(1) a)

D1 = [woof woof meow]

D2 = [woof woof squeak]

The set of distict words, w = [woof, meow, squeak]

$n = 3$

Term frequency:

$V_1 = [2, 1, 0]$

$V_2 = [2, 0, 1]$

$\sum V_1 \times V_2 = 2 \times 2 + 1 \times 0 + 0 \times 1 = 4$

$\|V_1\| = \sqrt{2^2 + 1^2 + 0^2} = \sqrt{5}$

$\|V_2\| = \sqrt{2^2 + 0^2 + 1^2} = \sqrt{5}$

$$\text{Sim}(V_1, V_2) = \frac{\sum V_1 V_2}{\|V_1\| \times \|V_2\|} = \frac{4}{5} = 0.8$$

1(b) D_1 = [woof woof meow]

D_2 = [woof woof squeak]

The set of distinct words W = [woof, meow, squeak]

$n = 3$.

Term frequency (TF):

V_1 = [2, 1, 0] , $n = 3$

V_2 = [2, 0, 1]

Invrse Document frequency (IDF): of each words:

$$IDF(woof) = \log_e \frac{N \text{ (number of documents)}}{n_i \text{ (number of documents that contains the word woof)}}$$

$$= \log_e\left(\frac{2}{2}\right) = 0$$

$$IDF(meow) = \log_e\left(\frac{2}{1}\right) = 0.69$$

$$IDF(squeak) = \log_e\left(\frac{2}{1}\right) = 0.69$$

Weight sets: $w_i = TF_i * IDF_i$

$V_1 = [2*0, 1*0.69, 0*0.69] = [0, 0.69, 0]$

$V_2 = [2*0, 0*0.69, 1*0.69] = [0, 0, 0.69]$

Cosine similarity:

$$Sim(V_1, V_2) = \frac{\sum v_{1_i} v_{2_i}}{\|V_1\| \ \|V_2\|} = \frac{0\times0 + 0.69*0 + 0*0.69}{\sqrt{0^2+0.69^2+0^2} * \sqrt{0^2+0^2+0.69^2}}$$

$$= 0$$

1(c)

D_1 = [woof woof meow]

D_2 = [woof woof squeak]

D_3 = [meow squeak]

Distinct word set W = [woof, meow, squeak], n=3

Term Frequency:

V_1 = [2, 1, 0]

V_2 = [2, 0, 1]

Inverse Document frequency (IDF) for each words:

woof $= \log_e \frac{N}{n_i} = \log_e \frac{3}{2} = 0.405$

meow $= \log_e \frac{3}{2} = 0.405$

squeak $= \log_e \frac{3}{2} = 0.405$.

Weight set: $w_i = tf_i * idf_i$

V_1 = [2*0.405, 1*0.405, 0*0.405]

= [0.81, 0.405, 0]

V_2 = [0.81, 0, 0.405]

$$\text{Similarity} = \frac{\sum v_{1i} v_{2i}}{\|V_1\| \times \|V_2\|} = \frac{0.81*0.81 + 0 + 0}{\sqrt{(0.81)^2+(0.405)^2} * \sqrt{(0.81)^2+(0.405)^2}}$$

$$= 0.8$$

So, the cosine similarity is no longer 0 like we found in (b).

2(a) 4.1 :

$$\prod P(\text{word} \mid \text{pos}) = 0.09 * 0.07 * 0.29 * 0.04 * 0.08$$
$$= 0.0000058$$

$$P(\text{pos}) = 0.5$$

So, $P(\text{pos} \mid \text{sentence}) = \prod P(\text{word} \mid \text{pos}) \cdot P(\text{pos})$
$$= 0.0000029$$

$$\prod P(\text{word} \mid \text{neg}) = 0.16 * 0.06 * 0.06 * 0.15 * 0.11$$
$$= 0.0000095$$

$$P(\text{neg}) = 0.5$$

So, $P(\text{neg} \mid \text{sentence}) = \prod P(\text{word} \mid \text{neg}) \cdot P(\text{neg})$
$$= 0.0000048$$

Since $P(\text{pos} \mid \text{sentence}) < P(\text{neg} \mid \text{sentence})$, Naive bayes will assign "neg" label to the sentence "I always like foreign films".

2(b) 4.2:

Probability of comedy class, $P(\text{comedy}) = \frac{2}{5}$

Probability of action class, $P(\text{action}) = \frac{3}{5}$

We need to find the most likely class for the new Document $D = [\text{fast}, \text{couple}, \text{shoot}, \text{fly}]$

$$P(\text{Fast} \mid \text{comedy}) = \frac{\#\text{ of "fast" word in comedy class} + 1}{\#\text{ of word in comedy class} + \#\text{ of "fast"}}$$

$$= \frac{1+1}{9+3} = \frac{1}{6}$$

$$P(\text{couple} \mid \text{comedy}) = \frac{2+1}{9+2} = \frac{3}{11}$$

$$P(\text{shoot} \mid \text{comedy}) = \frac{0+1}{9+4} = \frac{1}{13}$$

$$P(\text{fly} \mid \text{comedy}) = \frac{1+1}{9+2} = \frac{2}{11}$$

$$P(\text{comedy} \mid D) = \pi P(\text{word} \mid \text{comedy}) \cdot P(\text{comedy})$$

$$= \frac{1}{6} * \frac{3}{11} * \frac{1}{13} * \frac{2}{11} * \frac{2}{5} = 0.00025$$

$$P(\text{Fast} \mid \text{action}) = \frac{2+1}{11+3} = \frac{3}{14}$$

$$P(\text{couple} \mid \text{action}) = \frac{0+1}{11+2} = \frac{1}{13}$$

$$P(\text{shoot} \mid \text{action}) = \frac{4+1}{11+4} = \frac{1}{3}$$

$$P(\text{fly} \mid \text{action}) = \frac{1+1}{11+2} = \frac{2}{13}$$

$$P(\text{action} \mid D) = \pi P(\text{word} \mid \text{action}) \cdot P(\text{action})$$

$$= \frac{3}{14} * \frac{1}{13} * \frac{1}{3} * \frac{2}{13} * \frac{3}{5}$$

$$= 0.00051$$

Since, $P(\text{action} \mid D) > P(\text{comedy} \mid D)$, the most likely class is "action" for D.

3c.

```
# import nltk
# nltk.download('brown')

import gensim

from nltk.corpus import brown

model = gensim.models.Word2Vec(brown.sents())
word1="rebellion"
word2="slave"

similarities1= model.most_similar(positive=[word1], topn = 10)

similarities2= model.most_similar(positive=[word2], topn = 10)

similaritieswords1=[]
for p in similarities1:
   similaritieswords1.append(p[0])

similaritieswords2=[]
for p in similarities2:
   similaritieswords2.append(p[0])

cosinesimilaritie1=""
for p in similaritieswords1:
   a=model.similarity(word1,p)
   cosinesimilaritie1=cosinesimilaritie1+"\n"+ word1 + " - "+ p+ ": "+ str(a)

cosinesimilaritie2=""
for p in similaritieswords2:
   a=model.similarity(word2,p)
   cosinesimilaritie2=cosinesimilaritie2+"\n"+ word2 + " - "+ p+ ": "+ str(a)

print("Top ten Similar words: \n")
print(word1 + " = " + str(similaritieswords1))
print(word2 + " = " + str(similaritieswords2))
print("Cosine Similarities: \n")
print(cosinesimilaritie1)
print(cosinesimilaritie2)
```

Top ten Similar words:

rebellion = ['Dominican', 'incidence', 'lacks', 'warrant', 'applying', 'regime', 'appestat', 'Handler', 'simplified', 'vigorous']

slave = ['finding', 'female', 'exciting', 'unusual', 'consciousness', 'superiority', 'substance', 'rarely', 'childhood', 'equally']

Cosine Similarities:

rebellion - Dominican: 0.9648974
rebellion - incidence: 0.9647301
rebellion - lacks: 0.96454793
rebellion - warrant: 0.963092
rebellion - applying: 0.9614085
rebellion - regime: 0.96138084
rebellion - appestat: 0.9611717
rebellion - Handler: 0.95981103
rebellion - simplified: 0.95946807
rebellion - vigorous: 0.95889

slave - finding: 0.9645964
slave - female: 0.9578907
slave - exciting: 0.9574504
slave - unusual: 0.9561291
slave - consciousness: 0.9543508
slave - superiority: 0.9528072
slave - substance: 0.95238894
slave - rarely: 0.9521802
slave - childhood: 0.95091856
slave - equally: 0.9498176

3d.

```
import gensim
from gensim.models.word2vec import Word2Vec
# Load the binary model
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz',
binary = True);


from nltk.corpus import brown

#model = gensim.models.Word2Vec(brown.sents())
word1="rebellion"
word2="slave"

similarities1= model.most_similar(positive=[word1], topn = 10)

similarities2= model.most_similar(positive=[word2], topn = 10)

similaritieswords1=[]
for p in similarities1:
   similaritieswords1.append(p[0])

similaritieswords2=[]
for p in similarities2:
   similaritieswords2.append(p[0])

cosinesimilaritie1=""
for p in similaritieswords1:
   a=model.similarity(word1,p)
   cosinesimilaritie1=cosinesimilaritie1+"\n"+ word1 + " - "+ p+ ": "+ str(a)

cosinesimilaritie2=""
for p in similaritieswords2:
   a=model.similarity(word2,p)
   cosinesimilaritie2=cosinesimilaritie2+"\n"+ word2 + " - "+ p+ ": "+ str(a)

print("Top ten Similar words: \n")
print(word1 + " = " + str(similaritieswords1))
print(word2 + " = " + str(similaritieswords2))
print("Cosine Similarities: \n")
print(cosinesimilaritie1)
print(cosinesimilaritie2)
```

Top ten Similar words:

rebellion = ['revolt', 'insurrection', 'rebellions', 'uprising', 'revolts', 'mutiny', 'uprisings', 'rebellious', 'rebelled', 'insurrections']

slave = ['slaves', 'slavery', 'enslaved', 'enslaved_Africans', 'slaver', 'slavers', 'Abraham_Lincoln_emancipation', 'slave_masters', 'abhorred_slavery', 'slaveholders']

Cosine Similarities:

rebellion - revolt: 0.8357412
rebellion - insurrection: 0.78615975
rebellion - rebellions: 0.72444963
rebellion - uprising: 0.69388425
rebellion - revolts: 0.65813243
rebellion - mutiny: 0.6445442
rebellion - uprisings: 0.6043094
rebellion - rebellious: 0.58932734
rebellion - rebelled: 0.58750707
rebellion - insurrections: 0.5765192

slave - slaves: 0.8381532
slave - slavery: 0.72603637
slave - enslaved: 0.7017009
slave - enslaved_Africans: 0.6436935
slave - slaver: 0.6238722
slave - slavers: 0.59918255
slave - Abraham_Lincoln_emancipation: 0.57785845
slave - slave_masters: 0.5764069
slave - abhorred_slavery: 0.56875
slave - slaveholders: 0.5585118

Difference between the similar words of 3c and 3d:
3d gives more relevant words while 3c gives some arbitrary words that may or may not related to the given word.

** Thanks to these two incredibly detailed Kaggle Tutorials that help any beginner to have hands-on experience on NLP:
1. How to Develop Word Embeddings in Python with Gensim
2. word2vec gensim play! look for similar words