ĐẠI HỌC BÁCH KHOA HÀ NỘI
Trường Công nghệ Thông tin và Truyền thông

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: PROJECT I

MÃ LỚP: 733501

ĐỀ TÀI: SỬ DỤNG MÔ HÌNH HỌC MÁY SUPPORT VECTOR MACHINE ĐỂ GIẢI QUYẾT BÀI TOÁN PHÂN LOẠI, ĐÁNH GIÁ CẢM XÚC (SENTIMENT ANALYSIS)

Giảng viên hướng dẫn: PSG.TS Lê Thanh Hương

Sinh viên thực hiện: Nguyễn Long Nhật **MSSV: 20215440**

Lớp: Khoa học máy tính 05 – K66

HÀ NỘI, 1/2024

Mục lục

LỜI NÓI ĐẦU

Phân tích cảm xúc (Sentiment Analysis) là một trong những bài toán quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên và vô cùng cần thiết, hữu ích trong cuộc sống, đặc biệt là bối cảnh cuộc Cách mạng 4.0. Để tiếp cận và giải quyết bài toán này, chúng ta phải xây dựng một mô hình máy học để có thể phân loại, đánh giá cảm xúc.

PHẦN I: GIỚI THIỆU, MÔ TẢ BÀI TOÁN

CHƯƠNG 1: GIỚI THIỆU VỀ ĐỀ TÀI

KHÁI QUÁT CHUNG VỀ VẤN ĐỀ NGHIÊN CỨU

Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) đề cập đến việc áp dụng các kỹ thuật và phương pháp để thao tác trên ngôn ngữ tự nhiên bằng máy tính. Ngôn ngữ tự nhiên bao gồm các ngôn ngữ sử dụng trong giao tiếp hàng ngày như tiếng Việt, tiếng Anh, tiếng Nhật,… trong khi ngôn ngữ nhân tạo như ngôn ngữ lập trình, ngôn ngữ máy là những ngôn ngữ không phải là ngôn ngữ tự nhiên. NLP là một lĩnh vực kết hợp giữa khoa học máy tính, trí tuệ nhân tạo và ngôn ngữ học tính toán, tập trung vào tương tác giữa ngôn ngữ con người và máy tính. Trong lĩnh vực trí tuệ nhân tạo, xử lý ngôn ngữ tự nhiên được coi là một trong những thách thức khó khăn nhất do nó đòi hỏi khả năng hiểu ý nghĩa của ngôn ngữ, là công cụ tư duy và giao tiếp chính.

Phân loại văn bản, hay Text Classification, là một phần của quá trình Phân tích Cảm xúc (Sentiment Analysis). Phân tích cảm xúc, đề cập đến việc sử dụng xử lý ngôn ngữ tự nhiên và phân tích văn bản để nhận diện, trích xuất, đo lường, và nghiên cứu các trạng thái tâm lý và thông tin chủ quan một cách có hệ thống. Bài toán này được áp dụng rộng rãi trong việc đánh giá phản hồi từ khách hàng, trên các phương tiện truyền thông trực tuyến và xã hội, cũng như trong các dữ liệu liên quan đến y tế, tiếp thị, hay các tập dữ liệu về tài chính,...

Mục tiêu của dự án này là thực hiện Phân tích cảm xúc cho các đánh giá liên quan đến tài chính. Do đó, dữ liệu sẽ được sử dụng trong dự án này là các đánh giá về tài chính, được thu thập từ các nguồn dữ liệu tài chính.

PHẦN II: PHƯƠNG PHÁP THỰC HIỆN (SUPPORT VECTOR MACHINE)

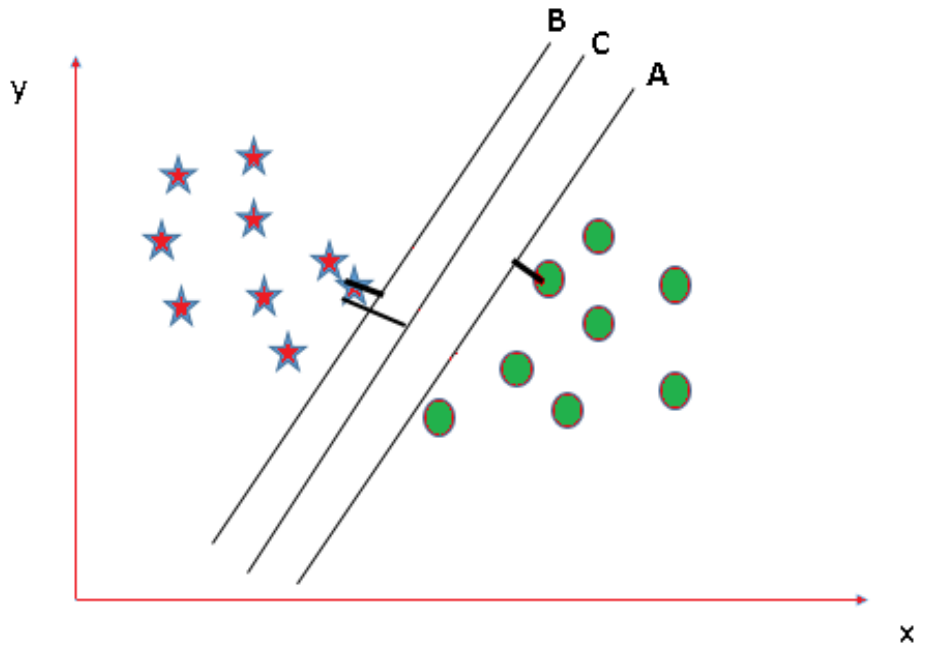
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Support Vector Machine

2.1.1 Support Vector Machine là gì?

Support Vector Machine (SVM) là một thuật toán học có giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồ thị dữ liệu là các điểm trong n chiều. Sau đó chúng ta thực hiện tìm "Hyper-plane" để phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.

Ưu điểm của thuật toán này là nó hoạt động tốt, hiệu quả trong các không gian có số chiều cao. Tuy nhiên nó cũng tồn tại những nhược điểm phải kể tới như khả năng xử lý với tập dữ liệu lớn, chưa thể hiện rõ tính xác suất

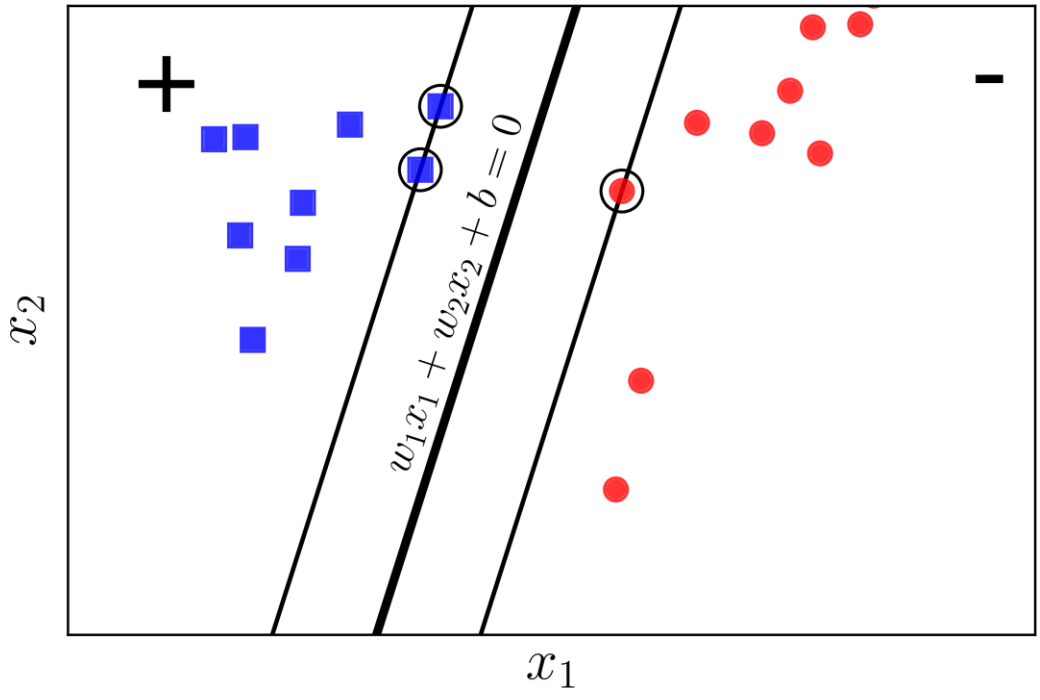


Từ hình trên ta có thể thấy có 3 đường A, B, C, và 2 lớp (ngôi sao đỏ và hình tròn xanh). Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class tới đường phân chia là như nhau. Khoảng cách như nhau này được gọi là *margin* (*lề*).

2.1.2 Cơ sở lý thuyết của thuật toán SVM

Giả sử rằng các cặp dữ liệu của tập huấn luyện (training set) là (x_1, y_1), (x_2, y_2),…, (x_N, y_N) với vector $x_i \, \epsilon \, \mathbb{R}^d$ thể hiện đầu vào của một điểm dữ liệu và y_i là nhãn của điểm dữ liệu đó, d là số chiều của dữ liệu và N là số điểm dữ liệu. Giả sử rằng nhãn của mỗi điểm dữ liệu được xác định bởi $y_i = 1 \, (class \, 1)$ hoặc $y_i = -1 \, (class \, 2)$

Để dễ hình dung, ví dụ như trong không gian 2 chiều như hình dưới đây.



Hình 1: Phân tích bài toán SVM

Giả sử rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộc class -1 và mặt $\boldsymbol{w^T x + b = \omega_1 x_1 + \omega_2 x_2 + b = 0}$ là mặt phân chia giữa hai classes (Hình 1). Hơn nữa,

class 1 nằm về phía dương, class -1 nằm về phía âm của mặt phân chia. Nếu ngược lại, ta chỉ cần đổi dấu của ***w*** và **b.** Ta cần đi tìm các hệ số ***w*** và **b**

Ta quan sát thấy một điểm quan trọng sau đây: với cặp dữ liệu (x_n, y_n) bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

Điều này có thể dễ nhận thấy vì theo điều ta đã giả sử ở trên,
y_n *luôn cùng dấu với phía của* x_n. Từ đó suy ra y_n *cùng dấu với* $(\mathbf{w^T x_n} + \mathbf{b})$, và tử số luôn là số không âm

Với mặt phân chia như trên, *margin* được tính là khoảng cách gần nhất từ 1 điểm tới mặt đó (bất kể điểm nào trong hai classes):

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

Bài toán tối ưu trong SVM chính là bài toán tìm **w** và **b** sao cho *margin* này đạt giá trị lớn nhất:

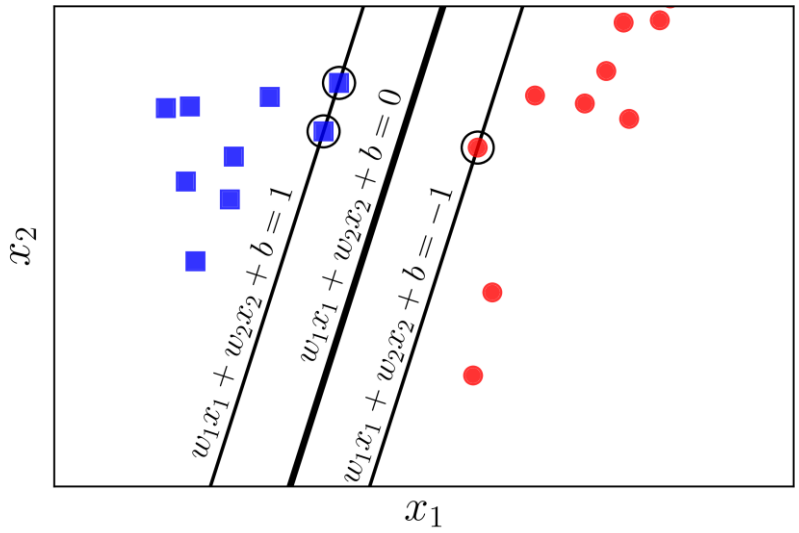
$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\} \quad (1)$$

Việc giải trực tiếp bài toán này sẽ rất phức tạp, nhưng chúng ta có thể để đưa nó về bài toán đơn giản hơn.

Nhận xét quan trọng nhất là nếu ta thay vector hệ số **w** bởi **kw** và **b** bởi **kb** trong đó **k** là một hằng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ từng điểm đến mặt phân chia không đổi, tức *margin* không đổi. Dựa trên tính chất này, ta có thể giả sử:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$$

với những điểm nằm gần mặt phân chia nhất như Hình 2 dưới đây:



Hình 2: Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn

Như vậy, với mọi n, ta có:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$$

Vậy bài toán tối ưu (1) có thể đưa về bài toán tối ưu có ràng buộc sau đây:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \frac{1}{||\mathbf{w}||_2}$$
$$\text{subject to:} \quad y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \forall n = 1, 2, \ldots, N \quad (2)$$

Bằng 1 vài phép biến đổi đơn giản, ta có thể đưa bài toán này về dạng sau:

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_2^2$$
$$\text{subject to:} \quad 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \ldots, N \quad (3)$$

Ở đây, chúng ta đã lấy nghịch đảo hàm mục tiêu, bình phương nó để được một hàm khả vi, và nhân với $\frac{1}{2}$ để biểu thức đạo hàm đẹp hơn

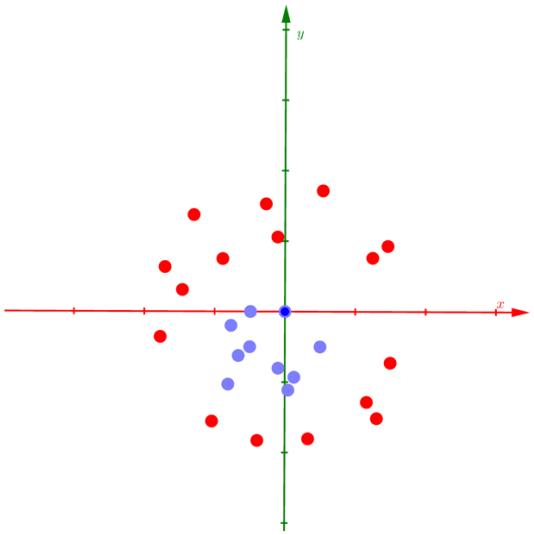
Để giải quyết bài toán này, chúng ta thường giải bài toán đối ngẫu của nó.

Trước hết, dễ thấy rằng bài toán tối ưu (3) là một bài toán lồi do hàm mục tiêu là một norm, các bất đẳng thức rang buộc là các hàm tuyến tính theo w và b, và từ đó việc giải bài toán này cũng trở nên dễ dàng hơn dựa vào các điều kiện, tiêu chuẩn Slater, Lagrange, điều kiện Karush – Kuhn – Tucker

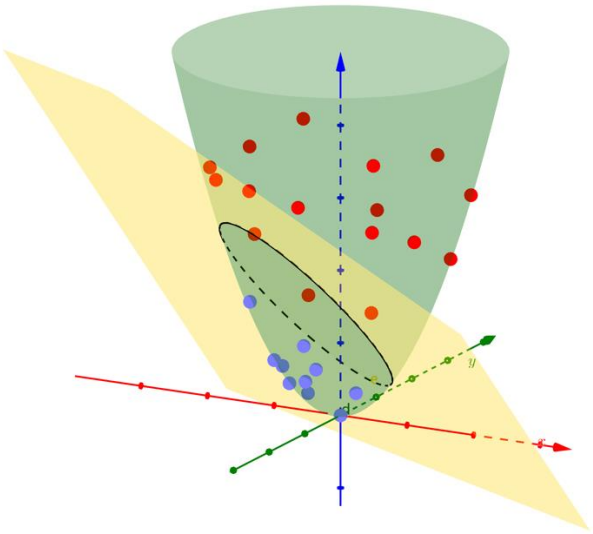
2.1.3 Kernel SVM

Ý tưởng cơ bản của Kernel SVM và các phương pháp kernel nói chung là tìm một phép biến đổi sao cho dữ liệu ban đầu là *không biệt tuyến tính* được biến sang không gian mới. Và ở không gian mới này, dữ liệu trở nên *phân biệt tuyến tính*

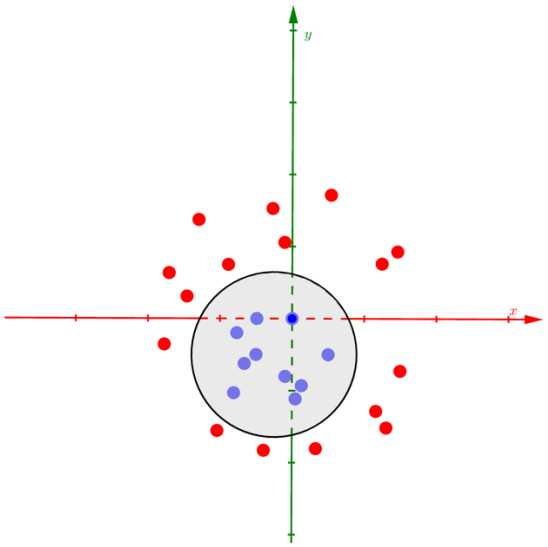
Xét ví dụ dưới đây với việc biến dữ liệu *không phân biệt tuyến tính* trong không gian hai chiều thành *phân biệt tuyến tính* trong không gian ba chiều bằng cách giới thiệu thêm một chiều mới:



Hình 3



Hình 4



Hình 5

Hình 3 là ví dụ về dữ liệu của 2 lớp không phân biệt tuyến tính trong không gian 2 chiều. Hình 4 chúng ta xét them chiều thứ ba là một hàm số của 2 chiều còn lại z = $x^2 + y^2$, các điểm dữ liệu sẽ được phân bố trên 1 parabolic và đã trở nên phân biệt tuyến tính với mặt phẳng màu vàng là mặt phân chia. Hình 5 là giao điểm của mặt phẳng phân cách đã tìm được và mặt parabolic là một đường ellipse, khi chiếu toàn bộ dữ liệu cũng như đường ellipse này xuống không gian hai chiều ban đầu, ta tìm được đường phân chia 2 lớp

CHƯƠNG 3: TIỀN XỬ LÝ DỮ LIỆU

Dữ liệu thu thập về sẽ có dạng thô do chưa được xử lý nên vẫn còn lỗi như có thể bị rỗng, sai chính tả, nhiễu với nhiều ký tự thừa, ký tự đặc biệt. Điều này sẽ làm ảnh hưởng đến quá trình và kết quả phân tích, vì vậy ta cần phải xử lý, làm sạch dữ liệu

3.1 Thống kê dữ liệu

```
In [124]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14780 entries, 0 to 14779
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   sentences   14780 non-null  object
 1   sentiments  14780 non-null  object
dtypes: object(2)
memory usage: 231.1+ KB
```

Tập dữ liệu gồm 14780 bản ghi, gồm 2 cột thuộc tính:

- Cột sentences là các dữ liệu thông tin tài chính bằng các văn bản bằng Tiếng Anh
- Cột sentiments là các nhãn “positive”, “neutral”, “negative” cho các sentences tương ứng

Dữ liệu không chứa bản ghi nào là “null”

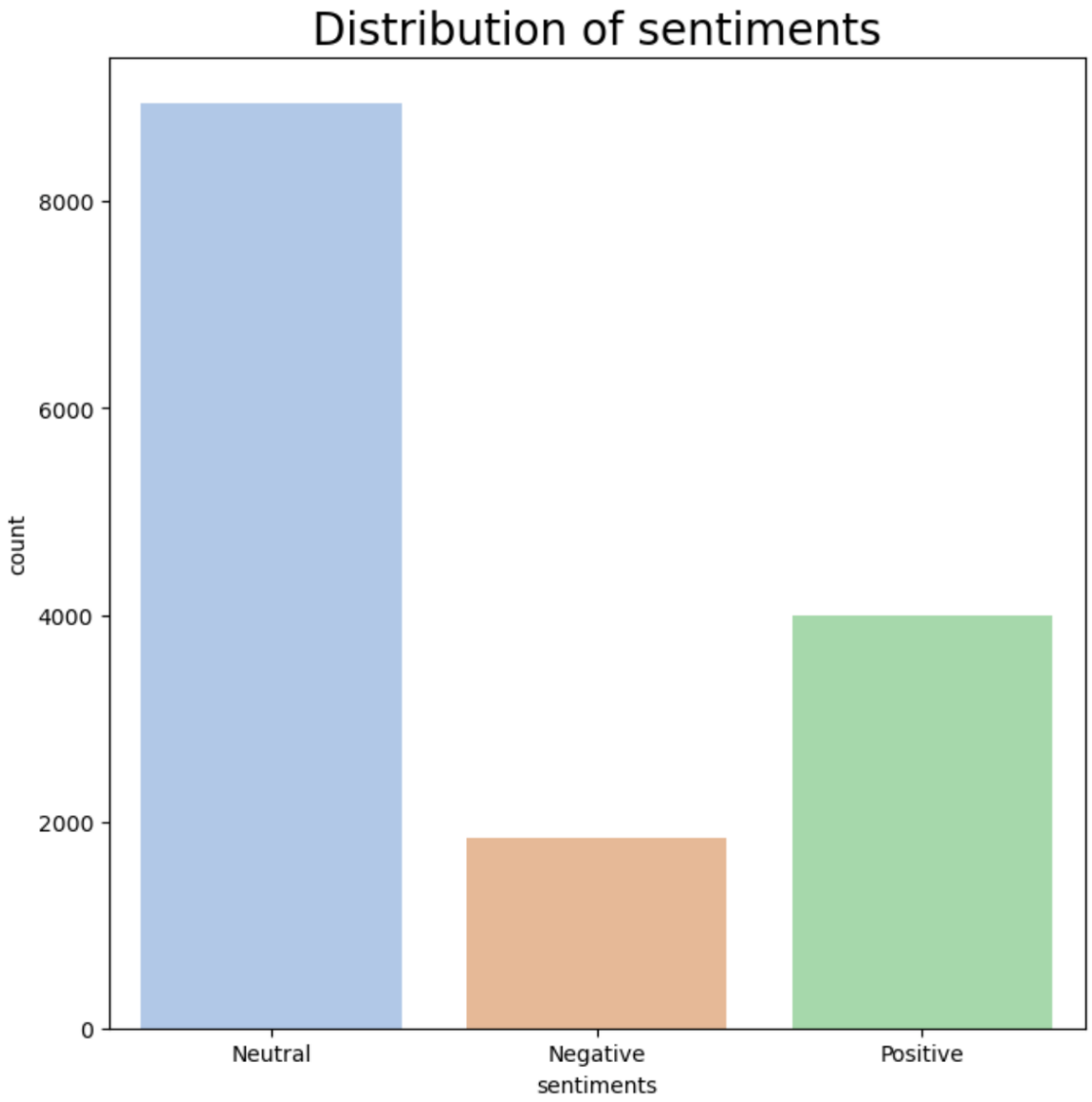
```
print(df.isnull().sum())
```

```
sentences     0
sentiments    0
dtype: int64
```

Hiển thị đồ thị phân tích sự phân bố dữ liệu

```
print(df.sentiments.value_counts())
plt.figure(figsize = (8, 8))
ax = sns.countplot(x = 'sentiments', data = df, palette = 'pastel')
ax.set_title(label = 'Distribution of sentiments', fontsize = 20)
ax.set_xticklabels(['Neutral','Negative', 'Positive'], rotation=0)
plt.show()
```

```
sentiments
neutral     8951
positive    3988
negative    1841
Name: count, dtype: int64
```



Từ biểu đồ trên ta thấy tập dữ liệu chưa được cân bằng khi phần lớn các bản ghi có nhãn cảm xúc “neutral” trong khi các bản ghi “positive” và “negative” lại có số lượng ít hơn , điều này sẽ làm ảnh hưởng tới quá trình và kết quả đầu ra của bài toán

Vì vậy, tiếp theo, ta cần phải cân bằng tập dữ liệu và bằng cách lấy mẫu các bản ghi “neutral” và “positive” xuống cùng mức với các bản ghi “negative”, tức là lấy 2000 mẫu từ tập dữ liệu với các bản ghi “positive” và “neutral” để cân bằng và mô hình sẽ cho hiệu

suất tốt.

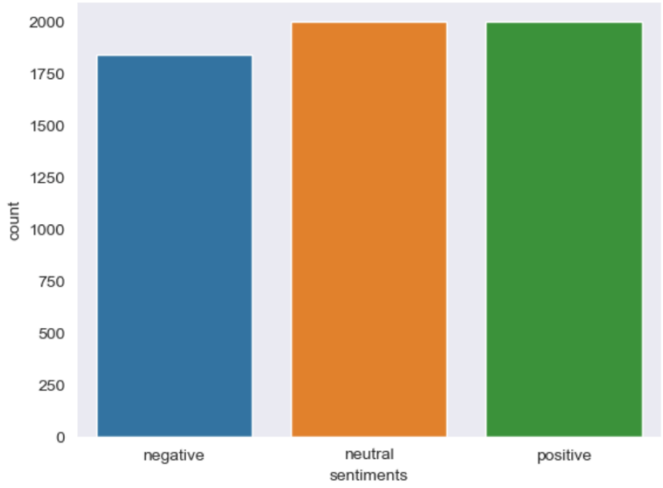
```
In [6]: #balance the data
df_pos = df[df.sentiments == 'positive'].head(2000)
df_neu = df[df.sentiments == 'neutral'].head(2000)
df_neg = df[df.sentiments == 'negative']

df_final = pd.concat([df_pos, df_neg], axis = 0)
df_final = pd.concat([df_final, df_neu], axis = 0)

# Convert 'sentiments' column to categorical type
df_final['sentiments'] = pd.Categorical(df_final['sentiments'])

# Plot the countplot
sns.set_style("dark")
sns.countplot(data=df_final, x='sentiments')
```

```
Out[6]: <Axes: xlabel='sentiments', ylabel='count'>
```



Biểu đồ trên cho thấy dữ liệu sau khi đã cân bằng để đạt được hiệu quả tốt hơn

Tiếp theo, ta xáo trộn dữ liệu để tránh việc sắp xếp dữ liệu có thể ảnh hưởng đến hiệu suất của mô hình. Nếu dữ liệu được sắp xếp theo một cách nào đó, mô hình có thể sẽ học được các mối quan hệ có thể không tồn tại trong dữ liệu tổng thể. Do đó, ta thực hiện quá trình xáo trộn để đảm bảo rằng mô hình được huấn luyện trên một tập dữ liệu ngẫu nhiên và đại diện cho toàn bộ dữ liệu.

```
In [7]: #shuffle

df_final = df_final.reindex(np.random.permutation(df_final.index))
df_final.head(20)
```

Thêm một trường dữ liệu vào data frame có ý nghĩa độ dài số từ trong câu (trường dữ liệu “sentences”)

```
In [9]: df_final['length'] = df_final['sentences'].apply(len)
        df_final.head()
```

Out[9]:

	sentences	sentiments	length
2890	Financial details were n't disclosed .	neutral	38
4895	Seppala 's revenue increased by 0.2 % to EUR10 .1 m. In Finland , revenue went down by 2.4 % to ...	positive	233
2273	EBIT totalled EUR 14.4 mn , compared to a loss of EUR 0.3 mn in the corresponding period in 2009 .	positive	98
4499	W+ñrtsil+ñ 's solution has been selected for its low fuel consumption , environmentally sound te...	positive	135
11921	Finnish communication electronics components supplier Scanfil Oyj Tuesday said sales in the firs...	negative	170

Tiếp theo, ta sẽ kiểm tra tổng thể sự phân bổ độ dài của văn bản trong tập dữ liệu.

```
In [10]: df_final.length.describe()
```

```
Out[10]: count    5841.000000
         mean      129.366547
         std        55.963152
         min         9.000000
         25%        85.000000
         50%       120.000000
         75%       165.000000
         max       315.000000
         Name: length, dtype: float64
```

Như chúng ta có thể thấy rằng độ dài cao nhất của câu trong tập dữ liệu là 315 trong khi độ dài tối thiểu là 9. Độ dài trung bình của trường “sentences” trong tập dữ liệu là 129.3

3.2 Tiền xử lý dữ liệu

Loại bỏ các dấu câu (Punctuations) và phân tích dữ liệu

Một kỹ thuật tiền xử lý văn bản phổ biến là loại bỏ các dấu câu từ dữ liệu văn bản. Đây là một quá trình tiêu chuẩn hóa văn bản sẽ giúp xử lý các ký tự ví dụ như ! "#$%& \ '()*+,-./:; <=>?@[\\]^_ {|} ~` trong văn bản

Word cloud biểu diễn dưới dạng đồ họa của tần suất xuất hiện của các từ, qua đó làm nổi bật các từ xuất hiện thường xuyên hơn trong tập dữ liệu. Từ trong hình ảnh càng lớn thì từ đó càng phổ biến và có ý nghĩa quan trọng.

Removing punctuation and WordCloud Display

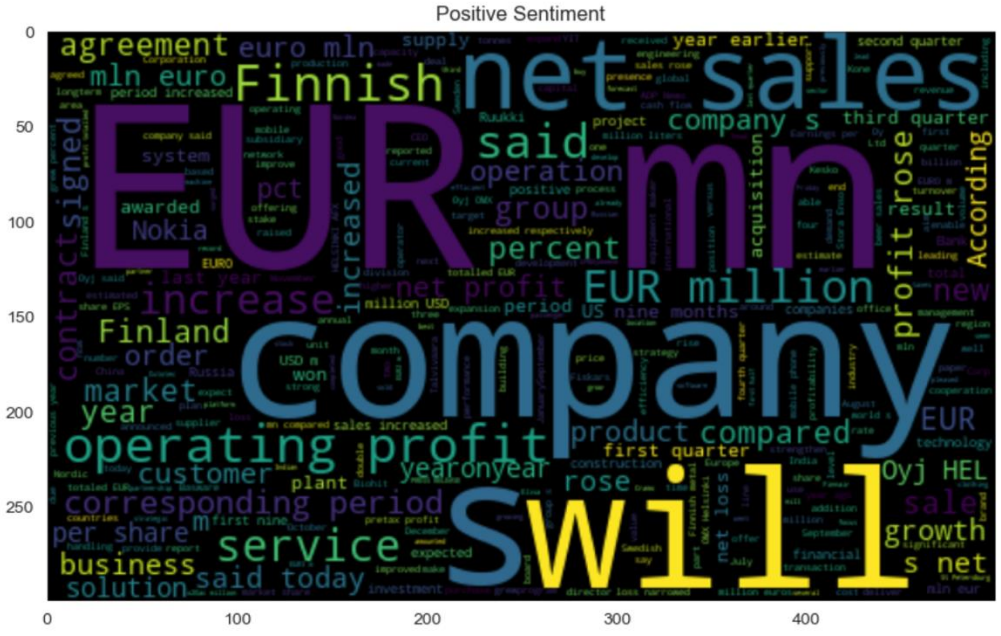
In [29]:
```
import string
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Creating list of punctuation
punctuation = list(string.punctuation)

# Removing punctuation from the financial text
def remove_punctuation(text):
    for punctuation_mark in punctuation:
        text = text.replace(punctuation_mark, '')
    return text

df_pos['sentences'] = df_pos['sentences'].apply(remove_punctuation)
df_neg['sentences'] = df_neg['sentences'].apply(remove_punctuation)
df_neu['sentences'] = df_neu['sentences'].apply(remove_punctuation)

# Plotting Positive sentiment wordcloud
plt.figure(figsize=(10, 12))
wc = WordCloud(max_words=500, width=500, height=300).generate(" ".join(df_pos.sentences))
plt.imshow(wc, interpolation='bilinear')
plt.title('Positive Sentiment')
plt.show()
```

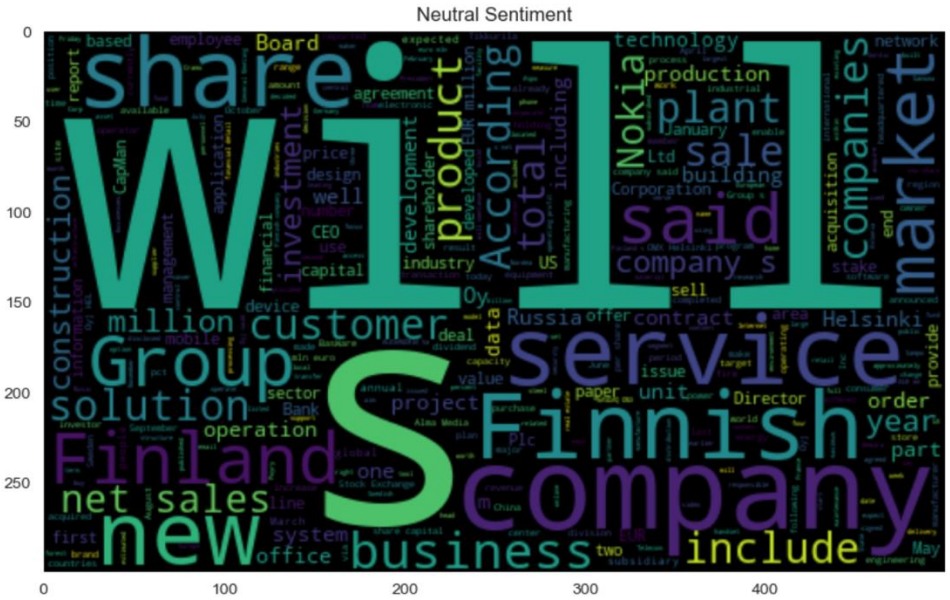


In [13]:
```
# plotting Negative sentiment wordcoud
plt.figure(figsize = (10,12))
wc = WordCloud(max_words = 500 , width = 500 , height = 300).generate(" ".join(df_neg.sentences))
plt.imshow(wc , interpolation = 'bilinear')
plt.title('Negative Sentiment')
```

Out[13]: Text(0.5, 1.0, 'Negative Sentiment')

In [14]:
```
# plotting Neutral sentiment wordcoud
plt.figure(figsize = (10,12))
wc = WordCloud(max_words = 500 , width = 500 , height = 300).generate(" ".join(df_neu.sentences))
plt.imshow(wc , interpolation = 'bilinear')
plt.title('Neutral Sentiment')
```

Out[14]: Text(0.5, 1.0, 'Neutral Sentiment')



Trên đây là WordCloud tương ứng cho 3 nhãn “positive”, “negative” và “neutral”

Bổ đề hóa (Lemmatization)

Ngược lại với từ gốc, quá trình bổ đề hóa (lemmatization) được xem là một phương pháp mạnh mẽ hơn đáng kể trong xử lý ngôn ngữ tự nhiên. Quá trình lemmatization giúp chuẩn hóa từ ngữ trong văn bản, giúp mô hình hiểu rõ hơn về ý nghĩa thực sự của từ, bởi vì nó giữ nguyên từ điển của từ đó. Nó có thể giúp giảm số lượng biến thể của một từ, làm cho các từ có cùng nguồn gốc được biểu diễn một cách thống nhất, từ đó cải thiện hiệu suất của các nhiệm vụ xử lý ngôn ngữ tự nhiên như phân loại văn bản

Lower Casting là một kỹ thuật tiền xử lý văn bản phổ biến. Ý tưởng là chuyển đổi văn bản đầu vào thành cùng một định dạng vỏ sao cho ' text ', 'Text ' và ' TEXT ' được xử lý theo cùng một cách

In [27]:
```
from nltk.stem import WordNetLemmatizer
import nltk
import re

lemma = WordNetLemmatizer()

# Creating list of possible stopwords from nltk library
stop = stopwords.words('english')

def cleanText(txt):
    # Lowercasing
    txt = txt.lower()
    # Tokenization
    words = nltk.word_tokenize(txt)
    # Lemmatizing the words
    words = ' '.join([lemma.lemmatize(word) for word in words])
    # Removing non-alphabetic characters
    txt = re.sub('[^a-z]',' ', words)
    return txt

# Applying cleanText function on the 'sentences' column
df_final['cleaned_text'] = df_final['sentences'].apply(cleanText)
df_final.head()
```

Out[27]:

	sentences	sentiments	length	cleaned_text	negation_count
3314	The parties have therefore agreed to leave Avena out of the deal .	neutral	66	the party have therefore agreed to leave avena out of the deal	0
5058	(ADP News) - Feb 12 , 2009 - Finnish IT solutions provider Affecto Oyj (HEL : AFE1V) said to...	positive	187	adp news feb finnish it solution provider affecto oyj hel afe v said tod...	0
1386	The handset also features a Media Bar for quick access to favorite media and applications , incl...	neutral	141	the handset also feature a medium bar for quick access to favorite medium and application incl...	0
2502	Last week , however , Nokia announced that it will pursue a long-term relationship with Microsof...	neutral	186	last week however nokia announced that it will pursue a long term relationship with microsof...	0
139	A structures BIM (building information modeling) software from Tekla , a model-based software ...	neutral	207	a structure bim building information modeling software from tekla a model based software p...	0

Hàm **cleanText** nhận vào một đoạn văn bản (**txt**) và thực hiện các bước tiền xử lý.
Đầu tiên, toàn bộ văn bản được chuyển về chữ thường.
Sau đó, từng từ được tách ra sử dụng **nltk.word_tokenize**.
Các từ được lemmatize bằng cách sử dụng **lemma.lemmatize**.
Cuối cùng, mọi ký tự không phải là chữ cái sẽ được loại bỏ.

CHƯƠNG 4: HUẤN LUYỆN MÔ HÌNH

Sử dụng hàm train_test_split ở thư viện scikit-learn để chia dữ liệu thành tập dữ liệu và tập kiểm thử.

Split Train - Test Data

```
In [16]: X=df_final.cleaned_text
         y = df_final.sentiments

In [17]: X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.20,stratify=y, random_state=0)
```

- **test_size=0.20**

Tham số này xác định tỷ lệ của dữ liệu sẽ được chia vào tập kiểm tra. Trong trường hợp này, 20% của dữ liệu sẽ được sử dụng cho tập kiểm thử, và 80% còn lại sẽ được sử dụng cho tập huấn luyện.

- **stratify=y**

Tham số này đảm bảo rằng phân phối của các lớp trong biến phụ thuộc **y** được duy trì trong cả tập huấn luyện và tập kiểm tra. Điều này là quan trọng khi dữ liệu có sự mất cân bằng giữa các lớp, và nó giúp đảm bảo rằng cả hai tập đều đại diện đúng cho các lớp khác nhau.

- **random_state=0**

Tham số này đặt một hạt giống (seed) cho quá trình tạo số ngẫu nhiên. Điều này làm cho kết quả của quá trình chia dữ liệu trở nên dự đoán được và lặp lại được. Khi sử dụng một giá trị cụ thể cho **random_state**, kết quả sẽ không thay đổi mỗi lần chạy mã, giúp dễ dàng so sánh và tái tạo kết quả.

Tiếp theo, sử dụng TfidfVectorizer từ thư viện scikit-learn để chuyển đổi các văn bản thành ma trận TF-IDF

TF-IDF Vectorizer

```
In [18]: tfidf_vectorizer = TfidfVectorizer( max_df=0.8, ngram_range=(1,2))
         tfidf_train = tfidf_vectorizer.fit_transform(X_train)
         tfidf_test = tfidf_vectorizer.transform(X_test)
```

- **max_df=0.8**

Tham số này là giới hạn tần suất tối đa của một từ trong các văn bản. Các từ có tần suất xuất hiện trong hơn 80% của tất cả các văn bản sẽ không được tính trong ma trận TF-IDF. Điều này giúp loại bỏ các từ phổ biến và có thể không mang lại nhiều thông tin giữa các văn bản.

- **ngram_range=(1,2)**

Tham số này xác định kích thước của các n-grams (tổ hợp các từ liên tiếp) mà **TfidfVectorizer** sẽ sử dụng. Trong trường hợp này, **ngram_range=(1,2)** chỉ định rằng

chúng ta muốn sử dụng các từ đơn (unigrams) và cặp từ (bigrams). Điều này có thể giúp bắt capture các mối quan hệ giữa các từ liên tiếp.

- **fit_transform(X_train)**

Phương thức này được sử dụng để học từ vựng từ tập huấn luyện (**X_train**) và biến đổi các văn bản trong tập huấn luyện thành ma trận TF-IDF. Tập từ vựng được xây dựng dựa trên tập huấn luyện và sau đó áp dụng cho cả tập huấn luyện và tập kiểm tra để đảm bảo cùng một từ vựng được sử dụng.

- **transform(X_test)**

Phương thức này sử dụng từ vựng đã học từ tập huấn luyện để biến đổi các văn bản trong tập kiểm tra (**X_test**) thành ma trận TF-IDF. Điều này đảm bảo rằng cả hai tập đều được biểu diễn dựa trên cùng một bộ từ vựng.

Cuối cùng, **tfidf_train** và **tfidf_test** là kết quả cuối cùng, là ma trận TF-IDF của tập huấn luyện và tập kiểm tra tương ứng. Các giá trị trong ma trận này thể hiện độ quan trọng của từng từ trong văn bản dựa trên tần suất xuất hiện của chúng trong toàn bộ tập dữ liệu.

Đặc trưng "negative count"

Đặc trưng này đo lường số lần xuất hiện của các từ phủ định trong mỗi câu. Cụ thể, danh sách **negation_words** chứa các từ như "not", "no", "never", "don't", "can't", "won't", "didn't".

Negative word feature

```
In [19]: negation_words = ["not", "no", "never", "don't", "can't", "won't", "didn't"]

def count_negation_words(text):
    words = nltk.word_tokenize(text)
    negation_count = sum(1 for word in words if word.lower() in negation_words)
    return negation_count

df_final['negation_count'] = df_final['sentences'].apply(count_negation_words)
X_train_nega_count = np.array([count_negation_words(sentence) for sentence in X_train])
X_test_nega_count = np.array([count_negation_words(sentence) for sentence in X_test])
```

Hàm **count_negation_words** được sử dụng để tính số lượng từ phủ định trong mỗi câu. Nó tokenizes câu thành các từ và đếm số từ nằm trong danh sách **negation_words**. Hai mảng **X_train_nega_count** và **X_test_nega_count** để lưu trữ số lượng từ phủ định trong mỗi câu tương ứng trong tập huấn luyện và tập kiểm thử

Đặc trưng "Sentiwordnet"

SentiWordNet là một từ điển cảm xúc, là một phần mở rộng của WordNet, được sử dụng để liên kết các từ với các điểm cảm xúc như tích cực, tiêu cực và trung tính.

Sentiwordnet feature

In [20]:
```
from nltk.corpus import sentiwordnet as swn
def get_sentiment_score(word):
    # Get positive and negative scores for a word from SentiWordNet
    try:
        sentiment = swn.senti_synset(word + '.n.01')
        return sentiment.pos_score() - sentiment.neg_score()
    except:
        return 0

def get_sentiment_feature(sentence):
    # Calculate the average sentiment score for all words in a sentence
    words = nltk.word_tokenize(sentence)
    sentiment_scores = [get_sentiment_score(word) for word in words]
    return np.mean(sentiment_scores)

def combine_features(tfidf_features, sentiwordnet_features, nega_count):
    # combine features TFIDF, SentiWordnet, Negation_count
    return np.hstack((tfidf_features.toarray(), sentiwordnet_features.reshape(-1, 1), nega_count.reshape(-1,1)) )
```

- **get_sentiment_score(word):**

Hàm này sử dụng SentiWordNet để lấy điểm cảm xúc cho một từ. Nếu từ không tồn tại trong SentiWordNet, hàm trả về 0.

- **get_sentiment_feature(sentence):**

Hàm này tính trung bình điểm cảm xúc cho tất cả các từ trong một câu, sử dụng hàm **get_sentiment_score**.

- **combine_features(tfidf_features, sentiwordnet_features, nega_count):**

Hàm này kết hợp các đặc trưng TFIDF, SentiWordNet và Negation_count thành một ma trận đặc trưng tổng cộng.

  - **tfidf_features** là ma trận đặc trưng TFIDF.
  - **sentiwordnet_features** là mảng chứa điểm cảm xúc trung bình cho mỗi câu.
  - **nega_count** là mảng chứa số lượng từ phủ định trong mỗi câu.

Trích xuất đặc trưng SentiWordnet cho tập huấn luyện và tập kiểm thử

In [21]:
```
# Extract SentiWordNet features for both training and testing sets
X_train_senti = np.array([get_sentiment_feature(sentence) for sentence in X_train])
X_test_senti = np.array([get_sentiment_feature(sentence) for sentence in X_test])
```

Kết hợp các đặc trưng

In [22]:
```
# combine features
X_train_combined = combine_features(tfidf_train, X_train_senti, X_train_nega_count)
X_test_combined = combine_features(tfidf_test, X_test_senti, X_test_nega_count)
```

SVM Model

In [23]:
```
from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score
from sklearn.svm import SVC

# Training SVM
svm_model = SVC(kernel='linear', C=1)
svm_model.fit(X_train_combined, y_train)

# Predict
y_pred = svm_model.predict(X_test_combined)

# Model evaluation
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Result display
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')

# Report
svm_tfidf_report = classification_report(y_test, y_pred, target_names=['negative', 'positive', 'neutral'])
print(svm_tfidf_report)
```

Sử dụng mô hình Support Vector Machine (SVM) với đặc trưng đã kết hợp (X_train_combined) và đánh giá mô hình trên tập kiểm thử (X_test_combined)

Kernel tuyến tính “Linear” thường được chọn khi dữ liệu có thể tách biệt tốt bằng một ranh giới tuyến tính. Điều này đặc biệt đúng khi số lượng chiều không lớn so với kích thước dữ liệu.

Tham số mặc định:

- C = 1

C là tham số rất quan trọng trong việc huấn luyện SVM.

• C là tham số phạt trong bài toán soft-margin giúp đưa các điểm dữ liệu nằm trong khoảng hai siêu phẳng lề được phân loại đúng vào class của chúng hay giúp control misclassification. Khi C càng lớn model được huấn luyện sẽ càng fit với dữ liệu trong tập train. Điều này cũng đồng nghĩa với việc model có thể bị Overfitting.

Vậy chúng ta có thể rút ra được một quy luật:

• C tăng, bias giảm, variance tăng

• C giảm, bias tăng, variance giảm

Đánh giá mô hình với các độ đo accuracy, precision, recall, và F1-score dựa vào các hàm từ thư viện scikit-learn. Các tham số **average='weighted'** được sử dụng để tính toán trung bình có trọng số dựa trên hỗn hợp các lớp.

PHẦN III: KẾT QUẢ ĐẠT ĐƯỢC

Cuối cùng, sử dụng hàm classification_report để hiển thị báo cáo kết quả chi tiết với các độ đo cho từng lớp, dưới đây là kết quả

```
Accuracy: 0.87
Precision: 0.87
Recall: 0.87
F1 Score: 0.87
              precision    recall  f1-score   support

    negative       0.97      0.95      0.96       369
    positive       0.84      0.81      0.82       400
     neutral       0.81      0.85      0.83       400

    accuracy                           0.87      1169
   macro avg       0.87      0.87      0.87      1169
weighted avg       0.87      0.87      0.87      1169
```

Kiểm thử mô hình

In [24]:
```python
# Select a random subset of examples from the test set
sample_indices = np.random.choice(len(X_test), size=5, replace=False)
sample_X = X_test.iloc[sample_indices]
sample_y_true = y_test.iloc[sample_indices]

# Preprocess the text in the selected examples
sample_X_cleaned = sample_X.apply(cleanText)

# Extract features for the selected examples
sample_tfidf_features = tfidf_vectorizer.transform(sample_X_cleaned)
sample_senti_features = np.array([get_sentiment_feature(sentence) for sentence in sample_X])
sample_nega_count = np.array([count_negation_words(sentence) for sentence in sample_X])
sample_combined_features = combine_features(sample_tfidf_features, sample_senti_features, sample_nega_count)

# Make predictions using the trained SVM model
sample_y_pred = svm_model.predict(sample_combined_features)

# Display the examples along with true and predicted labels
for i in range(len(sample_X)):
    print(f"Example {i + 1}:\n")
    print(f"Text: {sample_X.iloc[i]}\n")
    print(f"True Label: {sample_y_true.iloc[i]}\n")
    print(f"Predicted Label: {sample_y_pred[i]}\n")
    print("="*50)
```

Lấy ngẫu nhiên ví dụ 5 mẫu từ tập dữ liệu, đưa các dữ liệu ví dụ đó vào các đặc trưng rồi kết hợp các đặc trưng lại đưa vào mô hình dự đoán.

Sau đây là in ra kết quả:

```
Example 1:

Text: the company pledged that the new software would render e mail and other document much a they appear on deskto
p computer

True Label: neutral

Predicted Label: neutral

==================================================
Example 2:

Text: the employee negotiation are to address measure needed to adjust the operation to the present production situ
ation

True Label: neutral

Predicted Label: neutral

==================================================
Example 3:

Text: finnish medium group talentum ha issued a profit warning

True Label: negative

Predicted Label: negative

==================================================
Example 4:

Text: and the broker repeated it   buy   rating   based on expectation that current restructuring will lead to a cl
ear improvement in performance in europe in

True Label: positive

Predicted Label: positive

==================================================
Example 5:

Text: konecranes ha previously communicated an estimated reduction of about       employee on group level in

True Label: negative

Predicted Label: negative

==================================================
```

Tiếp theo, kiểm thử mô hình qua các dữ liệu bên ngoài cho thấy mô hình đánh giá đúng với nhãn cảm xúc gốc

In [32]:
```python
# Example sentence
new_sentence = "Foundries division reports its sales increased by 9.7 % to EUR 63.1 mn from EUR 57.5 mn in the corre

# Text preprocessing for the example sentence
cleaned_new_sentence = cleanText(new_sentence)

# Convert the new sentence to TF-IDF vector and extract SentiWordNet, nega_count features
tfidf_new = tfidf_vectorizer.transform([cleaned_new_sentence])
senti_new = get_sentiment_feature(cleaned_new_sentence)
nega_count_new = count_negation_words(cleaned_new_sentence)

# Combine features
combined_features_new = combine_features(tfidf_new, senti_new, np.array([nega_count_new]))

# Predict with the trained SVM model
prediction = svm_model.predict(combined_features_new)

# Print the result
print(f'New sentence: "{new_sentence}"')
print(f'Predicted sentiment: {prediction[0]}')
```

```
New sentence: "Foundries division reports its sales increased by 9.7 % to EUR 63.1 mn from EUR 57.5 mn in the corre
sponding period in 2006 , and sales of the Machine Shop division increased by 16.4 % to EUR 41.2 mn from EUR 35.4 m
n in the corresponding period in 2006 ."
Predicted sentiment: positive
```

TÀI LIỆU THAM KHẢO

[1] GeeksforGeeks (2020) Support Vector Machine (SVM) Algorithm [Online] Available at: https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[2] Machine Learning cơ bản (n.d). [Online] Available at: https://machinelearningcoban.com/

[3] Medium (n.d). Financial News Sentiment Analysis With Support Vector Machine [Online] Available at: https://medium.com/@pipietsetiowati/financial-news-sentiment-analysis-with-support-vector-machine-b8544d910b12

[4] Kaggle (2022) Financia Sentiment Analysis [Online] Available at: https://www.kaggle.com/datasets/sbhatti/financial-sentiment-analysis/data

[5] Scikit learn (n.d). [Online] Available at: https://scikit-learn.org/stable/