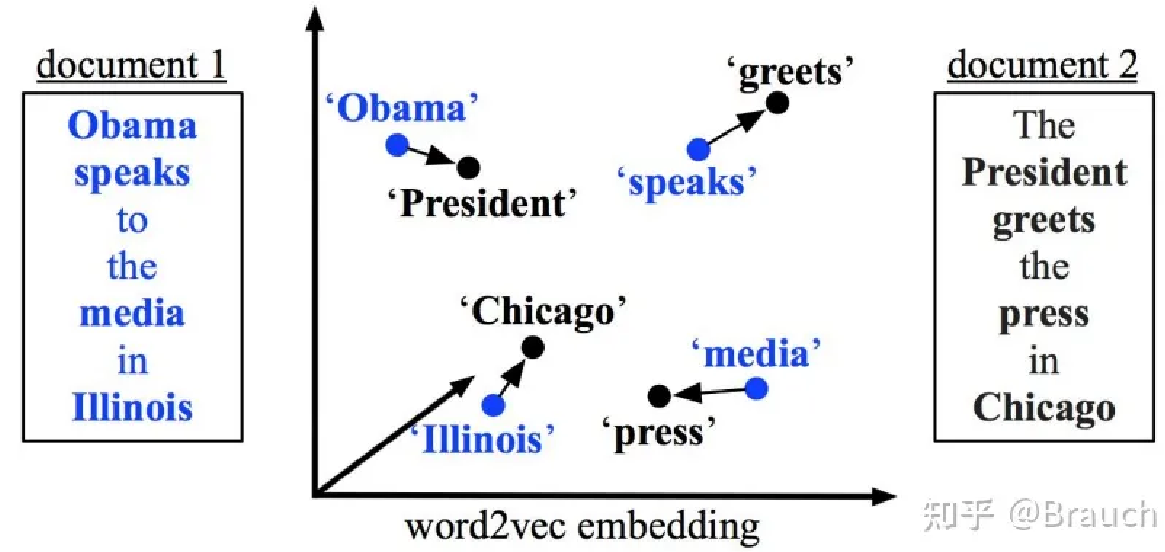
关于EMD和WMD

WMD(Word Mover's Distance）中文称作词移距离，是EMD(Earth Mover's Distance)在NLP领 域的延伸。

1.什么是EMD? 如果我们将分布想象为两个有一定存土量的土堆，那么EMD就是将一个土堆转换为另一个土堆所需的最小总工作量。工作量的定义是单位泥土的总量乘以它移动的距离。它是归一化的从一个分布变为另一个分布的最小代价, 可以用来测量两个分布(multi-dimensional distributions)之间的距离。

2.WMD。按照上EMD的概念，那么也可以计算从一个单词变为另一个单词的最小代价。 通过寻找两个文本之间所有词之间最小距离之和的配对来度量文本的语义相似度。如下图所示：



简单看：一般来说，可以计算出每个句子中（去掉stopwords后）最近单词之间的距离，就可以快速知道两个句子间的差距，但是这种情况下没有考虑词与词之间多多对应的关系，**需要将每个词进行一一对应，计算相似度，并且是有侧重地（分配的权重）计算相似度。**

WMD的优化公式如下：（线性优化问题，这个最小值就是文本之间的距离）

$$\min_{\mathbf{T}\geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j)$$

$$\text{subject to:} \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \quad \forall i \in \{1,\dots,n\} \qquad (1)$$

$$\sum_{i=1}^{n} \mathbf{T}_{ij} = d'_j \quad \forall j \in \{1,\dots,n\}.$$

$$d_i = \frac{c_i}{\sum_{j=1}^{n} c_j}$$

其中c(i,j) 表示单词ij之间的相似度计算，而Tij 表示对于这种相似度的权重。由于两个文本之间的单词两两之间都要进行相似度计算，可知 T是一个矩阵。

约束条件中： d 是normalized BOW：单词的重要程度与词的出现频率相关 （并且归一化），di 是指某个单词 i 在本文档d中出现的频率相关， di' 是文档d中的单词j分配到 d' 文档中的权重。这个地方就是一个**权重矩阵横向和纵向相加的约束**。举个例子：

> 文档1：我/在/我家/吃/我/做的/饭。
> [2,1,1,1,2,1,1] 计算d得出 [2,1,1,1,2,1,1]/9 '我'就是 2/9，这个权重分别分配到另外两个文档中去，假设权重分别是1/9 1/9，这两个值相加就是d的值，**这个权重如何分解是最优化求解出来的**
> 文档2：我在你家吃饭 文档3：你在我家吃饭
> 同样对于d'，也需要对每个词计算nBOW，而 d'中某 个词对应的被分配过来的权重加起来要等于这个词在 d'文档中的权重。

• 效果出色：充分利用了word2vec的领域迁移能力 • 无监督：不依赖标注数据，没有冷启动问题 • 模型简单：仅需要词向量的结果作为输入，没有任何超参数 • 可解释性：将问题转化成线性规划，有全局最优解 • 灵活性：可以人为干预词的重要性

• 词袋模型，没有保留语序信息 WMD认为『屡战屡败』和『屡败屡战』在语义上是完全一致的。 • 不能很好地处理词向量的OOV问题 由于词向量是离线训练的，应用于在线业务时会遇到OOV问题，用户query分 出来的词，有可能找不到对应的词向量。 • 处理否定词能力偏差 在训练好的词向量中，通常语义相反的词的词向量是比较相近的，这会导致语义相反的两个句子WMD距离很近。

实际效果

抛开算法设计与原理，决定WMD性能的重要一点就在于词向量的性能

找到了一个使用谷歌词向量的WMD实现

其计算单位是字符串，可能仍然需要先进行提取再计算

```
r"""
Word Mover's Distance
=====================

Demonstrates using Gensim's implemenation of the WMD.
实测，爬不到那个数据库，但是看起来还行，保留
"""

###############################################################################
# Word Mover's Distance (WMD) is a promising new tool in machine learning that
# allows us to submit a query and return the most relevant documents. This
# tutorial introduces WMD and shows how you can compute the WMD distance
# between two documents using ``wmdistance``.
#
```

```python
# WMD Basics
# ----------
#
# WMD enables us to assess the "distance" between two documents in a meaningful
# way even when they have no words in common. It uses `word2vec
# <http://rare-technologies.com/word2vec-tutorial/>`_ [4] vector embeddings of
# words. It been shown to outperform many of the state-of-the-art methods in
# k-nearest neighbors classification [3].
#
# WMD is illustrated below for two very similar sentences (illustration taken
# from `Vlad Niculae's blog
# <http://vene.ro/blog/word-movers-distance-in-python.html>`_). The sentences
# have no words in common, but by matching the relevant words, WMD is able to
# accurately measure the (dis)similarity between the two sentences. The method
# also uses the bag-of-words representation of the documents (simply put, the
# word's frequencies in the documents), noted as $d$ in the figure below. The
# intuition behind the method is that we find the minimum "traveling distance"
# between documents, in other words the most efficient way to "move" the
# distribution of document 1 to the distribution of document 2.
#

# Image from https://vene.ro/images/wmd-obama.png
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

###############################################################################
# This method was introduced in the article "From Word Embeddings To Document
# Distances" by Matt Kusner et al. (\ `link to PDF
# <http://jmlr.org/proceedings/papers/v37/kusnerb15.pdf>`_\ ). It is inspired
# by the "Earth Mover's Distance", and employs a solver of the "transportation
# problem".
#
# In this tutorial, we will learn how to use Gensim's WMD functionality, which
# consists of the ``wmdistance`` method for distance computation, and the
# ``WmdSimilarity`` class for corpus based similarity queries.
#
# .. Important::
#    If you use Gensim's WMD functionality, please consider citing [1] and [2].
#
# Computing the Word Mover's Distance
```

```python
# ------------------------------------
#
# To use WMD, you need some existing word embeddings.
# You could train your own Word2Vec model, but that is beyond the scope of this tutorial
# (check out :ref:`sphx_glr_auto_examples_tutorials_run_word2vec.py` if you're interested).
# For this tutorial, we'll be using an existing Word2Vec model.
#
# Let's take some sentences to compute the distance between.
#

# Initialize logging.
import logging

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

sentence_obama = 'Obama speaks to the media in Illinois'
sentence_president = 'The president greets the press in Chicago'

from nltk.corpus import stopwords
from nltk import download

download('stopwords')  # Download stopwords list.
stop_words = stopwords.words('english')


def preprocess(sentence):
    return [w for w in sentence.lower().split() if w not in stop_words]


sentence_obama = preprocess(sentence_obama)
sentence_president = preprocess(sentence_president)

import gensim.downloader as api


if __name__=='__main__':
    model = api.load('word2vec-google-news-300')
    distance = model.wmdistance(sentence_obama, sentence_president)
    print('distance = %.4f' % distance)
    sentence_orange = preprocess('Oranges are my favorite fruit')
    distance = model.wmdistance(sentence_obama, sentence_orange)
    print('distance = %.4f' % distance)

```