1. Bài toán phân lớp văn bản

1.1. Tổng quan

Phân lớp văn bản được coi là quá trình phân loại một văn bản bất kì vào một hay nhiều lớp cho trước. Quá trình này gồm hai bước. Ở bước thứ nhất, một mô hình phân lớp (classfication model) được xây dựng dựa trên tri thức kinh nghiệm. Ở đây, tri thức kinh nghiệm chính là một tập dữ liệu huấn luyện (training dataset) được cung cấp bởi con người bao gồm một tập văn bản và phân lớp tương ứng của chúng. Bước này còn gọi là bước xây dựng huấn luyện (training process) hay ước lượng mô hình phân lớp. Ở bước thứ hai, mô hình phân lớp xây dựng ở bước đầu sẽ được sử dụng để phân lớp cho những văn bản (chưa được phân loại) trong tương lai. Bước đầu tiên được xem như là việc học có giám sát mà chúng ta có thể sử dụng rất nhiều các kĩ thuật học máy đã có như: Naïve Bayes, k láng giềng gần nhất (kNN), cây quyết định (Decision Tree), Mục tiêu của bài toán phân lớp là nhằm xây dựng mô hình có khả năng gán nhãn cho một văn bản bất kì với độ chính xác cao nhất có thể.

1.2. Ứng dụng

Ứng dụng lớn nhất của bài toán phân lớp văn bản là áp dụng vào bài toán phânloại hay lọc nội dung. Trong bài toán lọc nội dung: một văn bản được phân loại vào nhóm: có ích hoặc không có ích. Sau đó lấy tất cả những văn bản thuộc nhóm có ích, nhóm còn lại bị loại bỏ. Các ứng dụng cụ thể như: lọc thư rác, lọc trang web phản động, … Một ứng dụng khác của bài toán phân lớp là xây dựng bộ phân lớpsau tìm kiếm, ứng dụng này rất hữu ích vì nó định vị nội dung thông tin cần tìm kiếm nhanh và dễ dàng hơn.

Tóm lại, với tất cả ý nghĩa thực tế trên, một lần nữa có thể khẳng định rằngtrong thời đại Internet được coi là một phần không thể thiếu trong cuộc sống, phânlớp văn bản luôn là vấn đề đáng được quan tâm để có thể phát triển và xây dựngđược những công cụ ngày càng hữu dụng hơn. Dựa trên nhu cầu cấp thiết đó, chúng em chọn đề tài "Phân lớp dữ liệu bình luận sản phẩm trên website thegioididong.com" để có thể nghiên cứu và phát triển ứng dụng này.

2. Chuẩn bị dữ liệu:

Dữ liệu là yếu tố quan trọng nhất và cũng là vấn đề mà chúng ta cần quan tâm nhất. Trong quá trình xây dựng một hệ thống phân lớp văn bản, bước chuẩn bị và tiền xử lý dữ liệu quyết định tới thành bại của hệ thống hơn cả.

Với bài toán "Phân lớp dữ liệu bình luận sản phẩm trên website thegioididong.com", dữ liệu cần chuẩn bị để thực hiện bao gồm:

+ Nội dung bình luận

+ Số lượng sao tương ứng với bình luận

Việc tải dữ liệu trên được thực hiện bằng đoạn code Python như sau:

```
import requests
from bs4 import BeautifulSoup
import validators
import json

resultSet = []

def append_new_line(file_name, text_to_append):
    with open(file_name, "a+", encoding='utf-8', errors='ignore') as file_obje
        file_object.seek(0)
        data = file_object.read(100)
        if len(data) > 0:
            file_object.write("\n")
        file_object.write(text_to_append)

def getComments(url):
    valid=validators.url(url)
    if valid==True:
        response = session.get(url, params = {}, stream = False)
        soupClass = BeautifulSoup(response.text, 'html.parser')
        ratingLst = soupClass.findAll("div", {"class": "rc"})
        for item in ratingLst:
            start = len(item.find_all('i','iconcom-txtstar'))
            comment = item.find('i','').text
            if start == 0 or comment == "":
                continue
            row = {}
            row["label"] = str(start)
            row["comment"] = comment
            resultSet.append(row)

URL = "https://www.thegioididong.com/aj/CategoryV6/Product"
session = requests.Session()
cats = [42, 44, 522] #Điện thoại, Máy tính bảng, Labtop
for cat in cats:
    payload = {
        "Category": cat,
        "PageSize": 1000,
        "PageIndex" : 1
    }
    response = session.post(
        URL, data=payload, headers=dict(referer=URL))
    soup = BeautifulSoup(response.text,'html.parser')
    for a in soup.find_all('a', href=True):
        getComments("https://www.thegioididong.com"+a['href']+"/danh-gia?p=1")
        getComments("https://www.thegioididong.com"+a['href']+"/danh-gia?p=2")
        getComments("https://www.thegioididong.com"+a['href']+"/danh-gia?p=3")
        getComments("https://www.thegioididong.com"+a['href']+"/danh-gia?p=4")
        getComments("https://www.thegioididong.com"+a['href']+"/danh-gia?p=5")

append_new_line('./data/data.json', json.dumps(resultSet, ensure_ascii=False))
```

Dữ liệu tải về được lưu thành file json với 2 thuộc tính:

+ Comment: nội dung bình luận

+ Label: số lượng sao tương ứng nội dung bình luận

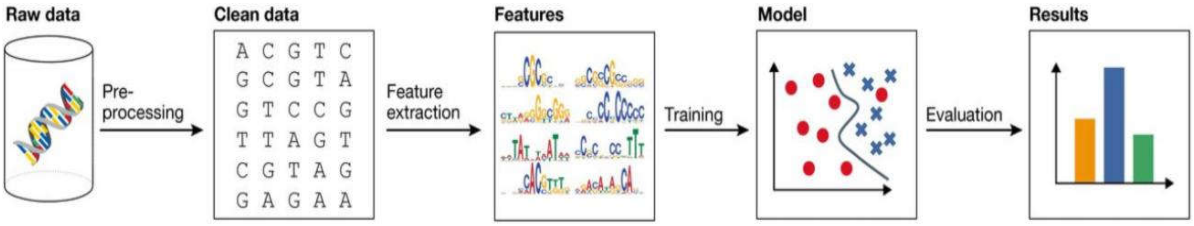
Ví dụ: nội dung file “data.json” sau khi chạy đoạn code Python trên:

[{"label": "2", "comment": "Pin tệ, tụt cực nhanh\nMình mới mua máy dc hơn 2 tháng, mà giờ thời lượng sd của pin tụt nhanh kinh khủng\nTầm hơn 2,5 tiếng.\nThegioididong kiểm tra giúp mình có phải pin có vấn đề ko?"}, {"label": "2", "comment": "Pin tệ, tụt cực nhanh\nMình mới mua máy dc hơn 2 tháng, mà giờ thời lượng sd của pin tụt nhanh kinh khủng\nTầm hơn 2,5 tiếng.\nThegioididong kiểm tra giúp mình có phải pin có vấn đề ko}]

3. Tiền xử lý dữ liệu

Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ-công cụ hoàn hảo nhất của tư duy và giao tiếp[1].

Bước đầu tiên trong xử lý ngôn ngữ tự nhiên là tiền xử lý dữ liệu. Vì văn bản trong ngôn ngữ tự nhiên không có cấu trúc, làm cho chương trình không thể hiểu được đâu là dữ liệu có giá trị. Quy trình tiền xử lý dữ liệu để có dữ liệu phù hợp được tổng kết như hình sau:



Hình 2.1. Quá trình tiền xử lý dữ liệu cho máy học[2]

Các bước tiền xử lý được sử dụng cho máy học thường gồm các bước: làm sạch dữ liệu, tách từ, chuẩn hóa từ, loại bỏ stopword… và được thể hiện như hình sau:

Hình 2.2. Các bước tiền xử lý dữ liệu

[1] wikipedia: https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%BD_ng%C3%B4n_ng%E1%BB%AF_t%E1%BB%B1_nhi%C3%AAn

[2] https://www.embopress.org/doi/full/10.15252/msb.20156651

3.1. Làm sạch dữ liệu

Dữ liệu được thu thập từ các website đôi khi vẫn còn sót lại các đoạn mã HTML hoặc các đoạn mã javascript. Các mã HTML code này là rác, chẳng những không có tác dụng cho việc phân loại mà còn làm kết quả phân loại văn bản bị kém đi, còn được gọi là noise.

Cách đơn giản nhất để làm sạch dữ liệu là sử dụng công cụ regx[3] để loại bỏ các đoạn mã.

```
def remove_html(self):
    return re.sub(r'<[^>]*>', '', self.text)
```

3.2. Tách từ

Đơn vị từ trong tiếng Việt bao gồm từ đơn và từ ghép. Nên chúng ta cần phải nói cho mô hình học máy biết đâu là từ đơn, đâu là từ ghép. Nếu không thì từ nào cũng sẽ là từ đơn hết.

Bởi vì mô hình của chúng ta sẽ coi các từ là đặc trưng, tách nhau theo dấu cách. Do đó, chúng ta phải nối các từ ghép lại thành một từ để không bị tách sai.

Ở phạm vi ứng dụng này, sử dụng thư viện nguồn mở pyvi để thực hiện tách từ:

```
def segmentation(self):
    return ViTokenizer.tokenize(self.text)

def split_words(self):
    text = self.segmentation()
    try:
        return [x.strip(settings.SPECIAL_CHARACTER).lower()
        for x in text.split()]
    except TypeError:
        return []
```

3.3. Chuẩn hóa

Mục đích là đưa văn bản từ các dạng không đồng nhất về cùng một dạng. Dưới góc độ tối ưu bộ nhớ lưu trữ và tính chính xác cũng rất quan trọng.

Trong tiếng Việt sử dụng 2 loại mã Unicode là Unicode tổ hợp và Unicode dựng sẵn, làm cho mô hình máy học khó phân biệt chính xác từ vựng. Ngoài ra, cách bỏ dấu tiếng Việt và cách viết tắt không giống nhau ở mỗi khu vực cũng làm ảnh hưởng đến tỷ lệ nhận dạng từ ngữ của máy.

[3] https://regex101.com/

Để chuẩn hóa dữ liệu, ta đưa về cùng 1 chuẩn Unicode dựng sẵn như câu lệnh bên dưới:

```
def covert_unicode(self):
    return re.sub(
    r'à|á|ả|ã|ạ|ầ|ấ|ẩ|ẫ|ậ|ằ|ắ|ẳ|ẵ|ặ|è|é|ẻ|ẽ|ẹ|ề|ế|ể|ễ|ệ|ì|í|ỉ
    |ĩ|ị|ò|ó|ỏ|õ|ọ|ồ|ố|ổ|ỗ|ộ|ờ|ớ|ở|ỡ|ợ|ù|ú|ủ|ũ|ụ|ừ|ứ|ử|ữ|ự|ỳ|y
    |ỷ|ỹ|ỵ|À|Á|Ả|Ã|Ạ|Ầ|Ấ|Ẩ|Ẫ|Ậ|Ằ|Ắ|Ẳ|Ẵ|Ặ|È|É|Ẻ|Ẽ|Ẹ|Ề|Ế|Ể|Ễ|Ệ|
    Ì|Í|Ỉ|Ĩ|Ị|Ò|Ó|Ỏ|Õ|Ọ|Ồ|Ố|Ổ|Ỗ|Ộ|Ờ|Ớ|Ở|Ỡ|Ợ|Ù|Ú|Ủ|Ũ|Ụ|Ừ|Ứ|Ử|Ữ
    |Ự|Ỳ|Ý|Ỷ|Ỹ|Ỵ',
        lambda x: dicchar[x.group()], self.txt)
```

Và sử dụng bộ từ điển để thay thế các kiểu gõ dấu và viết tắt

```
def replace_acronyms(self):
    list_text = self.text.split(" ")
    for i in range(len(list_text)):
        for j in range(len(acronyms)):
            if (list_text[i] == acronyms[j][0]):
                list_text[i] = acronyms[j][1]
    self.text = " ".join(list_text)
    return self.text
```

Với danh sách từ viết tắt được thống kê thủ công trong tập tin acronyms_vi:

```
1  ship    vận chuyển
2  shop    cửa hàng
3  m   mình
4  mik mình
5  ko  không
6  k   không
7  kh  không
8  khong   không
9  kg  không
```

Hình 3.1. Nội dung tập tin acronyms_vi

Đồng thời, việc đưa dữ liệu về chữ viết thường là rất cần thiết. Bởi vì đặc trưng này không có tác dụng ở bài toán phân loại văn bản. Đưa về chữ viết thường giúp giảm số lượng đặc trưng (vì máy tính hiểu hoa thường là 2 từ khác nhau) và tăng độ chính xác hơn cho mô hình.

```
def lowercase(self):
    self.text = self.text.lower()
    return self.text
```

Ngoài ra, các từ có ý nghĩa (không nằm trong danh sách Stopword) nhưng bị viết trùng lắp như: hoaa, caii… cũng làm ảnh hưởng đến tỷ lệ chính xác của mô hình, vì vậy cần phải chuẩn hóa:

```
def remove_loop_char(self):
    self.text = re.sub(r'([A-Z])\1+', lambda m: m.group(1).
    upper(), str(self.text),
    flags=re.IGNORECASE)
    self.text = re.sub(
    r'[^\s\wáàảãạăắằẳẵặâấầẩẫậéèẻẽẹêếềểễệóòỏõọôốồổỗộơớờởỡợíìỉĩ
    ịúùủũụưứừửữựýỳỷỹỵđ_]',
    ' ',self.text)
    return self.text
```

3.4. Loại bỏ stopword

StopWords là những từ xuất hiện nhiều trong ngôn ngữ tự nhiên, tuy nhiên lại không mang nhiều ý nghĩa. Ở tiếng việt StopWords là những từ như: để, này, kia... Tiếng anh là những từ như: is, that, this...

Có rất nhiều cách để loại bỏ StopWords nhưng có 2 cách chính là:

+ Dùng từ điển

+ Dựa theo tần suất xuất hiện của từ

Trong phạm vi ứng dụng này, dùng từ điển để loại bỏ stopword

```
def remove_stopwords(self):
    split_words = self.split_words()
    words = []
    for word in split_words:
        if word not in stopwords:
            words.append(word)
    return ' '.join(words)
```

4. Vector hóa văn bản

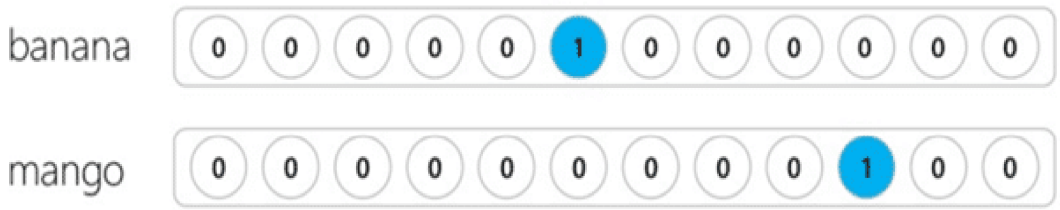
Thông thường, máy tính không thể hiểu được ý nghĩa các từ. Như vậy, để xử lý được ngôn ngữ tự nhiên, ta cần có một phương pháp để biểu diễn văn bản dưới dạng mà máy tính có thể hiểu được. Phương pháp tiêu chuẩn để biểu diễn văn bản đó là biểu diễn các văn bản theo vector. Trong đó, các từ/cụm từ thuộc kho tài liệu ngôn ngữ được ánh xạ thành những vector trên hệ không gian số thực.

Ta có thể chia các phương pháp Vector hóa văn bản thành hai nhóm chính: Phương pháp Word Embedding cổ điển và Neural Embedding (Vector hóa văn bản theo phương pháp mạng nơ-ron).

4.1. Phương pháp Word Embedding

4.1.1. Bag of Words (BoW)

Đây là cách biểu diễn vector truyền thống phổ biến nhất được sử dụng. Mỗi từ hoặc n-gram từ sẽ được mô tả là một vector có số chiều bằng đúng số từ trong bộ từ vựng. Tại vị trí tương ứng với vị trí của từ đó trong túi từ, phần tử trong vector đó sẽ được đánh dấu là 1. Những vị trí còn lại sẽ được đánh dấu là 0.



Ví dụ biểu diễn One-hot BOW của mỗi từ trong văn bản.

Phương pháp BoW thường được sử dụng trong những bài toán phân loại văn bản. Trong đó, tần suất của mỗi từ/ n-gram sẽ được coi là một feature trong văn bản phân loại.

Nhược điểm của phương pháp này là ta không thể xác định được nghĩa thực của mỗi từ và các từ tương quan với chúng.

Trong phương pháp BoW, từ giống nhau sẽ được đánh trọng số như nhau. Phương pháp này không xét đến tần suất xuất hiện của từ hay nhữ cảnh từ. Và trong thực tế, để cần hiểu được nghĩa của mỗi từ, ta cần xác định từ đó trong văn cảnh hơn là xét nghĩa độc lập từ.

4.1.2. TF-IDF

TF- IDF (term frequency–inverse document frequency) – tần suất- tần suất đảo nghịch từ. Đây là một phương pháp thống kê, nhằm phản ánh độ quan trọng của mỗi từ hoặc n-gram đối với văn bản trên toàn bộ tài liệu đầu vào. TF-IDF thể hiện trọng số của mỗi từ theo ngữ cảnh văn bản. TF-IDF sẽ có giá trị tăng tỷ lệ thuận với số lần xuất hiện của từ trong văn bản và số văn bản có chứa từ đó trên toàn bộ tập tài liệu. Phương pháp này giúp cho TF-IDF có tính phân loại cao hơn so với phương pháp trước.

$$tf_i = \frac{n_i}{N_i}$$

Trong đó:
i : 1 .. D
n_i : tần số xuất hiện của từ trong văn bản i.
N_i : tổng số từ trong văn bản i.

$$idf_i = \log_2 \frac{D}{d}$$

Trong đó:
D: tổng số documents trong tập dữ liệu.
d: số lượng documents có sự xuất hiện của từ.

$$tfidf_i = tf_i \times idf_i$$

Tuy nhiên, ngay cả khi phương pháp TF-IDF dựa trên BOW thể hiện được trọng số của các từ khác nhau trong văn bản, nhưng phương pháp này vẫn không biểu diễn được nghĩa của từ.

Đây chính là nhược điểm của hai phương pháp này.

Ta có thể trích dẫn câu của nhà ngôn ngữ học J.R. Firth: "The complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously." (tạm dịch: "Muốn hiểu được ý nghĩa thật sự của một từ, bạn phải dựa vào ngữ cảnh của câu nói")

4.2. Phương pháp Neural Embedding

4.2.1. Word2vec

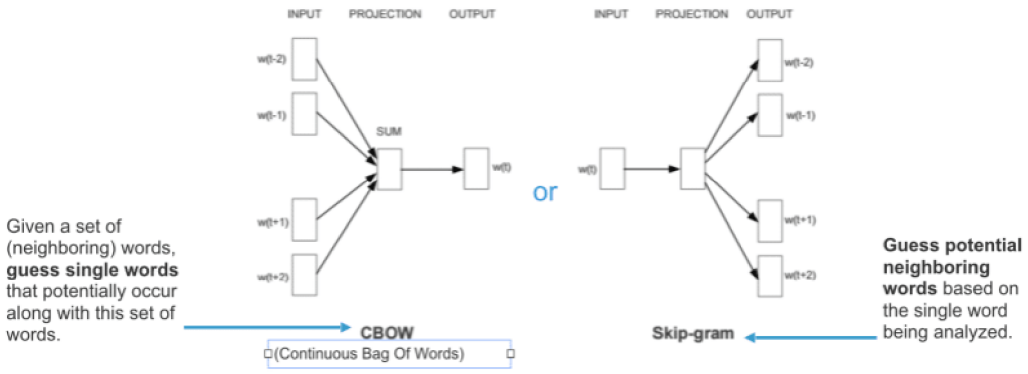
Word2vec là thuật toán theo phương pháp dự đoán (Prediction-based embedding). Mô hình dự đoán học biểu diễn vector từ thông qua những từ ngữ cảnh xung quanh nhằm cải thiện khả năng dự đoán ý nghĩa các từ.

Có hai cách xây dựng mô hình Word2vec để biểu diễn sự phân tán của từ trong không gian vector:

Sử dụng ngữ cảnh để dự đoán mục tiêu (CBOW): khi vị trí của các từ ngữ cảnh không ảnh hưởng tới việc dự đoán từ (giả định ban đầu của CBOW). Trong mô hình Skip-gram, mô hình sử dụng từ ngữ hiện tại để dự đoán những từ xung quanh trong ngữ cảnh đó.

Sử dụng một từ để dự đoán ngữ cảnh mục tiêu (Continuous skip-gram) xem xét những từ ngữ cảnh xung quanh sẽ được đánh giá tốt hơn so với những

từ trong ngữ cảnh nhưng ở vị trí xa hơn. Mặc dù thứ tự từ vẫn không được xem xét, mỗi vector của từ bối cảnh được xem xét và cân nhắc.



Thuật toán CBOW tốn ít thời gian huyến luyện mô hình hơn Skip-gram. Tuy nhiên, Skip-gram có độ chính xác cao hơn và có chứa cả những từ ít xuất hiện.

4.2.2. GloVe

Cả CBOW và Skip-Gram đều là các mô hình dự đoán. Trong đó, các thuật toán chỉ xem xét được ngữ cảnh xung quanh từ mục tiêu nhưng không đề cập được về ngữ cảnh toàn văn bản. Thuật toán GloVe dựa trên tương phản có lợi với cùng dự đoán của ma trận đồng xuất hiện sử dụng trong thuật toán Distributional Embedding, nhưng sử dụng phương pháp Neural Embedding để phân tích ma trận đồng xuất hiện thành những vector có ý nghĩa và tỷ trọng hơn.

Mặc dù thuật toán GloVe nhanh hơn Word2Vec, nhưng cả GloVe và Word2Vec đều không hiển thị để cung cấp kết quả tốt và rõ ràng hơn thay vì cả hai nên được đánh giá cho một tập dữ liệu nhất định.

4.2.3. FastText

FastText, được xây dựng trên Word2Vec bằng cách học các biểu diễn vectơ cho mỗi từ và n-gram được tìm thấy trong mỗi từ. Các giá trị của các biểu diễn sau đó được tính trung bình thành một vectơ ở mỗi bước đào tạo. Trong khi điều này bổ sung rất nhiều tính toán bổ sung cho việc đào tạo, nó cho phép nhúng từ để mã hóa thông tin từ phụ. Các vectơ FastText đã được chứng minh là chính xác hơn các vectơ Word2Vec bằng một số biện pháp khác nhau

Qua thực nghiệm 2 phương pháp Word Embedding cổ điển và Neural Embedding trên tập dữ liệu chúng tôi nhận thấy phương pháp Word Embedding với bộ trích đặc trưng sử dụng TF-IDF có độ chính xác cao nhất

do đó chúng tôi lựa chọn xây dựng bộ trích đặc trưng sử dụng TF-IDF để thực hiện đề tài.

4.3. Xây dựng bộ trích đặc trưng sử dụng TF-IDF

Với các thuật toán machine learning truyền thống sử dụng thư viện sklearn, xây dựng bộ trích xuất đặc trưng (feature extractor) sử dụng TF-IDF.

```
class FeatureExtractionTFIDF(object):

    def __init__(self, pathTrain = "", pathTest = "", text = ""):
        self.pathTrain = pathTrain
        self.pathTest = pathTest
        self.dataTrain = []
        self.dataTest = []
        self.text = text

    def __build_dataset(self):
        dict_data = pd.DataFrame(DataSource(filePath = self.pathTrain).load_data())
        self.labels = dict_data.label
        dict_data = dict_data.comment.tolist()

        for i in range(len(dict_data)):
            text_ = dict_data[i]
            text = PreProcessing(text = text_).text_util_final()
            self.dataTrain.append(text)

        vectorizer = TfidfVectorizer(ngram_range=(1,3),min_df=1,use_idf=True)
        return vectorizer.fit(self.dataTrain)

    def get_data_and_label(self):
        vectorizer = self.__build_dataset()
        self.features = vectorizer.transform(self.dataTrain)
        return vectorizer, self.features, self.labels
```

5. Phân loại dữ liệu văn bản

Mục tiêu của bài toán phân lớp là dự đoán một nhãn (label) *y* phù hợp nhất cho túi từ x. Để dự đoán một nhãn cho túi từ x, mỗi từ trong x đó sẽ được gán một giá trị điểm số (score) hay còn gọi là trọng số (weight hay θ) để đo lường sự tương thích (compatibility) với nhãn.

5.1. Một số bộ phân lớp

Một số bộ phân lớp dữ liệu văn bản

5.1.1. Bộ phân lớp Naïve Bayes

Naïve Bayes Classifier (NBC) là mô hình phân lớp văn bản dựa trên định lý Bayes.

Một số mô hình Naïve Bayes hay sử dụng là Multinomial Naive Bayes, Bernoulli Naive Bayes và Gaussian Naive Bayes.

Phương pháp phân lớp Multinomial Naïve Bayes hay còn lại là phương pháp phân lớp Bayes (Bayesian classification) thường được sử dụng để phân lớp văn bản.