PROJECT- KEYPHRASE EXTRACTION FROM SCIENTIFIC ARTICLES

IMPORTING LIBRABRIES

```
!pip install rake_nltk

from rake_nltk import Rake
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
Collecting rake_nltk
  Downloading rake_nltk-1.0.6-py3-none-any.whl (9.1 kB)
Requirement already satisfied: nltk<4.0.0,>=3.6.2 in /usr/local/lib/python3.10/dist-packages (from rake_nltk) (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk<4.0.0,>=3.6.2->rake_nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk<4.0.0,>=3.6.2->rake_nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk<4.0.0,>=3.6.2->rake_nltk) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk<4.0.0,>=3.6.2->rake_nltk) (4.66.2)
Installing collected packages: rake_nltk
Successfully installed rake_nltk-1.0.6
```

+ Code + Text

LOAD AND READ THE DATA

```
!pip install pymupdf
import fitz
file_path = '/content/scientific_article.pdf'
pdf_document = fitz.open(file_path)
pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)

    text = page.get_text("text")

    pdf_text += text
pdf_document.close()

print(pdf_text)
```

```
Streaming output truncated to the last 5000 lines.
article was coauthored by one of the coinvestigators involved
in this study (to avoid potential bias); one of the coinvesti-
gators involved in classifying article conclusions had previous
knowledge of the article’s sponsorship (also to avoid potential
bias): or both coinvestigators involved in classifying article
conclusions determined that all inclusion criteria were not
met. (A list of included articles is available from the authors
upon request.)
Classification of Article Conclusions
The study coordinator provided two coinvestigators (CBE
and DSL) with each article’s abstract and discussion/
conclusion section (as available). The coinvestigators were
given no information relating to the identity of the article
(e.g., journal, title, authors) or to financial sponsorship. When
electronic documents were available, a simple text file was
utilized. For articles without electronic versions, photocopies
were made that excluded or obscured any identifying
information. The coinvestigators classified article conclusions
independently and then met to resolve discrepancies, using
the categories outlined below.
Favorable—if both coinvestigators agreed that: (1) the
conclusions suggested beneficial health effects or absence of
expected adverse health effects, and (2) no statements were
made that cast the product in a negative light.
Unfavorable—if both coinvestigators agreed that: (1) the
conclusions suggested adverse health effects or absence of
expected beneficial health effects, and (2) no statements were
made that cast the product in a positive light.
Neutral—if the coinvestigators agreed that the conclusions
were neither favorable nor unfavorable, or if the coinvesti-
gators could not agree on classification.
Characterization of Financial Sponsorship
The study coordinator examined each article (and supple-
mental material, if relevant) in its entirety for information
about financial sponsorship. A coinvestigator (MG) was given
a list of all sponsors of each article linked to the type of
beverage under study (soft drinks, juice, or milk). The
coinvestigator was given no further information relating to
the identity of the article (e.g., journal, title, authors) or to its
methods or results.
The coinvestigator used generally available information,
obtained in part by Internet searches, to characterize each
sponsor as: (1) industry—including for profit and nonprofit
affiliations (e.g., US National Dairy Council), (2) industry-
associated—including governmental agencies that work with
industry to promote consumption of specific foods or
commodities (e.g., US Department of Agriculture), (3) non-
industry—including governmental agencies with no industry
association (e.g., US National Institutes of Health), university,
and independent foundations, philanthropies, and other
nonprofit organizations, and (4) unknown. Funding source
was then classified for each article as outlined below.
All industry—if all sponsors were classified as category (1)
above.
No industry—if all sponsors were classified as category (3)
above.
Mixed—if any sponsor was classified as category (2) or (4)
```

PREPROCESSING

EXTRA SPACES REMOVES

```
!pip install pymupdf
import re
import fitz
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)
pdf_text = ""
for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)

    text = page.get_text("text")

    cleaned_text = re.sub(r'\s+', ' ', text)
    pdf_text += cleaned_text
pdf_document.close()
print(pdf_text)
```

```
Requirement already satisfied: pymupdf in /usr/local/lib/python3.10/dist-packages (1.23.26)
Requirement already satisfied: PyMuPDFb==1.23.22 in /usr/local/lib/python3.10/dist-packages (from pymupdf) (1.23.22)
Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1262–1273, Baltimore, Maryland, USA, June 23-25 2014. c○2014 Association for Co
```

REMOVE ANY IRRELEVANT CONTENT SUCH AS HEADERS,FOOTERS AND REFERENCES

```
import re
import fitz
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)

pdf_text = ""
for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    cleaned_text = re.sub(r'\s+', ' ', text)
    cleaned_text = re.sub(r'^\[.*?\]', '', cleaned_text)
    cleaned_text = re.sub(r'^\(.{0,30}\)', '', cleaned_text)
    cleaned_text = re.sub(r'^.*?([A-Z][a-z]{2,} [0-9]{1,2},? [0-9]{4}):', '', cleaned_text)
    pdf_text += cleaned_text
pdf_document.close()
print(pdf_text)
```

```
Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1262–1273, Baltimore, Maryland, USA, June 23-25 2014. c○2014 Association for Co
```

NORMALISE THE TEXT

```
import re
import fitz
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)

normalized_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    # Extract text from the page
    text = page.get_text("text")
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    normalized_text += text
pdf_document.close()

print(normalized_text)
```

```
proceedings of the 52nd annual meeting of the association for computational linguistics, pages 1262–1273, baltimore, maryland, usa, june 23-25 2014. c○2014 association for co
```

LOWERCASING

```
import fitz
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)

normalized_text = ""
for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    text = text.lower()
    normalized_text += text

pdf_document.close()
print(normalized_text)
```

```
proceedings of the 52nd annual meeting of the association for computational linguistics, pages 1262–1273,
baltimore, maryland, usa, june 23-25 2014. c○2014 association for computational linguistics
automatic keyphrase extraction: a survey of the state of the art
kazi saidul hasan and vincent ng
human language technology research institute
university of texas at dallas
richardson, tx 75083-0688
{saidul,vince}@hlt.utdallas.edu
abstract
while automatic keyphrase extraction has
been examined extensively, state-of-the-
art performance on this task is still much
lower than that on many core natural lan-
guage processing tasks. we present a sur-
vey of the state of the art in automatic
keyphrase extraction, examining the major
sources of errors made by existing systems
```

```
and discussing the challenges ahead.
1
introduction
automatic keyphrase extraction concerns “the au-
tomatic selection of important and topical phrases
from the body of a document” (turney, 2000). in
other words, its goal is to extract a set of phrases
that are related to the main topics discussed in a
given document (tomokiyo and hurst, 2003; liu
et al., 2009b; ding et al., 2011; zhao et al., 2011).
document keyphrases have enabled fast and ac-
curate searching for a given document from a large
text collection, and have exhibited their potential
in improving many natural language processing
(nlp) and information retrieval (ir) tasks, such
as text summarization (zhang et al., 2004), text
categorization (hulth and megyesi, 2006), opin-
ion mining (berend, 2011), and document index-
ing (gutwin et al., 1999).
owing to its importance, automatic keyphrase
extraction has received a lot of attention. however,
the task is far from being solved: state-of-the-art
performance on keyphrase extraction is still much
lower than that on many core nlp tasks (liu et al.,
2010). our goal in this paper is to survey the state
of the art in keyphrase extraction, examining the
major sources of errors made by existing systems
and discussing the challenges ahead.
2
corpora
automatic keyphrase extraction systems have
been evaluated on corpora from a variety of
sources ranging from long scientific publications
to short paper abstracts and email messages. ta-
ble 1 presents a listing of the corpora grouped by
their sources as well as their statistics.1 there are
at least four corpus-related factors that affect the
difficulty of keyphrase extraction.
     length
     the difficulty of the task increases with
     the length of the input document as longer doc-
```

HANDLING SPECIAL CHARACTERS

```
import fitz
import re
file_path = '/content/scientific_article.pdf'
pdf_document = fitz.open(file_path)

normalized_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)

    normalized_text += text
pdf_document.close()
print(normalized_text)
```

```
     proceedings of the 52nd annual meeting of the association for computational linguistics pages 12621273
     baltimore maryland usa june 2325 2014 c2014 association for computational linguistics
     automatic keyphrase extraction a survey of the state of the art
     kazi saidul hasan and vincent ng
     human language technology research institute
     university of texas at dallas
     richardson tx 750830688
     saidulvincehltutdallasedu
     abstract
     while automatic keyphrase extraction has
     been examined extensively stateofthe
     art performance on this task is still much
     lower than that on many core natural lan
     guage processing tasks we present a sur
     vey of the state of the art in automatic
     keyphrase extraction examining the major
     sources of errors made by existing systems
     and discussing the challenges ahead
     1
     introduction
     automatic keyphrase extraction concerns the au
     tomatic selection of important and topical phrases
     from the body of a document turney 2000 in
     other words its goal is to extract a set of phrases
     that are related to the main topics discussed in a
     given document tomokiyo and hurst 2003 liu
     et al 2009b ding et al 2011 zhao et al 2011
     document keyphrases have enabled fast and ac
     curate searching for a given document from a large
     text collection and have exhibited their potential
     in improving many natural language processing
     nlp and information retrieval ir tasks such
     as text summarization zhang et al 2004 text
     categorization hulth and megyesi 2006 opin
     ion mining berend 2011 and document index
     ing gutwin et al 1999
     owing to its importance automatic keyphrase
     extraction has received a lot of attention however
     the task is far from being solved stateoftheart
     performance on keyphrase extraction is still much
     lower than that on many core nlp tasks liu et al
     2010 our goal in this paper is to survey the state
     of the art in keyphrase extraction examining the
     major sources of errors made by existing systems
     and discussing the challenges ahead
     2
     corpora
     automatic keyphrase extraction systems have
     been evaluated on corpora from a variety of
     sources ranging from long scientic publications
     to short paper abstracts and email messages ta
     ble 1 presents a listing of the corpora grouped by
```

```
their sources as well as their statistics1 there are
at least four corpusrelated factors that affect the
difculty of keyphrase extraction
length
the difculty of the task increases with
```

TOKENIZATION

```
import nltk
from nltk.tokenize import word_tokenize
import fitz

nltk.download('punkt')

file_path = '/content/scientific_article.pdf'
pdf_document = fitz.open(file_path)
pdf_text = ""
for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text

pdf_document.close()
tokens = word_tokenize(pdf_text)
print(tokens)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Proceedings', 'of', 'the', '52nd', 'Annual', 'Meeting', 'of', 'the', 'Association', 'for', 'Computational', 'Linguistics', ',', 'pages', '1262–1273', ',', 'Baltimore', ','
```

STOP WORD REMOVAL

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import nltk
from nltk.tokenize import word_tokenize
import fitz
file_path = '/content/scientific_article.pdf'
pdf_document = fitz.open(file_path)
pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text
pdf_document.close()

tokens = word_tokenize(pdf_text)
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print(filtered_tokens)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
['Proceedings', '52nd', 'Annual', 'Meeting', 'Association', 'Computational', 'Linguistics', ',', 'pages', '1262–1273', ',', 'Baltimore', ',', 'Maryland', ',', 'USA', ',', 'Ju
```

STEMMING

```
!pip install nltk fitz
!pip uninstall PyMuPDF
!pip install PyMuPDF
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import fitz
nltk.download('punkt')
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)

pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text
pdf_document.close()

tokens = word_tokenize(pdf_text)
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(word) for word in tokens]
print(stemmed_tokens)
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: fitz in /usr/local/lib/python3.10/dist-packages (0.0.1.dev2)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.2)
Requirement already satisfied: configobj in /usr/local/lib/python3.10/dist-packages (from fitz) (5.0.8)
Requirement already satisfied: configparser in /usr/local/lib/python3.10/dist-packages (from fitz) (6.0.1)
Requirement already satisfied: httplib2 in /usr/local/lib/python3.10/dist-packages (from fitz) (0.22.0)
Requirement already satisfied: nibabel in /usr/local/lib/python3.10/dist-packages (from fitz) (4.0.2)
Requirement already satisfied: nipype in /usr/local/lib/python3.10/dist-packages (from fitz) (1.8.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from fitz) (1.25.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from fitz) (1.5.3)
Requirement already satisfied: pyxnat in /usr/local/lib/python3.10/dist-packages (from fitz) (1.6.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from fitz) (1.11.4)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from configobj->fitz) (1.16.0)
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.10/dist-packages (from httplib2->fitz) (3.1.2)
Requirement already satisfied: packaging>=17.0 in /usr/local/lib/python3.10/dist-packages (from nibabel->fitz) (24.0)
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from nibabel->fitz) (67.7.2)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (3.2.1)
Requirement already satisfied: prov>=1.5.2 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (2.0.0)
Requirement already satisfied: pydot>=1.2.3 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (1.4.2)
Requirement already satisfied: python-dateutil>=2.2 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (2.8.2)
Requirement already satisfied: rdflib>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (7.0.0)
Requirement already satisfied: simplejson>=3.8.0 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (3.19.2)
Requirement already satisfied: traits!=5.0,<6.4,>=4.6 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (6.3.2)
Requirement already satisfied: filelock>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (3.13.1)
Requirement already satisfied: etelemetry>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (0.3.1)
Requirement already satisfied: looseversion in /usr/local/lib/python3.10/dist-packages (from nipype->fitz) (1.3.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->fitz) (2023.4)
Requirement already satisfied: lxml>=4.3 in /usr/local/lib/python3.10/dist-packages (from pyxnat->fitz) (4.9.4)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from pyxnat->fitz) (2.31.0)
Requirement already satisfied: pathlib>=1.0 in /usr/local/lib/python3.10/dist-packages (from pyxnat->fitz) (1.0.1)
Requirement already satisfied: ci-info>=0.2 in /usr/local/lib/python3.10/dist-packages (from etelemetry>=0.2.0->nipype->fitz) (0.3.0)
Requirement already satisfied: isodate<0.7.0,>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from rdflib>=5.0.0->nipype->fitz) (0.6.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->pyxnat->fitz) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->pyxnat->fitz) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->pyxnat->fitz) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->pyxnat->fitz) (2024.2.2)
WARNING: Skipping PyMuPDF as it is not installed.
Collecting PyMuPDF
  Downloading PyMuPDF-1.23.26-cp310-none-manylinux2014_x86_64.whl (4.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.4/4.4 MB 14.4 MB/s eta 0:00:00
Collecting PyMuPDFb==1.23.22 (from PyMuPDF)
  Downloading PyMuPDFb-1.23.22-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (30.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 30.6/30.6 MB 21.5 MB/s eta 0:00:00
Installing collected packages: PyMuPDFb, PyMuPDF
Successfully installed PyMuPDF-1.23.26 PyMuPDFb-1.23.22
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['proceed', 'of', 'the', '52nd', 'annual', 'meet', 'of', 'the', 'associ', 'for', 'comput', 'linguist', ',', 'page', '1262–1273', ',', 'baltimor', ',', 'maryland', ',', 'usa'
```

PHRASE EXTRACTION

```
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
import fitz
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)

pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text
pdf_document.close()
tokens = word_tokenize(pdf_text)
def generate_ngrams(tokens, n):
    n_grams = ngrams(tokens, n)
    return [' '.join(gram) for gram in n_grams]
n = 3
ngram_phrases = generate_ngrams(tokens, n)
for phrase in ngram_phrases:
    print(phrase)
```

```
Streaming output truncated to the last 5000 lines.
) . We
. We see
We see these
see these results
these results as
results as an
as an indication
an indication that
indication that shallow
that shallow discourse
shallow discourse processing
discourse processing with
processing with a
with a well-designed
a well-designed set
well-designed set of
set of surface-based
of surface-based indicators
surface-based indicators is
indicators is possible
is possible .
possible . 6.2
. 6.2 Limitations
6.2 Limitations and
Limitations and Future
and Future Work
Future Work The
Work The metadiscourse
The metadiscourse features
metadiscourse features ,
features , one
, one focus
one focus of
focus of our
of our work
our work ,
work , currently
, currently depend
currently depend on
depend on manual
on manual re-
manual re- sources
re- sources .
sources . The
. The experiments
The experiments reported
experiments reported here
reported here explore
here explore whether
explore whether metadiscourse
whether metadiscourse information
```

```
metadiscourse information is
information is useful
is useful for
useful for the
for the automatic
the automatic determination
```

WORD SCORES

```
from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
from nltk import FreqDist
import fitz
file_path = '/content/scientific_article.pdf'
pdf_document = fitz.open(file_path)
pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text
pdf_document.close()

tokens = word_tokenize(pdf_text)
def generate_ngrams(tokens, n):
    n_grams = ngrams(tokens, n)
    return [' '.join(gram) for gram in n_grams]

n = 3
ngram_phrases = generate_ngrams(tokens, n)
word_freq = FreqDist(tokens)

word_degree = {}
for token in tokens:
    if token not in word_degree:
        word_degree[token] = len(set(word for word in tokens if word != token))

word_scores = {}
for token in word_freq.keys():
    word_scores[token] = word_freq[token] / word_degree[token]

for word, score in word_scores.items():
    print(f"Word: {word}, Score: {score}")
```

```
Streaming output truncated to the last 5000 lines.
Word: Bohannon, Score: 8.037293039704228e-05
Word: Jacobsen, Score: 8.037293039704228e-05
Word: Puz, Score: 8.037293039704228e-05
Word: Weaver, Score: 8.037293039704228e-05
Word: Yerneni, Score: 8.037293039704228e-05
Word: 1277–1288, Score: 8.037293039704228e-05
Word: Corbett, Score: 8.037293039704228e-05
Word: Epstein, Score: 8.037293039704228e-05
Word: Frost, Score: 8.037293039704228e-05
Word: Furman, Score: 8.037293039704228e-05
Word: Curtiss, Score: 8.037293039704228e-05
Word: Becker, Score: 8.037293039704228e-05
Word: Bosman, Score: 8.037293039704228e-05
Word: Doroshenko, Score: 8.037293039704228e-05
Word: Grijincu, Score: 8.037293039704228e-05
Word: Jackson, Score: 0.00016074586079408456
Word: Kunnatur, Score: 8.037293039704228e-05
Word: 1150–1161, Score: 8.037293039704228e-05
Word: Big, Score: 0.00016074586079408456
Word: perspective, Score: 8.037293039704228e-05
Word: 110, Score: 0.00016074586079408456
Word: Simplified, Score: 8.037293039704228e-05
Word: Communications, Score: 0.00048223758238225364
Word: 107–113, Score: 8.037293039704228e-05
Word: DeCandia, Score: 8.037293039704228e-05
Word: Hastorun, Score: 8.037293039704228e-05
Word: Jampani, Score: 8.037293039704228e-05
Word: Kakulapati, Score: 8.037293039704228e-05
Word: Lakshman, Score: 0.00016074586079408456
Word: Pilchin, Score: 8.037293039704228e-05
Word: Sivasubramanian, Score: 8.037293039704228e-05
Word: Vosshall, Score: 8.037293039704228e-05
Word: Vogels, Score: 8.037293039704228e-05
Word: Bonaventura, Score: 8.037293039704228e-05
Word: Monte, Score: 8.037293039704228e-05
Word: Steffen, Score: 8.037293039704228e-05
Word: Zeuch, Score: 8.037293039704228e-05
Word: Tilmann, Score: 8.037293039704228e-05
Word: Rabl, Score: 8.037293039704228e-05
Word: Volker, Score: 8.037293039704228e-05
Word: Rethinking, Score: 0.00016074586079408456
Word: 1078–1092, Score: 8.037293039704228e-05
Word: Dragojević, Score: 8.037293039704228e-05
Word: Narayanan, Score: 8.037293039704228e-05
Word: Hodson, Score: 8.037293039704228e-05
Word: FaRM, Score: 8.037293039704228e-05
Word: Fast, Score: 0.00032149172158816913
Word: NSDI, Score: 0.00024111879119112682
Word: Fang, Score: 8.037293039704228e-05
Word: Mulder, Score: 8.037293039704228e-05
Word: Hidders, Score: 8.037293039704228e-05
Word: Lee, Score: 0.00048223758238225364
Word: Hofstee, Score: 0.00016074586079408456
Word: In-memory, Score: 8.037293039704228e-05
Word: VLDP, Score: 8.037293039704228e-05
Word: 33–59, Score: 8.037293039704228e-05
Word: Making, Score: 8.037293039704228e-05
```

PHRASE SCORE

```
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
from nltk import FreqDist
import fitz
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)
pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text
pdf_document.close()
tokens = word_tokenize(pdf_text)
def generate_ngrams(tokens, n):
    n_grams = ngrams(tokens, n)
    return [' '.join(gram) for gram in n_grams]

n = 3
ngram_phrases = generate_ngrams(tokens, n)
word_freq = FreqDist(tokens)
word_degree = {}
for token in tokens:
    if token not in word_degree:
        word_degree[token] = len(set(word for word in tokens if word != token))

word_scores = {}
for token in word_freq.keys():
    word_scores[token] = word_freq[token] / word_degree[token]

phrase_scores = {}
for phrase in ngram_phrases:
    phrase_tokens = phrase.split()
    phrase_scores[phrase] = sum(word_scores.get(token, 0) for token in phrase_tokens)
for phrase, score in phrase_scores.items():
    print(f"Phrase: {phrase}, Score: {score}")
```

```
Streaming output truncated to the last 5000 lines.
Phrase: and urine was, Score: 0.24256550393827359
Phrase: urine was collected, Score: 0.016797942452981836
Phrase: was collected for, Score: 0.10086802764828806
Phrase: collected for 24, Score: 0.08889246101912876
Phrase: for 24 h, Score: 0.08728500241118792
Phrase: 24 h for, Score: 0.08728500241118792
Phrase: h for the, Score: 0.4367465037775277
Phrase: the analysis of, Score: 0.6049670470985372
Phrase: analysis of urine, Score: 0.25333547661147726
Phrase: of urine albumin, Score: 0.2493972030220222
Phrase: urine albumin con-, Score: 0.0027326796334994374
Phrase: albumin con- centration, Score: 0.0024111879119112687
Phrase: con- centration and, Score: 0.22938434335315866
Phrase: centration and the, Score: 0.5797299469538659
Phrase: and the urine, Score: 0.5800514386754541
Phrase: the urine output, Score: 0.3551679794245298
Phrase: urine output ., Score: 0.498071049670471
Phrase: output . Then, Score: 0.49823179553126506
Phrase: Then , the, Score: 0.8405401060922681
Phrase: , the animals, Score: 0.840540106092268
Phrase: animals were weighed, Score: 0.015753094357820286
Phrase: were weighed and, Score: 0.2428066227294647
Phrase: weighed and anesthetized, Score: 0.2279376306060119
Phrase: and anesthetized with, Score: 0.27029416492525316
Phrase: anesthetized with ketamine, Score: 0.04275839897122649
Phrase: with ketamine and, Score: 0.2702137919948561
Phrase: ketamine and xylazine, Score: 0.2277768847452178
Phrase: and xylazine (, Score: 0.4029095000803729
Phrase: xylazine ( 100/10, Score: 0.17545410705674327
Phrase: mg/kg , i.p, Score: 0.4885066709532229
Phrase: , i.p ., Score: 0.982960938755827
Phrase: i.p . ), Score: 0.6707924770937148
Phrase: . ) ,, Score: 1.1586561646037614
Phrase: , and blood, Score: 0.7161228098376466
Phrase: and blood samples, Score: 0.2283394952579971
Phrase: blood samples were, Score: 0.015753094357820286
Phrase: samples were collected, Score: 0.01703906124417296
Phrase: were collected (, Score: 0.19209130364893104
Phrase: collected ( via, Score: 0.1781867866902427
Phrase: ( via cardiac, Score: 0.17641858222150777
Phrase: via cardiac puncture, Score: 0.0012859668863526765
Phrase: cardiac puncture ), Score: 0.17593634463912555
Phrase: puncture ) ., Score: 0.6707924770937148
Phrase: . The right, Score: 0.5370519209130364
Phrase: The right kidney, Score: 0.0458125703263141
Phrase: right kidney was, Score: 0.018968011573701978
Phrase: kidney was removed, Score: 0.019209130364893103
Phrase: was removed and, Score: 0.2431281144510529
Phrase: removed and then, Score: 0.2329207522906285
Phrase: and then transferred, Score: 0.2321170229866581
Phrase: then transferred into, Score: 0.015029737984246908
Phrase: transferred into RNAlater, Score: 0.010769972673203666
Phrase: into RNAlater solution, Score: 0.012136312489953385
Phrase: RNAlater solution (, Score: 0.176820446873493
Phrase: solution ( Sigma-Aldrich, Score: 0.17706156566468412
Phrase: USA ) for, Score: 0.2612120237903874
Phrase: for the real, Score: 0.4374698601511011
```

RANK PHRASES - RAKE METHOD

```
import nltk
nltk.download('stopwords')

!pip install rake-nltk
from rake_nltk import Rake
import fitz
file_path = '/content/scientific_article.pdf'

pdf_document = fitz.open(file_path)

pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text
pdf_document.close()

r = Rake()
r.extract_keywords_from_text(pdf_text)

ranked_phrases = r.get_ranked_phrases_with_scores()

for score, phrase in ranked_phrases:
    print(f"Score: {score}, Phrase: {phrase}")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
Streaming output truncated to the last 5000 lines.
Score: 1.3333333333333333, Phrase: hu
Score: 1.3333333333333333, Phrase: hu
Score: 1.3333333333333333, Phrase: happens
Score: 1.3333333333333333, Phrase: happens
Score: 1.3333333333333333, Phrase: happens
Score: 1.3333333333333333, Phrase: happens
Score: 1.3333333333333333, Phrase: guided
Score: 1.3333333333333333, Phrase: guided
Score: 1.3333333333333333, Phrase: gsi
Score: 1.3333333333333333, Phrase: gsi
Score: 1.3333333333333333, Phrase: gsi
Score: 1.3333333333333333, Phrase: gsi
Score: 1.3333333333333333, Phrase: gsi
Score: 1.3333333333333333, Phrase: french
Score: 1.3333333333333333, Phrase: french
Score: 1.3333333333333333, Phrase: felt
Score: 1.3333333333333333, Phrase: felt
Score: 1.3333333333333333, Phrase: f0
Score: 1.3333333333333333, Phrase: f0
Score: 1.3333333333333333, Phrase: experimented
Score: 1.3333333333333333, Phrase: experimented
Score: 1.3333333333333333, Phrase: exe
Score: 1.3333333333333333, Phrase: exe
Score: 1.3333333333333333, Phrase: evalu
Score: 1.3333333333333333, Phrase: evalu
Score: 1.3333333333333333, Phrase: essential
Score: 1.3333333333333333, Phrase: essential
Score: 1.3333333333333333, Phrase: emphasizes
Score: 1.3333333333333333, Phrase: emphasizes
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: edges
Score: 1.3333333333333333, Phrase: ecution
Score: 1.3333333333333333, Phrase: ecution
Score: 1.3333333333333333, Phrase: dif
Score: 1.3333333333333333, Phrase: dif
Score: 1.3333333333333333, Phrase: dichotomy
Score: 1.3333333333333333, Phrase: dichotomy
Score: 1.3333333333333333, Phrase: develop
Score: 1.3333333333333333, Phrase: develop
Score: 1.3333333333333333, Phrase: develop
Score: 1.3333333333333333, Phrase: develop
Score: 1.3333333333333333, Phrase: deal
Score: 1.3333333333333333, Phrase: deal
Score: 1.3333333333333333, Phrase: cut
Score: 1.3333333333333333, Phrase: cut
```

TOP KEYWORDS

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.2)
```

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
from rake_nltk import Rake
import fitz
import os

file_name = 'scientific_article.pdf'
file_path = os.path.join('/content', file_name)
pdf_document = fitz.open(file_path)
pdf_text = ""

for page_num in range(pdf_document.page_count):
    page = pdf_document.load_page(page_num)
    text = page.get_text("text")
    pdf_text += text
pdf_document.close()

r = Rake()
r.extract_keywords_from_text(pdf_text)

top_n = 50
keyphrases = r.get_ranked_phrases()[:top_n]
for keyphrase in keyphrases:
    print(keyphrase)
```

```
6 literature articles clinical trials topic dr pr nr dr pr nr 1 48 14 377 5 12 313 2 156 205 330 17 120 323 3 45 6 535 5 19 221 4 221 46 625 31 26 519 5 69 21 677 35 1 375 6
780 59 35 0 391 2nd gene exact 540 18 0 0 287 119 4 0 0 55 missing variant 230 8 0 0 218 127 4 0 0 121 different variant 91 3 0 0 83 17 1 0 0 14 missing gene 964 32 0 0 264 57
f p r f p r f p r f p r f p r f p r f p r system 52 44 65 26 34 20 61 57 66 86 84 88 45 40 50 38 37 40 44 52 39 baseline 11 30 7 17 31 12 23 56 15 83 78 90 22 32 17 7 15 5 44
industry 13 19 20 52 25 mixed industry 6 17 12 35 17 none stated 27 36 32 95 46 article type interventional 14 47 17 78 38 observational 26 17 38 81 39 scientific review 10 1
bkg bas oth total human aim 127 6 13 23 19 5 10 203 ctr 21 112 4 204 87 18 126 572 txt 14 1 145 46 6 2 6 220
240 bkg 14 31 1 222 370 5 101 744 bas 17 7 7 60 8 97 39 235 oth 6 70 10 828 215 72 773 1
487 bkg 5 13 115 20 153 bas 2 18 1 18 14 53 oth 1 18 2 55 10 1 412 499 total 48 173 39 2
419 47 3 0 464 452 15 5 0 108 different variant 560 19 10 0 110 243 8 2 0 91 missing gene 2
853 62 27 1 476 735 25 15 0 120
bkg bas oth total annotator c aim 35 2 1 19 3 2 62 ctr 86 31 16 23 156 txt 31 7 1 39
270 409 418 165 606 table 3
yes yes cluster graph processing systems pregel sys reg unknown passive yes yes cluster graphlab sys reg unknown passive yes yes cluster powergraph sys reg unknown passive ye
736 91 81 0 425 excludes 165 6 1 0 45 111 4 0 0 63 relevance definitely relevant 2
486 50 37 0 230 missing variant 1
736 258 226 37 578 376 13 1 0 197 excludes 880 29 17 0 141 364 12 3 0 57
149 138 120 9 506 1
959 132 119 18 428 animal pm 536 18 17 2 71 2 0 0 0 1
022 67 44 1 221 436 15 5 0 98 partially relevant 1
four acm document classifications dataset author reader combined trial 149 526 621 training 559 1824 2223 test 387 1217 1482 table 2
four acm document classifications dataset author reader combined trial 149 526 621 training 559 1824 2223 test 387 1217 1482 table 2
state management elem struct stor medium stor struct task st shared st st part repl repl consist repl prot update propag dataflow task deployment systems mapreduce n
systems system rank top 5 candidates top 10 candidates top 15 candidates p r f p r f p r f humb 1 39
systems system rank top 5 candidates top 10 candidates top 15 candidates p r f p r f p r f humb 1 39
state management elem struct stor medium stor struct task st shared st st part repl repl consist repl prot update propag nosql systems dynamo key
yes cluster new programming models sdg sys reg sync active yes yes cluster tensorflow client reg sync passive yes yes cluster tangram sys reg sync passive yes
24 system rank top 5 candidates top 10 candidates top 15 candidates p r f p r f p r f humb 1 30
24 system rank top 5 candidates top 10 candidates top 15 candidates p r f p r f p r f humb 1 30
spark reg dynamic task task activ dynamic shared spark streaming reg dynamic task task activ dynamic shared dataflow job deployment systems millwheel reg static job job compi
runs team id affiliation articles trials bitem bitem group 5 5 cbnu chonbuk national university 3 3 csiromed commonwealth science
state management elem struct stor medium stor struct task st shared st st part repl repl consist repl prot update propag computations
score system rank top 5 candidates top 10 candidates top 15 candidates p r f p r f p r f humb 1 21
score system rank top 5 candidates top 10 candidates top 15 candidates p r f p r f p r f humb 1 21
221 107 105 0 402
might 5 literature articles clinical trials type class total mean median min max total mean median min max pm human pm 8
data management elem struc temp elem bus conn bus impl bus persist bus partition bus repl bus inter dataflow task deployment systems mapreduce general
pr ),
nr
data management elem struct temp elem bus conn bus impl bus persist bus partition bus repl bus inter nosql systems dynamo key
j trial 40 10 10 10 10 training 144 34 39 35 36 test 100 25 25 25 25 table 1
j trial 40 10 10 10 10 training 144 34 39 35 36 test 100 25 25 25 25 table 1
2 ): 159 – 165
data management elem struc temp elem bus conn bus impl bus persist bus partition bus repl bus inter computations
17 march 2024 422 computational linguistics volume 28
j 12 6 7 8 1 2 3 4 9 10 11 13 14 15 16 17 18 19 20 5 figure 10 values
compil activ compil sys dep compil sys dep sys dep use resources info static static dynamic static
738 291 277 65 627 3
1 positive adjective appealing 68 negative adjective unsatisfactory 119
17 march 2024 433 teufel
1 1 ecnuica east china normal university 5 5 eth eth zurich 5 5 fdudmiip school
data manag new prog models hybrid state management elem struct class dep class dep – – graph sys dep gen
```