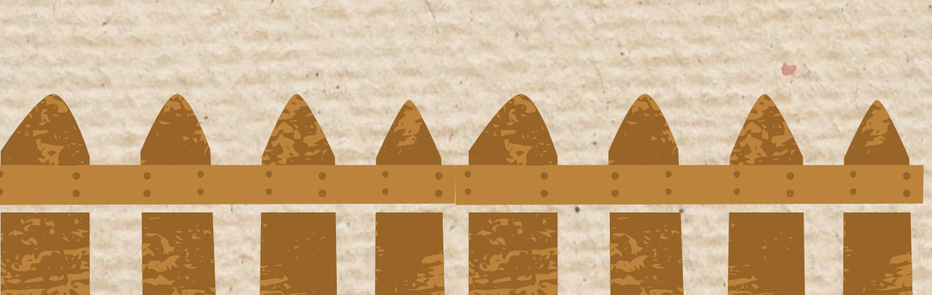
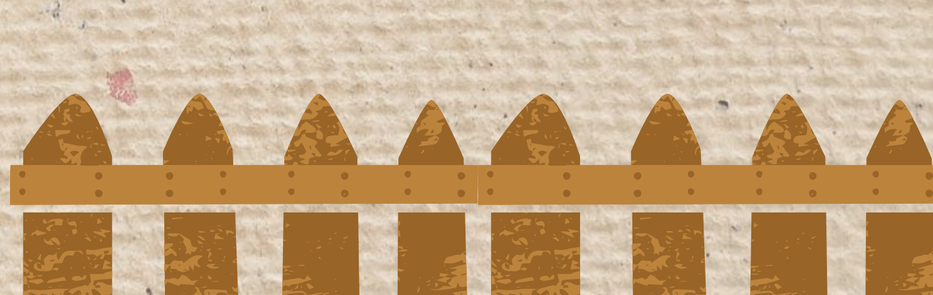
Data Science Week 2023

Fertilizer Company Product Mapping

Using Unsupervised Model





SobatNongsBekasi

Anzaldi Sulaiman Oemar

Computer Science – Institut Teknologi Bandung

Data Specialist – Sharing Vision Indonesia

Naufal Aditya Dirgandhavi

Computer Science – Institut Teknologi Bandung

Software Engineer – Samsung R&D Institute

Muhammad Athallah Rizki Putra

Telecommunication Engineering – Institut Teknologi Bandung

Network Engineer – PT. Bank Central Asia Tbk.

Background

A company that sells various fertilizer products just recently listed their product catalogs. But because the company are already running long before in many stores, that more than often not just sells fertilizer products, they have to map product names from various sources to the catalogs listed.

Data & Methodology

Datasets

There are two given datasets, Product Catalogs and Product Names

Datasets

Product Catalog

This dataset contains every product in the Stock Keeping Unit (SKU). It consists of Product SKU Name, Brand, Type, and Formula.

Product SKU	Brand	Type	Formula
Phonska Plus 15-15-15+9S+0.2Zn	PIHC	Majemuk	15-15-15
NPK Kebomas 12-12-17+2MgO+0.1Zn+0.2B+0.2Fe	PIHC	Majemuk	12-12-17
NPK Kebomas 12-6-22+3Mg	PIHC	Majemuk	12-6-22
NPK Kebomas 15-15-15	PIHC	Majemuk	15-15-15
PETROFERT 16-16-8+13S	PIHC	Majemuk	16-16-8
Petro Niphos 20-20+13S	PIHC	Majemuk	20-20-0
FERTIGRES 16-20+13S	PIHC	Majemuk	16-20-0
NPK PIM 15-15-15	PIHC	Majemuk	15-15-15
Nitroku 16-16-16	PIHC	Majemuk	16-16-16
NITROSKA 15-9-20+1.5MgO+3S	PIHC	Majemuk	15-9-20
NPK Kujang 15-15-15	PIHC	Majemuk	15-15-15
NPK Kujang 30-6-8	PIHC	Majemuk	30-6-8
Polivit-PIM 0-0-12+48S+9MgO	PIHC	Kalium	
Solution N 28-10-10 + TE	PIHC	Majemuk	28-10-10
Pelangi Agro 20-10-10	PIHC	Majemuk	20-10-10
Pelangi Jos 16-16-16 Coating Mikroba	PIHC	Majemuk	16-16-16
Pelangi 16-16-16	PIHC	Majemuk	16-16-16
Pelangi 15-15-15	PIHC	Majemuk	15-15-15
NPK Pusri 15-15-15	PIHC	Majemuk	15-15-15
Pelangi 20-10-18	PIHC	Majemuk	20-10-18
Pelangi 12-12-17-2	PIHC	Majemuk	12-12-17
NPK Pelangi 13-6-27-4	PIHC	Majemuk	13-6-27
Petro Nitrat 16-16-16	PIHC	Majemuk	16-16-16
Jeranti 18-10-14+2S+TE	PIHC	Majemuk	18-10-14
NPK Pusri 16-16-16	PIHC	Majemuk	16-16-16
NPK Pusri 13-6-27-4Mg+0.65B	PIHC	Majemuk	13-6-27

Datasets

Product Names

This dataset consists of free-text input products names from Point of Sale Transaction

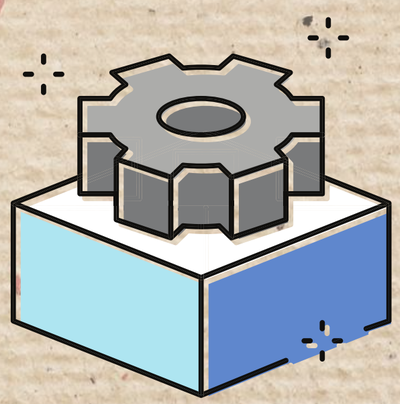
Product Name
Pupuk Urea N 46%
Pupuk Amonium Sulfat ZA
Pupuk Super Fosfat SP-36
Pupuk NPK Phonska
Pupuk NPK Formula Khusus
Pupuk Organik Granul
Pupuk Organik Cair
Produk Lain
Rondap
Sekor
abacel 250ml
nitrea
spontan
Starban
gramoxon
puradan
pastak
combitox
fujiwan
buldok
starban 100ml
postin/100ml
fastak
trebon
Ronsha
sidamethrin

Methodology

Steps

There are three main steps to solve the problems. We use python that is run in Google Collab

Preprocessing



Creating Model

Testing Model

Preprocessing

Vectorize Character

Convert a collection of character to a matrix of token counts.

```
def get_count_single_char(libraries):
    single_char_vectorizer = CountVectorizer(analyzer='char', ngram_range=(1,1))
    single_char_weights = single_char_vectorizer.fit_transform(libraries)
    charss = list(single_char_vectorizer.get_feature_names_out())

    chars_df = pd.DataFrame(single_char_weights.toarray(), columns=charss)
    chars_df_sum = chars_df.sum(axis=0)

    print(chars_df_sum.to_string())
```

	285
(	5
)	5
+	90
-	320
.	43
0	125
1	258
2	133
3	55
4	34
5	108
6	100
7	30
8	30
9	17

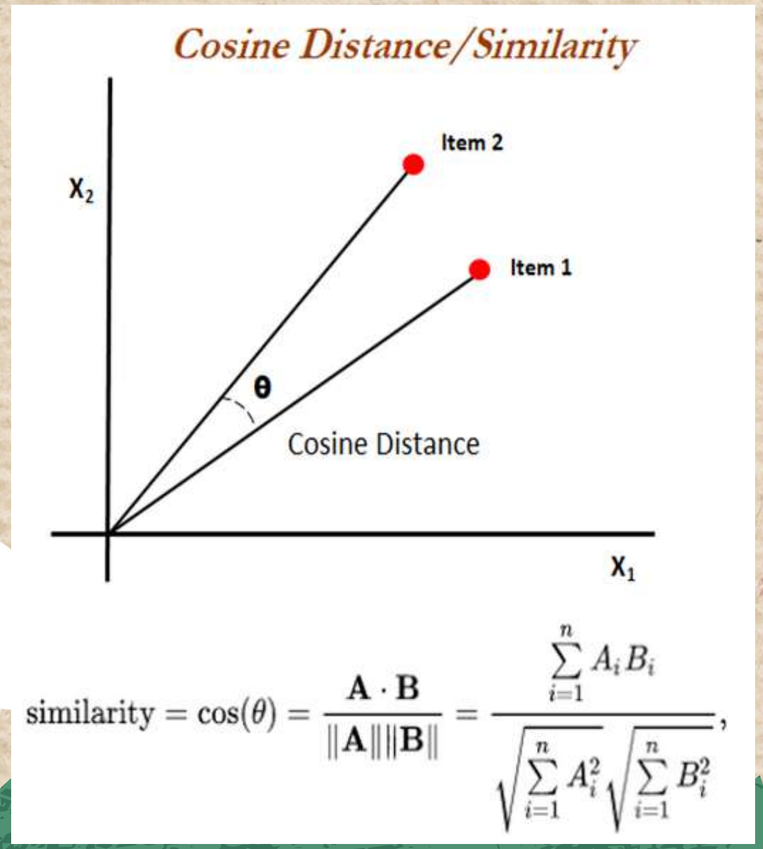
a	269
b	36
c	64
d	22
e	183
f	29
g	54
h	44
i	149
j	5
k	114
l	58
m	112
n	118
o	142
p	138
q	1
r	140
s	112
t	195
u	46
v	8
w	21
x	7
y	13
z	15

Preprocessing

Cleaning String

Find strange character and replace it with proper char

```
def get_clean_string(p_word, debug_mode=False):
    local_replacer = {
        "\\xa": " ",
        "ù": "u",
        "⁰": "0",
        "¹": "1",
        "²": "2",
        "³": "3",
    }

    for (k, v) in local_replacer.items():
        p_word = p_word.replace(k, v)
    if debug_mode: print(f"Penemuan simbol-simbol mirip                                  : {p_word}")

    p_word = re_partition_sub(r"[^0-9]1/4[^0-9]", r"1/4", "¼", p_word)
    p_word = re_partition_sub(r"[^0-9]1/2[^0-9]", r"1/2", "½", p_word)
    p_word = re_partition_sub(r"[^0-9]3/4[^0-9]", r"3/4", "¾", p_word)
    if debug_mode: print(f"Penemuan simbol pecahan                                      : {p_word}")

    p_word = p_word.lower()
    if debug_mode: print(f".lower() semua huruf                                          : {p_word}")

    p_word = re.sub(r"[^0-9a-z¼½¾]+", " ", p_word)
    if debug_mode: print(f"Replace non angka atau huruf                                 : {p_word}")

    p_word = re_partition_ins(r"[a-z][0-9¼½¾]", " ", 1, p_word)
    p_word = re_partition_ins(r"[0-9¼½¾][a-z]", " ", 1, p_word)
    if debug_mode: print(f"Pisahkan char angka yg menempel char huruf dan sebaliknya : {p_word}")

    p_word_splitter = p_word.split()
    #if debug_mode: print(p_word_splitter)
    p_word_splitter.sort()
    #if debug_mode: print(p_word_splitter)
    #p_word = " " + " ".join(p_word_splitter) + " "
    p_word = " ".join(p_word_splitter)
    if debug_mode: print(f"Urutkan kata atau bilangan yg sudah terpisah spasi         : {p_word}\n")

    #p_word = re_partition_sub(r"[a-z].*[a-z]", " ", "", p_word)
    #if debug_mode: print(p_word)

    return p_word
```



Creating Model

For the model, we use Cosine Similarity algorithm

Cosine similarity is a measure of similarity between two non-zero vectors defined in an inner product space.

```python
def get_cosine_similarity_score(vec1, vec2):
    # ASUMSI PANJANG SELALU SAMA
    norm = math.sqrt(np.dot(vec1, vec1) * np.dot(vec2, vec2))
    similarity_score = round((np.dot(vec1, vec2) / norm), 4)
    return similarity_score
```

```python
def get_similarity_score(vec1, vec2, formula='div', solve_both_zero=True, debug_mode=False):
    formulas = ["and", "cosine", "div", "pivot_first", "pivot_last"]
    if (formula in formulas) and (len(vec1) == len(vec2)):
        if (formula != "cosine"):
            vec_result = [
                int((v1 == 0) == (v2 == 0)) for (v1, v2) in zip(vec1, vec2)
            ] if (formula == "and") else [
                int(solve_both_zero) if ((v1 == 0) and (v2 == 0))
                else round(min(v1,v2)/max(v1,v2), 4) if (formula == "div")
                else 1.0 if ((v2 >= v1) and (v1 > 0) and (formula == "pivot_first"))
                else round(v2/v1, 4) if ((v2 < v1) and (formula == "pivot_first"))
                else 1.0 if ((v1 >= v2) and (v2 > 0) and (formula == "pivot_last"))
                else round(v1/v2, 4) if ((v1 < v2) and (formula == "pivot_last"))
                else 0.0 for (v1, v2) in zip(vec1, vec2)
            ]
        similarity_score = (
            get_cosine_similarity_score(vec1, vec2) if (formula == "cosine")
            else round(sum(vec_result)/len(vec_result), 4)
        )
    else:
        vec_result = []
        similarity_score = 0.0

    if debug_mode:
        print(f"\nRumus similarity yang digunakan : {formula.upper()}")
        if (formula in ["div", "pivot_first", "pivot_last"]):
            print(f"Handle elemen nol (0 vs 0)      : {str(solve_both_zero)}")
        if (formula != "cosine"):
            print(list(vec1))
            print(list(vec2))
            print(vec_result)
        print(similarity_score)

    return similarity_score
```

Testing Model

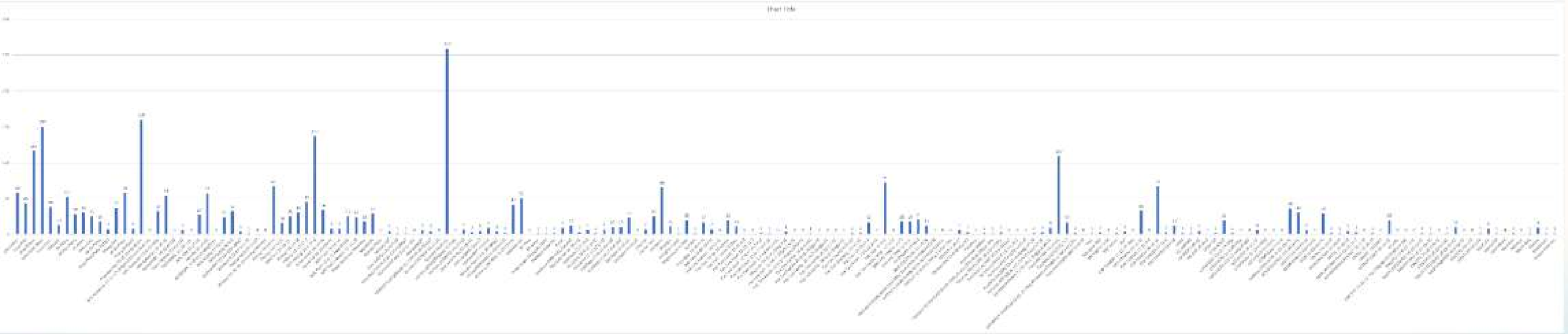
```
[ ]  def get_my_predict(
         p_string, targets, min_threshold=0.0001, formula='div',
         solve_both_zero=True, debug_mode=False
     ):
         start_dt = time.time()
         search = [(s, get_string_similarity_score(p_string, s, formula, solve_both_zero, False)) for s in targets]
         closest_res = max(search, key=lambda x: x[1])
         condition = (closest_res[1] > 0) and (closest_res[1] >= min_threshold)
         closest_str = (closest_res[0] if condition else "")
         closest_score = (closest_res[1] if condition else 0)

         end_dt = time.time()
         delta = round(end_dt - start_dt, 5)
         if debug_mode:
             condition = (closest_res[1] > 0)
             debug_str = (closest_res[0] if condition else "")
             debug_score = (
                 get_string_similarity_score(
                     p_string, closest_res[0], formula, solve_both_zero, debug_mode
                 ) if condition else 0
             )
             print(f"\nWaktu eksekusi pencarian : {round(1000*delta)} ms")
             if (closest_res[1] < min_threshold):
                 print(f"Target tidak ditemukan! Tingkat kemiripan belum memenuhi batas minimal, perlu lebih dari atau sama dengan {round(100*min_threshold)}%")

         return (closest_str, closest_score)
```

```python
def extract_batch_predict(
    datasets,
    targets,
    min_threshold=0.0001,
    formula='div',
    solve_both_zero=True,
    batch_size=557,
    filename='RESULTS SIMILARITY VERSION/transactions_prediction',
    first_file_id=0,
    last_file_id=sys.maxsize,
    sample_split=False,
    rand_state=0,
    debug_mode=False
):
    min_file_id = 0
    max_file_id = math.ceil(datasets.shape[0]/batch_size)
    max_row_id = datasets.shape[0]

    local_first_file_id = (min(first_file_id, max_file_id) if (first_file_id > 0) else min_file_id)
    local_last_file_id = (min(last_file_id, max_file_id) if (last_file_id > 0) else max_file_id)
    file_count = local_last_file_id - local_first_file_id

    first_row_id = min(local_first_file_id * batch_size, max_row_id)
    last_row_id = min(first_row_id + file_count*batch_size, max_row_id)
    delta_row_id = last_row_id - first_row_id
    row_count = min(delta_row_id, max_row_id)
    local_batch_size = (min(batch_size, row_count) if (batch_size > 0) else row_count)

    if debug_mode:
        print(f"{{min_file_id}}: {min_file_id}")
        print(f"{{first_file_id}}: {local_first_file_id}")
        print(f"{{last_file_id}}: {local_last_file_id}")
        print(f"{{max_file_id}}: {max_file_id}")
        print(f"{{file_count}}: {file_count}")
        print()
        print(f"{{first_row_id}}: {first_row_id}")
        print(f"{{last_row_id}}: {last_row_id}")
        print(f"{{max_row_id}}: {max_row_id}")
        print()
```

```python
print(f"\nJob executing {row_count} rows started at {datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]}\n")
remark = ""

if sample_split:
    local_df = datasets.sample(n=last_row_id, random_state=rand_state)
    remark = f"{remark}randstate{rand_state:03}_"
else:
    local_df = datasets

fid = local_first_file_id
rid = first_row_id
while (fid < local_last_file_id):
    next_fid = fid + 1
    next_rid = min(rid + local_batch_size, max_row_id)
    local_filename=f"{filename}_{remark}{local_batch_size}rows_{(next_fid):04}.csv"

    if (not debug_mode):
        df_batch = local_df.iloc[rid:next_rid].copy()
        df_batch['predict_product_name_to_sku'] = df_batch['Product Name'].apply(lambda x: get_my_predict(
            x, targets['Product SKU'], min_threshold, formula, solve_both_zero
        ))
        df_batch[['closest_product_sku', 'similarity_to_closest_product_sku']] = pd.DataFrame(
            list(df_batch['predict_product_name_to_sku']), index=df_batch.index
        )
        df_batch.drop(columns=['predict_product_name_to_sku'])
        #selected_cols = ['id', 'Product Name', 'closest_product_sku', 'similarity_to_closest_product_sku']
        df_batch_joined = pd.merge(
            df_batch.reset_index(names='id'), targets, how='left',
            left_on='closest_product_sku', right_on='Product SKU', sort=False
        )
        df_batch_joined.to_csv(local_filename, sep=";", index=False, quoting=csv.QUOTE_NONNUMERIC)

    print(f"File [{local_filename}] for rowid[{rid}:{next_rid}]\ngenerated at {datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]}\n")
    fid, rid = next_fid, next_rid

print(f"\nJob succeeded at {datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]}\n")
#res = ([] if debug_mode else df_batch_joined)
#return res
```

Results

To map the unmapped categories, we make ‘Misc’ category. Turns out the product dataset given have more than 90% unmapped products which is most of them are not a fertilizer product. To make a better visualization, we decide to use line chart that represent the frequencies of the categories and remove ‘Misc’ category from it.



Results

id	Product Name	similarity_	Product SKU	Brand	Type	Formula
0	Pupuk Urea N 46%	0.350572	Urea PIM	PIHC	Urea	
1	Pupuk Amonium Sulfat ZA	0.560898	Ammonium Sulfate	Yara	ZA	
2	Pupuk Super Fosfat SP-36	0.246922	Triple Super Phospate (TSP)	Mahkota	Fosfat	
3	Pupuk NPK Phonska	0.388887	Phonska Plus 15-15-15+9S+0.2Zn	PIHC	Majemuk	15-15-15
4	Pupuk NPK Formula Khusus	0.156481	NPK Pusri 15-15-15	PIHC	Majemuk	15-15-15
5	Pupuk Organik Granul	0.160102	NPK Petro Ningrat 12-11-20	PIHC	Majemuk	12-11-20
6	Pupuk Organik Cair	0.180769	Pak Tani 16-16-16 Biru	Pak Tani	Majemuk	16-16-16
7	Produk Lain	0.217391	Nitroplus (ZA)	Tawon	ZA	
8	Rondap	0.222222	HX-DAP 18-46-0	DGW/Hextar	Majemuk	18-46-0
9	Sekor	0.166667	MerokeTSP	Mutiara	Fosfat	
10	abacel 250ml	0.139423	COCKHEAD 13-6-27-4Mg+0.65B	DGW/Hextar	Majemuk	13-6-27
11	nitrea	0.619048	Urea Nitrea	PIHC	Urea	
12	spontan	0.213179	Pak Tani Singkong 15-15-15	Pak Tani	Majemuk	15-15-15
13	Starban	0.224138	Mestac	LaoYing	Nitrogen	
14	gramoxon	0.145161	Borat	Mahkota	Mikro	
15	puradan	0.277778	ULTRADAP 12-60-0	Pak Tani	Majemuk	12-60-0
16	pastak	0.256412	ZA Pak Tani	Pak Tani	ZA	
17	combitox	0.166667	Brucite	BASF	Mg	
18	fujiwan	0.148649	NPK Kujang 15-15-15	PIHC	Majemuk	15-15-15
19	buldok	0.114286	Nitroku 16-16-16	PIHC	Majemuk	16-16-16
20	starban 100ml	0.180556	Mestac	LaoYing	Nitrogen	
21	postin/100ml	0.24365	MESTIFOS 15-20-0+12S	LaoYing	Majemuk	15-20-0
22	fastak	00.25	Mestac	LaoYing	Nitrogen	
23	trebon	00.24	Borat	Mahkota	Mikro	
24	Ronsha	0.183333	Phosgro	Pak Tani	Fosfat	
25	sidamethrin	0.170732	Mestac	LaoYing	Nitrogen	

Conclusion

We decide to use similarity measure because the goal is just simply to map existing 4000 row of products to 180 categories.

For Future Development, we can use supervised learning to predict products to the existing categories, which can also predict new input in the future. But we need to add another step which is to give label to the data, train, and test the data.