
('i=', 16800)
('i=', 16900)
('i=', 17000)
('i=', 17100)
('i=', 17200)
('i=', 17300)
('i=', 17400)
('i=', 17500)
('i=', 17600)
('i=', 17700)
('i=', 17800)
('i=', 17900)
('i=', 18000)
('i=', 18100)
('i=', 18200)
('i=', 18300)
('i=', 18400)
('i=', 18500)
('i=', 18600)
('i=', 18700)
('i=', 18800)
('i=', 18900)
('i=', 19000)
('i=', 19100)
('i=', 19200)
('i=', 19300)
('i=', 19400)
('i=', 19500)
('i=', 19600)
('i=', 19700)
('i=', 19800)
('i=', 19900)
('i=', 20000)
('i=', 20100)
('i=', 20200)
('i=', 20300)
('i=', 20400)
('i=', 20500)
('i=', 20600)
('i=', 20700)
('i=', 20800)
('i=', 20900)
('i=', 21000)
('i=', 21100)
('i=', 21200)
('i=', 21300)
('i=', 21400)
('i=', 21500)
('i=', 21600)
('i=', 21700)
('i=', 21800)
('i=', 21900)
('i=', 22000)
('i=', 22100)
('i=', 22200)
('i=', 22300)
('i=', 22400)
('i=', 22500)
('i=', 22600)
('i=', 22700)
('i=', 22800)
('i=', 22900)
('i=', 23000)
('i=', 23100)
('i=', 23200)
('i=', 23300)
('i=', 23400)
('i=', 23500)
('i=', 23600)
('i=', 23700)
('i=', 23800)
('i=', 23900)
('i=', 24000)
('i=', 24100)
('i=', 24200)
('i=', 24300)
('i=', 24400)
('i=', 24500)
('i=', 24600)
('i=', 24700)
('i=', 24800)
('i=', 24900)
('i=', 25000)
('i=', 25100)
('i=', 25200)
('i=', 25300)
('i=', 25400)
('i=', 25500)
('i=', 25600)
('i=', 25700)
('i=', 25800)
('i=', 25900)
('i=', 26000)
('i=', 26100)
('i=', 26200)
('i=', 26300)
('i=', 26400)
('i=', 26500)
('i=', 26600)
('i=', 26700)
('i=', 26800)
('i=', 26900)
('i=', 27000)
('i=', 27100)
('i=', 27200)
('i=', 27300)
('i=', 27400)
('i=', 27500)
('i=', 27600)
('i=', 27700)
('i=', 27800)
('i=', 27900)
('i=', 28000)
('i=', 28100)
('i=', 28200)
('i=', 28300)
('i=', 28400)
('i=', 28500)
('i=', 28600)
('i=', 28700)
('i=', 28800)
('i=', 28900)
('i=', 29000)
('i=', 29100)
('i=', 29200)
('i=', 29300)
('i=', 29400)
('i=', 29500)
('i=', 29600)
('i=', 29700)
('i=', 29800)
('i=', 29900)
('i=', 30000)
('i=', 30100)
('i=', 30200)
('i=', 30300)
('i=', 30400)
('i=', 30500)
('i=', 30600)
('i=', 30700)
('i=', 30800)
('i=', 30900)
('i=', 31000)
('i=', 31100)
('i=', 31200)
('i=', 31300)
('i=', 31400)
('i=', 31500)
('i=', 31600)
('i=', 31700)
('i=', 31800)
('i=', 31900)
('i=', 32000)
('i=', 32100)
('i=', 32200)
('i=', 32300)
('i=', 32400)
('i=', 32500)
('i=', 32600)
('i=', 32700)
('i=', 32800)
('i=', 32900)
('i=', 33000)
('i=', 33100)
('i=', 33200)
('i=', 33300)
('i=', 33400)
('i=', 33500)
('i=', 33600)
('i=', 33700)
('i=', 33800)
('i=', 33900)
('i=', 34000)
('i=', 34100)
('i=', 34200)
('i=', 34300)
('i=', 34400)
('i=', 34500)
('i=', 34600)
('i=', 34700)
('i=', 34800)
('i=', 34900)
('i=', 35000)
('i=', 35100)
('i=', 35200)
('i=', 35300)
('i=', 35400)
('i=', 35500)
('i=', 35600)
('i=', 35700)
('i=', 35800)
('i=', 35900)
('i=', 36000)
('i=', 36100)
('i=', 36200)
('i=', 36300)
('i=', 36400)
('i=', 36500)
('i=', 36600)
('i=', 36700)
('i=', 36800)
('i=', 36900)
('i=', 37000)
('i=', 37100)
('i=', 37200)
('i=', 37300)
('i=', 37400)
('i=', 37500)
('i=', 37600)
('i=', 37700)
('i=', 37800)
('i=', 37900)
('i=', 38000)
('i=', 38100)
('i=', 38200)
('i=', 38300)
('i=', 38400)
('i=', 38500)
('i=', 38600)
('i=', 38700)
('i=', 38800)
('i=', 38900)
('i=', 39000)
('i=', 39100)
('i=', 39200)
('i=', 39300)
('i=', 39400)
('i=', 39500)
('i=', 39600)
('i=', 39700)
('i=', 39800)
('i=', 39900)
('i=', 40000)
('i=', 40100)
('i=', 40200)
('i=', 40300)
('i=', 40400)
('i=', 40500)
('i=', 40600)
('i=', 40700)
('i=', 40800)
('i=', 40900)
('i=', 41000)
('i=', 41100)
('i=', 41200)
('i=', 41300)
('i=', 41400)
('i=', 41500)
('i=', 41600)
('i=', 41700)
('i=', 41800)
('i=', 41900)
('i=', 42000)
('i=', 42100)
('i=', 42200)
('i=', 42300)
('i=', 42400)
('i=', 42500)
('i=', 42600)
('i=', 42700)
('i=', 42800)
('i=', 42900)
('i=', 43000)
('i=', 43100)
('i=', 43200)
('i=', 43300)
('i=', 43400)
('i=', 43500)
('i=', 43600)
('i=', 43700)
('i=', 43800)
('i=', 43900)
('i=', 44000)
('i=', 44100)
('i=', 44200)
('i=', 44300)
('i=', 44400)
('i=', 44500)
('i=', 44600)
('i=', 44700)
('i=', 44800)
('i=', 44900)
('i=', 45000)
('i=', 45100)
('i=', 45200)
('i=', 45300)
('i=', 45400)
('i=', 45500)
('i=', 45600)
('i=', 45700)
('i=', 45800)
('i=', 45900)
('i=', 46000)
('i=', 46100)
('i=', 46200)
('i=', 46300)
('i=', 46400)
('i=', 46500)
('i=', 46600)
('i=', 46700)
('i=', 46800)
('i=', 46900)
('i=', 47000)
('i=', 47100)
('i=', 47200)
('i=', 47300)
('i=', 47400)
('i=', 47500)
('i=', 47600)
('i=', 47700)
('i=', 47800)
('i=', 47900)
('i=', 48000)
('i=', 48100)
('i=', 48200)
('i=', 48300)
('i=', 48400)
('i=', 48500)
('i=', 48600)
('i=', 48700)
('i=', 48800)
('i=', 48900)
('i=', 49000)
('i=', 49100)
('i=', 49200)
('i=', 49300)
('i=', 49400)
('i=', 49500)
('i=', 49600)
('i=', 49700)
('i=', 49800)
('i=', 49900)
```

In [18]:
```python
path = '/Users/adilrhoulam/Desktop/test.txt'
def join_text(tokens,path):
    thefile = open(path, 'w')
    lines = [[] for i in range(len(tokens))]
    for i in range(len(tokens)):
        lines[i].append(" ".join(tokens[i]))
        thefile.write("%s\n" % lines[i])
    return lines

lines = join_text(corrected,path)
```