DAFTAR PUSTAKA

Alfred, V. A., Monica, S. L., & Jeffrey, D. U. (2007). Compilers Principles, Techniques & Tools. pearson Education.

Alkhateeb, F., Baget, J. F., & Euzenat, J. (2009). Extending SPARQL with regular expression patterns (for querying RDF). *Journal of web semantics*, 7(2), 57-73.

American Society for Indexing. (n.d.). *Publishers and Authors, How much is an Index Worth to You?*. https://www.asindexing.org/authors-how-much-is-an-index-worth-to-you/

Arslan, A. N. (2005, November). Multiple sequence alignment containing a sequence of regular expressions. In *2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology* (pp. 1-7). IEEE.

Ashikuzzaman MD. (2021, Agustus 12). *What is Indexing?*. Library & Information Science Community. https://www.lisedunetwork.com/indexing-principles-and-process/

Chapman, C., & Stolee, K. T. (2016, July). Exploring regular expression usage and context in Python. In *Proceedings of the 25th International Symposium on Software Testing and Analysis* (pp. 282-293).

Chiche, A., & Yitagesu, B. (2022). *Part of speech tagging: a systematic review of deep learning and machine learning approaches*. Journal of Big Data, 9(1), 1-25. https://doi.org/10.1186/s40537-022-00561-y

CLiPS Research Center. (2018). *Penn Treebank II tag set*. https://web.archive.org/web/20190206204307/https://www.clips.uantwerpen.be/pages/mbsp-tags

Davis, J. C., Coghlan, C. A., Servant, F., & Lee, D. (2018, October). The impact of regular expression denial of service (ReDoS) in practice: an empirical study at the ecosystem scale. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 246-256).

Flask. (n.d.). *Welcome to Flask — Flask Documentation (2.3.x)*. https://flask.palletsprojects.com/en/2.3.x/

Murata, M., Lee, D., Mani, M., & Kawaguchi, K. (2005). Taxonomy of XML schema languages using formal language theory. *ACM Transactions on Internet Technology (TOIT)*, 5(4), 660-704.

Next.js. (n.d.). *Next JS by Vercel - The React Framework*. https://nextjs.org/

Olympiou, K. (2023, Juni 7). *What Is an Index in a Book? Everything You Need to Know*. Reedsyblog. https://blog.reedsy.com/index-in-a-book/

Paxson, V. (2015). The Bro network security monitor. http://www.bro.org

Pennington, J., Socher, R., & Manning, C. D. (2014, October). *Glove: Global vectors for word representation*. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1), 1-47.

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013, August). Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers) (pp. 455-465).

Tellex, S., Katz, B., Lin, J., Fernandes, A., & Marton, G. (2003, July). Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (pp. 41-47).

Turian, J., Ratinov, L., & Bengio, Y. (2010, July). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384-394).

Vickers, J. (1996, direvisi 2018). *Guidance on indexing your book*. Society of Indexers. https://www2.societyofauthors.org/wp-content/uploads/2022/03/Guide-to-Indexing.pdf

Wondracek, G., Comparetti, P. M., Kruegel, C., Kirda, E., & Anna, S. S. S. (2008, February). Automatic Network Protocol Analysis. In *NDSS* (Vol. 8, pp. 1-14).

Yeole, A. S., & Meshram, B. B. (2011, February). Analysis of different technique for detection of SQL injection. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology* (pp. 963-966).

Zheng, L. X., Ma, S., Chen, Z. X., & Luo, X. Y. (2021). Ensuring the correctness of regular expressions: A review. International Journal of Automation and Computing, 18(4), 521-535.

LAMPIRAN

Lampiran 1. *Source code* program antenna_onlyFeature.py

Ini adalah program untuk menyimpan data skor indeks buku-AN dari 3 fitur (skor RAKE, skor *cosine similarity*, dan jumlah huruf kapital) ke file JSON untuk nantinya digunakan dalam percobaan skenario pembobotan yang berbeda untuk ketiga fitur tersebut.

```
jsonFileName = "antenna.json"

import fitx
import Vanilla_filter as MyRAKE
import WordEmbeddings as CountWordEmb
import math
import re
from ring import ring
import json

import spacy
spacy_model = spacy.load("en_core_web_md")

book_path = "C:\\Users\\LENOVO\\Documents\\Skripsi\\working\\tes yg
biasa cukup nda\\antenna.pdf"

pages = [
    {
        "f": 5, "l": 21, "d": 17
    },
    {
        "f": 23, "l": 42, "d": 16
    },
    {
        "f": 45, "l": 105, "d": 15
    },
    {
        "f": 107, "l": 187, "d": 14
    },
    {
        "f": 191, "l": 267, "d": 12
    }
]

# provide true index
from StructureIndex_antenna import Index as BoBI
```

```
# provide the titles
import TitleOfWholeBooks
pages_r = [
    {
        "f": 19, "l": 21, "d": 18
    },
    {
        "f": 22, "l": 38, "d": 17
    },
    {
        "f": 39, "l": 58, "d": 16
    },
    {
        "f": 59, "l": 120, "d": 15
    },
    {
        "f": 121, "l": 201, "d": 14
    },
    {
        "f": 202, "l": 279, "d": 12
    }
]
titles = TitleOfWholeBooks.titleAndPageNumber(19, 279, book_path,
pages_r)

def countCapitalLetters(string):
    return len(re.findall(r"[A-Z]", string))

bookTitle = "EM Modeling Antennas RF Components Wireless Communication
Systems"
tmpTitle = ""

DATA = []

# for each page,
#   generate the rake
for i in range(len(pages)):
    for j in range(pages[i]["f"], pages[i]["l"]+1):
        print(f"Page {j}")
        # initialize local value (page-specific variable)
        _BoBIofThePage = BoBI[j]

        page_text = fitx.extract_text_from_a_page(book_path,
j+pages[i]["d"]-1)
        rakeIndex = MyRAKE.all(page_text)
```

```
        # for each keyphrases generated, count the cosine similarity and
the capital, then total
        for rakeIdx in rakeIndex:
            rakeMark = rakeIndex[rakeIdx]
            rakeIndex[rakeIdx] = {"rake": rakeMark}

            # compute its mark against the word embeddings
            cosMark = 0
            if (len(titles[j]) != 0):
                for k in range(len(titles[j]) + 1):
                    if (k == len(titles[j])): # klo iteration trkhr mi,
compare ke bookTitle
                        Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
bookTitle)
                        if not math.isnan(Cos_AvF):
                            cosMark += abs(Cos_AvF)
                    else:
                        tmpTitle = titles[j][k][1] if titles[j][k] else
tmpTitle
                        if (tmpTitle != ""):
                            Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
tmpTitle)
                            if not math.isnan(Cos_AvF):
                                cosMark += abs(Cos_AvF)
            else: # klo kosong (tidak ada judul di page tsb, maka pake
last title saja)
                if tmpTitle:
                    Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
tmpTitle)
                    if not math.isnan(Cos_AvF):
                        cosMark += abs(Cos_AvF)
                Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx, bookTitle)
                if not math.isnan(Cos_AvF):
                    cosMark += abs(Cos_AvF)
            rakeIndex[rakeIdx]["cos"] = cosMark

            # compute also the point of its capitalization
            capAmnt = countCapitalLetters(rakeIdx)
            rakeIndex[rakeIdx]["cap"] = capAmnt

        for r, rakeIdx in enumerate(rakeIndex):
            # check if it has the corresponding index in BoBI
            for BoBIIdx in _BoBIofThePage[:]:
                rake_delParentheses = re.sub(r'\s*\(.*\)', '', rakeIdx)
                bobi_delParentheses = re.sub(r'\s*\(.*\)', '', BoBIIdx)
                list_rake = MyRAKE.splitEachWord(rake_delParentheses)
                list_bobi = MyRAKE.splitEachWord(bobi_delParentheses)
```

```
                if [spacy_model(t)[0].lemma_ for t in list_rake] ==
[spacy_model(t)[0].lemma_ for t in list_bobi]:
                    rakeIndex[rakeIdx]["true"] = BoBIIdx
                    break

        DATA.append({
            "page": j,
            "indexes": rakeIndex
        })

with open(jsonFileName, "w", encoding="utf-8") as file:
    json.dump(DATA, file, indent=4)

ring()
```

Lampiran 2. Cuplikan *file* antenna.json


```
[
    {
        "page": 5,
        "indexes": {
            "time-dependent differential form": {
                "rake": 16.0,
                "cos": 0.23061101138591766,
                "cap": 0
            },
            "volume charge density Equation": {
                "rake": 13.5,
                "cos": 0.5969633013010025,
                "cap": 1
            },
            "non-conductive material": {
                "rake": 9.0,
                "cos": 0.10563267581164837,
                "cap": 0
            },
            "conduction current density": {
                "rake": 8.75,
                "cos": 0.2957608252763748,
                "cap": 0
            },
            "displacement current density": {
                "rake": 8.75,
                "cos": 0.1840440034866333,
                "cap": 0
            },
            "total current density": {
                "rake": 8.75,
                "cos": 0.12508918531239033,
```

Lampiran 3. *Source code* program scientific_onlyFeature.py

Ini adalah program untuk menyimpan data skor indeks buku-SC dari 3 fitur (skor RAKE, skor *cosine similarity*, dan jumlah huruf kapital) ke file JSON untuk nantinya digunakan dalam percobaan skenario pembobotan yang berbeda untuk ketiga fitur tersebut.

```
jsonFileName = "scientific.json"

import fitx
import Vanilla_filter as MyRAKE
import WordEmbeddings as CountWordEmb
import math
import re
from ring import ring
import json

import spacy
spacy_model = spacy.load("en_core_web_md")

book_path = "C:\\Users\\LENOVO\\Documents\\Skripsi\\working\\tes yg
biasa cukup nda\\scientific.pdf"

pages = [
    {
        "f": 1, "l": 38, "d": 12
    },
    {
        "f": 45, "l": 73, "d": 12
    },
    {
        "f": 75, "l": 95, "d": 11
    },
    {
        "f": 97, "l": 121, "d": 10
    },
    {
        "f": 123, "l": 145, "d": 10
    },
    {
        "f": 147, "l": 169, "d": 10
    },
    {
        "f": 171, "l": 182, "d": 9
    },
    {
```

```
        "f": 185, "l": 197, "d": 9
    },
    {
        "f": 199, "l": 215, "d": 8
    },
    {
        "f": 219, "l": 246, "d": 7
    },
    {
        "f": 251, "l": 259, "d": 7
    },
    {
        "f": 261, "l": 278, "d": 7
    },
    {
        "f": 281, "l": 292, "d": 6
    },
    {
        "f": 295, "l": 309, "d": 5
    },
    {
        "f": 311, "l": 324, "d": 5
    },
    {
        "f": 327, "l": 342, "d": 5
    }
]

# provide true index
from StructureIndex_scientific import index_tmp as BoBI

# provide the titles
import TitleOfWholeBooks
pages_r = [
    {
        "f": 13, "l": 85, "d": 12
    },
    {
        "f": 86, "l": 106, "d": 11
    },
    {
        "f": 107, "l": 179, "d": 10
    },
    {
        "f": 180, "l": 206, "d": 9
    },
    {
```

```
        "f": 207, "l": 225, "d": 8
    },
    {
        "f": 226, "l": 286, "d": 7
    },
    {
        "f": 287, "l": 299, "d": 6
    },
    {
        "f": 300, "l": 347, "d": 5
    }
]
titles = TitleOfWholeBooks.titleAndPageNumber(13, 347, book_path,
pages_r)

def countCapitalLetters(string):
    return len(re.findall(r"[A-Z]", string))

bookTitle = "Scientific Computing"
tmpTitle = ""

DATA = []

# for each page,
#   generate the rake
for i in range(len(pages)):
    for j in range(pages[i]["f"], pages[i]["l"]+1):
        print(f"Page {j}")
        # initialize local value (page-specific variable)
        _BoBIofThePage = BoBI[j]

        page_text = fitx.extract_text_from_a_page(book_path,
j+pages[i]["d"]-1)
        rakeIndex = MyRAKE.all(page_text)

        # for each keyphrases generated, count the cosine similarity and
the capital, then total
        for rakeIdx in rakeIndex:
            rakeMark = rakeIndex[rakeIdx]
            rakeIndex[rakeIdx] = {"rake": rakeMark}

            # compute its mark against the word embeddings
            cosMark = 0
            if (len(titles[j]) != 0):
                for k in range(len(titles[j]) + 1):
                    if (k == len(titles[j])): # klo iteration trkhr mi,
compare ke bookTitle
```

```
                        Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
bookTitle)
                        if not math.isnan(Cos_AvF):
                            cosMark += abs(Cos_AvF)
                    else:
                        tmpTitle = titles[j][k][1] if titles[j][k] else
tmpTitle
                        if (tmpTitle != ""):
                            Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
tmpTitle)
                            if not math.isnan(Cos_AvF):
                                cosMark += abs(Cos_AvF)
            else: # klo kosong (tidak ada judul di page tsb, maka pake
last title saja)
                if tmpTitle:
                    Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
tmpTitle)
                    if not math.isnan(Cos_AvF):
                        cosMark += abs(Cos_AvF)
                Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx, bookTitle)
                if not math.isnan(Cos_AvF):
                    cosMark += abs(Cos_AvF)
            rakeIndex[rakeIdx]["cos"] = cosMark

            # compute also the point of its capitalization
            capAmnt = countCapitalLetters(rakeIdx)
            rakeIndex[rakeIdx]["cap"] = capAmnt

        for r, rakeIdx in enumerate(rakeIndex):
            # check if it has the corresponding index in BoBI
            for BoBIIdx in _BoBIofThePage[:]:
                rake_delParentheses = re.sub(r'\s*\(.*\)', '', rakeIdx)
                bobi_delParentheses = re.sub(r'\s*\(.*\)', '', BoBIIdx)
                list_rake = MyRAKE.splitEachWord(rake_delParentheses)
                list_bobi = MyRAKE.splitEachWord(bobi_delParentheses)
                if [spacy_model(t)[0].lemma_ for t in list_rake] ==
[spacy_model(t)[0].lemma_ for t in list_bobi]:
                    rakeIndex[rakeIdx]["true"] = BoBIIdx
                    break

        DATA.append({
            "page": j,
            "indexes": rakeIndex
        })

with open(jsonFileName, "w", encoding="utf-8") as file:
    json.dump(DATA, file, indent=4)
```

Lampiran 4. Cuplikan *file* scientific.json


```
[
    {
        "page": 1,
        "indexes": {
            "Springer-Verlag London Limited": {
                "rake": 16.0,
                "cos": 0.35628342628479004,
                "cap": 4
            },
            "High-Performance Scientific Computing": {
                "rake": 15.5,
                "cos": 0.8194358348846436,
                "cap": 4
            },
            "Saied Computer Science Department": {
                "rake": 10.916666666666666,
                "cos": 0.6190155744552612,
                "cap": 4
            },
            "Parallel Numerical Computing": {
                "rake": 9.5,
                "cos": 0.746430516242981,
                "cap": 3
            },
            "Greece e-mail": {
                "rake": 8.6,
                "cos": 0.08344364166259766,
                "cap": 1
            },
            "Saied e-mail": {
                "rake": 8.6,
                "cos": 0.0011989613994956017,
```


Lampiran 5. *Source code* program matlab_onlyFeature.py

Ini adalah program untuk menyimpan data skor indeks buku-MT dari 3 fitur (skor RAKE, skor *cosine similarity*, dan jumlah huruf kapital) ke file JSON untuk nantinya digunakan dalam percobaan skenario pembobotan yang berbeda untuk ketiga fitur tersebut.

```
jsonFileName = "matlab.json"

import fitx
import Vanilla_filter as MyRAKE
import WordEmbeddings as CountWordEmb
import math
import re
```

```
from ring import ring
import json

import spacy
spacy_model = spacy.load("en_core_web_md")

book_path = "C:\\Users\\LENOVO\\Documents\\Skripsi\\working\\tes yg
biasa cukup nda\\matlab.pdf"

pages = [
    {
        "f": 1, "l": 37, "d": 13
    },
    {
        "f": 39, "l": 67, "d": 13
    },
    {
        "f": 71, "l": 98, "d": 12
    },
    {
        "f": 101, "l": 119, "d": 11
    },
    {
        "f": 123, "l": 164, "d": 11
    },
    {
        "f": 167, "l": 183, "d": 11
    },
    {
        "f": 187, "l": 234, "d": 10
    },
    {
        "f": 237, "l": 263, "d": 10
    },
    {
        "f": 267, "l": 305, "d": 9
    }
]

# provide true index
from StructureIndex_matlab import index_tmp as BoBI

# provide the titles
import TitleOfWholeBooks
pages_r = [
    {
        "f": 14, "l": 82, "d": 13
```

```
    },
    {
        "f": 83, "l": 111, "d": 12
    },
    {
        "f": 112, "l": 196, "d": 11
    },
    {
        "f": 197, "l": 275, "d": 10
    },
    {
        "f": 276, "l": 314, "d": 9
    }
]
titles = TitleOfWholeBooks.titleAndPageNumber(14, 314, book_path,
pages_r)

def countCapitalLetters(string):
    return len(re.findall(r"[A-Z]", string))

bookTitle = "Scientific Computing with MATLAB and Octave"
tmpTitle = ""

DATA = []

# for each page,
#   generate the rake
for i in range(len(pages)):
    for j in range(pages[i]["f"], pages[i]["l"]+1):
        print(f"Page {j}")
        # initialize local value (page-specific variable)
        _BoBIofThePage = BoBI[j]

        page_text = fitx.extract_text_from_a_page(book_path,
j+pages[i]["d"]-1)
        rakeIndex = MyRAKE.all(page_text)

        # for each keyphrases generated, count the cosine similarity and
the capital, then total
        for rakeIdx in rakeIndex:
            rakeMark = rakeIndex[rakeIdx]
            rakeIndex[rakeIdx] = {"rake": rakeMark}

            # compute its mark against the word embeddings
            cosMark = 0
            if (len(titles[j]) != 0):
                for k in range(len(titles[j]) + 1):
```

```
                    if (k == len(titles[j])): # klo iteration trkhr mi,
compare ke bookTitle
                        Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
bookTitle)
                        if not math.isnan(Cos_AvF):
                            cosMark += abs(Cos_AvF)
                    else:
                        tmpTitle = titles[j][k][1] if titles[j][k] else
tmpTitle
                        if (tmpTitle != ""):
                            Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
tmpTitle)
                            if not math.isnan(Cos_AvF):
                                cosMark += abs(Cos_AvF)
            else: # klo kosong (tidak ada judul di page tsb, maka pake
last title saja)
                if tmpTitle:
                    Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
tmpTitle)
                    if not math.isnan(Cos_AvF):
                        cosMark += abs(Cos_AvF)
                Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx, bookTitle)
                if not math.isnan(Cos_AvF):
                    cosMark += abs(Cos_AvF)
            rakeIndex[rakeIdx]["cos"] = cosMark

            # compute also the point of its capitalization
            capAmnt = countCapitalLetters(rakeIdx)
            rakeIndex[rakeIdx]["cap"] = capAmnt

        for r, rakeIdx in enumerate(rakeIndex):
            # check if it has the corresponding index in BoBI
            for BoBIIdx in _BoBIofThePage[:]:
                rake_delParentheses = re.sub(r'\s*\(.*\)', '', rakeIdx)
                bobi_delParentheses = re.sub(r'\s*\(.*\)', '', BoBIIdx)
                list_rake = MyRAKE.splitEachWord(rake_delParentheses)
                list_bobi = MyRAKE.splitEachWord(bobi_delParentheses)
                if [spacy_model(t)[0].lemma_ for t in list_rake] ==
[spacy_model(t)[0].lemma_ for t in list_bobi]:
                    rakeIndex[rakeIdx]["true"] = BoBIIdx
                    _BoBIofThePage.remove(BoBIIdx)
                    break

        DATA.append({
            "page": j,
            "indexes": rakeIndex
        })
```

```
with open(jsonFileName, "w", encoding="utf-8") as file:
    json.dump(DATA, file, indent=4)

ring()
```

Lampiran 6. Cuplikan *file* matlab.json


```
[
    {
        "page": 1,
        "indexes": {
            "GNU General Public License": {
                "rake": 15.0,
                "cos": 0.3712409436702728,
                "cap": 6
            },
            "elementary mathematical concepts": {
                "rake": 8.0,
                "cos": 0.2950824499130249,
                "cap": 0
            },
            "short explanation notice": {
                "rake": 8.0,
                "cos": 0.013449111953377724,
                "cap": 0
            },
            "GNU Octave": {
                "rake": 4.125,
                "cos": 0.5605891942977905,
                "cap": 4
            },
            "Numerical Analysis": {
                "rake": 4.0,
                "cos": 0.595086932182312,
                "cap": 2
            },
            "MATrix LABoratory": {
                "rake": 4.0,
                "cos": 0.0,
```


Lampiran 7. *Source code* program DL_onlyFeature.py

Ini adalah program untuk menyimpan data skor indeks buku-DL dari 3 fitur (skor RAKE, skor *cosine similarity*, dan jumlah huruf kapital) ke file JSON untuk nantinya digunakan dalam percobaan skenario pembobotan yang berbeda untuk ketiga fitur tersebut.

```
jsonFileName = "DL.json"
```

```
import fitx
import Vanilla_filter as MyRAKE
import WordEmbeddings as CountWordEmb
import math
import re
from ring import ring
import json

import spacy
spacy_model = spacy.load("en_core_web_md")

book_path = "C:\\Users\\LENOVO\\Documents\\Skripsi\\working\\tes yg
biasa cukup nda\\DL.pdf"

pages = [
    {
        "f": 1, "l": 15, "d": 13
    },
    {
        "f": 17, "l": 49, "d": 13
    },
    {
        "f": 51, "l": 76, "d": 12
    },
    {
        "f": 79, "l": 102, "d": 11
    },
    {
        "f": 107, "l": 119, "d": 10
    },
    {
        "f": 121, "l": 132, "d": 10
    },
    {
        "f": 135, "l": 151, "d": 9
    },
    {
        "f": 153, "l": 162, "d": 9
    },
    {
        "f": 165, "l": 172, "d": 8
    },
    {
        "f": 175, "l": 181, "d": 7
    },
    {
```

```
        "f": 185, "l": 186, "d": 6
    }
]

# provide true index
from StructureIndex_DL import index_tmp as BoBI

# provide the titles
import TitleOfWholeBooks
pages_r = [
    {'f': 14, 'd': 13, 'l': 62},
    {'f': 63, 'd': 12, 'l': 89},
    {'f': 90, 'd': 11, 'l': 116},
    {'f': 117, 'd': 10, 'l': 143},
    {'f': 144, 'd': 9, 'l': 172},
    {'f': 173, 'd': 8, 'l': 181},
    {'f': 182, 'd': 7, 'l': 190},
    {'f': 191, 'd': 6, 'l': 193}
]
titles = TitleOfWholeBooks.titleAndPageNumber(14, 193, book_path,
pages_r)

bookTitle = "Introduction Deep Learning"
tmpTitle = ""

DATA = []

# for each page,
#   generate the rake
for i in range(len(pages)):
    for j in range(pages[i]["f"], pages[i]["l"]+1):
        print(f"Page {j}")
        # initialize local value (page-specific variable)
        _BoBIofThePage = BoBI[j]

        page_text = fitx.extract_text_from_a_page(book_path,
j+pages[i]["d"]-1)
        rakeIndex = MyRAKE.all(page_text)

        # for each keyphrases generated, count the cosine similarity
and the capital, then total
        for rakeIdx in rakeIndex:
            rakeMark = rakeIndex[rakeIdx]
            rakeIndex[rakeIdx] = {"rake": rakeMark}

            # compute its mark against the word embeddings
            cosMark = 0
```

```
            if (len(titles[j]) != 0):
                for k in range(len(titles[j]) + 1):
                    if (k == len(titles[j])): # klo iteration trkhr
mi, compare ke bookTitle
                        Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
bookTitle)
                        if not math.isnan(Cos_AvF):
                            cosMark += abs(Cos_AvF)
                    else:
                        tmpTitle = titles[j][k][1] if titles[j][k]
else tmpTitle
                        if (tmpTitle != ""):
                            Cos_AvF =
CountWordEmb.count_CosAvF(rakeIdx, tmpTitle)
                            if not math.isnan(Cos_AvF):
                                cosMark += abs(Cos_AvF)
            else: # klo kosong (tidak ada judul di page tsb, maka
pake last title saja)
                if tmpTitle:
                    Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
tmpTitle)
                    if not math.isnan(Cos_AvF):
                        cosMark += abs(Cos_AvF)
                Cos_AvF = CountWordEmb.count_CosAvF(rakeIdx,
bookTitle)
                if not math.isnan(Cos_AvF):
                    cosMark += abs(Cos_AvF)
            rakeIndex[rakeIdx]["cos"] = cosMark

            # compute also the point of its capitalization
            capAmnt = MyRAKE.countCapitalLetters(rakeIdx)
            rakeIndex[rakeIdx]["cap"] = capAmnt

        for r, rakeIdx in enumerate(rakeIndex):
            # check if it has the corresponding index in BoBI
            for BoBIIdx in _BoBIofThePage[:]:
                rake_delParentheses = re.sub(r'\s*\(.*\)', '',
rakeIdx)
                bobi_delParentheses = re.sub(r'\s*\(.*\)', '',
BoBIIdx)
                list_rake =
MyRAKE.splitEachWord(rake_delParentheses)
                list_bobi =
MyRAKE.splitEachWord(bobi_delParentheses)
                if [spacy_model(t)[0].lemma_ for t in list_rake] ==
[spacy_model(t)[0].lemma_ for t in list_bobi]:
                    rakeIndex[rakeIdx]["true"] = BoBIIdx
```

```
                    _BoBIofThePage.remove(BoBIIdx)
                    break

        DATA.append({
            "page": j,
            "indexes": rakeIndex
        })

with open(jsonFileName, "w", encoding="utf-8") as file:
    json.dump(DATA, file, indent=4)

ring()
```

Lampiran 8. Cuplikan *file* DL.json

```
{} DLjson > {} 23 > {} indexes > {} vector space
[
    {
        "page": 1,
        "indexes": {
            "great seventeenth-century philosopher": {
                "rake": 16.0,
                "cos": 0.06876692175865173,
                "cap": 0
            },
            "classical nineteenth century worksinlogic": {
                "rake": 16.0,
                "cos": 0.08979621529579163,
                "cap": 0
            },
            "Springer International Publishing AG": {
                "rake": 15.0,
                "cos": 0.7800828218460083,
                "cap": 5
            },
            "Springer Nature": {
                "rake": 5.0,
                "cos": 0.8555644154548645,
                "cap": 2
            },
            "characteristica universalis": {
                "rake": 4.0,
                "cos": 0.2303551360964775,
```

Lampiran 9. *Source code* program fitx.py

```
import fitz  # PyMuPDF
import re

def replace_ligatures(text):
```

```
    # Define a dictionary mapping ligatures to their respective
replacements
    ligature_replacements = {
        "ﬁ": "fi",
        "ﬂ": "fl",
        "ﬀ": "ff",
        "ﬃ": "ffi",
        "ﬄ": "ffl",
        "–": "-",
        "¨o": "o",
        "–": "-",
        "¨u": "u",
    }

    # Construct a regular expression pattern to match any ligature
    pattern = re.compile('|'.join(re.escape(ligature) for ligature in
ligature_replacements.keys()))

    # Replace ligatures in the text using the dictionary
    replaced_text = pattern.sub(lambda match:
ligature_replacements[match.group(0)], text)

    return replaced_text

def extract_text_with_mupdf(pdf_path, page_start, page_end):
    doc = fitz.open(pdf_path)
    text = ""

    for page_num in range(page_start, page_end):
        page = doc[page_num]
        text += replace_ligatures(page.get_text())

    return text

def extract_text_from_a_page(pdf_path, page_number):
    doc = fitz.open(pdf_path)

    page = doc[page_number]
    text = page.get_text()
    text = text.split("\n")
    textTmp = ""
    for t in text:
        if t.endswith("-"):
            textTmp += t[:-1]
        else:
            textTmp += (t + " ")
```

```
    return replace_ligatures(textTmp)
```

Lampiran 10. *Source code* program Vanilla_filter.py

```
import unicodedata
import regex as re
import spacy

spacy_model = spacy.load("en_core_web_md")

def is_symbol_or_greek(char):
    name =  unicodedata.name(char)
    return ('GREEK' in name) or ('SIGN' in name) or ('INCREMENT' in
name) or ('REPLACEMENT CHAR' in name)

def countCapitalLetters(string):
    return len(re.findall(r"[A-Z]", string))

# return an integer
def countPuncts(stringg):
    count = 0
    adaBukaKurung = 0
    ygDalamKurung = ""

    if bool(re.search(r"[-+]", stringg)):
        if len(stringg) <= 5:
            return 3
        if bool(re.search(r'[^A-Za-z \-]', stringg)) and
(len(stringg) <= 7):
            return 3

    if bool(re.search(r'f\(x', stringg, re.IGNORECASE)):
        return 3

    for i,char in enumerate(stringg):
        if adaBukaKurung > 0:
            ygDalamKurung += char

        if (char == "(" or (char == ")" and adaBukaKurung<1)):
            count += 3
            if char == "(":
                adaBukaKurung += 1
        elif char == ")" and adaBukaKurung>0:
            count -= 3
            adaBukaKurung -= 1
        elif (not char.isalpha()) and (not char.isdigit()) and (not
char.isspace()) and (char != "-") or (is_symbol_or_greek(char)):
```

```
            count += 1

    if (re.search(r'\(.*\)', stringg) and len(ygDalamKurung) < 4 and
countCapitalLetters(ygDalamKurung) < 2):
        return 3

    return count

# return a dictionary
# key    (string) : word
# value  (number) : amount of time that word appears in the text
def countWordFreq(text):
    words = re.findall(r'\b\w+\b', text.lower())
    wordFreq = {}
    for w in words:
        if w in wordFreq:
            wordFreq[w] += 1
        else:
            wordFreq[w] = 1
    return wordFreq


# return list of words
def splitEachWord(text):
    return re.findall(r'\b\w+\b', text.lower())


# '.' and ',' is also got into the list
def splitEachWord_(text):
    return re.findall(r'\b\w+\b|[.,]', text)


# return list of candidate keyphrase
# split by: stopwords | [.,] | words which length <= 3
# we use splitEachWord_ here because the [.]/[,] will also separate
/ split the text
def splitTextByStopwords_scratch(text, stopwords):
    words = splitEachWord_(text)
    # print("words", words)
    chunks = []
    tmp = []
    for w in words:
        if (w.lower() in stopwords) or w[-1]=="." or len(w)<3:
            if tmp==[]:
                continue
            addToChunk = " ".join(tmp)
            chunks.append(addToChunk)
```

```
            tmp = []
        else:
            tmp.append(w)
    if tmp!=[]:
        addToChunk = " ".join(tmp)
        chunks.append(addToChunk)

    return chunks

spacy_tags_to_avoid = ["CC", "CD", "DT", "EX", "IN", "JJR", "MD",
"PDT", "POS", "PRP", "PRP$", "RB", "RBR", "RBS", "RP", "TO", "UH",
"VBZ", "VBN", "VBD", "VBG", "WDT", "WP", "WP$", "WRB", "ADD", "BES",
"HVS", "VBP", "JJS", "SYM", ",", "."]

# equivalent to splitTextByStopwords_scratch
def splitTextByTAG_scratch(text):
    doc = spacy_model(text) # printed like a string, but not of
string type -> the length is equal to the number of tagged
    chunks = []
    tmp = ""
    i = 0

    while i < len(doc):
        the_word = doc[i].text
        the_tag = doc[i].tag_
        is_stopwords = doc[i].is_stop

        nextChar = doc[i+1].text if i!=len(doc)-1 else ""

        # if the previous character is -
        if i!=0 and doc[i-1].text=="-":
            if "-" in tmp:
                tmp += the_word + ("-" if nextChar=="-" else " ")
                if nextChar == "-":
                    i += 2
                else:
                    i += 1
                continue
            else:
                i += 1
                continue

        # if the next character is -
        if i != len(doc)-1:
            if nextChar == "-":
                if (the_tag not in spacy_tags_to_avoid):
                    tmp += the_word + "-"
```

```
                    i += 2
                    continue

        if (
            (the_tag in spacy_tags_to_avoid)
            or (is_stopwords)
            or (len(the_word)==1)
            or (((len(the_word)==2) or (len(the_word)==3)) and not
the_word.isupper())
            or ("." in the_word)
            or not (the_word[0].isalpha() or the_word[0].isdigit())
        ):
            if len(tmp) <= 3: # cands shouldnt be less than 3
characters
                tmp = ""
            else:
                chunks.append(tmp.strip())
                tmp = ""
        else:
            possess = bool(re.search(r"[^A-Za-z0-9]s",
doc[i+1].text)) if i != len(doc)-1 else False
            tmp += (the_word + ("'s " if possess else " "))
            if possess:
                i += 2
                continue
        i += 1

    if tmp != "":
        chunks.append(tmp.strip())

    # make sure tidak ada keyError saat generate cooccurence matrix
    # filter chunks that has too many punctuations
    # filter chunks yang punya 
    will_be_in_wordfreqs = splitEachWord(text)
    for c in chunks[:]:
        if (bool(re.search(r"[^‖|⋆→¯×;=+Δτ]", c))):
            chunks.remove(c)
            continue

        jumlahPunc = countPuncts(c)
        if (jumlahPunc > 1) or ((jumlahPunc == 1) and (len(c) <=
5)):
            chunks.remove(c)
            continue

        chunk_tmp = splitEachWord(c)
        for eachword in chunk_tmp:
```

```
            if eachword not in will_be_in_wordfreqs:
                chunks.remove(c)
                break

    return chunks


# return list of unique word of candidate keyphrases returned from
splitTextByStopwords_scratch or splitTextByTAG_scratch
def getUniqueKeywordFromCandidates(cands):
    # cands is returned from splitTextByStopwords_scratch
    uniKeyword = []
    for i in range(len(cands)):
        words = splitEachWord(cands[i])
        for w in words:
            if w not in uniKeyword:
                uniKeyword.append(w)

    return uniKeyword


# return list of lists of number
def generateCoOccurenceMatrix(uniqueKeywordFromCandidates, cands,
text):
    n = len(uniqueKeywordFromCandidates)
    matrix = [[-7 for _ in range(n)] for _ in range(n)]
    wordFreqs = countWordFreq(text)
    # isi dlu yang diagonal
    for i in range(n):
        matrix[i][i] = wordFreqs[uniqueKeywordFromCandidates[i]]
    # print(matrix)

    for i in range(n):
        for j in range(n):
            if (i == j):
                continue
            elif matrix[j][i] != -7:
                matrix[i][j] = matrix[j][i]
            else:
                tmp = 0
                for k in range(len(cands)):
                    if uniqueKeywordFromCandidates[i].lower() in
splitEachWord(cands[k]) and uniqueKeywordFromCandidates[j].lower()
in splitEachWord(cands[k]):
                        tmp += 1
                matrix[i][j] = tmp
```

```
    return matrix


# return a dictionary of
# key    (string) : word
# value  (number) : degree score of that word
def countDegreeScore(uniqueKeywordFromCandidates,
coOccurenceMatrix):
    word_degreeScore = {}
    for i in range(len(uniqueKeywordFromCandidates)):
        degree_of_word = sum(coOccurenceMatrix[i])
        word_frequency = coOccurenceMatrix[i][i]
        degree_score = degree_of_word / word_frequency
        word_degreeScore[uniqueKeywordFromCandidates[i]] =
degree_score
    return word_degreeScore


def sum_ascii(string):
    return sum(ord(char) for char in string)


def removeDuplicationInCands(cands): # also remove keyphrase with
more than 5 words
    nonDuplCands = []

    for cand in cands:
        if len(splitEachWord(cand)) >= 5:
            continue

        if cand.lower() not in [ndpl.lower() for ndpl in
nonDuplCands]:
            idx__ = next((j for j, ndpls in enumerate(nonDuplCands)
if splitEachWord(cand) == splitEachWord(ndpls)), None)
            if idx__ is None:
                nonDuplCands.append(cand)
            else: #klo sdh ada di nonDuplCands
                if "-" in cand:
                    nonDuplCands[idx__] = cand
                # klo "-" in nonDuplCands[idx__] brti cckmi, jgnmi
gntiki

        else: # kalau sdh ada mi di nonDuplCands
            idx = next((i for i, ndpl in enumerate(nonDuplCands) if
ndpl.lower() == cand.lower()), None)
            # cek! kalau lebih kapital ki, gnti dgn yg ini
            if (sum_ascii(cand) < sum_ascii(nonDuplCands[idx])):
```

```
                nonDuplCands[idx] = cand
                continue

    return nonDuplCands


# return a dictionary
# key   (string) : candidate keyphrase (non duplicate one)
# value (number) : total degree score of each word in the phrase
def generateFinalScore(nonDuplCands, degreeScore):
    finalScores = {}
    # print(f"nonDuplCands = {nonDuplCands}\ndegreeScore =
{degreeScore}")
    for i in range(len(nonDuplCands)):
        words = splitEachWord(nonDuplCands[i])
        score = 0
        for w in words:
            score += degreeScore[w.lower()]
        finalScores[nonDuplCands[i]] = score
    return finalScores


# connect all of the functions
# variant : using stopwords to split
def all_useSW(text, stopwords):
    cands = splitTextByStopwords_scratch(text, stopwords)
    uniqueKeywordFromCandidates =
getUniqueKeywordFromCandidates(splitTextByStopwords_scratch(text,
stopwords))
    coOccurenceMatrix =
generateCoOccurenceMatrix(uniqueKeywordFromCandidates, cands, text)
    finalScores =
generateFinalScore(removeDuplicationInCands(cands),

countDegreeScore(uniqueKeywordFromCandidates, coOccurenceMatrix))
    return dict(sorted(finalScores.items(), key=lambda item:
item[1], reverse=True))


# connect all of the functions
# variant : using tag to split
def all(text):
    cands = splitTextByTAG_scratch(text)
    uniqueKeywordFromCandidates =
getUniqueKeywordFromCandidates(splitTextByTAG_scratch(text))
    coOccurenceMatrix =
generateCoOccurenceMatrix(uniqueKeywordFromCandidates, cands, text)
```

```
    finalScores =
generateFinalScore(removeDuplicationInCands(cands),

countDegreeScore(uniqueKeywordFromCandidates, coOccurenceMatrix))
    sortedRake = dict(sorted(finalScores.items(), key=lambda item:
item[1], reverse=True))
    copy_sortedRake = sortedRake.copy()
    for idx in copy_sortedRake:
        if copy_sortedRake[idx] == 0:
            del sortedRake[idx]
    return sortedRake
```

Lampiran 11. *Source code* program WordEmbeddings.py

```
import torch
import torchtext
import torch.nn.functional as F
import re
import numpy as np
from nltk.corpus import stopwords

glove_file =
"C:\\Users\\LENOVO\\Downloads\\glove.840B.300d\\glove.840B.300d.txt"
glove = torchtext.vocab.Vectors(glove_file)

stop_words = stopwords.words("english")

def cos_sim(w1, w2):
    return float(torch.cosine_similarity(glove[w1].unsqueeze(0),
                                         glove[w2].unsqueeze(0)))

def eucli_dist(w1, w2):
    return float(torch.norm(glove[w2] - glove[w1]))

def count(w1, w2):
    cossim = cos_sim(w1, w2)
    euclid = eucli_dist(w1, w2)
    print(f"{w1} & {w2}\nCos = {cossim}\nEucl = {euclid}")

def print_closest_words_eucl(vec, n=5):
    dists = torch.norm(glove.vectors - vec, dim=1) # compute distances
to all words
    lst = sorted(enumerate(dists.numpy()), key=lambda x: x[1]) # sort by
distance
    for idx, difference in lst[1:n+1]:
        print(glove.itos[idx], difference)
```

```
def print_closest_words_cossim(vec, n=5):
    # Compute cosine similarity between input vector and all word
vectors
    sims = F.cosine_similarity(glove.vectors, vec.unsqueeze(0), dim=1)

    # Sort by similarity (descending order)
    lst = sorted(enumerate(sims.numpy()), key=lambda x: x[1],
reverse=True)

    # Print the closest words
    for idx, similarity in lst[:n]:
        print(glove.itos[idx], similarity)

def toTokens(text):
    tokens = re.findall(r"\b(\w+)\b", text)
    # for i in range(len(tokens)):
    #     if tokens[i] in stop_words:
    #         del tokens[i]
    for t in tokens[:]:
        if t in stop_words:
            tokens.remove(t)
    return tokens

def count_vectors_average(sentence):
    word_list = toTokens(sentence)
    vector_list = [glove[word] for word in word_list]

    vector_list_np = [vector.numpy() for vector in vector_list]

    average_vector_np = np.mean(vector_list_np, axis=0)
    return torch.tensor(average_vector_np)

def average_first(text1, text2):
    ave1 = count_vectors_average(text1)
    ave2 = count_vectors_average(text2)

    dist_cos = float(torch.cosine_similarity(ave1.unsqueeze(0),
ave2.unsqueeze(0)))
    dist_eucl = float(torch.norm(ave2 - ave1))
    return dist_cos, dist_eucl

def count_CosAvF(text1, text2):
    ave1 = count_vectors_average(text1)
    ave2 = count_vectors_average(text2)

    dist_cos = float(torch.cosine_similarity(ave1.unsqueeze(0),
ave2.unsqueeze(0)))
```

```
    return dist_cos

def each_then_average(text1, text2):
    tokens1 = toTokens(text1)
    tokens2 = toTokens(text2)
    dists_cossim = []
    dists_eucl = []
    for i in range(len(tokens1)):
        for j in range(len(tokens2)):
            dists_eucl.append(eucli_dist(tokens1[i], tokens2[j]))
            dists_cossim.append(cos_sim(tokens1[i], tokens2[j]))

    np_eucl = np.array(dists_eucl)
    np_cossim = np.array(dists_cossim)

    ave_eucl = np.mean(np_eucl)
    ave_cossim = np.mean(np_cossim)

    return ave_cossim, ave_eucl

# return tokens which doesnt have embeddings
def checkIfHasEmbeddings(tokens):
    doesntHaveEmbeddings = []
    for token in tokens:
        tokenVec = glove[token]
        if (tokenVec == torch.zeros_like(tokenVec)).all():
            doesntHaveEmbeddings.append(token)

    return doesntHaveEmbeddings
```

Lampiran 12. *Source code* program TitleOfWholeBooks.py

```
import fitx
import re

def getTitleOfAPage(book_path, pageNumber_whole, previousDigit,
previous2ndDigit):
    page_text = fitx.extract_text_with_mupdf(book_path,
pageNumber_whole-1, pageNumber_whole)

    lines = page_text.split("\n")
    prev = previousDigit
    firstnumber = 1
    prevSec = previous2ndDigit
    secondnumber = 1

    res = []
```

```
    for line in lines:
        regex = re.findall(r"^([1-9]\d*\.[\d\.]+)\s*([A-Z].+)", line)
        regex_init = re.findall(r"^\d+", line)
        regex_second = re.findall(r"^\d+\.(\d*)", line)
        firstnumber = int(regex_init[0]) if len(regex_init) > 0 else
firstnumber
        try:
            secondnumber = int(regex_second[0])
        except:
            pass

        if (regex != [] and secondnumber==1 and firstnumber==prev+1):
            prevSec = 1

        if (regex != []) and (firstnumber == prev or firstnumber ==
prev+1) and (secondnumber == prevSec or secondnumber == prevSec+1):
            prevSec = secondnumber
            prev = firstnumber
            res.append(regex[0])

    return res, prev, prevSec

# first_page -> what page is the page number 1
# last_page  -> the last absolute content page
# pages_r    -> array of dict {"f", "l", "d"}
#               f: absolute first page number,
#               l: absolute last page number,
#               d: difference between absolute and labeled page number
def titleAndPageNumber(first_page, last_page, book_path, pages_r):
    _prev = 1
    _prevSec = 1

    p_idx = 0
    all = [[] for i in range(pages_r[-1]["l"] - pages_r[-1]["d"] + 1)]
    for i in range(first_page, last_page+1):
        res, prev, prevSec = getTitleOfAPage(book_path, i, _prev,
_prevSec)
        _prev = prev
        _prevSec = prevSec
        if (i > pages_r[p_idx]["l"]):
            p_idx += 1
        num = i - pages_r[p_idx]["d"]
        all[num] = res

    return all
```

Lampiran 13. *Source code* program tesbobot1to10.ipynb

Di sini, peneliti melakukan percobaan skenario pembobotan dari 1 sampai 10 untuk masing-masing fitur tambahan (*cosine similarity* dan jumlah huruf kapital).

```
import json
import importlib
from ring import ring
import numpy as np
import StructureIndex

def do_retrank(JSONfilename, TXTfilename, trueBobiName, bobot_cos,
bobot_kap):
    print(f"{TXTfilename} ---")
    BoBI = StructureIndex.call(JSONfilename[:-5])

    with open(JSONfilename, "r") as filee:
        DATA = json.load(filee)

    _TCORRECT = 0     # number of the correct one
    _TRANK = 0        # number of rank totaled: the lower this is, the
better
    _TMARK = 0        # total mark of the correct one
    _TUNDETECTED = 0  # bobi which undetected
    _TGENERATED = 0   # the number of index that rake generated
    _RANK = []

    with open(TXTfilename, "w", encoding="utf-8") as file:
        for pageData in DATA:
            _Tcorrect = 0
            _Trank = 0
            _Tmark = 0
            _Tundetected = 0
            _Tgenerated = len(pageData["indexes"])
            _BoBIofThePage = BoBI[pageData["page"]]
            _rank = []

            text_items = ""

            for index in pageData["indexes"]:
                rake = pageData["indexes"][index]["rake"]
                cos = pageData["indexes"][index]["cos"]
                cap = pageData["indexes"][index]["cap"]
```

```
                pageData["indexes"][index]["total"] = rake +
(bobot_cos*cos) + (bobot_kap*cap)

            sortedIndex = dict(sorted(pageData["indexes"].items(),
key=lambda item: item[1]["total"], reverse=True))

            for i,idx in enumerate(sortedIndex):
                inquote = f"'{idx}'"
                text_items += f"\t{i+1}) {inquote.ljust(50)}"

                if "true" in sortedIndex[idx]:
                    theFounded = sortedIndex[idx]["true"]
                    try:
                        _BoBIofThePage.remove(theFounded)
                        _Tcorrect += 1
                        _Trank += i+1
                        _rank.append(i+1)
                        _Tmark += sortedIndex[idx]["total"]
                        text_items += f">>>>>>>>>>  '{theFounded}'"
                    except:
                        # print("\tDobel found:", theFounded)
                        pass

                text_items += f"\n\t\t\t{sortedIndex[idx]['rake']:.3f}
 +  {sortedIndex[idx]['cos']:.3f}*{bobot_cos}  +
 {sortedIndex[idx]['cap']}*{bobot_kap}    =
 {sortedIndex[idx]['total']:.4f}\n"

            _Tundetected = len(_BoBIofThePage)

            _TCORRECT += _Tcorrect
            _TRANK += _Trank
            _TMARK += _Tmark
            _TUNDETECTED += _Tundetected
            _TGENERATED += _Tgenerated
            _RANK += _rank

            file.write(f"\nPage {pageData['page']} {{\n")
            file.write(f"\tCorrect : {_Tcorrect}  |  Rank : {_rank} =>
{_Trank}  |  TMark : {_Tmark:.3f}  |  Undetected : {_Tundetected}  |
 Generated : {_Tgenerated}\n\n")
            file.write(f"\tUndetected = {_BoBIofThePage}\n\n")
            file.write(text_items)
            file.write("}\n")

        file.write("\n\n")
        file.write("--------WHOLLY-------\n")
```

```
        file.write(f"\tCorrect     = {_TCORRECT}\n")
        file.write(f"\tAve Rank    = {(_TRANK / _TCORRECT):.3f}     *
the lesser the better\n")
        file.write(f"\tMark        = {_TMARK:.3f}\n")
        file.write(f"\tUndetected  = {_TUNDETECTED}\n")
        file.write(f"\tGenerated   = {_TGENERATED}\n\n")

        precision = _TCORRECT/_TGENERATED
        recall = _TCORRECT / (_TCORRECT + _TUNDETECTED)
        fmeasure = (2*recall*precision) / (recall+precision)

        file.write(f"Precision     = {_TCORRECT} / {_TGENERATED} =
{precision}\n")
        file.write(f"Recall        = {_TCORRECT} / ({_TCORRECT} +
{_TUNDETECTED}) = {recall}\n")
        file.write(f"F-Measure     = 2*recall*precision /
(recall+precision) = {fmeasure}\n\n\n")

        file.write(f"Rank distributions = {_RANK}\n\n")

    return (_TRANK / _TCORRECT)
```

```
testing = [(i,j) for i in range(1,11) for j in range(1,11)]
```

```
DATA = [
    {
        "bobot": t,
        "antenna": 0,
        "scientific": 0,
        "matlab": 0,
        "DL": 0,
        "total": 0
    } for t in testing
]
```

```
buku = ["antenna", "scientific", "matlab", "DL"]
```

```
for i,d in enumerate(DATA):
    for b in buku:
        bobot_cos = DATA[i]["bobot"][0]
        bobot_cap = DATA[i]["bobot"][1]
        DATA[i][b] = do_retrank(f"{b}.json",
f"./evalAll/{b}_{bobot_cos}_{bobot_cap}.txt", f"StructureIndex_{b}",
bobot_cos, bobot_cap)
```

```
    DATA[i]["total"] = DATA[i]["antenna"] + DATA[i]["scientific"] +
DATA[i]["matlab"]  + DATA[i]["DL"]
```

```
sortedDATA = sorted(DATA, key=lambda x: x["total"])

print(sortedDATA)
```

Lampiran 14. Tautan *source code* data indeks benar dari buku evaluasi

https://drive.google.com/drive/folders/1YtQ_teDCJHAeXT-XvawZ7K8pPK-kAXpe?usp=sharing

Lampiran 15. Tautan dokumentasi skenario

https://drive.google.com/drive/folders/1oBx_lojJcEMyEc_4-zkAE5S0ej_fApww?usp=sharing

Lampiran 16. Tautan *repository* github kode *front-end* web

https://github.com/christiechindy/FE_generateBoBI

Lampiran 17. Tautan *repository* github kode *back-end* web

https://github.com/christiechindy/BE_generateBoBI

KARTU BIMBINGAN SKRIPSI

Prodi S1 Teknik Informatika Universitas Hasanuddin

Stb.	Nama Mahasiswa
D121201077	Chindy Christie Davina

Pembimbing	Nama Pembimbing	Paraf & Tgl. Persetujuan Ujian Akhir
I	Dr. Ir. Ingrid Nurtanio, M.T.	1 - 5 - 2024
No. SK Pemb	No. 478/UN4.7.1/TD.06/2024	

Judul Skripsi	IMPLEMENTASI ALGORITMA *RAPID AUTOMATIC KEYWORD EXTRACTION (RAKE)* PADA PEMBUATAN INDEKS BUKU

No	Tanggal Bimbingan	Uraian Kegiatan Bimbingan	Paraf Pemb.
1	Rabu, 17 Jan 2024	Konsultasi mengenai phrase deliniter	
2	Jumat, 26 Jan 2024	konsultasi mengenai cara meningkatkan akurasi	
3	Rabu, 7 Feb 2024	konsultasi mengenai percobaan menggunakan word Embeddings	
4	Jumat, 23 Feb 2024	konsultasi mengenai hasil percobaan menggunakan normalisasi skor	
5	Kamis, 29 Feb 2024	konsultasi mengenai kendala pada buku dengan nomor halaman yang terlangkahi	
6	Jumat, 1 Mar 2024	konsultasi mengenai hasil percobaan menggunakan jenis splitter yang berbeda	
7	Rabu, 6 Mar 2024	Konsultasi mengenai penambahan fitur	
8	Rabu, 13 Mar 2024	konsultasi mengenai tata cara dan struktur penulisan	
9	Rabu, 20 Mar 2024	konsultasi mengenai struktur penjabaran skenario	
10	Jumat, 22 Mar 2024	konsultasi mengenai blok diagram alur skenario penelitian	
11	Senin, 25 Mar 2024	konsultasi mengenai review penulisan	
12	Rabu, 27 Mar 2024	konsultasi mengenai perbaikan penulisan	
13	Selasa, 2 Apr 2024	konsultasi PPT seminar hasil	
14	Kamis, 4 Apr 2024	konsultasi persrapan presentasi seminar hasil	
15	Rabu, 17 Apr 2024	konsultasi mengenai revisian	
16			

LEMBAR PERBAIKAN SKRIPSI

"IMPLEMENTASI ALGORITMA *RAPID AUTOMATIC KEYWORD EXTRACTION* (*RAKE*) PADA PEMBUATAN INDEKS BUKU"

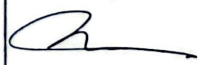
OLEH:

CHINDY CHRISTIE DAVINA
D121201077

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 8 Mei 2024.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Dr. Ir. Ingrid Nurtanio, M.T.	
Anggota	Mukarramah Yusuf, B.Sc., M.Sc., Ph.D.	
	Anugrayani Bustamin, ST., MT.	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr. Ir. Ingrid Nurtanio, M.T.	

KEMENTERIAN PENDIDIKAN , KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus FakultasTeknikUnhas, Jl. PorosMalino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

DAFTAR HADIR SEMINAR HASIL

Nama/Stambuk : 1.Chindy Christie Davina D121201077

Judul Skipsi/T.A : **"Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku"**

Hari/Tanggal : Jumat, 5 April 2024
Jam : 11.00 Wita – Selesai
Tempat : Ruang **Lab. AIMP** Departeman Teknik Informatika Gowa

No.	Jabatan	Nama Dosen	Tanda Tangan
L.	Pembimbing I	1. Dr. Ir. Ingrid Nurtanio, M.T	1.....
II.	Anggota Penguji	2. MukarramahYusuf,B.Sc.,M.Sc.,Ph.D	2 .....
		3. Anugrayani Bustamin,ST.,M.T	3 .....

PANITIA UJIAN

Ketua,

Dr. Ir. Ingrid Nurtanio, M.T

KEMENTERIAN PENDIDIKAN , KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus FakultasTeknikUnhas, Jl. PorosMalino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

BERITA ACARA SEMINAR HASIL

Pada hari ini **Jumat,** tanggal **5 April 2024** Pukul **11.00 WITA - Selesai** bertempat di **Ruang Lab. AIMP Departemen Teknik Informatika**, telah dilaksanakan Seminar Hasil bagi Saudara :

Nama : Chindy Christie Davina
No. Stambuk : D121201077
Fakultas/Departemen : Teknik/Teknik Informatika
Judul Skripsi : **"Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku"**

Yang dihadiri oleh Tim Penguji Seminar Hasil sebagai berikut :

No.	N a m a	Jabatan	Tanda tangan
1.	Dr. Ir. Ingrid Nurtanio, M.T	Pemb I/Ketua	1.....
2.	MukarramahYusuf,B.Sc.,M.Sc.,Ph.D	Anggota	2.....
3.	Anugrayani Bustamin,ST.,M.T	Anggota	3.....

Hasil keputusan Tim Penguji Seminar Hasil : **Lulus / Tidak lulus** dengan nilai angka ..... dan huruf .....

Gowa, 5 April 2024

Ketua/Sekretaris Panitia Ujian,

Dr. Ir. Ingrid Nurtanio, M.T

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus Fakultas Teknik Unhas, Jl. Poros Malino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

Nomor : 458/UN4.7.7/TD.06/2024
Lamp : -
Hal : Penerbitan Surat Penugasan Panitia/Penguji Seminar Hasil Strata Satu (S1)

Kepada Yth :

Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Universitas Hasanuddin

Di-

Gowa

Dengan hormat,
Berdasarkan Persetujuan Pembimbing Mahasiswa, Bersama ini diusulkan susunan Panitia/Penguji Seminar Hasil Strata Satu (S1) bagi mahasiswa Departemen Teknik Informatika Fakultas Teknik tersebut di bawah ini :

Nama / Stambuk : Chindy Christie Davina D121201077
Judul TA : Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku

Dengan ini kami sampaikan Susunan Panitia Seminar Hasil Program Strata Satu (S1) Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin dengan susunan sebagai berikut :

Pembimbing I/ Ketua : 1. Dr. Ir. Ingrid Nurtanio, M.T
Penguji/ Anggota : 2. Mukarramah Yusuf, B.Sc., M.Sc., Ph.D.
3. Anugrayani Bustamin, ST., M.T.

Untuk dapat diterbitkan surat penugasannya

Demikian penyampaian kami, atas perhatian dan kerjasamanya diucapkan terima kasih.

Gowa, 2 April 2024
Ketua Departemen Tek.Informatika,

Prof. Dr. Ir. Indrabayu, ST, MT., M.Bus.Sys., IPM, ASEAN.Eng
Nip.19750716 200212 1 004

Tembusan :
1. Arsip

Jumat, 5 April 2024
Jam : 11^00
Lab : AIMP

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
Poros Malino Km.6Bontomarannu(92172) Gowa, Sulawesi Selatan 92172, Sulawesi Selatan
Telp. (0411) 586015, 586262 Fax (0411) 586015
http://eng.unhas.ac.id, Email : teknik@unhas.ac.id

SURAT PENUGASAN
No. 7573/UN4.7.1/TD.06/2024

Dari : Dekan Fakultas Teknik Universitas Hasanuddin

Kepada : Mereka yang tercantum namanya dibawah ini

Isi :

1. Bahwa merujuk kepada Peraturan Rektor Universitas Hasanuddin **Nomor : 29/UN4.1/2023 tentang Penyelenggaraan Program Sarjana Universitas Hasanuddin** , dengan ini menugaskan Saudara sebagai PENGUJI/PANITIA SEMINAR HASIL Program Strata Satu (S1) Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin dengan susunan sebagai berikut :

   Pembimbing I/ Ketua : 1. Dr. Ir. Ingrid Nurtanio, M.T
   Penguji/ Anggota : 2. Mukarramah Yusuf, B.Sc., M.Sc., Ph.D.
   3. Anugrayani Bustamin, ST., M.T.

   Untuk menguji bagi mahasiswa tersebut dibawah ini :

   Nama/NIM : Chindy Christie Davina D121201077
   Program Studi : Teknik Informatika
   Judul thesis/Skrips : Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku

2. Waktu seminar ditetapkan oleh Panitia Seminar Hasil Program Strata Satu (S1)
3. Agar Surat Penugasan ini dilaksanakan sebaik-baiknya dengan penuh rasa tanggung jawab.
4. Surat penugasa ini berlaku sejak tanggal ditetapkan sampai dengan berakhirnya seminar tersebut dengan ketentuan bahwa segala sesuatunya akan ditinjau dan diperbaiki sebagaimana mestinya apabila dikemudia hari terdapat kekeliruan dalam keputusan ini.

Ditetapkan di Gowa
Pada tanggal 2 April 2024
a.n. Dekan,
Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Unhas

Dr. Amil Ahmad Ilham, ST., M.IT
NIP. 197310101998021001

Tembusan :
1. Dekan Fak. Teknik Unhas
2. Ketua Departemen Teknik Informatika FT-UH
3. Mahasiswa yang bersangkutan

KEMENTERIAN PENDIDIKAN , KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus FakultasTeknikUnhas, Jl. PorosMalino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

DAFTAR HADIR UJIAN SKRIPSI MAHASISWA FAKULTAS TEKNIK UNHAS

Nama/Stambuk : 1. .Chindy Christie Davina D121201077

Judul Skipsi/T.A : **"Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku"**

Hari/Tanggal : Rabu, 8 Mei 2024
Jam : 10- 30 Wita – Selesai
Tempat : Ruang Lab. AIMP Departemen Teknik Informatika Gowa

No.	Jabatan	Nama Dosen	Tanda Tangan
L.	Pembimbing I	1. Dr.Ir.Ingrid Nurtanio,M.T	1.....
II.	Anggota Penguji	3. MukarramahYusuf,B.Sc.,M.Sc.,Ph.D	3 .....
		4. Anugrayani Bustamin,ST.,M.T	4 .....

PANITIA UJIAN

Ketua,

Dr.Ir.Ingrid Nurtanio,M.T

KEMENTERIAN PENDIDIKAN , KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus FakultasTeknikUnhas, Jl. PorosMalino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

BERITA ACARA UJIAN SKRIPSI

Pada hari ini Rabu, tanggal 8 Mei 2024 Pukul **10.30 WITA - Selesai** bertempat di **Lab. AIMP Departemen Teknik Informatika Gowa** , telah dilaksanakan Ujian Skripsi bagi Saudara :

Nama : Chindy Christie Davina
No. Stambuk : D121201077
Fakultas/Departemen : Teknik /Teknik Informatika
Judul Skripsi : **"Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku"**

Yang dihadiri oleh Tim Penguji Ujian Skripsi sebagai berikut :

No.	N a m a	Jabatan	Tanda tangan
1.	Dr.Ir.Ingrid Nurtanio,M.T	Pemb I/Ketua	1.
2.	MukarramahYusuf,B.Sc.,M.Sc.,Ph.D	Anggota	2.
3.	Anugrayani Bustamin,ST.,M.T	Anggota	3.

Hasil keputusan Tim Penguji Ujian Skripsi/Tugas Akhir : **Lulus** / ~~**Tidak lulus**~~ dengan nilai angka .....88..... dan huruf .....A.....

Gowa, 8 Mei 2024

Ketua/Sekretaris Panitia Ujian,

Dr.Ir.Ingrid Nurtanio,M.T

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus Fakultas Teknik Unhas, Jl. Poros Malino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

Gowa, 2 Mei 2024

Nomor : 559/UN4.7.7.1/TD.06/2024
Lamp : -
Hal : Usulan Susunan Panitia/Penguji Ujian Sarjana

Yth. : Bapak Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Unhas
Di
Gowa

Dalam rangka penyelesaian studi pada Departemen Teknik Informatika Fakultas Teknik Unhas, bersama ini kami usulkan susunan Panitia/Penguji Ujian Sarjana Program Strata Satu (S1) bagi mahasiswa Departemen Teknik Informatika Fakultas Teknik Uniersitas Hasanuddin atas nama :

Pembimbing I / Ketua : 1. Dr. Ir. Ingrid Nurtanio, M.T
Penguji / Anggota : 2. Mukarramah Yusuf, B.Sc., M.Sc., Ph.D
: 3. Anugrayani Bustamin, ST., M.T

Untuk Bertugas sebagai Penguji/ Penanggap Ujian Sarjana bagi Mahasiswa :

Nama : Chindy Christie Davina
Stambuk : D121 20 1077

Dengan Judul Skripsi
" Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku "

Pada :
Hari/Tanggal : Rabu, 8 Mei 2024
Jam : 10.30 Wita - Selesai
Tempat : Ruang Sidang Lab. AIMP

Demikian penyampaian kami, atas perhatiannya diucapkan terimah kasih.

Ketua Departemen Tek.Informatika,

Prof. Dr. Ir. Indrabayu.,ST, MT, M.Bus.Sys., IPM, ASEAN.Eng
Nip.197507016 200212 1 004

Tembusan :
1. Arsip

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
Poros Malino Km.6Bontomarannu(92172) Gowa, Sulawesi Selatan 92172, Sulawesi Selatan
Telp. (0411) 586015, 586262 Fax (0411) 586015
http://eng.unhas.ac.id, Email : teknik@unhas.ac.id

SURAT PENUGASAN

No. 9569/UN4.7.1/TD.06/2024

Dari : Dekan Fakultas Teknik Universitas Hasanuddin.

Kepada. : Mereka yang tercantum namanya di bawah ini.

Isi : 1. Bahwa merujuk kepada Peraturan Rektor Universitas Hasanuddin **Nomor : 29/UN4.1/2023 tentang Penyelenggaraan Program Sarjana Universitas Hasanuddin**, dengan ini menugaskan Saudara sebagai PENGUJI/PANITIA UJIAN SARJANA Program Strata Satu (S1) Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin dengan susunan sebagai berikut :

Pembimbing I / Ketua : 1. Dr. Ir. Ingrid Nurtanio, M.T
Penguji / Anggota : 2. Mukarramah Yusuf, B.Sc., M.Sc., Ph.D
: 3. Anugrayani Bustamin, ST., M.T

untuk menguji bagi mahasiswa tersebut di bawah ini :

Nama/NIM : Chindy Christie Davina D121 20 1077
Program Studi : Teknik Informatika
Judul Thesis/Skripsi : " Implementasi Algoritma Rapid Automatic Keyword Extraction (RAKE) pada Pembuatan Indeks Buku "

2. Waktu Ujian ditetapkan oleh Panitia Ujian Sarjana Program Strata Satu (S1).
3. Agar Surat penugasan ini dilaksanakan sebaik-baiknya dengan penuh rasa tanggung jawab.
4. Surat penugasan ini berlaku sejak tanggal ditetapkan sampai dengan berakhirnya Ujian Sarjana tersebut, dengan ketentuan bahwa segala sesuatunya akan ditinjau dan diperbaiki sebagaimana mestinya apabila dikemudian hari ternyata terdapat kekeliruan dalam keputusan ini.

Ditetapkan di Gowa,
Pada tanggal 2 Mei 2024
a.n. Dekan
Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Unhas

Dr. Amil Ahmad Ilham, ST., M.IT
NIP.197310101998021001

Tembusan :
1. Dekan Fak. Teknik Unhas
2. Ketua Departemen Teknik Informatika FT-UH
3. Kasubag. Umum dan Perlengkapan FT-UH