1. python安装

windows环境下直接下载相应版本安装即可，linux下查看默认版本，升级win2.7+或者3.4+

1. 安装pip

Windows下载后解压到python库下执行 python setup.py install

Linux下下载文件wget <https://bootstrap.pypa.io/get-pip.py>，python get-pip.py

1. 安装numpy包，pip install numpy
2. 安装ntlk。pip install ntlk
3. 下载corenlp包。
4. 分词实现。

NLTK 与 Stanford NLP

NLTK 是一款著名的 Python 自然语言处理(Natural Language Processing, NLP)工具包，在其收集的大量公开数据集、模型上提供了全面、易用的接口，涵盖了分词、词性标注(Part-Of-Speech tag, POS-tag)、命名实体识别(Named Entity Recognition, NER)、句法分析(Syntactic Parse)等各项 NLP 领域的功能。

Stanford NLP 是由斯坦福大学的 NLP 小组开源的 Java 实现的 NLP 工具包，同样对 NLP 领域的各个问题提供了解决办法。

斯坦福大学的 NLP 小组是世界知名的研究小组，如果能将 NLTK 和 Stanford NLP 这两个工具包结合起来使用，那自然是极好的！在 2004 年 Steve Bird 在 NLTK 中加上了对 Stanford NLP 工具包的支持，通过调用外部的 jar 文件来使用 Stanford NLP 工具包的功能。

现在的 NLTK 中，通过封装提供了 Stanford NLP 中的以下几个功能:

分词

词性标注

命名实体识别

句法分析，依存句法分析

Nltk封装了以下几个类

1. 分词: StanfordSegmenter
2. 词性标注: StanfordPOSTagger
3. 命名实体识别: StanfordNERTagger
4. 句法分析: StanfordParser
5. 依存句法分析: StanfordDependencyParser, StanfordNeuralDependencyParser

下面是对文本分词的程序代码：

分词: StanfordSegmenter  
# coding: utf-8

from nltk.tokenize.stanford\_segmenter import StanfordSegmenter

from nltk.tag import StanfordPOSTagger

segmenter = StanfordSegmenter(

java\_class='edu.stanford.nlp.ie.crf.CRFClassifier',

path\_to\_slf4j="/home/lcy/stanford-segmenter/slf4j-api.jar",

path\_to\_jar="/home/lcy/stanford-segmenter/stanford-segmenter.jar",

path\_to\_sihan\_corpora\_dict="/home/lcy/stanford-segmenter/data/",

path\_to\_model="/home/lcy/stanford-segmenter/data/pku.gz",

path\_to\_dict="/home/lcy/stanford-segmenter/data/dict-chris6.ser.gz"

)

f = open("wiki.zh.text", "r") # 读取文本

string = f.read().decode("utf-8")

result = segmenter.segment(string)

f = open("fenci.txt", "w") # 将结果保存到另一个文档中

f.write(result.encode("utf-8"))

f.close()

7、安装gensim包，pip install gensim

Gensim包含以下模型

（- LSI - LDA - HDP - DTM - DIM - TF-IDF - word2vec、paragraph2vec）

8、训练词向量模型

from gensim.models import word2vec

import logging

logging.basicConfig(format='%(asctime)s:%(levelname)s: %(message)s', level=logging.INFO)

sentences = word2vec.Text8Corpus(u"fenci\_wiki.txt") # 加载分词后的语料

model = word2vec.Word2Vec(sentences, size=200, sg=1)

model.save(u"wiki.model") #保存模型

9、使用训练好的模型

# coding: utf-8

from gensim.models import word2vec

import logging

logging.basicConfig(format='%(asctime)s:%(levelname)s: %(message)s', level=logging.INFO)

model = word2vec.Word2Vec.load("wiki.model") # 加载模型

y2 = model.most\_similar(u"保险", topn=20) # 20个与“保险”最相关的词

print u"和【保险】最相关的词有：\n"

for item in y2:

print item[0], item[1]

try:

y1 = model.similarity(u"中国", u"韩国") # 两个词的相似度

except KeyError:

y1 = 0

print u"【中国】和【韩国】的相似度为：", y1

# 计算某个词的相关词列表

print model[u'中国'] # 打印词的向量值