Keyword Extraction – Question 2

OPTION 1

Tagging for Keyword Extraction or Named Entity Recognition (NER) using models such as YAKE, BERT-derived models, spacy, or Google NLP API

Keyword extraction is defined as the task that automatically identifies a set of the terms that best describe the subject of document.

Automatic Keyword extraction algorithms used:

- Rapid Automatic Keyword Extraction (RAKE). Python implementations
- Gensim implementation of Text-Rank
- Yet Another Keyword Extractor (YAKE)

Dataset used:

NIPS Papers Dataset

Neural Information Processing Systems (NIPS) is one of the top machine learning conferences in the world.

This dataset includes the title, authors, abstracts, and extracted text for all NIPS papers to date (ranging from the first 1987 conference to the current 2016 conference).

PROCESS:

First, pre-processing the "paper-text" feature from the data which provides the full content of the paper, that will be used for the keyword extraction process.

Using the abstract of the paper provide in the data, we can perform any algorithm for keyword extraction, such as

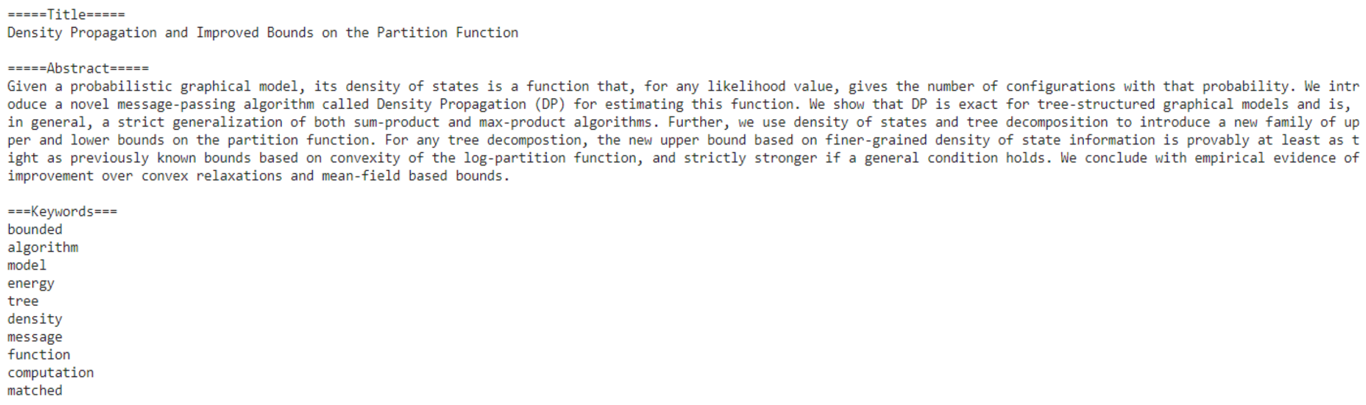
Gen-sim - It is a free Python library designed to automatically extract semantic topics from documents. The gen-sim implementation is based on the popular Text-Rank algorithm.

Rapid Automatic Keyword Extraction algorithm (RAKE) – It is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occur with other words in the text.

Yet Another Keyword Extractor (YAKE) - YAKE! is a light-weight unsupervised automatic keyword extraction method which rests on text statistical features extracted from single documents to select the most important keywords of a text.

Results & Observation:

By using the **Gensim implementation of Text-Rank summarization algorithm** the result received is,

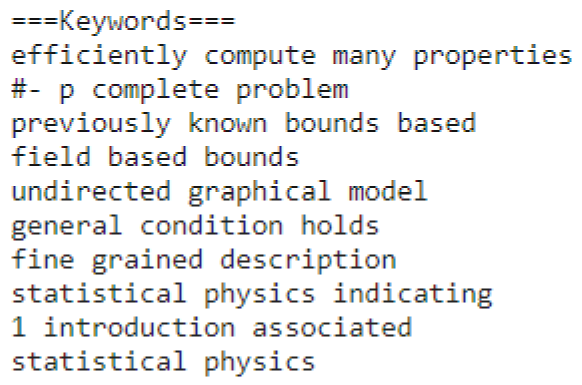


```
=====Title=====
Density Propagation and Improved Bounds on the Partition Function

=====Abstract=====
Given a probabilistic graphical model, its density of states is a function that, for any likelihood value, gives the number of configurations with that probability. We intr
oduce a novel message-passing algorithm called Density Propagation (DP) for estimating this function. We show that DP is exact for tree-structured graphical models and is,
in general, a strict generalization of both sum-product and max-product algorithms. Further, we use density of states and tree decomposition to introduce a new family of up
per and lower bounds on the partition function. For any tree decompostion, the new upper bound based on finer-grained density of state information is provably at least as t
ight as previously known bounds based on convexity of the log-partition function, and strictly stronger if a general condition holds. We conclude with empirical evidence of
improvement over convex relaxations and mean-field based bounds.

===Keywords===
bounded
algorithm
model
energy
tree
density
message
function
computation
matched
```

Observation:

The keywords provided from this algorithm it got to highlight the main point, but still miss valuable information. So, we use another method which is Rapid Automatic Keyword Extraction algorithm (RAKE) using NLTK

Rapid Automatic Keyword Extraction algorithm (RAKE) using NLTK

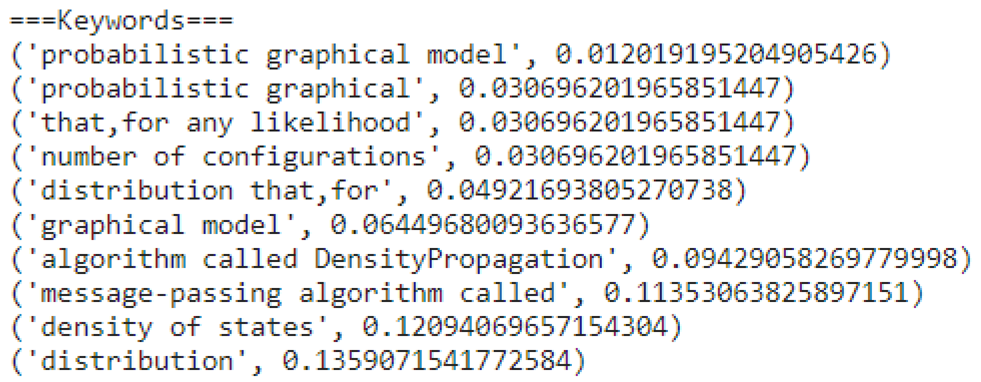


```
===Keywords===
efficiently compute many properties
#- p complete problem
previously known bounds based
field based bounds
undirected graphical model
general condition holds
fine grained description
statistical physics indicating
1 introduction associated
statistical physics
```

Observation:

This implementation works well on sentences, but it is not flexible enough for large text. However, those who are interested in RANK can expand the capabilities of this code to their needs. We will consider next options.

Yet Another Keyword Extractor (Yake)



```
===Keywords===
('probabilistic graphical model', 0.012019195204905426)
('probabilistic graphical', 0.030696201965851447)
('that,for any likelihood', 0.030696201965851447)
('number of configurations', 0.030696201965851447)
('distribution that,for', 0.04921693805270738)
('graphical model', 0.06449680093636577)
('algorithm called DensityPropagation', 0.09429058269779998)
('message-passing algorithm called', 0.11353063825897151)
('density of states', 0.12094069657154304)
('distribution', 0.1359071541772584)
```

Observation:

The results are better, but key phrases are repeated, and the text needs pre-processing to remove stop words.