1. **Preprocessing of Ingredient Phrases**
    a. Convert to Lowercase and convert chilli(es) to chilli

```
for i in df.index:
  df.at[i, 'Sentence'] = df.at[i, 'Sentence'].lower()
  sentence_tokens = (df.at[i, 'Sentence']).split()
  for j in range(len(sentence_tokens)):
    word=sentence_tokens[j]
    if(len(word)>4 and word[-4:]=='(es)'):
      sentence_tokens[j]=word[:-4]
  df.at[i,'Sentence']=' '.join(sentence_tokens)
```

    b. Removal of Stopwords

```
stop_words = set(stopwords.words('english'))

for i in df.index:
    sentence_tokens = word_tokenize(df.at[i, 'Sentence'])
    sentence = ""
    for token in sentence_tokens:
        if token not in stop_words:
            sentence += token + " "
    df.at[i, 'Sentence'] = sentence
```

    c. Lemmatisation

```
lemmatizer = WordNetLemmatizer()

for i in df.index:
  sentence_tokens = word_tokenize(df.at[i, 'Sentence'])
  # Lemmatize list of words and join
  lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in sentence_tokens])
  df.at[i, 'Sentence'] = lemmatized_output
```

2. **POS Tagging**
    a. We used 26 Tags (https://github.com/amishaagg/RecipeDB-2.0/blob/main/tags_meaning) to tag each word in an ingredient phrase.

```
tags=[]
for i in df.index:
  text_tok = nltk.word_tokenize(df.at[i, 'Sentence'])
  pos_tagged = nltk.pos_tag(text_tok)
  pos_tagged_sentence=""

  for word,word_class in pos_tagged:
      pos_tagged_sentence+=word + "_" + word_class+" "
  tags.append(pos_tagged_sentence)
df['POS tagging']=tags
```

Original	Sentence	POS tagging
2 cups chickpea flour seived	2 cup chickpea flour seived	2_CD cup_NN chickpea_NN flour_NN seived_VBD

b. Next we made a vector of size 26 for each ingredient phrase storing the tag frequency. For example, a phrase like '2_CD cup_NN chickpea_NN flour_NN seived_VBD' has a vector of the form [2,0,0,0,1,0,0,0,0,0,0,0,0,0..] where 2 denoted NN and 1 denoted VBD.

```
vector_dictionary = {}
vector_format = VectorFormat()
fullvector = []

#Storing the vector corresponding to each ingredient phrase
for i in df.index:
    sentenceID=df.at[i,"Unnamed: 0"]
    taggedSentence = df.at[i,"POS tagging"]
    vector = make_vector(taggedSentence,vector_format)
    vector_dictionary[sentenceID] = vector
    fullvector.append(vector)
```

3. K means Clustering.

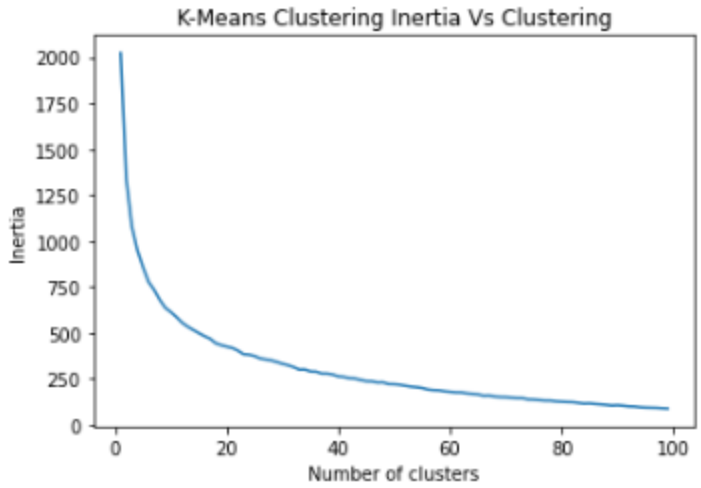
a. We use clustering to cluster the vectors of the ingredient phrases. This ensures that the ingredient phrases with similar linguistic structure are clustered together. We also plot the inertia corresponding to a fixed number of clusters.

```
inertias = []
nfclustersarr = []

for nfclusters in range(1,100):
    kmeans = KMeans(n_clusters = nfclusters, random_state = 0).fit(fullvector)
    inertia = kmeans.inertia_
    labels = kmeans.labels_
    inertias.append(inertia)
    nfclustersarr.append(nfclusters)
    print("Number of Clusters = ",nfclusters)
    print("Inertia = ",inertia)
    print("--------------------------------------------------")

plt.plot(nfclustersarr,inertias)
plt.xlabel("Number of clusters")
plt.ylabel("Inertia")
plt.title("K-Means Clustering Inertia Vs Clustering")
plt.show()
```

b. We get a plot of inertia (The sum of squared distances of samples to their closest cluster center) vs number of clusters like this



c. We find the optimum number of clusters using elbow method (https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/) and perform clustering for the optimum number of clusters.

4. **Test train split for NER tagging**
   a. Next we split each cluster into training and testing samples, and make two excel files train and test.

   b. Next we manually annotate them, to perform NER tagging, The tags we used are as follows:

Tag	Significance	Example
NAME	Name of Ingredient	salt, pepper
STATE	Processing State of Ingredient.	ground, thawed
UNIT	Measuring unit(s)	gram, cup
QUANTITY	Quantity associated with the unit(s).	1, $1\frac{1}{2}$, 2-4
SIZE	Portion sizes mentioned	small, large
TEMP	Temperature applied prior to cooking	hot, frozen
DRY/FRESH	Fresh otherwise as mentioned.	dry, fresh

2	QUANTITY
cup	UNIT
mushroom	NAME
diced	STATE

5. **Training our custom NER model**
   We trained our own NER tagging model with NLTK and Stanford NER Tagger by following this article.
   https://www.sicara.fr/blog/2018-04-25-python-train-model-ntlk-stanford-ner-tagger

   a. Create a text file named ‘test.txt’ and paste the testing data in the file.

b. We train our model using Stanford NER Tagger on the training dataset we manually annotated earlier and test it on the testing dataset (without annotations).

```
jar = './stanford-ner-tagger/stanford-ner.jar'
model = './stanford-ner-tagger/recipedb-corpus.ser.gz'

ner_tagger = StanfordNERTagger(model, jar, encoding='utf8')

words = nltk.word_tokenize(sentence)
pred = ner_tagger.tag(words)
```

c. Next, we calculated the accuracy of the testing dataset that we annotated manually.

```
Accuracy on Test: 88.10623556581986 %
```