Compsci 361 Assignment 2

Name: Joanne Chen UPI: jehc820

Part 1: Report

1. Chosen representation and data preprocessing

To begin with, Pandas DataFrame is used for inputting and preprocessing the trg csv document. Motivation: it is convenient for visualising and manipulating the tabular data structure, and it integrates well with other Python libraries. Representation: we choose occurrence frequency for the representation of the text in X instead of 0-1 attributes. Motivation: the occurrence frequency of each word is needed in the model training stage. It is written as a function, so it can be easily reused for handling other datasets in the modelling steps. Also, we split the document into X (abstract part) and y (class column) for further processing. Pre-processing techniques including text preprocessing and data splitting are performed. Text preprocessing: We identify all occurring different words. We count the total amount of them which is called |Vocabulary| in Mitchell, T. M. (1997). Data splitting: we use sklearn library to do single 80/20 training/validation split because there are 4000 abstracts in the original data set, which is large enough. More data for training would generate a better classifier, so I decided to use 80% of the raw data for the model training data.

2. The idea behind the model implementation and their improvements

In the second part in the code, we train a Standard Naive Bayse (SNB) classifier using X_train and y_train. The idea of this model is the algorithm provided in Mitchell, T. M. (1997). Machine learning. Firstly, in the training function, we use a formula to calculate the conditional probability of each word in the text: **P(wk|vj) = (nk + 1) / (n + |Vocabulary|)**, where wk is a specific word in the text, nk refers to the amount of times wk appears in the text, and n is total amount of different word with different target class. This formula uses Laplace Smoothing to avoid the very tiny probability and to prevent the situation where an unseen word has a probability of 0, resulting in the entire posterior probability becoming 0. Secondly, we use **P(vj) = |docsj| / |Examples|** to calculate the Prior Probability for each class, where Examples refer to the text with their target class, and docsj refers to the subset from Examples with their class vj. Finally, in the prediction stage, we multiply prior probability by the conditional probability of all words provided by the classifier. The class with the highest posterior probability will be the predicted result. The procedure will be discussed in the following part.

To improve my model, I concatenate some relevant words into one word. Also, I set the occurrence frequency of some most frequent words to 0, to make these words look as same as unseen or removed word. Motivation is from Mitchell, T. M. (1997). Machine learning, and Rennie at al. (2003) "Tackling the Poor Assumptions of Naive

Bayes Text Classifiers", the most frequent and common words are less likely to be relevant to the actual class of text.

3. Explanation of the evaluation procedure

20% of the raw data has been randomly selected to be the validation data, named X_valid and y_valid which is utilised to evaluate the SNB classifier. The dataset has been split into two subsets: a training set and a validation set in the data preprocessing part. The model has been trained on the training set to get the SNB classifer and will be evaluated on the validation set. This method provides an independent dataset for evaluating performance of the model and helps prevent overfitting.

To begin with, we use the SNB classifier to get the conditional probabilities of all word, and multiply the prior probability by the conditional probabilities of all word, using **VNB =argmax P(vj)πP(ai|vj)**. The result is posterior probability given each of the four classes for text or abstract in validation or test data. Then the predict function compares four posterior probabilities and selects the class with the highest posterior probability as the predicted result for the specific text. Afterward, we calculate accuracy. Accuracy is the probability of correct predictions given the total size of samples, used to measure how well the SNB classifier perform. The prediction is compared with y_valid which is the actual class of the abstract. Finally we get 94.375% accuracy for the initial SNB model.

4. Training/validation results for the SNB and improved model

The initial model is applied to the unseen Kaggle test data and luckily we get the accuracy or the public score of 0.953, a bit lower than the threshold 0.96. Therefore, we have to improve our initial model to increase the accuracy.

To improve the model, I concatenated some words including "homo sapiens" by adding "-" between these words because they are not independent as the assumption of Naive Bayse. Following that, I selected some common words to remove like "an", "a" and "the"and removed them from the classifier by setting their occurrence times to 0. As a result, improvement did work on validation data: the accuracy increased to 95.5%. Thus, I believe that this improvement is a good idea. The accuracy of the prediction on the Kaggel test data increases to 0.98 which was much higher than expected. Furthermore, I tried to only remove the most frequent word "the". However, the result on Kaggle was 0.973 which was slightly worse than removing three common words. Therefore, finally I decide to keep the model improved by removing "an", "a" and "the".

SNB Classifier	Validation data	Kaggle test data
Standard	94.375%	95.3%
Improved v1	95.5%	98%
Improved v2	95.5%	97.3%

Part 2: Code and implementation

Task 1: Data pre-process and representations

Pre-processing data part

1. We use Pandas to input the data and represent the data in Pandas data frame.
2. We use occurence frequency to represent the data. We count and save the times of occuring of each distinct word.
3. We collect all unique word from the document and calculate the total amount of occurrence of the distinct word.

In [318…

```
import pandas as pd

#Data pre-process
# Read the data file
df = pd.read_csv('trg.csv')

# Because there is 'class' as a word in the abstracts, we rename the colu
df = df.rename(columns={'class': 'target_feature'})

# Check whether data is loaded in the right way or not
#df.head()

y = df['target_feature']
X = df['abstract']

#split X into 4 subset along with their target values A,B,E,V
def split_X(X,target_class,y):
    indices = []
    for i, label in enumerate(y):
        if label == target_class:
            indices.append(i)
    return X.iloc[indices]

#test
X_class_A = split_X(X,"A",y)
X_class_V = split_X(X,"V",y)
print(X_class_A.index)
X_class_V.head()
```

```
Index([   1,   18,   39,   46,   99,  104,  110,  269,  283,  309,
       ...
       3689, 3694, 3767, 3772, 3821, 3896, 3937, 3945, 3977, 3985],
      dtype='int64', length=128)
```

Out[318…

```
8     the complete dna sequence of the a2 strain of ...
34    sequencing of the ecori n' fragment of african...
63    we have sequenced a dna fragment containing th...
70    the complete nucleotide sequence of two human ...
87    hardy-zuckerman 2 feline sarcoma virus hz2-fes...
Name: abstract, dtype: object
```

Text preprocessing: split text into distinct words and count all the occurrence frequency of a word in the abstract in the following part:

In [319…
```
#|vocabulary| or |X|: set of all distinct words in the abstract part in X
unique_word = []
for abstract in X_train:
    for w in abstract.split():
        if w not in unique_word:
            unique_word.append(w)
size_vocab = len(unique_word)
print(size_vocab)
```

```
28232
```

Using occurrence frequency of each distinct word as representation for the text in the abstract:

In [320…
```
#This function is used to create a dict to record the occurrence frequenc
def occur_freq_count(X):
    words_freq = {}
    #loop through every row in the file
    for row in X:
        #loop through every word in each row, and the delimiter is space
        for w in row.split():
            #if the word not in the dictionary, add to the dict
            if w not in words_freq:
                words_freq[w] = 1
            #if the word in the dict, add one to the freqency
            else:
                words_freq[w] += 1
    return words_freq

#test
print(occur_freq_count(X_class_A))
```

{'the': 1783, 'complete': 93, '1751377-bp': 17, 'sequence': 209, 'of': 128
8, 'genome': 283, 'thermophilic': 23, 'archaeon': 64, 'methanobacterium':
22, 'thermoautotrophicum': 74, 'deltah': 20, 'has': 98, 'been': 122, 'dete
rmined': 65, 'by': 172, 'a': 500, 'whole-genome': 30, 'shotgun': 26, 'sequ
encing': 43, 'approach': 23, 'total': 60, '1855': 17, 'open': 72, 'readin
g': 73, 'frames': 65, 'orfs': 198, 'have': 176, 'identified': 100, 'that':
251, 'appear': 23, 'to': 524, 'encode': 60, 'polypeptides': 103, '844': 1
7, '46': 28, 'which': 128, 'assigned': 89, 'putative': 69, 'functions': 7
9, 'based': 54, 'on': 107, 'their': 54, 'similarities': 25, 'database': 2
0, 'sequences': 232, 'with': 289, '514': 17, '28': 17, 'orf-encoded': 17,
'are': 444, 'related': 92, 'unknown': 35, 'and': 1018, '496': 17, '27': 1
9, 'little': 34, 'or': 41, 'no': 45, 'homology': 20, 'in': 604, 'public':
33, 'databases': 63, 'comparisons': 45, 'eucarya-': 17, 'bacteria-': 17,
'archaea-specific': 17, 'reveal': 17, '1013': 17, 'gene': 234, 'products':
58, '54': 17, 'most': 92, 'similar': 105, 'polypeptide': 19, 'described':
19, 'previously': 28, 'for': 198, 'other': 62, 'organisms': 55, 'domain':
37, 'archaea': 81, 'methanococcus': 41, 'jannaschii': 125, 'data': 63, 'un
derline': 17, 'extensive': 26, 'divergence': 17, 'occurred': 19, 'betwee
n': 33, 'these': 129, 'two': 159, 'methanogens': 36, 'only': 75, '352': 1
7, '19': 19, 'm': 202, '50': 19, 'identical': 23, 'there': 71, 'is': 187,
'conservation': 18, 'relative': 19, 'locations': 17, 'orthologous': 17, 'g
enes': 266, 'when': 19, 'compared': 19, 'from': 109, 'eucaryal': 88, 'bact
erial': 121, 'domains': 17, '786': 17, '42': 19, 'more': 88, '241': 17, '1
3': 18, 'domain-like': 17, 'include': 45, 'majority': 47, 'those': 52, 'pr
edicted': 76, 'be': 74, 'involved': 69, 'cofactor': 19, 'small': 17, 'mole
cule': 21, 'biosyntheses': 17, 'intermediary': 17, 'metabolism': 51, 'tran
sport': 34, 'nitrogen': 17, 'fixation': 17, 'regulatory': 23, 'interaction
s': 17, 'environment': 23, 'proteins': 139, 'dna': 135, 'transcription': 4
1, 'translation': 43, 'structure': 35, 'organization': 28, 'features': 50,
'typical': 40, 'bacteria': 48, 'including': 66, 'closely': 21, '24': 21,
'could': 33, 'form': 22, 'two-component': 18, 'sensor': 19, 'kinase-respon
se': 17, 'regulator': 17, 'systems': 43, 'homologs': 44, 'hsp70-response':
17, 'dnak': 17, 'dnaj': 17, 'notably': 17, 'absent': 17, 'replication': 4
8, 'initiation': 40, 'chromosome': 36, 'packaging': 17, 'presence': 80, 'c
dc6': 17, 'three': 41, 'histones': 17, 'however': 44, 'an': 181, 'ftsz': 1
7, 'indicates': 30, 'type': 53, 'cell': 36, 'division': 30, 'polymerases':
21, 'x-family': 17, 'repair': 24, 'unusual': 18, 'archaeal': 101, 'b': 35,
'formed': 19, 'separate': 23, 'dna-dependent': 17, 'rna': 48, 'polymeras
e': 45, 'rnap': 34, 'subunits': 33, "a'": 34, "b'": 17, 'h': 21, 'encode
d': 26, 'operon': 39, 'although': 62, 'second': 39, 'subunit-encoding': 1
7, 'present': 26, 'at': 112, 'remote': 17, 'location': 34, 'rrna': 56, 'op
erons': 18, '39': 21, 'trna': 65, 'dispersed': 17, 'around': 19, 'occur':
17, 'clusters': 41, 'introns': 26, 'trnapro': 17, 'ggg': 17, 'contains': 4
6, 'intron': 35, 'unprecedented': 17, 'selenocysteinyl-trna': 17, 'nor': 1
7, 'evidence': 32, 'classically': 17, 'organized': 23, 'elements': 46, 'pr
ophages': 17, 'plasmids': 19, 'one': 37, 'intein': 24, 'extended': 17, 're
peats': 20, '36': 23, '86': 17, 'kb': 21, 'members': 17, 'family': 23, '1
8': 17, 'representatives': 17, 'date': 3, 'eight': 2, 'escherichia': 9, 'c
oli': 15, 'umuc': 2, 'protein': 32, 'all': 33, 'play': 1, 'critical': 1,
'roles': 1, 'damage-inducible': 1, 'mutagenesis': 3, 'enterobacteriaceae':
1, 'recently': 1, 'distantly': 1, 'umuc-homolog': 1, 'dinb': 3, 'also': 2
2, 'e': 6, 'using': 17, 'chain': 13, 'reaction': 10, 'together': 6, 'degen
erate': 1, 'primers': 1, 'designed': 1, 'against': 17, 'conserved': 15, 'r
egions': 41, 'found': 45, 'umuc-like': 2, 'we': 45, 'new': 19, 'member':
6, 'umuc-superfamily': 1, 'archeon': 2, 'sulfolobus': 13, 'solfataricus':
8, 'this': 106, 'homolog': 3, 'shows': 15, 'high': 30, 'similarity': 58,
'lower': 2, 'level': 11, 'as': 95, 'consequence': 1, 'called': 2, 'dbh':
4, 'analysis': 34, 'approximately': 5, 'encompassing': 1, 'region': 6, 're
vealed': 12, 'several': 7, 'encoding': 9, 'ribokinase': 1, 'was': 82, 'loc
ated': 6, 'immediately': 2, 'upstream': 4, 'orf': 2, 'overlaps': 1, '4':

2, 'bp': 24, 'suggesting': 11, 'both': 17, 'might': 23, 'coordinately': 1, 'expressed': 5, 'further': 5, 'ribokinase-dbh': 1, 'locus': 6, 'another': 9, 'potential': 25, 'atpase': 2, 'homologous': 5, 'uncharacterized': 7, 's': 8, 'cerevisiae': 3, 'yd934602c': 1, 'sc38kcxvi20': 1, 'ruvb': 1, 'while': 4, 'first': 11, 'report': 19, 'detected': 3, 'additional': 8, 'gram-positive': 7, 'cyanobacteria': 1, 'among': 39, 'human': 5, 'est': 1, 'indicating': 10, 'umuc-related': 1, 'comprise': 3, 'ubiquitous': 1, 'superfamily': 1, 'probably': 9, 'used': 16, 'n-terminal': 6, 'amino': 45, 'acid': 25, 'dihydrolipoamide': 4, 'dehydrogenase': 21, 'haloferax': 1, 'volcanii': 5, 'design': 6, 'synthesize': 1, 'oligonucleotide': 1, 'probes': 3, 'were': 187, 'identify': 2, 'clone': 3, '43': 1, 'kilobase': 1, 'pair': 28, 'kbp': 1, 'fragment': 4, 'mboi': 1, 'restriction': 14, 'endonuclease': 4, 'digestion': 1, 'hf': 1, 'genomic': 67, 'nucleotide': 5, '15-kbp': 1, 'frame': 8, 'translated': 1, 'into': 25, 'good': 1, 'sources': 16, '48': 1, 'acids': 23, 'obtained': 2, 'purified': 13, 'primary': 4, 'halophilic': 8, 'analyzed': 3, 'terms': 1, 'its': 68, 'homologies': 1, 'dehydrogenases': 13, 'molecular': 13, 'adaptations': 1, 'intracellular': 6, 'ionic': 1, 'strength': 1, 'hyper-thermophilic': 12, 'archaebacterium': 11, 'pyrococcus': 29, 'horikoshii': 11, 'ot3': 7, 'assembling': 7, 'physical': 8, 'map-based': 7, 'contigs': 7, 'fosmid': 7, 'clones': 7, 'long': 30, 'pcr': 20, 'gap-filling': 7, 'entire': 24, 'length': 13, '1738505': 7, 'authenticity': 12, 'supported': 12, 'directly': 14, 'amplified': 13, 'protein-coding': 32, '2061': 7, 'search': 23, '406': 7, '197': 7, 'function': 33, '453': 7, '220': 7, 'registered': 16, 'but': 26, 'remaining': 16, '1202': 7, '583': 7, 'did': 19, 'not': 50, 'show': 30, 'any': 38, 'significant': 31, 'comparison': 21, 'provided': 11, 'considerable': 12, 'number': 12, 'generated': 12, 'duplication': 21, '11': 7, 'assumed': 7, 'contain': 18, 'single': 22, '16s-23s': 16, '5s': 16, 'coding': 28, 'occupied': 12, '9125': 7, 'whole': 33, 'presented': 17, 'paper': 17, 'available': 39, 'internet': 16, 'httpwwwnitegojp': 7, 'hsp70dnak': 3, 'moderate': 2, 'methanosarcina': 31, 'thermophila': 5, 'tm-1': 4, 'cloned': 13, 'sequenced': 32, 'tested': 1, 'vitro': 1, 'measure': 1, 'induction': 1, 'heat': 1, 'ammonia': 12, 'ie': 6, 'stressors': 2, 'pertinent': 1, 'biotechnological': 1, 'ecosystem': 1, 'methanogen': 26, 'plays': 13, 'key': 5, 'role': 39, 'anaerobic': 2, 'bioconversions': 1, "locus'": 1, "5'-grpe-hsp70dnak-hsp40": 1, "dnaj-trka-3'": 1, 'same': 3, 'mesophile': 1, 'mazei': 21, 's-6': 3, 'different': 21, 'comparable': 1, 'exist': 1, 'thermophile': 2, 'genus': 2, 'trka': 3, 'part': 5, 'very': 5, 'considerably': 2, 'less': 4, '23-amino': 1, 'deletion--by': 1, 'gram-negative': 1, 'later': 1, 'gram': 1, 'positives': 1, 'responded': 1, 'temperature': 7, 'elevation': 1, 'manner': 1, 'demonstrated': 3, 'they': 24, 'heat-shock': 4, 'functionally': 6, 'active': 10, 'vivo': 3, 'induced': 1, 'response': 13, 'suggest': 5, 'hsp70dnak-chaperone': 1, 'machine': 1, 'contrast': 8, 'hyperthermophilic': 10, 'stress': 11, 'inasmuch': 1, 'it': 29, 'responds': 1, 'like': 2, 'classic': 1, 'induce': 1, 'methanogenesis': 28, 'biological': 12, 'production': 37, 'methane': 45, 'pivotal': 13, 'global': 25, 'carbon': 36, 'cycle': 28, 'contributes': 12, 'significantly': 22, 'warming': 12, 'nature': 19, 'derived': 12, 'acetate': 34, 'here': 13, 'acetate-utilizing': 12, 'acetivorans': 36, 'c2a': 12, 'methanosarcineae': 24, 'metabolically': 22, 'diverse': 14, 'thrive': 12, 'broad': 12, 'range': 12, 'environments': 18, 'unique': 17, 'forming': 19, 'complex': 25, 'multicellular': 12, 'structures': 20, 'diversity': 29, 'reflected': 13, '5751492': 12, 'base': 28, 'pairs': 27, 'far': 19, 'largest': 12, 'known': 24, '4524': 12, 'code': 12, 'strikingly': 12, 'wide': 12, 'unanticipated': 12, 'variety': 13, 'metabolic': 37, 'cellular': 25, 'capabilities': 12, 'novel': 16, 'methyltransferases': 12, 'likelihood': 13, 'undiscovered': 12, 'natural': 12, 'energy': 30, 'whereas': 19, 'single-subunit': 12, 'monoxide': 14, 'raises': 12, 'possibility': 13, 'nonmethanogenic': 12, 'growth': 26, 'motility': 14, 'observed': 14, 'flagellin': 18, 'cluster': 20, 'chemotaxis': 12, 'availability': 18, 'genetic': 29, 'methods': 22, 'coupled': 12, 'physiological': 12, 'makes': 12, 'powerful': 12, 'model': 22, 'organis

m': 27, 'study': 20, 'biology': 16, 'annotations': 12, 'analyses': 22, 'ht
tpwww-genomewimitedu': 12, 'acetyl-coa': 3, 'decarbonylasesynthase': 1, 'a
cds': 3, 'catalyzes': 1, 'central': 3, 'acetyl': 3, 'c-c': 4, 'bond': 4,
'cleavage': 2, 'growing': 5, 'responsible': 6, 'synthesis': 4, 'units': 1,
'during': 4, 'c-1': 1, 'substrates': 2, 'beta': 11, 'subunit': 10, 'nicke
l': 11, 'fes': 7, 'center': 3, 'reacts': 1, 'acetyl-enzyme': 2, 'intermedi
ate': 1, 'presumably': 1, 'activation': 5, 'investigate': 1, 'process': 3,
'forms': 2, 'overexpressed': 1, 'anaerobically': 3, 'grown': 2, 'containe
d': 17, 'lacked': 6, 'inactive': 1, 'formation': 3, 'redox-dependent': 1,
'acetyltransferase': 1, 'assays': 2, 'activity': 9, 'developed': 1, 'incub
ation': 1, 'nicl2': 1, 'native': 3, 'nickel-reconstituted': 1, 'iron': 2,
'21': 2, 'ratio': 1, 'insignificant': 1, 'levels': 3, 'metals': 1, 'coppe
r': 1, 'binding': 2, 'elicited': 1, 'marked': 1, 'changes': 8, 'uv-visibl
e': 1, 'spectrum': 2, 'intense': 1, 'charge': 1, 'transfer': 17, 'bands':
2, 'multiple': 2, 'thiolate': 1, 'ligation': 2, 'kinetics': 1, 'incorporat
ion': 1, 'matched': 1, 'time': 3, 'course': 1, 'enzyme': 26, 'divalent':
1, 'metal': 2, 'ions': 1, 'substitute': 1, 'yielding': 2, 'catalytic': 3,
'reactions': 4, 'coa': 1, 'co': 8, 'methylcobalamin': 1, 'demonstrating':
1, 'absence': 2, 'indispensable': 1, 'too': 1, 'needed': 1, 'characteristi
c': 5, 'epr-detectable': 1, 'enzyme-carbonyl': 1, 'adduct': 1, 'epr': 5,
'signal': 9, 'require': 1, 'addition': 1, 'reducing': 3, 'agent': 1, 'indi
rect': 1, 'involvement': 1, 'paramagnetic': 1, 'species': 18, 'site-direct
ed': 1, 'indicated': 2, 'cys-278': 1, 'cys-280': 1, 'coordinate': 1, 'cys-
189': 1, 'essential': 9, 'results': 5, 'consistent': 4, 'ni2fe4s4': 1, 'ar
rangement': 4, 'site': 4, 'mechanism': 6, 'proposed': 3, 'includes': 1, 's
pecific': 2, 'fe4s4': 3, 'accounts': 1, 'absolute': 1, 'requirement': 1,
'extreme': 12, 'halophile': 18, 'halobacterium': 21, 'sp': 8, 'nrc-1': 13,
'harboring': 6, 'dynamic': 7, '2571010-bp': 6, 'containing': 9, '91': 6,
'insertion': 14, 'representing': 7, '12': 9, 'families': 7, 'large': 20,
'2': 15, 'minichromosomes': 6, 'codes': 8, '2630': 6, 'unrelated': 6, 'rep
orted': 12, 'pathways': 29, 'uptake': 14, 'utilization': 6, 'sodium-proto
n': 6, 'antiporter': 6, 'potassium': 6, 'sophisticated': 6, 'photosensor
y': 6, 'transduction': 6, 'resembling': 8, 'eukaryotic': 10, 'proteome':
6, 'definite': 6, 'bacillus': 7, 'subtilis': 7, 'ease': 6, 'culturing': 6,
'manipulation': 6, 'laboratory': 6, 'construction': 6, 'knockouts': 6, 're
placements': 6, 'indicate': 27, 'can': 7, 'serve': 6, 'excellent': 6, 'sys
tem': 18, 'aerobic': 16, 'crenarchaeon': 15, 'aeropyrum': 6, 'pernix': 6,
'k1': 6, 'optimally': 9, 'grows': 9, '95': 5, 'degrees': 23, 'c': 26, 'met
hod': 10, 'some': 25, 'modifications': 9, '1669695': 5, '2694': 5, '633':
5, '235': 5, '523': 5, '194': 5, 'tca': 9, 'except': 7, 'alpha-ketoglutara
te': 10, 'included': 9, 'instead': 5, '2-oxoacidferredoxin': 5, 'oxidoredu
ctase': 5, '1538': 5, '571': 5, 'suggested': 10, '47': 5, '14': 5, '8912':
5, 'homepage': 9, 'httpwwwmildnitegojp': 5, 'great': 14, 'ecological': 10,
'importance': 10, 'fermenting': 10, 'methylamines': 20, 'methanol': 10, 'd
ioxide': 10, 'case': 11, 'since': 11, 'precursor': 11, '60': 15, 'produce
d': 11, 'earth': 10, 'contribute': 10, 'greenhouse': 10, 'gas': 10, 'eg':
15, 'rice': 10, 'paddies': 10, '4096345': 10, 'circular': 10, 'than': 24,
'twice': 10, 'genomes': 21, 'methanogenic': 11, 'currently': 20, 'complete
ly': 14, 'bult': 10, 'et': 20, 'al': 20, '1996': 10, 'smith': 10, '1997':
10, '3371': 10, '376': 10, 'methanosarcina-specific': 10, '1043': 10, 'fin
d': 10, 'closest': 10, 'homologue': 10, '544': 10, 'reach': 10, 'values':
14, '56': 10, '102': 10, 'transposases': 10, 'gluconeogenesis': 10, 'proli
ne': 10, 'biosynthesis': 14, 'processes': 12, 'dna-repair': 10, 'environme
ntal': 10, 'sensing': 10, 'regulation': 13, 'striking': 10, 'examples': 1
0, 'occurrence': 11, 'groelgroes': 10, 'chaperone': 10, 'tetrahydrofolate-
dependent': 10, 'enzymes': 25, 'findings': 18, 'lateral': 14, 'played': 1
0, 'important': 12, 'evolutionary': 14, 'forging': 10, 'physiology': 14,
'versatile': 10, 'thermoplasma': 12, 'acidophilum': 3, 'thermoacidophili
c': 7, 'thrives': 1, '59': 1, 'ph': 5, 'isolated': 5, 'self-heating': 1,
'coal': 1, 'refuse': 1, 'piles': 1, 'solfatara': 1, 'fields': 1, 'do': 5,

'possess': 1, 'rigid': 1, 'wall': 1, 'delimited': 1, 'plasma': 2, 'membran
e': 3, 'many': 16, 'macromolecular': 1, 'assemblies': 1, 'primarily': 1,
'proteases': 3, 'chaperones': 1, 'elucidating': 1, 'homologues': 1, 'our':
3, 'interest': 1, 'folding': 3, 'degradation': 3, 'led': 1, 'us': 1, 'see
k': 1, 'representation': 1, 'determining': 1, '1564905-base-pair': 1, 'jus
t': 1, '7855': 1, 'strategy': 1, '1509': 1, 'euryarchaeon': 5, 'substantia
l': 6, 'complement': 1, 'bacteria-related': 1, 'much': 2, 'phylogeneticall
y': 1, 'distant': 2, 'inhabiting': 1, 'least': 2, '252': 1, 'pathway': 3,
'various': 1, 'resemble': 1, '1694969-nt': 3, 'gc-rich': 3, 'methanopyru
s': 3, 'kandleri': 24, 'direct': 3, 'unlinking': 3, 'thermofidelase': 3,
'version': 3, 'topoisomerase': 3, 'v': 3, 'directed': 3, "2'-modified": 3,
'oligonucleotides': 3, 'fimers': 3, 'redundancy': 3, '33x': 3, 'sufficien
t': 3, 'assemble': 3, 'error': 3, 'per': 5, '40': 6, 'combination': 3, 'se
arches': 3, 'prediction': 3, '1692': 3, 'structural': 5, 'rnas': 6, 'unusu
ally': 4, 'content': 13, 'negatively': 3, 'charged': 3, 'adaptation': 10,
'salinity': 3, 'previous': 3, 'phylogenetic': 10, '16s': 5, 'belonged': 3,
'deep': 3, 'branch': 5, 'close': 3, 'root': 7, 'tree': 9, 'trees': 6, 'con
structed': 4, 'concatenated': 3, 'alignments': 3, 'ribosomal': 4, 'consist
ently': 4, 'groups': 4, 'shares': 4, 'set': 3, 'implicated': 4, 'methanoth
ermobacter': 3, 'monophyletic': 3, 'distinctive': 3, 'feature': 4, 'paucit
y': 3, 'signaling': 3, 'expression': 8, 'appears': 10, 'fewer': 8, 'acquir
ed': 3, 'via': 4, 'reflect': 3, 'habitat': 3, 'archaeoglobus': 6, 'fulgidu
s': 16, 'sulphur-metabolizing': 5, '2178400': 5, '2436': 5, 'information':
5, 'processing': 9, 'biosynthetic': 6, 'components': 13, 'nucleotides': 5,
'cofactors': 5, 'correlation': 5, 'counterparts': 5, 'dramatic': 5, 'diffe
rences': 8, 'way': 5, 'sense': 5, 'perform': 6, 'gain': 5, 'restriction-mo
dification': 6, 'none': 5, 'inteins': 5, 'quarter': 10, '651': 5, 'encode
s': 19, 'yet': 5, 'two-thirds': 5, 'shared': 7, '428': 5, 'annotated': 3,
'22-megabase': 3, 'pyrobaculum': 3, 'aerophilum': 6, 'facultatively': 4,
'nitrate-reducing': 3, 'topt': 3, '100': 5, 'clues': 3, 'explanations': 3,
"organism's": 3, 'surprising': 3, 'intolerance': 3, 'sulfur': 5, 'may': 1
2, 'aid': 3, 'development': 3, 'studies': 6, 'interesting': 7, 'worthy':
3, 'computational': 3, 'confirmed': 5, 'experiments': 3, 'showing': 3,
'p': 13, 'perhaps': 3, 'crenarchaea': 5, 'lack': 3, "5'": 7, 'untranslate
d': 3, 'mrnas': 4, 'thus': 4, 'use': 3, 'ribosome-binding': 4, 'shine-dalg
arno-based': 3, 'end': 4, 'transcripts': 9, 'inspection': 3, 'lengths': 5,
'distribution': 3, 'mononucleotide': 6, 'repeat-tracts': 6, 'instance': 3,
'seen': 3, 'gs': 3, 'cs': 3, 'highly': 6, 'unstable': 4, 'pattern': 7, 'ex
pected': 5, 'deficient': 3, 'mismatch': 3, 'result': 8, 'independent': 3,
'mutation': 4, 'rates': 4, 'suggests': 8, 'mutator': 3, 'phenotype': 3, 'c
dhabc': 1, 'respective': 1, 'alpha': 13, 'epsilon': 3, 'five-subunit': 1,
'gamma': 1, 'delta': 1, 'dehydrogenaseacetyl-coenzyme': 1, 'synthase': 1,
'codhacs': 4, 'northern': 3, 'blot': 3, 'cdh': 4, 'five': 2, 'orf1': 2, 'c
otranscribed': 2, 'primer': 1, 'extension': 2, 'mrna': 2, 'promoters': 3,
'transcript': 5, 'cdha': 2, 'cdhb': 2, 'each': 6, 'identity': 6, 'methanos
aeta': 1, 'soehngenii': 1, 'cdhc': 4, '397': 1, 'c-terminal': 1, 'half':
1, 'acetogenic': 1, 'anaerobe': 1, 'clostridium': 1, 'thermoaceticum': 1,
'deduced': 6, 'cdhd': 1, '29': 2, 'nifh2': 1, 'ivanovii': 1, 'abstract':
1, 'cyp119': 1, 'cytochrome': 11, 'p450': 2, 'acidothermophilic': 1, 'pred
icts': 1, '368': 1, 'consensus': 2, 'heme-binding': 1, 'phe-gly-xaa-gly-xa
a-his-xaa-cys-xaa-gly-': 1, 'xaa3-ala-arg-xaa-glu': 1, 'resembles': 1, 'p4
50s': 1, 'bacterium': 1, '129': 1, 'residues': 7, '35': 2, 'represents':
1, 'step': 1, 'tracing': 1, 'history': 1, 'biologically': 1, '166-megabas
e': 13, 'autotrophic': 13, '58-': 13, '16-kilobase': 13, 'extrachromosoma
l': 14, 'random': 13, '1738': 13, 'minority': 13, '38': 14, 'percent': 13,
'confidence': 15, 'eukaryotes': 19, 'mutants': 1, 'resistant': 1, 'dihydro
folate': 3, 'reductase': 5, 'inhibitor': 1, 'trimethoprim': 1, 'amplificat
ions': 1, 'describes': 1, 'cloning': 1, 'nucleic': 1, 'mutant': 2, 'wr21
5': 1, 'reductases': 4, 'amplification': 1, 'trimethoprim-resistant': 1,
'overproduces': 1, 'homogeneity': 1, 'ammonium': 1, 'sulfate-mediated': 1,

'chromatographies': 1, 'shown': 3, 'comprises': 1, '5': 3, '15': 3, 'fit
s': 1, 'preliminary': 1, 'biochemical': 5, 'characterization': 2, 'salt':
1, 'concentrations': 2, 'increases': 1, 'increase': 2, 'kcl': 1, 'nacl':
1, 'abyssi': 12, 'furiosus': 8, 'whose': 5, 'presently': 4, 'laboratorie
s': 4, 'develop': 4, 'tools': 4, 'hyperthermophiles': 4, 'performed': 4,
're-annotation': 4, 'obtain': 4, 'integrated': 4, 'view': 5, 'phylogeny':
4, 'informational': 4, 'operational': 4, 'moreover': 6, 'candidate': 8, 'm
issing': 9, 'links': 4, 'agrees': 4, 'position': 5, 'near': 4, 'origin':
6, 'mesophilic': 4, 'catalase-peroxidase': 2, 'haloarcula': 1, 'marismortu
i': 1, 'edman': 1, 'mass': 7, 'spectrometry': 2, 'proteolytic': 1, 'fragme
nts': 1, 'corresponds': 1, '731': 1, 'calculated': 3, 'mature': 5, 'delete
d': 2, 'methionine': 1, '8125365': 1, 'da': 5, 'reasonable': 1, 'agreemen
t': 1, 'value': 1, '81292': 1, '-': 1, '9': 1, 'measured': 2, 'southern':
1, 'showed': 7, 'monocistronic': 1, 'catalase-peroxidases': 1, 'strongly':
1, 'heme': 7, 'histidines': 1, 'similarly': 1, 'soluble': 2, 'excess': 1,
'acidic': 6, 'associated': 2, 'solvation': 1, 'volcanium': 5, 'possessin
g': 15, 'optimum': 6, 'ogt': 25, 'systematically': 5, 'comparing': 6, 'hig
her': 7, 'strong': 8, 'correlations': 5, 'characteristics': 6, 'increasin
g': 10, 'frequency': 6, 'clustering': 5, 'purines': 5, 'pyrimidines': 5,
'dinucleotides': 5, 'rises': 5, 'often': 5, 'aa': 5, 'tt': 5, 'avoiding':
5, 'ta': 5, 'coded': 5, 'divided': 6, 'distinct': 7, 'subpopulations': 5,
'isoelectric': 5, 'points': 6, 'ranges': 5, 'basic': 12, 'size': 14, 'subp
opulation': 5, 'becomes': 5, 'larger': 9, 'mediating': 5, 'synthesizing':
5, 'coenzymes': 5, 'such': 7, 'start': 5, 'provide': 6, 'insights': 5, 'in
dividual': 5, 'well': 9, 'principles': 5, 'coordinating': 5, 'designs': 5,
'tokodaii': 8, 'strain7': 8, '80': 4, 'low': 7, 'under': 5, 'conditions':
5, 'slight': 4, '2694756': 4, 'g': 6, '328': 4, 'following': 4, 'rna-codin
g': 4, 'intron-containing': 4, 'repetitive': 16, 'sr-type': 4, 'dispersed-
type': 4, 'tn-like': 4, '2826': 4, '911': 4, '322': 4, 'functional': 11,
'921': 4, '326': 4, '145': 4, '51': 4, 'motifs': 4, '849': 4, '300': 4, 'a
ssignments': 4, 'sulfide': 4, 'respiratory': 4, 'integration': 4, 'plasmi
d': 5, 'rearrangement': 4, 'eukaryote-type': 4, 'cca': 4, 'strain': 6, 'cl
oser': 4, 'strains': 6, 'so': 6, 'httpwwwbionitegojpe-homegenomelist-ehtm
l': 4, 'formate': 8, 'dehydrogenase-encoding': 1, 'fdhcab': 4, 'flanking':
3, 'thermoformicicum': 3, 'z-245': 2, 'fdh': 4, 'initiated': 1, 'fdhc': 3,
'terminated': 1, 'processed': 1, 'fdha': 1, 'resulting': 1, 'fdhab': 1, 's
tages': 1, 'cells': 5, 'barely': 1, 'detectable': 5, 'early': 1, 'exponent
ial': 1, 'h2': 6, 'plus': 3, 'co2': 4, 'dramatically': 1, 'coincident': 1,
'decrease': 1, 'rate': 6, 'onset': 1, 'constant': 1, 'culture': 1, 'densit
ies': 1, 'reached': 1, 'optical': 1, 'density': 1, '600': 2, 'nm': 4, '0
5': 1, 'mth': 3, 'h2-dependent': 1, 'methenyl-h4': 1, 'mpt': 1, 'frh': 4,
'mvh': 3, 'coenzyme': 1, 'f420-reducing': 1, 'nonreducing': 1, 'hydrogenas
es': 2, 'respectively': 15, 'exogenously': 1, 'supplied': 1, 'supply': 2,
'reduced': 3, 'ch4': 2, 'increased': 4, 'mtd': 1, 'mer': 1, 'mcr': 2, 'cat
alyze': 2, 'steps': 1, '7': 1, 'reduction': 1, 'insufficient': 1, 'resulte
d': 1, 'disappearance': 1, 'incapable': 1, 're-examined': 1, 'inducible':
1, 'b558566': 3, 'acidocaldarius': 3, 'dsm': 1, '639': 1, 'formerly': 1,
'thought': 2, 'component': 1, 'terminal': 1, 'oxidase': 2, 'becker': 1, 's
chfer': 1, '1991': 1, 'febs': 1, 'lett': 1, '291': 1, '331-335': 1, 'impro
ved': 1, 'purification': 1, 'yield': 2, 'allowed': 1, 'detailed': 3, 'inve
stigations': 1, '64210': 1, '1': 3, 'mol': 3, 'hememol': 1, 'oxygen': 1,
'tensions': 1, 'composition': 1, 'medium': 1, 'exerts': 1, 'influence': 1,
'membranes': 1, 'exhibits': 2, 'extremely': 5, 'redox': 1, '400': 1, 'mv':
1, 'reactivity': 1, 'hishis-coordination': 1, 'axial': 1, 'ligands': 1, 'l
ikely': 3, 'turned': 1, 'out': 1, 'glycosylated': 1, '20': 1, 'sugar': 4,
'exposed': 1, 'outer': 1, 'surface': 1, 'moiety': 1, 'consists': 1, 'o-gly
cosidically': 1, 'linked': 3, 'mannoses': 2, 'n-glycosidically': 2, 'hexas
accharide': 1, 'comprising': 1, 'glucoses': 1, 'n-acetyl-glucosamines': 1,
'cbsa': 1, 'revealing': 1, 'secondary': 2, 'alpha-helical': 1, 'anchors':
1, 'mainly': 1, 'beta-pleated': 1, 'sheet': 1, 'amounts': 1, 'serine': 1,

'threonine': 1, 'cbsb': 1, 'latter': 4, 'displays': 1, 'hydrophobicity':
1, 'unit': 1, 'copurify': 1, 'entry': 1, 'banks': 1, 'indeed': 1, 'kind':
1, 'b-type': 1, 'hemoprotein': 1, 'analogous': 1, 'pseudoperiplasmic': 1,
'space': 1, 'discussed': 4, 'dissimilatory': 3, 'sulphite': 3, 'sulphate-r
educing': 1, 'common': 3, 'characterized': 6, '2-structure': 1, 'sirohae
m': 2, 'non-haem': 1, 'atoms': 1, 'labile': 1, 'sulphide': 1, 'oxidized':
3, 'exhibited': 3, 'absorption': 1, 'maxima': 1, '281': 1, '394': 1, '54
5': 1, '593': 1, 'weak': 1, 'band': 1, '715': 1, 'designate': 1, 'dsra':
4, 'dsrb': 5, 'contiguous': 1, 'order': 1, 'preceded': 2, 'followed': 1,
'termination': 2, 'signals': 5, 'sulphur-dependent': 1, '474': 1, 'kda': 1
1, '417': 1, 'peptides': 3, '256': 1, 'arisen': 1, 'ancestral': 2, 'peptid
e': 2, 'cysteine': 3, 'postulated': 1, 'bind': 2, 'sirohaem-fe4s4': 2, 'co
mplexes': 1, 'nitrite': 1, 'lacks': 1, 'residue': 1, 'binds': 1, 'chemica
l': 2, 'sirohaems': 1, 'ferredoxins': 1, 'six': 3, 'would': 2, 'four': 7,
'ferredoxin-like': 1, 'sites': 4, 'flourish': 1, 'hypersaline': 1, 'replic
ons': 1, 'replicon': 2, 'pnrc100': 5, 'undergoes': 1, 'high-frequency': 1,
'element-mediated': 3, 'insertions': 1, 'deletions': 1, 'inversions': 1,
'recombination': 3, '39-kb-long': 1, 'inverted': 1, 'irs': 2, 'now': 1, '1
91346-bp': 1, 'circle': 1, '176': 1, '850-bp': 1, 'average': 1, 'repeate
d': 1, 'within': 4, 'one-half': 1, 'represent': 1, 'copies': 2, 'partition
ing': 1, 'evolved': 2, 'fusions': 1, 'smaller': 1, 'idea': 1, 'typically':
1, 'arsenic': 1, 'resistance': 2, 'buoyant': 1, 'gas-filled': 1, 'vesicle
s': 1, 'element': 1, 'electron': 3, 'd': 1, 'thioredoxin': 2, 'eukaryotic-
like': 1, 'tata-binding': 1, 'factors': 1, 'chromosomal': 3, 'initiator':
1, 'multi-step': 1, 'account': 1, 'acquisition': 1, 'finding': 1, 'propert
y': 2, 'curing': 1, 'evolving': 1, 'ef-2': 2, 'woesei': 1, 'desulfurococcu
s': 1, 'mobilis': 1, 'phylogenies': 3, 'inferred': 4, 'alternative': 1, 't
ree-making': 1, 'ef-2g': 2, 'contrasted': 1, 'shorter': 2, 'ef-1': 7, 'alp
hatu': 4, 'monophyly': 2, 'sensu': 1, 'hennig': 1, 'subdivision': 1, 'king
doms': 1, 'crenarchaeota': 1, 'euryarchaeota': 1, 'usually': 1, 'bootstra
p': 3, 'tend': 1, 'inconsistent': 1, 'distance': 2, 'maximum': 1, 'parsimo
ny': 2, 'maximum-likelihood': 1, 'topologies': 1, 'receive': 1, 'support':
1, '25': 1, 'retinal': 1, 'halophiles': 1, 'establish': 1, 'relationship':
4, 'basis': 2, 'apparently': 1, 'designated': 2, 'rhodopsin': 11, 'arf':
1, 'protein-coupled': 1, 'receptors': 1, 'pump': 2, 'bacteriorhodopsin':
3, 'cl-': 1, 'halorhodopsin': 3, 'kinds': 1, 'sensory': 4, 'phoborhodopsi
n': 4, 'seemed': 1, 'duplications': 1, 'before': 1, 'generic': 1, 'speciat
ion': 4, 'therefore': 2, 'simply': 1, 'divergent': 1, 'evolution': 5, 'aft
er': 4, 'reconstituted': 1, 'rhodopsins': 1, 'bacteriorhodopsinhalorhodops
insensory': 1, '54310': 1, 'branching': 1, 'topology': 1, 'grouped': 1, 'v
ersus': 1, 'mapping': 1, 'outgroup': 1, 'point': 3, 'rhodopsinphoborhodops
in': 1, 'calculating': 1, 'speculate': 1, 'upon': 1, 'pre-sensory': 3, 'pr
e-phoborhodopsin': 2, 'fivefold': 1, 'faster': 1, 'original': 2, 'accumula
tion': 1, 'mutations': 1, 'differentiation': 1, 'halobacterial': 2, 'stamp
s': 1, 'better': 1, 'understand': 1, 'endonucleases': 2, '37': 1, 'homodim
er': 1, 'recombinant': 2, 'cleave': 1, 'trnas': 1, 'lacking': 1, 'full':
1, 'comparative': 1, 'existed': 1, 'yeast': 1, 'elongation': 2, 'factor':
2, 'tuf-gene': 2, 'downstream': 1, "3'": 2, 'highest': 1, 't': 1, 'vanniel
ii': 2, 'arns-arnl': 1, 'disclosed': 1, 'hitherto': 1, 'undescribed': 1,
'positions': 3, '699-nucleotide': 1, 'nt': 2, 'ialpha': 2, '908': 1, 'numb
ering': 2, 'f': 1, 'noller': 1, 'annu': 1, 'rev': 1, 'biochem': 1, '53119-
162': 1, '1984': 1, '202-nt': 1, 'ibeta': 1, '575-nt': 1, 'igamma': 2, '10
85': 1, '1927': 1, '23s': 2, 'crucial': 1, 'remarkably': 1, 'rich': 1, '41
5': 1, '431': 1, 'gc': 2, 'rrnas': 2, '677': 1, '692': 1, 'obvious': 1, 't
hem': 1, 'splicing': 3, 'quantity': 1, 'intronic': 1, 'stably': 1, 'retain
ed': 1, 'linear': 1, 'monomeric': 1, 'trace': 1, 'topoisomeric': 1, 'molec
ules': 1, 'appeared': 1, 'behavior': 1, 'two-dimensional': 1, 'gel': 5, 'e
lectrophoresis': 3, 'models': 1, 'ialpha-': 1, 'ibeta-': 1, 'igamma-contai
ning': 1, 'precursors': 1, 'agree': 1, 'bulge-helix-bulge': 1, 'motif': 2,
'overall': 2, 'share': 1, 'lagli-dadg': 1, 'homing': 1, '281-kbp': 1, 'con

tig': 2, 'crenarchaeote': 1, 'p2': 3, 'analysed': 1, 'notable': 1, 'archae
al-type': 1, 'histidine': 1, 'pyrimidine': 1, 'arginine': 1, 'numerous':
1, 'lipopolysaccharide': 1, 'crenarchaeotes': 1, 'euryarchaeotes': 1, 'for
micicum': 1, 'had': 1, 'appropriate': 1, 'mobility': 1, 'sds': 1, 'cross-r
eacted': 1, 'antibodies': 1, 'raised': 1, 'overlap': 1, 'eubacterial': 2,
'archaebacterial': 3, 'iron-sulfur': 1, 'centers': 1, 'background': 1, 're
spects': 1, 'prokaryotes': 1, 'respect': 1, 'few': 1, 'identification': 1,
'add': 1, 'knowledge': 2, 'mechanisms': 1, 'composed': 2, 'dp1': 1, 'dp2':
1, 'weights': 1, '69294': 1, '143161': 1, 'corresponding': 2, 'tandemly':
2, 'arranged': 2, 'pol': 3, 'transcribed': 1, 'additionally': 1, 'cdc18cdc
6': 1, 'dmc1rad51': 1, 'expressing': 1, 'possesses': 1, '--': 1, 'exonucle
olytic': 1, 'template-primer': 1, 'preference': 1, 'replicative': 2, 'conc
lusion': 1, 'neither': 2, 'display': 1, 'enzymatic': 1, 'properties': 1,
'raise': 1, '2992245': 2, '2977': 2, 'one-third': 2, 'archaeal-specific':
4, '23': 2, 'exclusively': 2, 'eukarya': 4, 'plasticity': 2, '200': 2, 'no
nautonomous': 2, 'mobile': 3, 'integrase-mediated': 2, 'events': 2, 'regul
arly': 2, 'spaced': 2, 'tandem': 3, 'inorganic': 2, 'organic': 2, 'solute
s': 2, 'wealth': 2, 'extracellular': 2, 'metabolizing': 2, 'major': 4, 'ca
rrier': 2, 'nadh': 2, 'ferredoxin': 2, 'required': 2, 'transcriptional':
2, 'eukaryal': 2, 'character': 3, 'illustrate': 2, 'euryarchaea': 2, 'espe
cially': 2, 'translational': 2, 'apparatus': 2, 'kod1': 1, 'kod': 6, '501
3': 1, 'bases': 1, '1671': 1, 'genbank': 1, 'accession': 1, 'd29671': 1,
"3'-5'": 2, 'exonuclease': 2, 'in-frame': 1, 'intervening': 2, '1080': 1,
'360': 1, 'intein-1': 1, '1611': 1, '537': 1, 'intein-2': 1, 'middle': 1,
'alpha-like': 1, "polymerase's": 1, '75': 1, 'x': 1, '10-3': 1, 'pfu': 2,
'130': 1, 'nucleotidess': 1, 'times': 2, 'processivity': 1, 'persistence':
1, 'sequential': 1, 'polymerization': 1, '10': 1, 'enabled': 1, 'accurat
e': 1, 'thermococcus': 1, 'litoralis': 1, 'split': 1, 'ivss': 2, 'continuo
us': 1, 'exons': 2, 'ivs': 2, 'group': 1, 'i': 5, 'ivs2': 4, 'i-tli': 1,
'cleaves': 1, 'exon': 2, '2-exon': 1, '3': 1, 'junction': 2, 'self-splice
s': 1, 'abolished': 1, 'if': 2, 'disrupted': 1, 'silent': 1, '2-ivs2': 1,
'maintain': 1, 'inhibit': 1, 'rather': 1, 'frisia': 2, 'g1': 2, 'ni2-': 2,
'fe2-': 2, 's2--containing': 2, 'alpha2beta2': 2, 'heterotetramer': 2, '21
4': 2, 'pi': 3, '52': 2, '94': 2, 'vmax': 2, '03': 2, 'mmol': 2, 'min-1':
2, 'mg-1': 2, 'km': 2, 'methyl': 2, 'viologen': 2, '09': 2, 'mm': 4, '01
2': 2, 'spectroscopy': 2, 'overlapping': 2, 'indicative': 2, '4fe-4s': 2,
'atypical': 2, 'standard': 2, 'high-spin': 2, 'tentatively': 2, 'nuclearit
y': 2, 'flagellar': 1, 'filaments': 1, 'halobium': 1, 'sodium': 2, 'dodecy
l': 2, 'sulfate-polyacrylamide': 2, 'flagellins': 1, 'fla': 5, 'ii': 2, 'i
ii': 1, 'sulfated': 1, 'glycoproteins': 1, 'oligosaccharides': 1, 'glca-1-
---4-glca-1----4-glca-1----4-glc': 1, 'immunologically': 1, 'cross-reactiv
e': 1, 'vector': 1, 'antibody': 1, 'probe': 1, 'subcloned': 1, 'flg': 5,
'a1': 2, '-2': 2, 'b1': 1, '-3': 1, 'immunological': 1, 'a2': 1, 'glutamat
e': 2, 'reconstructed': 1, 'automated': 1, 'trypsin': 1, 'cyanogen': 1, 'b
romide': 1, 'staphylococcus': 1, 'aureus': 1, 'v8': 1, 'protease': 1, 'pep
sin': 1, '421': 1, '46078': 1, 'n-epsilon-methyllysine': 1, 'gives': 1, '9
2': 1, 'symmetrical': 1, 'vertebrate': 1, 'side': 2, 'eucaryote': 1, 'subs
titutions': 1, 'possible': 2, 'n-epsilon-methylation': 1, 'lysine': 1, 'cu
rrent': 1, 'hypotheses': 1, 'thermal': 1, 'haloalkaliphilic': 1, 'natrialb
a': 1, 'magadii': 1, 'flagella': 1, 'cotranscription': 1, 'histories': 1,
'exchange': 1, 'argg': 2, 'barkeri': 3, 'ms': 2, 'argininosuccinate': 1,
'synthetase': 3, '45': 1, '31': 1, 'carbamyl': 2, 'phosphate': 2, 'sacchar
omyces': 2, 'example': 1, 'linkage': 1, 'blue': 2, 'named': 1, 'ambineel
a': 2, 'extract': 1, 'acidianus': 1, 'ambivalens': 2, 'solution': 1, 'mono
mer': 1, '87': 1, 'electronic': 1, 'centred': 1, '395': 1, '625': 1, 'a625
a395': 1, '07': 1, 'does': 1, 'transition': 1, 'colour': 1, 'due': 1, 'uni
dentified': 1, 'non-fluorescent': 1, 'covalently': 1, 'bound': 1, 'adp-bin
ding': 1, 'analogue': 1, '3382': 1, 'ligase': 3, 'acidophilic': 1, 'desulf
urolobus': 1, '67619': 1, 'weight': 1, '30-34': 1, 'atp-dependent': 3, 'li
gases': 5, 'schizosaccharomyces': 1, 'pombe': 1, 'vaccinia': 1, 'bacteriop

```
hages': 1, 't3': 1, 't4': 2, 't6': 1, 't7': 1, 'african': 1, 'swine': 1,
'fever': 1, 'virus': 1, 'nad-dependent': 1, 'eubacteria': 1, 'thermus': 1,
'thermophilus': 1, 'rna-ligase': 1, 'bacteriophage': 1, 'trna-ligase': 1,
'scerevisiae': 1, 'alignment': 1, 'vivo-transcription': 1, 'dambivalens':
1, 'mapped': 1, '1904-1911': 1, 'peptidyl': 2, 'prolyl': 1, 'cis-trans':
1, 'isomerase': 1, 'ppiase': 2, 'thermolithotrophicus': 3, 'inhibited': 1,
'fk506': 2, 'cyclosporine': 1, 'estimated': 1, '16': 1, 'filtration': 1,
'thermostable': 2, 'half-lives': 1, '90': 2, 'being': 1, '30': 1, 'min':
1, 'efficiencies': 1, 'kcatkm': 1, 'n-succinyl-ala-leu-pro-phe-p-nitroanil
ide': 1, 'n-succinyl-ala-ala-pro-phe-p-nitroanilide': 1, '035': 1, '020':
1, 'microm-1': 1, 's-1': 1, 'chymotrypsin-coupled': 1, 'sensitive': 1, 'mt
fk': 5, 'fk506-binding': 2, '462': 1, 'library': 1, 'ppiases': 1, '13-amin
o-acid': 1}
```

We split the data into training and validation set. As we have seperate test data, we only need to split data into training and validation data. We use 80%/20% train/validation single split because the data size is large enought with 4000 rows.

In [312…

```
from sklearn.model_selection import train_test_split

#in case this function will be used to split other data in the programme,
def split_data(df):
    # Split data into features (X) and target variable (y)
    y = df['target_feature']
    X = df['abstract']

    # training / validation split
    X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size

    return X_train, X_valid, y_train, y_valid
```

In [321…

```
# training / validation split
X_train, X_valid, y_train, y_valid = split_data(df)

print(y_valid)
```

```
1655    B
1866    E
2526    B
2650    E
318     E
       ..
2435    E
874     E
476     B
2533    B
28      E
Name: target_feature, Length: 800, dtype: object
```

Task 2: Implement the standard Naive Bayes algorithm

Training part

This function is used to train a Standard Naive Bayse model and save the conditional probabilities in a dictionary. We will use laplace smoothing to calculate P(w|c) =

(count(w,c)+1) / (sum(count(x,c)) + |X|) according to the pesudo code provided in Mitchell, T. M. (1997). Machine learning Page 183:

In [314…
```
def train_naive_bayse_model (X_train,y_train,size_vocab):

    #calculate and record count(w,c) in the dict
    #initial a dict with four classes as key, and value would be the amou
    count_w_c_dict = {"A":{},"B":{},"E":{},"V":{}}

    #loop through four keys (four classes)
    for c, count in count_w_c_dict.items():
        X_C = split_X(X_train,c,y_train)
        #record nk (symbol in the text book) or count(w,c) (symbol in the
        count_w_c_dict[c] = occur_freq_count(X_C)

    #calculate and record P(w|c) or P(Wk|Vj) = (count(w,v)+1) / (sum(coun
    result = {"A":{},"B":{},"E":{},"V":{}}
    #key is each distinct word, and value is P(w|c)
    for c, dict in count_w_c_dict.items():
        for word, freq in count_w_c_dict[c].items():
            #using Laplace Smoothing to calculate the conditional probabi
            result[c][word] = (count_w_c_dict[c][word] + 1) / (sum(count_

    return result
```

This function will be used to calculate priors probability by using P(class) = count(class) / count(samples):

In [322…
```
def cal_priors(y):
    #total amount of training samples (the amount of rows in the training
    total = len(y)
    #use dictionary to record the amount of each class in the training da
    cls_counts = {}
    #loop through all class labels and count unique class
    for c in y:
        #if the class not in the dictionary then add a new class as a new
        if c not in cls_counts:
            cls_counts[c] = 1
        else:
            cls_counts[c] += 1
    #calculate priors for all unique class, and record in a result dict
    priors = {}
    for cls_label, count in cls_counts.items():
        #Prior function: P(class) = count(class) / count(samples)
        priors[cls_label] = count / total
    return priors

#test
df = pd.DataFrame({"class": ["A", "B", "E", "E", "E", "V"]})
print(cal_priors(df['class']))
```

```
{'A': 0.16666666666666666, 'B': 0.16666666666666666, 'E': 0.5, 'V': 0.1666
6666666666666}
```

Prediction part

The following function is used to predict class for each row of the validation or unseen test dataset, by using Standard Naive Bayse (SNB) model trained and

calculating the Posterior probabilities. Then we will choose the highest Posterior probabilities and the corresponding class will be the final predicted result for the specific text in the document.

In [311…
```
import math

def naive_bayse_predict(X_test, y_train, word_cond_prob, size_vocabulary)

    #prediction result will be saved in a list
    prediction_list = []

    #calculate priors for all unique class: P(Vj) = |docsj| / |Examples|
    priors_dict = cal_priors(y_train)

    #count freqency for each word in each class
    count_w_c_dict = {"A":{},"B":{},"E":{},"V":{}}
    for c, count in count_w_c_dict.items():
        X_C = split_X(X_train,c,y_train)
        count_w_c_dict[c] = occur_freq_count(X_C)

    #calculate posterior probablities to the predict class: P(Class|test_
    #loop through each row (text) in the document
    for row in X_test:
        #use .split() and use space as dilimiter to split a text into a l
        word_list = row.split()
        predicted_class = None
        #initialize the max posterior probablity is negative infinity
        #so that any posterior probability is larger then update the prob
        max_posterior_prob = float('-inf')
        for cls, word_prob in word_cond_prob.items():
            #to avoid computational issues of multiplying tiny numbers, w
            posterior_prob = math.log(priors_dict[cls])
            for w in word_list:
                if w not in word_prob:
                    posterior_prob += math.log(1 / (sum(count_w_c_dict[cl
                else:
                    posterior_prob += math.log(word_prob[w])
            #we compare and choose the largest posterior probability, and
            if posterior_prob > max_posterior_prob:
                max_posterior_prob = posterior_prob
                #finally the corresponding class as the predicted result
                predicted_class = cls
        #update the predicted result into the result list
        prediction_list.append(predicted_class)

    return prediction_list
```

Evaluating part

The following function is used to evaluate the accuracy of the predicted result of validation data, compared with the actual classes of validation data. The function of accuracy is Accuracy = True Prediction / Count(samples):

In [315…
```
#calculate accurary of the predicted result and the true result
def cal_accuracy(y_true, y_pred):
    #y_true is the  true class of validation data
    #in case y_true is not a python list, we convert y_true into a list,
```

```
    y_true = y_true.tolist()
    correct_counts = 0

    #loop through both true class and predicted class of the validation d
    for y_t,y_p in zip(y_true,y_pred):
        #if the
        if y_t == y_p:
            correct_counts += 1
    accuracy = correct_counts / len(y_true)
    return accuracy
```

1. Create a SNB classifier using the training data and the function defined above
2. Predict the result for validation data set using the SNB classifier trained in the first step
3. Calculate accuracy of prediction of validation data set, use the accuracy to evaluate our SNB model and see if the accuracy is high enough to predict the final test dataset. If the accuracy is lower than 90%, that means our SNB model is not good enough. We will have to refine the SNB model or rewrite the train SNB function.

In [219…

```
#train a Naive Bayse classifier
word_cond_prob = train_naive_bayse_model(X_train,y_train,size_vocab)

#predict validation data set
y_pred = naive_bayse_predict(X_valid, y_train, word_cond_prob, size_vocab
#print(y_pred)

#calculate accuracy of prediction of validation data set
acc = cal_accuracy(y_valid, y_pred)
print("Accuracy: " + str(acc*100) +"%")
```

```
Accuracy: 94.375%
```

The accuracy is higher than 90%. Therefore, we are confident to use this SNB model to predict our test dataset. The final stage is to predict the unseen test dataset and save into a csv file. The true class of the test dataset is unknown. The result csv file will contain two columns: id and class. This file be uploaded to Kaggle and the accuracy of the prediction will be calculated by Kaggle.

In [313…

```
#predict test data set

#read test file
df_test = pd.read_csv('tst.csv')

#generate Pandas data frame for easier handling
X_test = df_test['abstract']

#use SNB classifier to predict result
test_pred = naive_bayse_predict(X_test, y_train, word_cond_prob, size_voc

#test
#the number of line should be 1000
print(len(test_pred))

#write the predicted result for test data to a csv file, used to upload t
```

```
import csv
#a list of ids from 1 to the end of test_pred
ids = list(range(1, len(test_pred) + 1))
#create a file with a file name
csv_file = 'test_prediction_jehc820.csv'
with open(csv_file, 'w', newline='') as file:
    w = csv.writer(file)
    w.writerow(['id', 'class'])  # Write header
    for i, p in zip(ids, test_pred):
        w.writerow([i, p])
```

```
1000
```

Task 3: Improve the model obtained using standard Naive Bayes algorithm.

1. We handle some relevant words, like homo sapiens, escherichia coli, human immunodeficiency virus. We concatenate these sepearate but dependent words into one word, to avoid wrong classification
2. We remove some of them (these include words such as "the" and "an") Eg. 'the' in class A in X_train appear 1446 times, which does not help with any prediction

1. Concatenate some known correlated words into one word

In [272…

```
#1. concatenate homo sapiens, escherichia coli, human immunodeficiency vi

def concat_word(X_train):
    for row in X_train:
        # concatenate "homo" and "sapiens" into "homo-sapiens"
        row = row.replace("homo sapiens", "homo-sapiens")
        row = row.replace("escherichia coli", "escherichia-coli")
        row = row.replace("human immunodeficiency virus", "human-immunode
    return X_train

X_train = concat_word(X_train)
X_valid = concat_word(X_valid)
```

2. Remove the most frequent word

In [323…

```
#2.remove the most frequent word that have no effect or even negative eff

count_w_c_dict = {"A":{},"B":{},"E":{},"V":{}} #count freqency for each w

for c, count in count_w_c_dict.items():
    X_C = split_X(X_train,c,y_train)
    count_w_c_dict[c] = occur_freq_count(X_C) #record nk in text book or

#initial a list with two very common words which are not relevant to the
words_to_be_removed = ["a","an"]

#record the most frequent word to be remove
for k,v in count_w_c_dict.items():
    #find the most frequent word
    for i in range(1):
```

```
        #sort all subset in descending order, to find the most frequent w
        sorted_subset = sorted(count_w_c_dict[k].items(), key=lambda item
        word = sorted_subset[i][0]
        if word not in words_to_be_removed:
            words_to_be_removed.append(word)
print(words_to_be_removed)

def improved_train (X_train,y_train,size_vocab):

    #calculate and record count(w,c) and sum(count(x,c)) in the dict resp
    count_w_c_dict = {"A":{},"B":{},"E":{},"V":{}} #count freqency for ea
    for c, count in count_w_c_dict.items():
        X_C = split_X(X_train,c,y_train)
        count_w_c_dict[c] = occur_freq_count(X_C) #record nk in text book

    #calculate and record P(w|c) or P(Wk|Vj) = (count(w,v)+1) / (sum(coun
    result = {"A":{},"B":{},"E":{},"V":{}} #key:value - word:P(w|c)
    for c, dict in count_w_c_dict.items():
        for word, freq in count_w_c_dict[c].items():
            #if the word is one of the most frequent words, its condition
            if word not in words_to_be_removed:
                result[c][word] = (count_w_c_dict[c][word] + 1) / (sum(co
            else:
                result[c][word] = 1 / (sum(count_w_c_dict[c].values()) +

    return result

def improved_predict(X_test, y_train, word_cond_prob, size_vocabulary):

    prediction_list = []

    #calculate priors for all unique class: P(Vj) = |docsj| / |Examples|
    priors_dict = cal_priors(y_train)

    #count freqency for each word in each class
    count_w_c_dict = {"A":{},"B":{},"E":{},"V":{}}
    for c, count in count_w_c_dict.items():
        X_C = split_X(X_train,c,y_train)
        count_w_c_dict[c] = occur_freq_count(X_C) #record nk in text book

    #calculate posterior probablities to the predict class: P(Class|test_
    for row in X_test:
        word_list = row.split()
        predicted_class = None
        #initialize the max posterior probablity is negative number
        max_posterior_prob = float('-inf')
        for cls, word_prob in word_cond_prob.items():
            #to avoid computational issues of multiplying tiny numbers, w
            posterior_prob = math.log(priors_dict[cls])
            for w in word_list:
                if w not in word_prob:
                    posterior_prob += math.log(1 / (sum(count_w_c_dict[cl
                else:
                    posterior_prob += math.log(word_prob[w])
            #we compare and choose the largest posterior probability, so
            if posterior_prob > max_posterior_prob:
                max_posterior_prob = posterior_prob
                predicted_class = cls
        prediction_list.append(predicted_class)
```

```
    return prediction_list
```

```
['a', 'an', 'the']
```

In [292…

```
#train an improved Naive Bayse classifier
improved_word_prob = improved_train(X_train,y_train,size_vocab)
#print(improved_word_prob)

#predict validation data set using improved Naive Bayse classifier
y_pred_improved = improved_predict(X_valid, y_train, improved_word_prob,
#print(y_pred)

#calculate accuracy of improved prediction of validation data set
acc_improved = cal_accuracy(y_valid, y_pred_improved)
print("Accuracy: " + str(acc_improved*100) +"%")
#It did improve!!
```

```
Accuracy: 95.5%
```

In [300…

```
test_pred_improved = improved_predict(X_test, y_train, improved_word_prob
#write the improved predicted result for test data to a csv file, used to
#a list of ids from 1 to the end of test_pred
ids = list(range(1, len(test_pred) + 1))
#create a file with a file name
csv_file = 'improved_test_prediction_jehc820.csv'
with open(csv_file, 'w', newline='') as file:
    w = csv.writer(file)
    w.writerow(['id', 'class'])  # Write header
    for i, p in zip(ids, test_pred_improved):
        w.writerow([i, p])
```