HarvestText

Release 0.8.2.1

Sep 03, 2023

Contents:

本文档目前记录了部分函数的参数含义，具体例子请见项目主页：https://github.com/blmoistawinde/HarvestText

CHAPTER 1

harvesttext package

1.1 Submodules

1.2 harvesttext.download_utils module

class harvesttext.download_utils.**RemoteFileMetadata**(*filename*, *url*, *checksum*)

Bases: tuple

checksum

Alias for field number 2

filename

Alias for field number 0

url

Alias for field number 1

harvesttext.download_utils.**check_download_resource**(*remote*, *use_proxy=False*, *proxies=None*)

harvesttext.download_utils.**clear_data_home**(*data_home=None*)

Delete all the content of the data home cache. Parameters ———- data_home : str | None

The path to data dir.

harvesttext.download_utils.**download**(*remote*, *file_path=None*, *use_proxy=False*, *proxies={'http': 'socks5h://127.0.0.1:1080', 'https': 'socks5h://127.0.0.1:1080'}*)

harvesttext.download_utils.**get_data_home**(*data_home=None*)

Return the path of the scikit-learn data dir. This folder is used by some large dataset loaders to avoid downloading the data several times. By default the data dir is set to a folder named 'scikit_learn_data' in the user home folder. Alternatively, it can be set by the 'SCIKIT_LEARN_DATA' environment variable or programmatically by giving an explicit folder path. The '~' symbol is expanded to the user home folder. If the folder does not already exist, it is automatically created. Parameters ———- data_home : str | None

The path to data dir.

1.3 harvesttext.ent_network module

class harvesttext.ent_network.**EntNetworkMixin**

Bases: object

实体网络模块：- 根据实体在文档中的共现关系

- 建立全局社交网络
- 建立以某一个实体为中心的社交网络

build_entity_ego_graph(*docs*, *word*, *min_freq=0*, *other_min_freq=-1*, *inv_index={}*, *used_types=[]*)

Entity only version of build_word_ego_graph()

build_entity_graph(*docs*, *min_freq=0*, *inv_index={}*, *used_types=[]*)

build_word_ego_graph(*docs*, *word*, *standard_name=True*, *min_freq=0*, *other_min_freq=-1*, *stopwords=None*)

根据文本和指定限定词，获得以限定词为中心的各词语的关系。限定词可以是一个特定的方面（衣食住行这类文档），这样就可以从词语中心图中获得关于这个方面的简要信息

Parameters

- **docs** – 文本的列表
- **word** – 限定词
- **standard_name** – 把所有实体的指称化为标准实体名
- **stopwords** – 需要过滤的停用词
- **min_freq** – 作为边加入到图中的与中心词最小共现次数，用于筛掉可能过多的边
- **other_min_freq** – 中心词以外词语关系的最小共现次数

Returns G（networxX 中的 Graph）

1.4 harvesttext.ent_retrieve module

class harvesttext.ent_retrieve.**EntRetrieveMixin**

Bases: object

实体检索模块: - 基于倒排索引快速检索包括某个实体的文档，以及统计出现某实体的文档数目

build_index(*docs*, *with_entity=True*, *with_type=True*)

get_entity_counts(*docs*, *inv_index*, *used_type=[]*)

search_entity(*query*, *docs*, *inv_index*)

1.5 harvesttext.harvesttext module

class harvesttext.harvesttext.**HarvestText**(*standard_name=False*, *language='zh_CN'*)

Bases: *harvesttext.ent_network.EntNetworkMixin*, *harvesttext.ent_retrieve.EntRetrieveMixin*, *harvesttext.parsing.ParsingMixin*, *harvesttext.sentiment.SentimentMixin*, *harvesttext.summary.SummaryMixin*, *harvesttext.word_discover.WordDiscoverMixin*

主模块：- 主要保留了与实体分词、分句，预处理相关的代码 - 还有存取、状态管理等基础代码 - 其他功能在各个 mixin 里面 - 主模块的功能是会被各个子模块最频繁调用的，也体现了本库以实体为核心，基于实体展开分析或改进算法的理念

add_entities(*entity_mention_dict=None*, *entity_type_dict=None*, *override=False*, *load_path=None*)

登录的实体信息到 ht，或者从 save_entities 保存的文件中读取（如果指定了 load_path）

Parameters

- **entity_mention_dict** – dict, {entity:[mentions]} 格式，
- **entity_type_dict** – dict, {entity:entity_type} 格式，
- **override** – bool, 是否覆盖已登录实体，默认 False
- **load_path** – str, 要读取的文件路径（默认不使用）

Returns None

add_new_entity(*entity0*, *mention0=None*, *type0='添加词'*)

add_new_mentions(*entity_mention_dict*)

add_new_words(*new_words*)

add_typed_words(*type_word_dict*)

build_trie(*new_word*, *entity*, *entity_type*)

check_prepared()

choose_from(*surface0*, *entity_types*)

choose_from_multi_mentions(*mention_cands*, *sent=''*)

clean_text(*text*, *remove_url=True*, *email=True*, *weibo_at=True*, *stop_terms=('转发微博',)*, *emoji=True*, *weibo_topic=False*, *markdown_hyperlink=True*, *deduplicate_space=True*, *norm_url=False*, *norm_html=False*, *to_url=False*, *remove_puncts=False*, *remove_tags=True*, *t2s=False*, *expression_len=(1, 6)*, *linesep2space=False*, *custom_regex=None*)

进行各种文本清洗操作，微博中的特殊格式，网址，email，html 代码，等等

Parameters

- **text** – 输入文本
- **remove_url** – （默认使用）是否去除网址
- **email** – （默认使用）是否去除 email
- **weibo_at** – （默认使用）是否去除微博的 @ 相关文本
- **stop_terms** – 去除文本中的一些特定词语，默认参数为（“转发微博”,）
- **emoji** – （默认使用）去除 [] 包围的文本，一般是表情符号
- **weibo_topic** – （默认不使用）去除 ## 包围的文本，一般是微博话题
- **markdown_hyperlink** – （默认使用）将类似 markdown 超链接的格式“[文本内容](链接)”清洗为只剩下”文本内容”
- **deduplicate_space** – （默认使用）合并文本中间的多个空格为一个
- **norm_url** – （默认不使用）还原 URL 中的特殊字符为普通格式，如 (%20 转为空格)
- **norm_html** – （默认不使用）还原 HTML 中的特殊字符为普通格式，如 (转为空格)
- **to_url** – （默认不使用）将普通格式的字符转为还原 URL 中的特殊字符，用于请求，如 (空格转为%20)
- **remove_puncts** – （默认不使用）移除所有标点符号
- **remove_tags** – （默认使用）移除所有 html 块
- **t2s** – （默认不使用）繁体字转中文
- **expression_len** – 假设表情的表情长度范围，不在范围内的文本认为不是表情，不加以清洗，如 [加上特别番外荞麦花开时共五册]。设置为 None 则没有限制
- **linesep2space** – （默认不使用）把换行符转换成空格
- **custom_regex** – （默认 None）一个正则表达式或一个列表的正则表达式，会优先根据这些表达式将对应内容替换为空

Returns 清洗后的文本

clear()

cut_sentences(*para*, *drop_empty_line=True*, *strip=True*, *deduplicate=False*)

Parameters

- **para** – 输入文本
- **drop_empty_line** – 是否丢弃空行
- **strip** – 是否对每一句话做一次 strip
- **deduplicate** – 是否对连续标点去重，帮助对连续标点结尾的句子分句

Returns sentences: list of str

decoref(*sent*, *entities_info*)

deprepare()

dig_trie(*sent*, *l*)

entity_linking(*sent*, *pinyin_tolerance=None*, *char_tolerance=None*, *keep_all=False*, *with_ch_pos=False*)

Parameters

- **sent** – 句子/文本
- **pinyin_tolerance** – {None, 0, 1} 搜索拼音相同 (取 0 时) 或者差别只有一个 (取 1 时) 的候选词链接到现有实体，默认不使用 (None)
- **char_tolerance** – {None, 1} 搜索字符只差 1 个的候选词 (取 1 时) 链接到现有实体，默认不使用 (None)
- **keep_all** – if True, keep all the possibilities of linked entities
- **with_ch_pos** – if True, also returns ch_pos

Returns entities_info：依存弧, 列表中的列表。if not keep_all: [([l, r], (entity, type)) for each linked mention m] else: [([l, r], set((entity, type) for each possible entity of m)) for each linked mention m] ch_pos: 每个字符对应词语的词性标注（不考虑登录的实体，可用来过滤实体，比如去掉都由名词组成的实体，有可能是错误链接）

get_linking_mention_candidates(*sent*, *pinyin_tolerance=None*, *char_tolerance=None*)

get_pinyin_correct_candidates(*word*, *tolerance=1*)

hanlp_prepare()

load_entities(*load_path='./ht_entities.txt'*, *override=True*)

从 save_entities 保存的文件读取实体信息

Parameters

- **load_path** – str, 读取路径（默认：./ht_entities.txt）
- **override** – bool, 是否重写已登录实体，默认 True

Returns None, 实体已登录到 ht 中

mention2entity(*mention*)

找到单个指称对应的实体

Parameters **mention** – 指称

Returns 如果存在对应实体，则返回（实体, 类型），否则返回 None, None

posseg(*sent*, *standard_name=False*, *stopwords=None*)

prepare()

remove_entity(*entity*)

remove_mention(*mention*)

save_entity_info(*save_path='./ht_entities.txt'*, *entity_mention_dict=None*, *entity_type_dict=None*)

保存 ht 已经登录的实体信息，或者外部提供的相同格式的信息，目前保存的信息包括 entity,mention,type.

如果不提供两个 dict 参数，则默认使用模型自身已登录信息，否则使用提供的对应 dict

格式:

entity|| 类别 mention|| 类别 mention|| 类别

entity|| 类别 mention|| 类别

每行第一个是实体名，其后都是对应的 mention 名，用一个空格分隔，每个名称后面都对应了其类别。

保存这个信息的目的是为了便于手动编辑和导入:

- 比如将某个 mention 作为独立的新 entity，只需剪切到某一行的开头，并再复制一份再后面作为 mention

Parameters

- **save_path** – str, 要保存的文件路径（默认: ./ht_entities.txt）
- **entity_mention_dict** – dict, {entity:[mentions]} 格式，
- **entity_type_dict** – dict, {entity:entity_type} 格式，

Returns None

search_word_trie(*word*, *tolerance=1*)

Parameters

- **word** –
- **tolerance** –

Returns

seg(*sent*, *standard_name=False*, *stopwords=None*, *return_sent=False*)

set_linking_strategy(*strategy*, *lastest_mention=None*, *entity_freq=None*, *type_freq=None*)

为实体链接设定一些简单策略，目前可选的有：’None’,’freq’,’latest’,’latest&freq’

‘None’：默认选择候选实体字典序第一个

‘freq’：对于单个字面值，选择其候选实体中之前出现最频繁的一个。对于多个重叠字面值，选择其中候选实体出现最频繁的一个进行连接【每个字面值已经确定唯一映射】。

‘latest’：对于单个字面值，如果在最近有可以确定的映射，就使用最近的映射。

‘latest’- 对于职称等作为代称的情况可能会比较有用。

比如” 经理” 可能代指很多人，但是第一次提到的时候应该会包括姓氏。我们就可以记忆这次信息，在后面用来消歧。

‘freq’- 单字面值例：’市长’+{‘A 市长’:5,’B 市长’:3} -> ‘A 市长’

重叠字面值例，’xx 市长江 yy’+{‘xx 市长’:5,’长江 yy’:3}+{‘市长’:’xx 市长’}+{‘长江’:’长江 yy’} -> ‘xx 市长’

Parameters

- **strategy** – 可选‘None’,’freq’,’latest’,’latest&freq’中的一个
- **lastest_mention** – dict, 用于’latest’, 预设
- **entity_freq** – dict, 用于’freq’, 预设某实体的优先级（词频）
- **type_freq** – dict, 用于’freq’, 预设类别所有实体的优先级（词频）

:return None

1.6 harvesttext.parsing module

class harvesttext.parsing.ParsingMixin

Bases: object

文本解析模块：- 依存句法分析 - 基于依存句法分析的三元组抽取 - 基于 Texttile 的文本自动分段算法

cut_paragraphs(*text*, *num_paras=None*, *block_sents=3*, *std_weight=0.5*, *align_boundary=True*, *stopwords='baidu'*, *remove_puncts=True*, *seq_chars=-1*, ***kwargs*)

Parameters

- **text** –
- **num_paras** – (默认为 None) 可以手动设置想要划分的段落数，也可以保留默认值 None，让算法自动确定
- **block_sents** – 算法的参数，将几句句子分为一个 block。一般越大，算法自动划分的段落越少
- **std_weight** – 算法的参数。一般越大，算法自动划分的段落越多
- **align_boundary** – 新划分的段落是否要与原有的换行处对齐
- **stopwords** – 字符串列表/元组/集合，或者’baidu’为默认百度停用词，在算法中引入的停用词，一般能够提升准确度
- **remove_puncts** – （默认为 True）是否在算法中去除标点符号，一般能够提升准确度
- **seq_chars** – （默认为-1）如果设置为 >=1 的值，则以包含这个数量的字符为基本单元，代替默认的句子。
- ****kwargs** – passed to ht.cut_sentences, like deduplicate

Returns

dependency_parse(*sent*, *standard_name=False*, *stopwords=None*)

依存句法分析，调用 pyhanlp 的接口，并且融入了 harvesttext 的实体识别机制。不保证高准确率。

Parameters

- **sent** –
- **standard_name** –
- **stopwords** –

Returns arcs：依存弧, 列表中的列表。[[词语 id, 词语字面值或实体名 (standard_name 控制), 词性，依存关系，依存子词语 id] for 每个词语]

triple_extraction(*sent*, *standard_name=False*, *stopwords=None*, *expand='all'*)

利用主谓宾等依存句法关系，找到句子中有意义的三元组。很多代码参考：https://github.com/liuhuanyong/EventTriplesExtraction 不保证高准确率。

Parameters

- **sent** –
- **standard_name** –
- **stopwords** –
- **expand** – 默认”all”：扩展所有主谓词，”exclude_entity”：不扩展已知实体，可以保留标准的实体名，用于链接。”None”：不扩展

Returns

1.7 harvesttext.resources module

`harvesttext.resources.get_baidu_stopwords()`

获得百度停用词列表来源，网上流传的版本: https://wenku.baidu.com/view/98c46383e53a580216fcfed9.html 包含了中英文常见词及部分标点符号

Returns stopwords: set of string

`harvesttext.resources.get_english_senti_lexicon(type='LH')`

获得英语情感词汇表

目前默认为来自这里的词汇表 https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon

If you use this list, please cite the following paper:

Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA,

Returns sent_dict = { "pos" :[words]," neg" :[words]}

`harvesttext.resources.get_jieba_dict(min_freq=0, max_freq=inf, with_pos=False, use_proxy=False, proxies=None)`

获得 jieba 自带的中文词语词频词典

Params min_freq 选取词语需要的最小词频

Params max_freq 选取词语允许的最大词频

Params with_pos 返回结果是否包括词性信息

Return if not with_pos, dict of {wd freq}, else, dict of {(wd, pos): freq}

`harvesttext.resources.get_nltk_en_stopwords()`

来自 nltk 的英语停用词

Returns stopwords: set of string

`harvesttext.resources.get_qh_sent_dict()`

获得参考褒贬义词典：褒贬义词典清华大学李军

此资源被用于以下论文中: Jun Li and Maosong Sun, Experimental Study on Sentiment Classification of Chinese Review using Machine Learning Techniques, in Proceding of IEEE NLPKE 2007 李军中文评论的褒贬义分类实验研究硕士论文清华大学 2008

Returns qh_sent_dict = { "pos" :[words]," neg" :[words]}

harvesttext.resources.**get_qh_typed_words**(*used_types=['IT', '动物', '医药', '历史人名', '地名', '成语', '法律', '财经', '食物']*)

THUOCL：清华大学开放中文词库 http://thuocl.thunlp.org/ IT 财经成语地名历史名人诗词医学饮食法律汽车动物

Parameters **used_types** –

Returns typed_words: 字典，键为类型，值为该类的词语组成的 set

harvesttext.resources.**get_sanguo**()

获得三国演义原文

Returns [“章节 1 文本”,”章节 2 文本”,…]

harvesttext.resources.**get_sanguo_entity_dict**()

获得三国演义中的人名、地名、势力名的知识库。自行搭建的简单版，一定有遗漏和错误，仅供参考使用

Returns entity_mention_dict,entity_type_dict

1.8 harvesttext.sentiment module

class harvesttext.sentiment.**SentimentMixin**

Bases: object

情感分析模块：- 基于 SO-PMI 的情感词典挖掘和情感分析算法

analyse_sent(*sent*, *avg=True*)

输入句子，输出其情感值，默认使用句子中，在情感词典中的词语的情感值的平均来计算

Parameters

- **sent** – string, 句子
- **avg** – (default True) 是否使用平均值计算句子情感值

Returns float 情感值 (if avg == True), 否则为词语情感值列表

build_sent_dict(*sents*, *method='PMI'*, *min_times=5*, *scale='None'*, *pos_seeds=None*, *neg_seeds=None*, *stopwords=None*)

利用种子词，构建情感词典

Parameters

- **sents** – list of string, 一般建议为句子，是计算共现 PMI 的基本单元
- **method** – “PMI”，使用的算法，目前仅支持 PMI
- **min_times** – int, 默认为 5，在所有句子中出现次数少于这个次数的词语将被过滤

- **scale** – { “None” ,” 0-1” ,” +-1” }, 默认为” None”，否则将对情感值进行变换若为” 0-1”，按照最大为 1，最小为 0 进行线性伸缩，0.5 未必是中性若为” +-1” ，在正负区间内分别伸缩，保留 0 作为中性的语义
- **pos_seeds** – list of string, 积极种子词，如不填写将默认采用清华情感词典
- **neg_seeds** – list of string, 消极种子词，如不填写将默认采用清华情感词典
- **stopwords** – list of string, stopwords 词，如不填写将不使用

Returns sent_dict: dict, 可以查询单个词语的情感值

1.9 harvesttext.summary module

class harvesttext.summary.**SummaryMixin**

Bases: object

文本摘要模块：- 基于 textrank+MMR 的无监督抽取式摘要方法

get_summary(*sents*, *topK=5*, *stopwords=None*, *with_importance=False*, *standard_name=True*, *maxlen=None*, *avoid_repeat=False*, *sim_func='default'*)

使用 Textrank 算法得到文本中的关键句

Parameters

- **sents** – str 句子列表
- **topK** – 选取几个句子, 如果设置了 maxlen，则优先考虑长度
- **stopwords** – 在算法中采用的停用词
- **with_importance** – 返回时是否包括算法得到的句子重要性
- **standard_name** – 如果有 entity_mention_list 的话，在算法中正规化实体名，一般有助于提升算法效果
- **maxlen** – 设置得到的摘要最长不超过多少字数，如果已经达到长度限制但未达到 topK 句也会停止
- **avoid_repeat** – 使用 MMR principle 惩罚与已经抽取的摘要重复的句子，避免重复
- **sim_func** – textrank 使用的相似度量函数，默认为基于词重叠的函数（原论文），也可以是任意一个接受两个字符串列表参数的函数

Returns 句子列表，或者 with_importance=True 时，(句子，分数）列表

1.10 harvesttext.word_discover module

class harvesttext.word_discover.WordDiscoverMixin

Bases: object

新词、关键词发现模块: - 基于凝聚度和左右熵的新词发现 - 基于模式的专有名词发现 - 命名实体识别 - 实验性质的实体别名发现算法

entity_discover(*text*, *return_count=False*, *method='NFL'*, *min_count=5*, *pinyin_tolerance=0*, ***kwargs*)

无监督地从较大量文本中发现实体的类别和多个同义 mention。建议对千句以上的文本来挖掘，并且文本的主题

效率：在测试环境下处理一个约 10000 句的时间大约是 20 秒。另一个约 200000 句的语料耗时 2 分半精度：算法准确率不高，但是可以初步聚类，建议先 save_entities 后，再进行手动进行调整，然后 load_entities 再用于进一步挖掘

ref paper: Mining Entity Synonyms with Efficient Neural Set Generation(https://arxiv.org/abs/1811.07032v1)

Parameters

- **text** – string or list of string
- **return_count** – (default False) 是否再返回每个 mention 的出现次数
- **method** – 使用的算法，目前可选 “NFL” (NER+Fasttext+Louvain+ 模式修复，基于语义和规则发现同义实体，但可能聚集过多错误实体)，“NERP” (NER+ 模式修复, 仅基于规则发现同义实体)
- **min_count** – (default 5) mininum freq of word to be included
- **pinyin_tolerance** – {None, 0, 1} 合并拼音相同 (取 0 时) 或者差别只有一个 (取 1 时) 的候选词到同一组实体，默认使用 (0)
- **kwargs** – 根据算法决定的参数，目前，“NERP” 不需要额外参数，而” NFL” 可接受的额外参数有：

  emb_dim: (default 50) fasttext embedding’ s dimensions

  threshold: (default 0.98) [比较敏感，调参重点]larger for more entities, threshold for add an edge between 2 entities if cos_dim exceeds

  ft_iters: (default 20) larger for more entities, num of iterations used by fasttext

  use_subword: (default True) whether to use fasttext’ s subword info

  min_n: (default 1) min length of used subword

  max_n: (default 4) max length of used subword

Returns entity_mention_dict, entity_type_dict

`extract_keywords`(*text*, *topK*, *with_score=False*, *min_word_len=2*, *stopwords='baidu'*, *allowPOS='default'*, *method='jieba_tfidf'*, ***kwargs*)

用各种算法抽取关键词（目前均为无监督），结合了 ht 的实体分词来提高准确率

目前支持的算法类型（及额外参数）：

- jieba_tfidf: （默认）jieba 自带的基于 tfidf 的关键词抽取算法，idf 统计信息来自于其语料库
- **textrank: 基于 textrank 的关键词抽取算法**
  - block_type: 默认" doc"。支持三种级别，" sent" ，"para" ，"doc"，每个 block 之间的临近词语不建立连边
  - window: 默认 2, 邻接的几个词语之内建立连边
  - weighted: 默认 False, 时候使用加权图计算 textrank
  - 构建词图时会过滤不符合 min_word_len, stopwords, allowPOS 要求的词语

Params text 从中挖掘关键词的文档

Params topK int, 从每个文档中抽取的关键词（最大）数量

Params with_score bool, 默认 False, 是否同时返回算法提供的分数（如果有的话）

Params min_word_len 默认 2, 被纳入关键词的词语不低于此长度

Parameters `stopwords` – 字符串列表/元组/集合，或者' baidu' 为默认百度停用词，在算法中引入的停用词，一般能够提升准确度

Params allowPOS iterable of str，关键词应当属于的词性，默认为" default" { 'n' , 'ns' , 'nr' , 'nt' , 'nz' , 'vn' , 'v' , 'an' , 'a' , 'i' } 以及已登录的实体词类型

Params method 选择用于抽取的算法，目前支持" jieba_tfidf" ，"tfidf" ，"textrank"

Params kwargs 其他算法专属参数

`find_entity_with_rule`(*text*, *rulesets=[]*, *add_to_dict=True*, *type0='添加词'*)

利用规则从分词结果中的词语找到实体，并可以赋予相应的类型再加入实体库

Parameters

- `text` – string, 一段文本
- `rulesets` – list of (tuple of rules or single rule) from match_patterns, list 中包含多个规则，满足其中一种规则的词就认为属于这个 type 而每种规则由 tuple 或单个条件 (pattern) 表示，一个词必须满足其中的一个或多个条件。
- `add_to_dict` – 是否把找到的结果直接加入词典
- `type0` – 赋予满足条件的词语的实体类型, 仅当 add_to_dict 时才有意义

Returns found_entities

named_entity_recognition(*sent*, *standard_name=False*, *return_posseg=False*)

利用 pyhanlp 的命名实体识别，找到句子中的（人名，地名，机构名，其他专名）实体。harvesttext 会预先链接已知实体

Parameters

- **sent** – string, 文本
- **standard_name** – bool, 是否把连接到的已登录转化为标准名
- **return_posseg** – bool, 是否返回包括命名实体识别的，带词性分词结果
- **book** – bool, 预先识别

Returns entity_type_dict: 发现的命名实体信息，字典 {实体名: 实体类型}（return_posseg=True 时）possegs: list of (单词, 词性)

word_discover(*doc*, *threshold_seeds=[]*, *auto_param=True*, *excluding_types=[]*, *excluding_words='baidu_stopwords'*, *max_word_len=5*, *min_freq=5e-05*, *min_entropy=1.4*, *min_aggregation=50*, *ent_threshold='both'*, *mem_saving=None*, *sort_by='freq'*, *exclude_number=True*)

新词发现，基于 http://www.matrix67.com/blog/archives/5044 实现及微调

Parameters

- **doc** – (string or list) 待进行新词发现的语料，如果是列表的话，就会自动用换行符拼接
- **threshold_seeds** – list of string, 设定能接受的“质量”最差的种子词，更差的词语将会在新词发现中被过滤
- **auto_param** – bool, 使用默认的算法参数
- **excluding_types** – list of str, 设定要过滤掉的特定词性或已经登录到 ht 的实体类别
- **excluding_words** – list of str, 设定要过滤掉的特定词
- **max_word_len** – 允许被发现的最长的新词长度
- **min_freq** – 被发现的新词，在给定文本中需要达到的最低频率
- **min_entropy** – 被发现的新词，在给定文本中需要达到的最低左右交叉熵
- **min_aggregation** – 被发现的新词，在给定文本中需要达到的最低凝聚度
- **ent_threshold** – “both”：(默认) 在使用左右交叉熵进行筛选时，两侧都必须超过阈值；“avg”：两侧的平均值达到阈值即可
- **mem_saving** – bool or None, 采用一些过滤手段来减少内存使用，但可能影响速度。如果不指定，对长文本自动打开，而对短文本不使用
- **sort_by** – 以下 string 之一：{‘freq’：词频，‘score’：综合分数，‘agg’：凝聚度} 按照特定指标对得到的词语信息排序，默认使用词频

- **exclude_number** – （默认 True）过滤发现的纯数字新词

Returns info: 包含新词作为 index, 以及对应各项指标的 DataFrame

1.11 Module contents

harvesttext.**loadHT**(*filename*)

harvesttext.**saveHT**(*htModel*, *filename*)

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

L

M

N

P

R

S

T

U

W