source

April 28, 2024

[ ]:
```
!wget -q http://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.
↪1-bin-hadoop3.2.tgz
```

[ ]:
```
!tar xf spark-3.1.1-bin-hadoop3.2.tgz
```

[ ]:
```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

[ ]:
```
!pip install -q findspark
```

[ ]:
```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
import findspark as fs
fs.init()
```

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
dataset = "/content/drive/MyDrive/Colab Notebooks/MMDS/QT2/WebOfScience-5736.
↪txt"
```

1 Task 1: In-memory MinHashLSH

[4]:
```
import pandas as pd
import numpy as np
from random import shuffle
```

[5]:
```
class InMemoryMinHashLSH():
    def __init__(self, documents: pd.DataFrame) :
        self.documents = documents
        self.shingle_size = 2
        self.bool_vectors = None
        self.num_hashes = 100
        self.permutations = []
```

```
        self.signatures = None
        self.lsh_buckets = None
        self.num_bands = 25
        self.shingles_set = set()
        self.similarity_threshold = 0.5

    def shingling(self, documents : pd.DataFrame):
        for doc in documents['text']:
            for i in range(len(doc) - self.shingle_size + 1):
                shingle = doc[i:i+self.shingle_size]
                self.shingles_set.add(shingle)
        bool_vectors = []
        for doc in documents['text']:
            document_vector = [1 if shingle in doc else 0 for shingle in self.
↪shingles_set]
            bool_vectors.append(document_vector)
        return bool_vectors

    def create_permutation(self, size):
        permutation = list(range(1, size + 1))
        shuffle(permutation)
        return permutation

    def minhash(self, bool_vector, permutation):
        minhash_value = float('inf')
        for idx, val in enumerate(bool_vector):
            if val == 1:
                minhash_value = min(minhash_value, permutation[idx])
        return minhash_value

    def minhashing(self,bool_vectors):
        num_docs = len(bool_vectors)
        num_features = len(bool_vectors[0])
        signatures = np.zeros((self.num_hashes, num_docs),dtype=int)
        for i in range(self.num_hashes):
            if len(self.permutations) <= i:
                self.permutations.append(self.create_permutation(num_features))
            permutation = self.permutations[i]
            # print(f"Permutation {i+1}: {permutation}")
            for j in range(num_docs):
                signatures[i, j] = self.minhash(self.bool_vectors[j],␣
↪permutation)
        return signatures

    def jaccard_similarity(self,sequence1, sequence2):
        intersection_count = 0
        union_count = len(sequence1)
```

```
        for elem1, elem2 in zip(sequence1, sequence2):
            if elem1 == elem2:
                intersection_count += 1
        similarity = intersection_count / union_count
        return similarity

    def locality_sensity_hashing(self, signatures):
        num_buckets = 102233
        r = self.num_hashes // self.num_bands
        num_docs = signatures.shape[1]
        results = {}
        for band in range(self.num_bands):
            start_row = band * r
            end_row = (band + 1) * r
            for col in range(num_docs):
                band_signature = signatures[start_row:end_row, col]
                hash_value = hash(tuple(band_signature)) % num_buckets
                signature_str = ''.join(map(str, band_signature))
                signature_hash_str = f"{signature_str}-{hash_value}"
                if col not in results:
                    results[col] = []
                results[col].append(signature_hash_str)
        return results

    def run(self):
        self.bool_vectors = self.shingling(self.documents)
        self.signatures = self.minhashing(self.bool_vectors)
        self.lsh_buckets = self.locality_sensity_hashing(self.signatures)


    def approxNearestNeighbors(self, key, n):
        textqueried_df = pd.DataFrame({"text": [key]})
        bool_vectors_queried=self.shingling(textqueried_df)
        signature_queried = np.zeros((self.num_hashes, 1), dtype=int)
        for i in range(self.num_hashes):
            permutation = self.permutations[i]
            signature_queried[i, 0] = self.minhash(bool_vectors_queried[0],␣
↪permutation)
        lsh_signature_queried = self.locality_sensity_hashing(signature_queried)
        list_query_bucket_id = [item.split('-')[1] for item in [item for␣
↪sublist in list(lsh_signature_queried.values()) for item in sublist]]
        candidate_docs = []
        for doc_id, lsh_signatures in self.lsh_buckets.items():
            current_bucket_ids = [item.split('-')[1] for item in lsh_signatures]
            if any(bucket_id in list_query_bucket_id for bucket_id in␣
↪current_bucket_ids):
                candidate_docs.append(doc_id)
```

```
        similarities = []
        for doc_id in candidate_docs:
            signature_set_queried =  [item for sublist in signature_queried.T␣
↪for item in sublist]
            lsh_signature_existing= self.signatures.T[doc_id]
            jaccard_similarity = self.jaccard_similarity(signature_set_queried,␣
↪lsh_signature_existing)
            if jaccard_similarity >= self.similarity_threshold:
                similarities.append((doc_id, jaccard_similarity))
        similarities.sort(key=lambda x: -x[1])
        return similarities[:n]
```

[20]:
```
with open(dataset) as file:
    lines = file.readlines()
dataframes = pd.DataFrame(lines, columns=['text'])
dataframes['text'] = dataframes['text'].str.strip()
minhash_lsh = InMemoryMinHashLSH(dataframes)
minhash_lsh.run()
```

[7]:
```
dataframes.head()
```

[7]:

	text
0	Phytoplasmas are insect-vectored bacteria that…
1	Background: (-)-alpha-Bisabolol, also known as…
2	A universal feature of the replication of posi…
3	1,2-Dichloropropane (1,2-DCP) and dichlorometh…
4	This paper presents the simulation results of …

[13]:
```

```

```
query = "Many recent publications highlight the large role of the pivotal␣
 ↪eukaryotic nuclear export protein exportin-1 (XPO1) in the oncogenesis of␣
 ↪several malignancies, and there is emerging evidence that XPO1 inhibition is␣
 ↪a key target against cancer. The clinical validation of the pharmacological␣
 ↪inhibition of XPO1 was recently achieved with the development of the␣
 ↪selective inhibitor of nuclear export compounds, displaying an interesting␣
 ↪anti-tumor activity in patients with massive pre-treated hematological␣
 ↪malignancies. Recent reports have shown molecular alterations in the gene␣
 ↪encoding XPO1 and showed a mutation hotspot (E571K) in the following two␣
 ↪hematological malignancies with similar phenotypes and natural histories:␣
 ↪primary mediastinal diffuse large B cell lymphoma and classical Hodgkin's␣
 ↪lymphoma. Emerging evidence suggests that the mutant XPO1 E571K plays a role␣
 ↪in carcinogenesis, and this variant is quantifiable in tumor and plasma␣
 ↪cell-free DNA of patients using highly sensitive molecular biology␣
 ↪techniques, such as digital PCR and next-generation sequencing. Therefore,␣
 ↪it was proposed that the XPO1 E571K variant may serve as a minimal residual␣
 ↪disease tool in this setting. To clarify and summarize the recent findings␣
 ↪on the role of XPO1 in B cell hematological malignancies, we conducted a␣
 ↪literature search to present the major publications establishing the␣
 ↪landscape of XPO1 molecular alterations, their impact on the XPO1 protein,␣
 ↪their interest as biomarkers, and investigations into the development of new␣
 ↪XPO1-targeted therapies in B cell hematological malignancies."
n = 3 # số lượng tài liệu tương tự
document_similarity = minhash_lsh.approxNearestNeighbors(query, 3) #5720
print(document_similarity)
```

```
[(5720, 1.0), (3741, 0.67), (4104, 0.67)]
```

[21]:

```
query_1 = "Background: (-)-alpha-Bisabolol, also known as levomenol, is an␣
 ↪unsaturated sesquiterpene alcohol that has mainly been used in␣
 ↪pharmaceutical and cosmetic products due to its anti-inflammatory and␣
 ↪skin-soothing properties. (-)-alpha-Bisabolol is currently manufactured␣
 ↪mainly by steam-distillation of the essential oils extracted from the␣
 ↪Brazilian candeia tree that is under threat because its natural habitat is␣
 ↪constantly shrinking. Therefore, microbial production of (-)-alpha-bisabolol␣
 ↪plays a key role in the development of its sustainable production from␣
 ↪renewable feedstock. Results: Here, we created an Escherichia coli strain␣
 ↪producing (-)-alpha-bisabolol at high titer and developed an in situ␣
 ↪extraction method of (-)-alpha-bisabolol, using natural vegetable oils. We␣
 ↪expressed a recently identified (-)-alpha-bisabolol synthase isolated from␣
 ↪German chamomile (Matricaria recutita) (titer: 3 mg/L), converted the␣
 ↪acetyl-CoA to mevalonate, using the biosynthetic mevalonate pathway (12.8 mg/
 ↪L), and overexpressed farnesyl diphosphate synthase to efficiently supply␣
 ↪the (-)-alpha-bisabolol precursor farnesyl diphosphate. Combinatorial␣
 ↪expression of the exogenous mevalonate pathway and farnesyl diphosphate␣
 ↪synthase enabled a dramatic increase in (-)-alpha-bisabolol production in␣
 ↪the shake flask culture (80 mg/L) and 5 L bioreactor culture (342 mg/L) of␣
 ↪engineered E. coli harboring (-)-alpha-bisabolol synthase. Fed-batch␣
 ↪fermentation using a 50 L fermenter was conducted after optimizing culture␣
 ↪conditions, resulting in efficient (-)-alpha-bisabolol production with a␣
 ↪titer of 9.1 g/L. Moreover, a green, downstream extraction process using␣
 ↪vegetable oils was developed for in situ extraction of (-)-alpha-bisabolol␣
 ↪during fermentation and showed high yield recovery (>98%). Conclusions: The␣
 ↪engineered E. coli strains and economically viable extraction process␣
 ↪developed in this study will serve as promising platforms for further␣
 ↪development of microbial production of (-)-alpha-bisabolol at large scale."
query_20 = "3D printing has shown promise for neural regeneration by providing␣
 ↪customized nerve scaffolds to structurally support and bridge the defect gap␣
 ↪as well as deliver cells or various bioactive substances. Low-level light␣
 ↪therapy (LLLT) exhibits positive effects on rehabiliation of degenerative␣
 ↪nerves and neural disorders. With this in mind, we postulate that 3D printed␣
 ↪neural scaffold coupling with LLLT will generate a new strategy to repair␣
 ↪neural degeneration. To achieve this goal, we applied red laser light to␣
 ↪stimualte neural stem cells on 3D printed scaffolds and investigated the␣
 ↪subsequent cell response with respect to cell proliferation and␣
 ↪differentiation. Here we show that cell prolifeartion rate and intracellular␣
 ↪reactive oxgen species synthesis were significantly increased after 15 s␣
 ↪laser stimulation follwed by 1 d culture. Over culturing time of 14 d in␣
 ↪vitro, the laser stimulation promoted neuronal differentiation of neural␣
 ↪stem cells, while the glial differentiation was suppressed based on results␣
 ↪of both immunocytochemistry studies and real-time quantitative reverse␣
 ↪transcription polymerase chain reaction testing. These findings suggest that␣
 ↪integration of 3D printing and LLLT might provide a powerful methodology for␣
 ↪neural tissue engineering."
query = query_1+query_20
```

```
n = 5 # số lượng tài liệu tương tự
document_similarity = minhash_lsh.approxNearestNeighbors(query, 3) #1,20
print(document_similarity)
```

```
[(1, 0.88), (20, 0.65), (2479, 0.64)]
```

2 Task 2: LargDataMinHashLSH

[ ]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr, explode, collect_set, array, col, udf,␣
↪lit, collect_list, concat, desc, row_number, split, \
                                    array_intersect,␣
↪monotonically_increasing_id, max as spark_max, min as spark_min
from pyspark.sql import DataFrame
from pyspark.sql.types import  ArrayType, StringType, BooleanType, IntegerType,␣
↪FloatType
from pyspark.sql.window import Window
import random
```

[ ]:
```
spark = SparkSession.builder \
    .appName("QT2") \
    .getOrCreate()
```

[ ]:
```
class LargDataMinHashLSH():
    def __init__(self, documents: DataFrame, shingles_size = 3, num_hashes =␣
↪100, num_bands = 50, num_buckets = 1024, similarity_threshold = 0.6 ):
        self.documents = documents
        self.shingles_size = shingles_size
        self.shingles_vector = None
        self.shingles = None
        self.bool_vectors = None
        self.num_hashes = num_hashes
        self.signature_matrix = None
        self.buckets = None
        self.num_bands = num_bands
        self.num_buckets = num_buckets
        self.similarity_threshold = similarity_threshold
        self.lsh = None
        self.indexed_df =  None

    def shingling(self, documents: DataFrame):
        shingles_size = self.shingles_size
        shingling_udf = udf(lambda text: [text[i:i+shingles_size] for i in␣
↪range(len(text)-shingles_size+1)], ArrayType(StringType()))
        shingles_vector = documents.withColumn("shingles",␣
↪shingling_udf(col("document"))).cache()
```

```
        shingles = shingles_vector.select(explode("shingles") \
                                          .alias("shingle")).groupBy() \
                                          .agg( collect_set("shingle") \
                                          .alias("vocab")).
↪collect()[0]["vocab"]
        shingles_vector.unpersist()
        return shingles

    def bool_vector(self, documents: DataFrame, shingles):
        bool_vectors_udf = udf(lambda text: [True if shingle in text else False␣
↪for shingle in shingles] , ArrayType(BooleanType()))
        return documents.withColumn("bool_vector",␣
↪bool_vectors_udf(col("document")))

    def minhashing(self, bool_vectors):
        shingles = self.shingles
        num_shingles = len(shingles)
        num_hashes = self.num_hashes
        def random_permutation():
            return random.sample(range(1, num_shingles+1), num_shingles)
        permutations = [random_permutation() for _ in range(num_hashes)]

        def minhash_udf(bool_vector, permutations):
            min_hashes = []
            for permutation in permutations:
                hashed_vector = [permutation[idx] for idx, present in␣
↪enumerate(bool_vector) if present]
                if hashed_vector:
                    min_hash = min(hashed_vector)
                else:
                    min_hash = float('inf')
                min_hashes.append(min_hash)
            return min_hashes

        udf_minhash = udf(lambda bool_vector: minhash_udf(bool_vector,␣
↪permutations) , ArrayType(IntegerType()))
        signature_matrix = bool_vectors.withColumn("signature",␣
↪udf_minhash(col("bool_vector")))
        return signature_matrix

    def locality_sensity_hashing(self, signatures):
        num_bands = self.num_bands
        num_buckets = self.num_buckets

        def hash_band(band):
```

```
            band_hash = hash(tuple(band)) % num_buckets
            return f"{band}-{band_hash}"

        def split_vector(signature):
            sub_vecs = []
            r = int(len(signature) / num_bands)
            for i in range(0, len(signature), r):
                sub_vecs.append(signature[i: i+r])
            return sub_vecs

        def hashing_signature(signature):
            hashes_value = []
            sub_vecs = split_vector(signature)
            for sub in sub_vecs:
                hash_value = hash_band(sub)
                hashes_value.append(hash_value)
            return hashes_value


        hashing_signature_udf  = udf(lambda signature:␣
↪hashing_signature(signature), ArrayType(StringType()))
        hashed_value = signatures.withColumn("hashed_value",␣
↪hashing_signature_udf(col("signature")))

        hashed_value = hashed_value.withColumn("temp_order", lit(1))
        indexed_df = hashed_value.withColumn("document_index", row_number().
↪over(Window.orderBy("temp_order"))).drop("temp_order")

        results = indexed_df.select(explode(col("hashed_value")).
↪alias("hashed_value"), ("document_index"))

        return results, indexed_df

    def run(self):
        self.shingles = self.shingling(self.documents)
        self.bool_vectors = self.bool_vector(self.documents, self.shingles)
        self.signature_matrix = self.minhashing(self.bool_vectors)
        self.lsh, self.indexed_df = self.locality_sensity_hashing(self.
↪signature_matrix)

    def approxNearestNeighbors(self, key, n):
        query_document = spark.createDataFrame([(key,)], ['document'])
        docs = self.documents.union(query_document)
        shingles = self.shingling(docs)
        bool_vector = self.bool_vector(docs, shingles)
        signature_matrix = self.minhashing(bool_vector)
        lsh, indexed_df = self.locality_sensity_hashing(signature_matrix)
```

```
        max_document_index = indexed_df.selectExpr("max(document_index)").
↪first()[0]

        lsh_query = lsh.filter(col("document_index") == max_document_index)
        indexed_df_query = indexed_df.filter(col("document_index") ==␣
↪max_document_index)

        lsh = lsh.filter(col("document_index") != max_document_index)
        indexed_df = indexed_df.filter(col("document_index") !=␣
↪max_document_index)

        similar_doc = lsh.alias("lsh").join(
                    lsh_query.alias("lsh_query"),
                    col("lsh.hashed_value") == col("lsh_query.hashed_value"),
                    "inner"
                ).select(col("lsh.document_index")).distinct()

        def jaccard_similarity(sequence1, sequence2):
            intersection_count = 0
            union_count = len(sequence1)

            for elem1, elem2 in zip(sequence1, sequence2):
                if elem1 == elem2:
                    intersection_count += 1

            similarity = intersection_count / union_count
            return similarity

        jaccard_udf = udf(jaccard_similarity, FloatType())

        similarity_threshold = self.similarity_threshold

        indexed_df = indexed_df.alias("indexed_df").join(
                similar_doc.alias("similar_doc"),
                col("indexed_df.document_index") == col("similar_doc.
↪document_index"),
                "left"
            ).filter(col("similar_doc.document_index").isNotNull()).
↪drop("similar_doc")

        df_with_similarity = indexed_df.crossJoin(indexed_df_query.
↪select(col("signature").alias("signature_query"))) \
                                .withColumn("jaccard_similarity",
                                            jaccard_udf(col("signature"),
                                            col("signature_query"))) \
```

```
                             .select("document", "jaccard_similarity") \
                             .filter(col("jaccard_similarity") >=␣
↪similarity_threshold) \
                             .orderBy(desc("jaccard_similarity")) \
                             .limit(n)
        df_with_similarity.show()
        return df_with_similarity
```

2.1 Test

[ ]:
```
documents = spark.read.text(dataset).withColumnRenamed("value", "document")
```

[ ]:
```
documents.show(5, truncate = False)
```

```
+-------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------+
|document
|
+-------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```


----------------------+
|Phytoplasmas are insect-vectored bacteria that cause disease in a wide range of plant species. The increasing availability of molecular DNA analyses, expertise and additional methods in recent years has led to a proliferation of discoveries of phytoplasma-plant host associations and in the numbers of taxonomic groupings for phytoplasmas. The widespread use of common names based on the diseases with which they are associated, as well as separate phenetic and taxonomic systems for classifying phytoplasmas based on variation at the 16S rRNA-encoding gene, complicates interpretation of the literature. We explore this issue and related trends through a focus on Australian pathosystems, providing the first comprehensive compilation of information for this continent, covering the phytoplasmas, host plants, vectors and diseases. Of the 33 16Sr groups reported internationally, only groups I, II, III, X, XI and XII have been recorded in Australia and this highlights the need for ongoing biosecurity measures to prevent the introduction of additional pathogen groups. Many of the phytoplasmas reported in Australia have not been sufficiently well studied to assign them to 16Sr groups so it is likely that unrecognized groups and sub-groups are present. Wide host plant ranges are apparent among well studied phytoplasmas, with multiple crop and non-crop species infected by some. Disease management is further complicated by the fact that putative vectors have been identified for few phytoplasmas, especially in Australia. Despite rapid progress in recent years using molecular approaches, phytoplasmas remain the least well studied group of plant pathogens, making them a "crouching tiger" disease threat.
|
|Background: (-)-alpha-Bisabolol, also known as levomenol, is an unsaturated sesquiterpene alcohol that has mainly been used in pharmaceutical and cosmetic products due to its anti-inflammatory and skin-soothing properties. (-)-alpha-Bisabolol is currently manufactured mainly by steam-distillation of the

essential oils extracted from the Brazilian candeia tree that is under threat because its natural habitat is constantly shrinking. Therefore, microbial production of (-)-alpha-bisabolol plays a key role in the development of its sustainable production from renewable feedstock. Results: Here, we created an Escherichia coli strain producing (-)-alpha-bisabolol at high titer and developed an in situ extraction method of (-)-alpha-bisabolol, using natural vegetable oils. We expressed a recently identified (-)-alpha-bisabolol synthase isolated from German chamomile (Matricaria recutita) (titer: 3 mg/L), converted the acetyl-CoA to mevalonate, using the biosynthetic mevalonate pathway (12.8 mg/L), and overexpressed farnesyl diphosphate synthase to efficiently supply the (-)-alpha-bisabolol precursor farnesyl diphosphate. Combinatorial expression of the exogenous mevalonate pathway and farnesyl diphosphate synthase enabled a dramatic increase in (-)-alpha-bisabolol production in the shake flask culture (80 mg/L) and 5 L bioreactor culture (342 mg/L) of engineered E. coli harboring (-)-alpha-bisabolol synthase. Fed-batch fermentation using a 50 L fermenter was conducted after optimizing culture conditions, resulting in efficient (-)-alpha-bisabolol production with a titer of 9.1 g/L. Moreover, a green, downstream extraction process using vegetable oils was developed for in situ extraction of (-)-alpha-bisabolol during fermentation and showed high yield recovery (>98%). Conclusions: The engineered E. coli strains and economically viable extraction process developed in this study will serve as promising platforms for further development of microbial production of (-)-alpha-bisabolol at large scale.|
|A universal feature of the replication of positive-strand RNA viruses is the association with intracellular membranes. Carnation Italian ringspot virus (CIRV) replication in plants occurs in vesicles derived from the mitochondrial outer membrane. The product encoded by CIRV ORF1, p36, is required for targeting the virus replication complex to the outer mitochondrial membrane both in plant and yeast cells. Here the yeast Saccharomyces cerevisiae was used as a model host to study the effect of CIRV p36 on cell survival and death. It was shown that p36 does not promote cell death, but decreases cell growth rate. In addition, p36 changed the nature of acetic acid-induced cell death in yeast by increasing the number of cells dying by necrosis with concomitant decrease of the number of cells dying by programmed cell death, as judged by measurements of phosphatidylserine externalization. The tight association of p36 to membranes was not affected by acetic acid treatment, thus confirming the peculiar and independent interaction of CIRV p36 with mitochondria in yeast. This work proved yeast as an invaluable model organism to study both the mitochondrial determinants of the type of cell death in response to stress and the molecular pathogenesis of (+)RNA viruses.
|
|1,2-Dichloropropane (1,2-DCP) and dichloromethane (DCM) are possible causative agents associated with the development of cholangiocarcinoma in employees working in printing plant in Osaka, Japan. However, few reports have demonstrated an association between these agents and cholangiocarcinoma in rodent carcinogenicity studies. Moreover, the combined effects of these compounds have not been fully elucidated. In the present study, we evaluated the in vivo mutagenicity of 1,2-DCP and DCM, alone or combined, in the livers of gpt

delta rats. Six-week-old male F344 gpt delta rats were treated with 1,2-DCP, DCM or 1,2-DCP+DCM by oral administration for 4weeks at the dose (200mgkg(-1) body weight 1,2-DCP and 500mgkg(-1) body weight DCM) used in the carcinogenesis study performed by the National Toxicology Program. In vivo mutagenicity was analyzed by gpt mutation/Spi(-) assays in the livers of rats. In addition, gene and protein expression of CYP2E1 and GSTT1, the major enzymes responsible for the genotoxic effects of 1,2-DCP and DCM, were analyzed by quantitative polymerase chain reaction and western blotting. Gpt and Spi(-) mutation frequencies were not increased by 1,2-DCP and/or DCM in any group. Additionally, there were no significant changes in the gene and protein expression of CYP2E1 and GSTT1 in any group. These results indicated that 1,2-DCP, DCM and 1,2-DCP+DCM had no significant impact on mutagenicity in the livers of gpt delta rats under our experimental conditions. Copyright (c) 2016 John Wiley & Sons, Ltd.
|
|This paper presents the simulation results of a linear, fully integrated, two-stage digitally programmable 130 nm CMOS power amplifier (PA) operating at 2.4 GHz. Its power stage is composed of a set of amplifying cells which can be enabled or disabled independently by a digital control circuit. All seven operational modes are univocal in terms of 1 dB output compression point (OCP1dB), saturated output power (P-SAT) and power gain at 2.4 GHz. The lowest power mode achieves an 8.1 dBm P-SAT, a 13.5 dB power gain and consumes 171 mW DC power (P-DC) at an OCP1dB of 6 dBm, whereas the highest power mode reaches an 18.9 dBm P-SAT and a 21.1 dB power gain and consumes 415 mW P-DC at an OCP1dB of 18.2 dBm.
|
+----------


```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------+
only showing top 5 rows
```

[ ]:
```
large_data_minhash_LSH = LargDataMinHashLSH(documents)
large_data_minhash_LSH.run()
```

[ ]:
```
query_document = "Many recent publications highlight the large role of the␣
 ↪pivotal eukaryotic nuclear export protein exportin-1 (XPO1) in the␣
 ↪oncogenesis of several malignancies, and there is emerging evidence that␣
 ↪XPO1 inhibition is a key target against cancer. The clinical validation of␣
 ↪the pharmacological inhibition of XPO1 was recently achieved with the␣
 ↪development of the selective inhibitor of nuclear export compounds,␣
 ↪displaying an interesting anti-tumor activity in patients with massive␣
 ↪pre-treated hematological malignancies. Recent reports have shown molecular␣
 ↪alterations in the gene encoding XPO1 and showed a mutation hotspot (E571K)␣
 ↪in the following two hematological malignancies with similar phenotypes and␣
 ↪natural histories: primary mediastinal diffuse large B cell lymphoma and␣
 ↪classical Hodgkin's lymphoma. Emerging evidence suggests that the mutant␣
 ↪XPO1 E571K plays a role in carcinogenesis, and this variant is quantifiable␣
 ↪in tumor and plasma cell-free DNA of patients using highly sensitive␣
 ↪molecular biology techniques, such as digital PCR and next-generation␣
 ↪sequencing. Therefore, it was proposed that the XPO1 E571K variant may serve␣
 ↪as a minimal residual disease tool in this setting. To clarify and summarize␣
 ↪the recent findings on the role of XPO1 in B cell hematological␣
 ↪malignancies, we conducted a literature search to present the major␣
 ↪publications establishing the landscape of XPO1 molecular alterations, their␣
 ↪impact on the XPO1 protein, their interest as biomarkers, and investigations␣
 ↪into the development of new XPO1-targeted therapies in B cell hematological␣
 ↪malignancies."
df_with_similarity = large_data_minhash_LSH.
 ↪approxNearestNeighbors(query_document, 5)
```

```
+--------------------+------------------+
|            document|jaccard_similarity|
+--------------------+------------------+
|Many recent publi…|               1.0|
|The integration o…|              0.66|
|Six different com…|              0.65|
|Protein-protein i…|              0.64|
|Quantitative PCR(…|              0.64|
+--------------------+------------------+
```

```
[ ]: query_1 = "Background: (-)-alpha-Bisabolol, also known as levomenol, is an␣
      ↪unsaturated sesquiterpene alcohol that has mainly been used in␣
      ↪pharmaceutical and cosmetic products due to its anti-inflammatory and␣
      ↪skin-soothing properties. (-)-alpha-Bisabolol is currently manufactured␣
      ↪mainly by steam-distillation of the essential oils extracted from the␣
      ↪Brazilian candeia tree that is under threat because its natural habitat is␣
      ↪constantly shrinking. Therefore, microbial production of (-)-alpha-bisabolol␣
      ↪plays a key role in the development of its sustainable production from␣
      ↪renewable feedstock. Results: Here, we created an Escherichia coli strain␣
      ↪producing (-)-alpha-bisabolol at high titer and developed an in situ␣
      ↪extraction method of (-)-alpha-bisabolol, using natural vegetable oils. We␣
      ↪expressed a recently identified (-)-alpha-bisabolol synthase isolated from␣
      ↪German chamomile (Matricaria recutita) (titer: 3 mg/L), converted the␣
      ↪acetyl-CoA to mevalonate, using the biosynthetic mevalonate pathway (12.8 mg/
      ↪L), and overexpressed farnesyl diphosphate synthase to efficiently supply␣
      ↪the (-)-alpha-bisabolol precursor farnesyl diphosphate. Combinatorial␣
      ↪expression of the exogenous mevalonate pathway and farnesyl diphosphate␣
      ↪synthase enabled a dramatic increase in (-)-alpha-bisabolol production in␣
      ↪the shake flask culture (80 mg/L) and 5 L bioreactor culture (342 mg/L) of␣
      ↪engineered E. coli harboring (-)-alpha-bisabolol synthase. Fed-batch␣
      ↪fermentation using a 50 L fermenter was conducted after optimizing culture␣
      ↪conditions, resulting in efficient (-)-alpha-bisabolol production with a␣
      ↪titer of 9.1 g/L. Moreover, a green, downstream extraction process using␣
      ↪vegetable oils was developed for in situ extraction of (-)-alpha-bisabolol␣
      ↪during fermentation and showed high yield recovery (>98%). Conclusions: The␣
      ↪engineered E. coli strains and economically viable extraction process␣
      ↪developed in this study will serve as promising platforms for further␣
      ↪development of microbial production of (-)-alpha-bisabolol at large scale."
     query_20 = "3D printing has shown promise for neural regeneration by providing␣
      ↪customized nerve scaffolds to structurally support and bridge the defect gap␣
      ↪as well as deliver cells or various bioactive substances. Low-level light␣
      ↪therapy (LLLT) exhibits positive effects on rehabiliation of degenerative␣
      ↪nerves and neural disorders. With this in mind, we postulate that 3D printed␣
      ↪neural scaffold coupling with LLLT will generate a new strategy to repair␣
      ↪neural degeneration. To achieve this goal, we applied red laser light to␣
      ↪stimualte neural stem cells on 3D printed scaffolds and investigated the␣
      ↪subsequent cell response with respect to cell proliferation and␣
      ↪differentiation. Here we show that cell prolifeartion rate and intracellular␣
      ↪reactive oxgen species synthesis were significantly increased after 15 s␣
      ↪laser stimulation follwed by 1 d culture. Over culturing time of 14 d in␣
      ↪vitro, the laser stimulation promoted neuronal differentiation of neural␣
      ↪stem cells, while the glial differentiation was suppressed based on results␣
      ↪of both immunocytochemistry studies and real-time quantitative reverse␣
      ↪transcription polymerase chain reaction testing. These findings suggest that␣
      ↪integration of 3D printing and LLLT might provide a powerful methodology for␣
      ↪neural tissue engineering."
     key = query_1+query_20
```

```
df_with_similarity = large_data_minhash_LSH.approxNearestNeighbors(key, 3)
```

```
+--------------------+------------------+
|            document|jaccard_similarity|
+--------------------+------------------+
|Background: (-)-a…|              0.84|
|3D printing has s…|               0.6|
+--------------------+------------------+
```