TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**
**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN GIỮA KÌ MÔN**

**NHẬP MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

# MIDTERM PROJECT

*Người hướng dẫn*: **TS LÊ ANH CƯỜNG**

*Người thực hiện*: **LÊ GIA BẢO – 52000629**

**NGUYỄN THÀNH CHẤT – 52000636**

**PHAN QUỐC TOÀN - 52000814**

Lớp : **N05**

Khoá : **24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**
**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN GIỮA KÌ MÔN**

**NHẬP MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

# MIDTERM PROJECT

Người hướng dẫn: **TS LÊ ANH CƯỜNG**

Người thực hiện: **LÊ GIA BẢO – 52000629**

**NGUYỄN THÀNH CHẤT – 52000636**

**PHAN QUỐC TOÀN - 52000814**

Lớp **: N05**

Khoá **: 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

# LỜI CẢM ƠN

- Nhóm chúng em xin gửi lời cảm ơn sâu sắc và lòng biết ơn tới Thầy về sự hỗ trợ và sự chỉ dẫn tận tình mà Thầy đã dành cho nhóm chúng trong quá trình thực hiện đồ án lần này. Sự tận tâm và kiến thức sâu rộng của Thầy đã giúp nhóm chúng em vượt qua những thách thức và hoàn thành đồ án một cách thành công.
- Nhóm chúng em rất may mắn và tự hào được học hỏi từ Thầy, và chúng em cam kết sẽ tiếp tục áp dụng những bài học và kiến thức mà Thầy đã truyền đạt cho chúng em vào tương lai.
- Một lần nữa, xin chân thành cảm ơn Thầy về tất cả những điều tốt lành mà Thầy đã mang đến trong việc học tập của nhóm chúng em.

# ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của chúng tôi và được sự hướng dẫn của TS.Lê Anh Cường;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 5 tháng 4 năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Bao*

*Lê Gia Bảo*

*Chat*

*Nguyễn Thành Chất*

*Toan*

*Phan Quốc Toàn*

# PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày  tháng  năm
(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày  tháng  năm
(kí và ghi họ tên)

# TÓM TẮT

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được, những phát hiện cơ bản trong vòng 1 -2 trang.

# MỤC LỤC

# CHƯƠNG 1 – WORD2VEC,FASTTEXT VÀ ỨNG DỤNG CBOW & SKIP-GRAM

- Word2Vec là một mô hình nổi tiếng trong lĩnh vực xử lý ngôn ngữ tự nhiên, giúp tạo ra các biểu diễn embedding của từ trong một không gian có số chiều thấp hơn nhiều lần so với số từ trong từ điển. Ý tưởng của Word2Vec đã được áp dụng trong nhiều bài toán khác nhau, không chỉ liên quan đến ngôn ngữ.

- Là một mạng neural 2 lớp với duy nhất 1 tầng ẩn, lấy đầu vào là một corpus lớn và sinh ra không gian vector(với số chiều khoảng vài trăm), với mỗi từ duy nhất trong corpus được gắn với một vector tương ứng trong không gian.

- Các word vectors được xác định trong không gian vector sao cho những từ có chung ngữ cảnh trong corpus được đặt gần nhau trong không gian. Dự đoán chính xác cao về ý nghĩa của một từ dựa trên những lần xuất hiện trước đây.

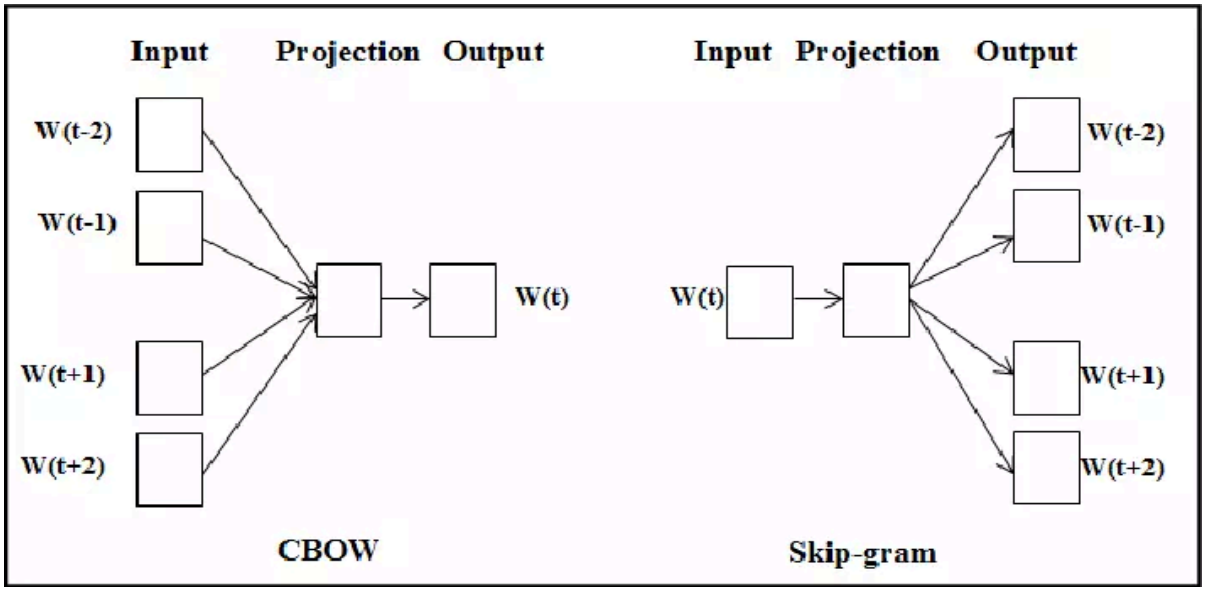
| | | Vua | Hoàng hậu | Phụ nữ | Công chúa |
|---|---|---|---|---|---|
| Hoàng gia | | 0.99 | 0.99 | 0.02 | 0.98 |
| Nam tính | | 0.99 | 0.05 | 0.01 | 0.02 |
| Nữ tính | | 0.05 | 0.93 | 0.999 | 0.94 |
| Tuổi | | 0.7 | 0.6 | 0.5 | 0.1 |

Hình 1: Vector được biểu diễn theo word2vec

- Phương pháp Word2Vec là một trong những phương pháp quan trọng trong xử lý ngôn ngữ tự nhiên và biểu diễn từ vựng dưới dạng vector.

Có 2 cách xây dựng word2vec:

- Sử dụng ngữ cảnh để dự đoán mục tiêu(CBOW).
- Sử dụng một từ để dự đoán ngữ cảnh mục tiêu(skip-gram)(cho kết quả tốt hơn với dữ liệu lớn)




Hình 2: Mô hình CBOW & Skip-gram

## I.Phương pháp CBOW:

-CBOW là viết tắt của "Continuous Bag of Words," một phương pháp trong lĩnh vực xử lý ngôn ngữ tự nhiên và học sâu. CBOW là một mô hình trong lớp mô hình Word Embedding (nhúng từ), mà mục tiêu chính là học cách biểu diễn từ vựng dưới dạng các vector có chiều thấp trong không gian số học, sao cho các vector này bắt chước được mối quan hệ ngữ nghĩa giữa các từ.

-Ý tưởng chính của CBOW là dự đoán từ mục tiêu dựa trên ngữ cảnh xung quanh nó. Nó hoạt động bằng cách sử dụng một cửa sổ trượt qua văn bản, và từ thông qua các từ còn lại trong cửa sổ này để dự đoán từ mục tiêu. Cụ thể, CBOW cố gắng học cách dự đoán từ mục tiêu (hoặc "từ trung tâm") dựa trên các từ xung quanh (hoặc "từ ngữ cảnh") trong một văn bản.

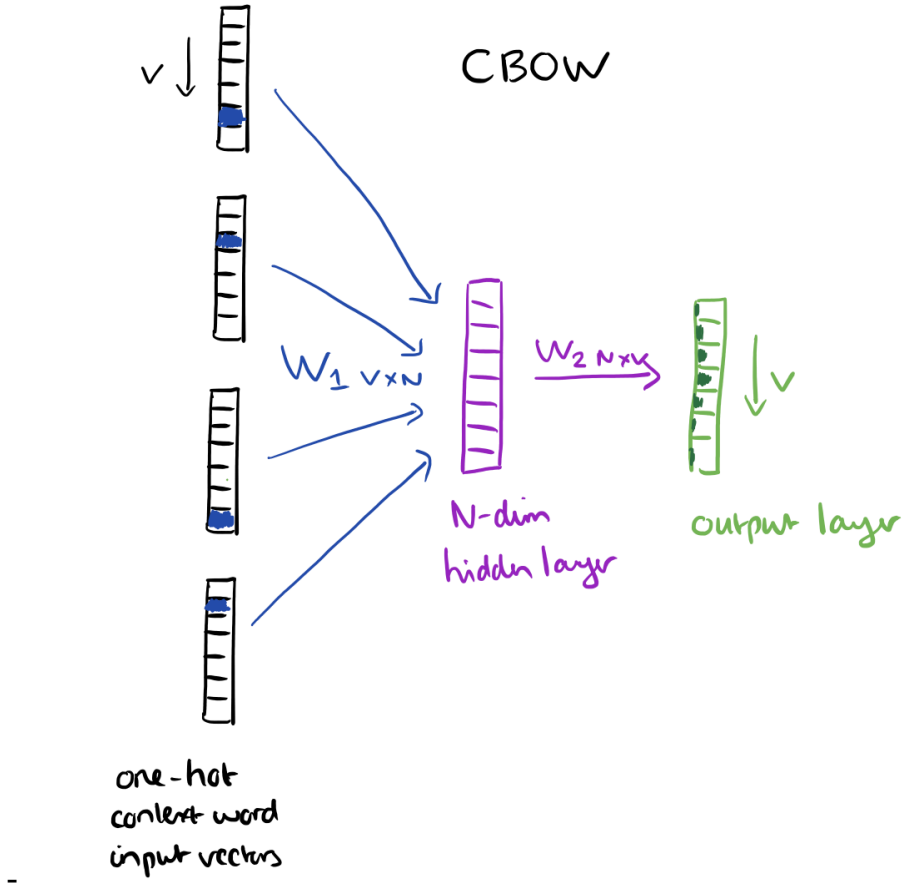
-CBOW thường được sử dụng trong các ứng dụng như tạo ra các vector nhúng từ cho các từ trong từ điển, các bước tiền xử lý cho các mô hình học máy văn bản, hoặc trong các nhiệm vụ như phân loại văn bản, dịch máy, và tìm kiếm thông tin.

## 1.Ý nghĩa CBOW:

- Cung cấp một cách để biểu diễn từ vựng dưới dạng các vector có chiều thấp trong không gian số học, giúp mô hình hiểu và xử lý ngôn ngữ tự nhiên hiệu quả hơn.

### 2.Mô hình CBOW:

- Mô hình CBOW sẽ cố gắng dự đoán từ trung tâm (center word hoặc target word) dựa trên ngữ cảnh được tạo ra từ các từ xung quanh nó (surrounding words). Chúng ta hay xem xét một câu đơn giản "the quick brown fox jumps over the lazy dog", chúng ta có thể có các cặp (context_window, target_word) nếu chọn context_window = 2 ta sẽ có ([quick, fox], brown), ([the, brown], quick), ([the,dog], lazy). Như vậy ta có thể dự đoán target_word dựa trên context_window từ.




Hình 3: Mô hình CBOW

## 3.Huấn luyện mô hình CBOW:

### 3.1.Chuẩn bị dữ liệu:

Để bắt đầu, trước tiên chúng ta sẽ xây dựng kho từ vựng trong đó trích xuất từng từ duy nhất khỏi từ vựng của mình và ánh xạ một mã định danh số duy nhất cho nó.

```
from keras.preprocessing import text
from keras.utils import np_utils
from keras.preprocessing import sequence

tokenizer = text.Tokenizer()
```

```
tokenizer.fit_on_texts(norm_bible)
word2id = tokenizer.word_index

# build vocabulary of unique words
word2id['PAD'] = 0
id2word = {v:k for k, v in word2id.items()}
wids = [[word2id[w] for w in text.text_to_word_sequence(doc)] for doc in
norm_bible]

vocab_size = len(word2id)
embed_size = 100
window_size = 2 # context window size

print('Vocabulary Size:', vocab_size)
print('Vocabulary Sample:', list(word2id.items())[:10])
```

**Output**

**------**

**Vocabulary Size: 12425**

**Vocabulary Sample: [('perceived', 1460), ('flagon', 7287), ('gardener', 11641), ('named', 973), ('remain', 732), ('sticketh', 10622), ('abstinence', 11848), ('rufus', 8190), ('adversary', 2018), ('jehoiachin', 3189)]**

### 3.2.Xây dựng từ điển:

Chúng ta cần các cặp bao gồm từ trung tâm mục tiêu và các từ ngữ cảnh bao quanh. Trong quá trình triển khai, *từ target word* có độ dài 1 và surrounding context có độ dài 2 x window_size trong đó chúng em lấy window_size các từ trước và sau từ mục tiêu trong kho văn bản. Điều này sẽ trở nên rõ ràng hơn với ví dụ sau.

```
def generate_context_word_pairs(corpus, window_size, vocab_size):
    context_length = window_size*2
    for words in corpus:
        sentence_length = len(words)
```

```
    for index, word in enumerate(words):
        context_words = []
        label_word   = []
        start = index - window_size
        end = index + window_size + 1
            context_words.append([words[i]
                    for i in range(start, end)
                    if 0 <= i < sentence_length
                    and i != index])
        label_word.append(word)

        x = sequence.pad_sequences(context_words, maxlen=context_length)
        y = np_utils.to_categorical(label_word, vocab_size)
        yield (x, y)
```

**Context (X): ['old','testament','james','bible'] -> Target (Y): king**
**Context (X): ['first','book','called','genesis'] -> Target(Y): moses**
**Context(X):['beginning','god','heaven','earth'] -> Target(Y):created**
**Context (X):['earth','without','void','darkness'] -> Target(Y): form**
**Context (X): ['without','form','darkness','upon'] -> Target(Y): void**
**Context (X): ['form', 'void', 'upon', 'face'] -> Target(Y): darkness**
**Context (X): ['void', 'darkness', 'face', 'deep'] -> Target(Y): upon**
**Context (X): ['spirit', 'god', 'upon', 'face'] -> Target (Y): moved**
**Context (X): ['god', 'moved', 'face', 'waters'] -> Target (Y): upon**
**Context (X): ['god', 'said', 'light', 'light'] -> Target (Y): let**
**Context (X): ['god', 'saw', 'good', 'god'] -> Target (Y): light**

### 3.3.Chuẩn bị dữ liệu đầu vào và đầu ra:

Kết quả trước đó sẽ giúp mở rộng cái nhìn về cách **X** tạo ra ngữ cảnh và chúng ta cố gắng dự đoán từ trung tâm mục tiêu **Y** dựa trên ngữ cảnh đó. Ví dụ, nếu văn bản gốc là "ban đầu thần tạo ra trời và đất", sau khi loại bỏ các mật khẩu, chúng ta có "ban

đầu thần tạo ra trời đất". Chúng ta đang cố gắng hiểu rằng từ trung tâm mục tiêu trong trường hợp này là "**tạo ra**", dựa trên ngữ cảnh của "**sự khởi đầu**", "**thần**", "**trời**", "**đất**".

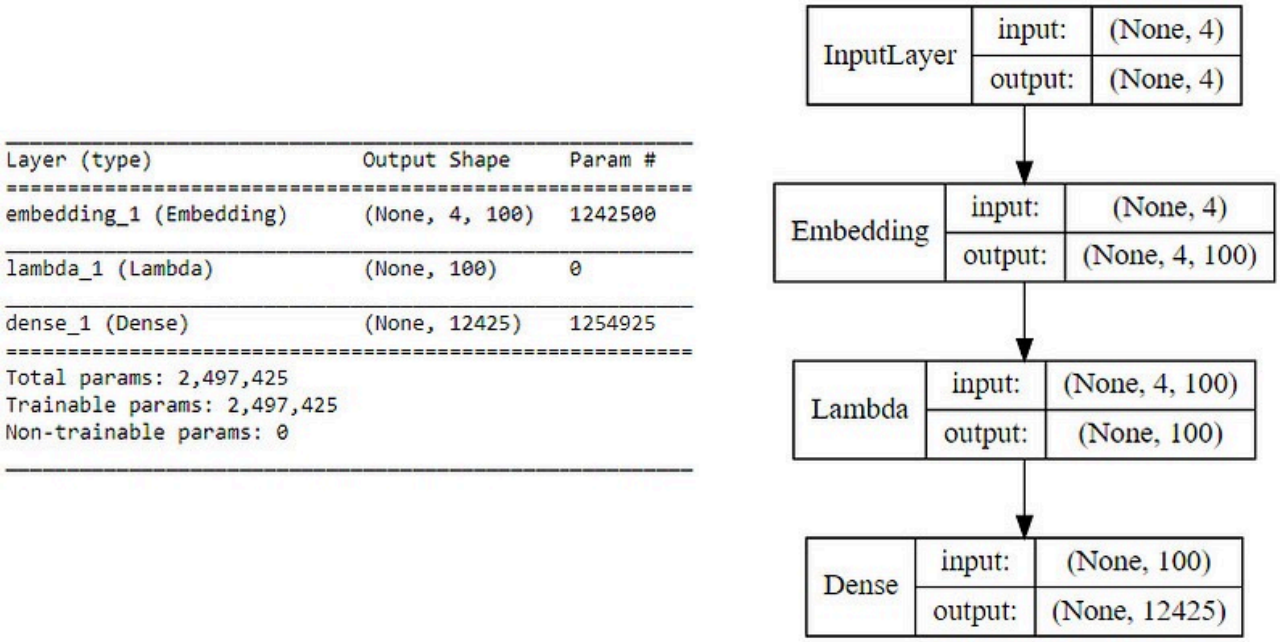
### 3.4.Xây dựng mô hình CBOW:

Chúng em sử dụng thư viện **keras** của **tensorflow** để xây dựng kiến trúc **deep learning** cho mô hình **CBOW**. Đầu vào là các từ ngữ cảnh được chuyển đến một lớp nhúng, mà lớp này được khởi tạo với trọng số ngẫu nhiên. Các từ nhúng sau đó được truyền vào một lớp **lambda** để tính trung bình (do đó được gọi là **CBOW** vì không xem xét thứ tự hoặc trình tự trong các từ ngữ cảnh khi tính trung bình). Tiếp theo, chúng em đưa ngữ cảnh trung bình này vào một lớp **softmax** dày đặc để dự đoán từ mục tiêu. Tiếp theo, so sánh dự đoán này với từ mục tiêu thực tế bằng cách tính toán tổn thất sử dụng **categorical_crossentropy** và thực hiện lan truyền ngược để cập nhật lớp nhúng trong quá trình. Mã dưới đây sẽ mô tả kiến trúc mô hình:

```
import keras.backend as K
from keras.models import Sequential
from keras.layers import Dense, Embedding, Lambda

# build CBOW architecture
cbow = Sequential()
cbow.add(Embedding(input_dim=vocab_size, output_dim=embed_size,
input_length=window_size*2))
cbow.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(embed_size,)))
cbow.add(Dense(vocab_size, activation='softmax'))
cbow.compile(loss='categorical_crossentropy', optimizer='rmsprop')

# view model summary
print(cbow.summary())
```



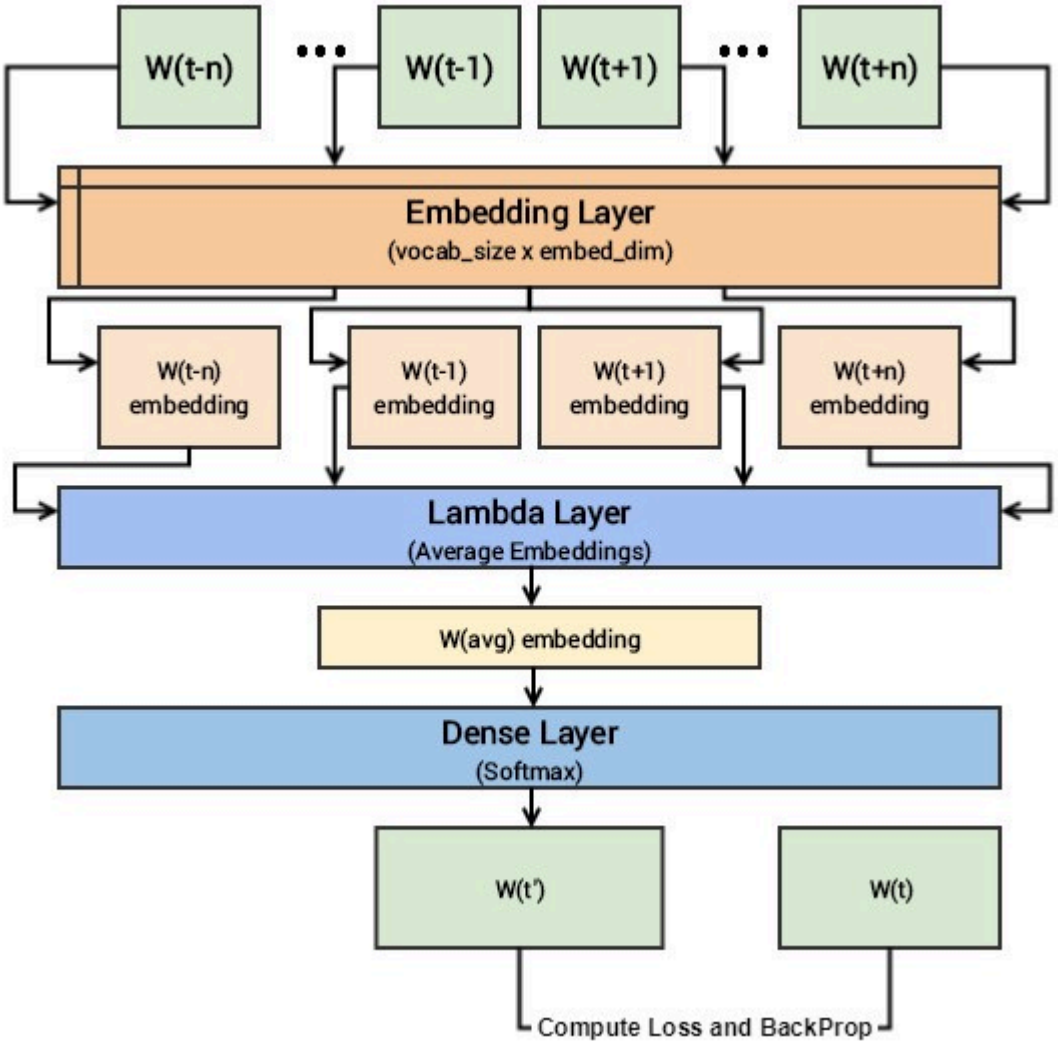
| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_1 (Embedding) | (None, 4, 100) | 1242500 |
| lambda_1 (Lambda) | (None, 100) | 0 |
| dense_1 (Dense) | (None, 12425) | 1254925 |

Total params: 2,497,425
Trainable params: 2,497,425
Non-trainable params: 0



Hình 4: Tóm tắt và kiến trúc mô hình CBOW

Các từ ngữ cảnh đầu vào có kích thước (**2 x window_size**), chuyển chúng vào một lớp nhúng với kích thước (**vocab_size x embed_size**) để tạo ra các từ nhúng dày đặc cho mỗi từ trong ngữ cảnh (**1 x embed_size** cho mỗi từ). Sau đó, sử dụng một lớp lambda để tính trung bình các từ nhúng này, tạo ra một phần nhúng dày đặc trung bình (**1 x embed_size**) được đưa vào một lớp softmax dày đặc để dự đoán từ mục tiêu có khả năng nhất. Chúng em so sánh dự đoán này với từ mục tiêu thực tế, tính toán tổn thất, và sử dụng thuật toán truyền ngược để điều chỉnh trọng số (trong lớp nhúng). Quy trình này được lặp lại cho tất cả các cặp (ngữ cảnh, mục tiêu) trong nhiều kỷ nguyên. Hình dưới đây em sẽ giải thích quy trình này một cách trực quan.




Hình 5: Mô tả trực quan mô hình học sâu CBOW

**3.5.Huấn luyện mô hình:**

- Sử dụng các cặp từ mục tiêu và ngữ cảnh để huấn luyện mô hình.

- Đối với mỗi cặp từ, đưa ngữ cảnh vào mô hình và huấn luyện mô hình để dự đoán từ mục tiêu.

- Tối thiểu hóa hàm mất mát bằng các thuật toán tối ưu như Gradient Descent hoặc Adam.

```
for epoch in range(1, 6):
    loss = 0.
    i = 0
    for x, y in generate_context_word_pairs(corpus=wids, window_size=window_size,
vocab_size=vocab_size):
        i += 1
        loss += cbow.train_on_batch(x, y)
        if i % 100000 == 0:
            print('Processed {} (context, word) pairs'.format(i))

    print('Epoch:', epoch, '\tLoss:', loss)
    print()
```

**Output:**

**Epoch: 1 Loss: 4257900.60084**
**Epoch: 2 Loss: 4256209.59646**
**Epoch: 3 Loss: 4247990.90456**
**Epoch: 4 Loss: 4225663.18927**
**Epoch: 5 Loss: 4104501.48929**

### 3.6.Đánh giá và điều chỉnh:

- Đánh giá hiệu suất của mô hình trên các tập dữ liệu kiểm tra.

- Điều chỉnh các siêu tham số và kiến trúc mô hình để cải thiện hiệu suất nếu cần.

### 3.7.Sử dụng mô hình:

Để có được các từ nhúng cho toàn bộ từ vựng, chúng ta có thể trích xuất từ đó từ lớp nhúng của mình bằng cách tận dụng đoạn mã sau. Chúng em không thực hiện việc nhúng ở vị trí 0 vì nó thuộc về **(PAD)** thuật ngữ đệm.

```
weights = cbow.get_weights()[0]
weights = weights[1:]
print(weights.shape)

pd.DataFrame(weights, index=list(id2word.values())[1:]).head()
```

**(12424, 12424)**

**{'egypt': ['destroy', 'none', 'whole', 'jacob', 'sea'],**
**'famine': ['wickedness', 'sore', 'countries', 'cease', 'portion'],**
**'god': ['therefore', 'heard', 'may', 'behold', 'heaven'],**
**'gospel': ['church', 'fowls', 'churches', 'preached', 'doctrine'],**
**'jesus': ['law', 'heard', 'world', 'many', 'dead'],**
**'john': ['dream', 'bones', 'held', 'present', 'alive'],**
**'moses': ['pharaoh', 'gate', 'jews', 'departed', 'lifted'],**
**'noah': ['abram', 'plagues', 'hananiah', 'korah', 'sarah']}**

Có thể thấy rõ rằng một số trong này có ý nghĩa theo ngữ cảnh ***(god, heaven)***, ***(gospel, church)*** , v.v. và một số có thể không. Việc đào tạo nhiều thường mang lại kết quả tốt hơn. Bây giờ chúng ta sẽ khám phá kiến trúc Skip-gram thường cho kết quả tốt hơn so với CBOW.

## II.Phương pháp Skip-gram

Skip-gram thì ngược lại với CBOW, dùng target word để dự đoán các từ xung quanh. Skip-gram huấn luyện chậm hơn. Thường làm việc khá tốt với các tập data nhỏ, đặc biệt do đặc trưng của mô hình nên khả năng vector hóa cho các từ ít xuất hiện tốt hơn CBOW.

### 1.Ý nghĩa của skip-gram phân tích như sau:

#### 1.1.Dự đoán ngữ cảnh từ mục tiêu:

- Trong phương pháp Skip-gram, mục tiêu là dự đoán ngữ cảnh (context) dựa trên từ mục tiêu (target).

- Ngược lại với CBOW, Skip-gram cố gắng học cách biểu diễn một từ bằng cách dự đoán các từ xung quanh nó.

### 1.2.Biểu diễn từ vựng dưới dạng ngữ cảnh:

- Skip-gram cố gắng học các biểu diễn cho các từ bằng cách xem xét ngữ cảnh trong đó chúng xuất hiện.

- Mỗi từ được biểu diễn bằng một vector nhúng (embedding vector) sao cho khi đưa vào mô hình, từ này có khả năng dự đoán được ngữ cảnh xung quanh.

### 1.3.Mô hình dùng cho từ điển lớn:

- Skip-gram thường hiệu quả hơn CBOW đối với các từ điển lớn với các từ hiếm và có ý nghĩa đặc biệt.

- Bởi vì Skip-gram tập trung vào việc học biểu diễn cho từng từ cụ thể, nó thích hợp hơn cho việc xử lý các từ hiếm hoặc không phổ biến.

### 1.4.Ứng dụng trong các bài toán tương tự:

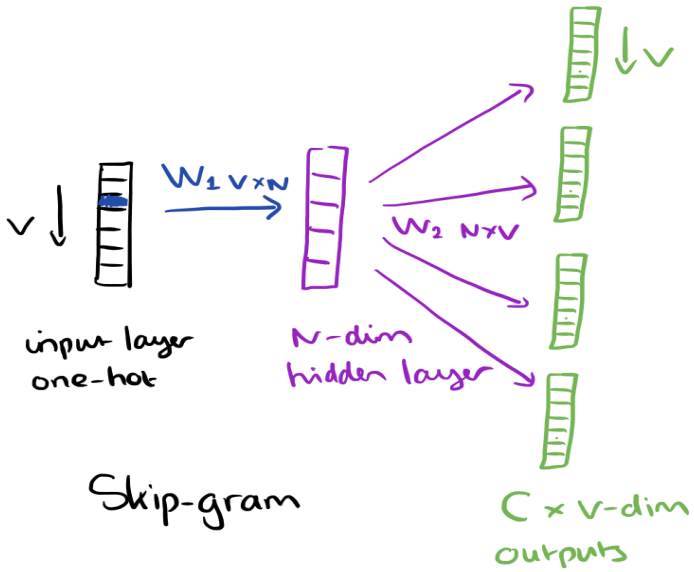
- Skip-gram cũng được sử dụng trong các bài toán tương tự như CBOW, bao gồm phân loại văn bản, tìm kiếm thông tin, và dịch máy.

- Biểu diễn từ vựng dưới dạng các vector nhúng từ Skip-gram có thể được sử dụng để đo lường sự tương đồng ngữ nghĩa giữa các từ và tìm ra các mối quan hệ ngữ nghĩa.

- Tóm lại, phương pháp Skip-gram là một cách hiệu quả để học biểu diễn từ vựng dưới dạng các vector nhúng, đặc biệt là đối với các từ trong các văn bản lớn và có từ điển phức tạp. Phương pháp này cung cấp một cách tiếp cận linh hoạt và mạnh mẽ để nắm bắt ý nghĩa của từ thông qua ngữ cảnh mà chúng xuất hiện.

## 2.Mô hình Skip-gram

- Mô hình Skip-gram là một trong những mô hình quan trọng trong Word Embedding, được sử dụng để học biểu diễn từ vựng dưới dạng các vector nhúng trong không gian nhiều chiều. Dưới đây là cấu trúc cơ bản của mô hình Skip-gram:

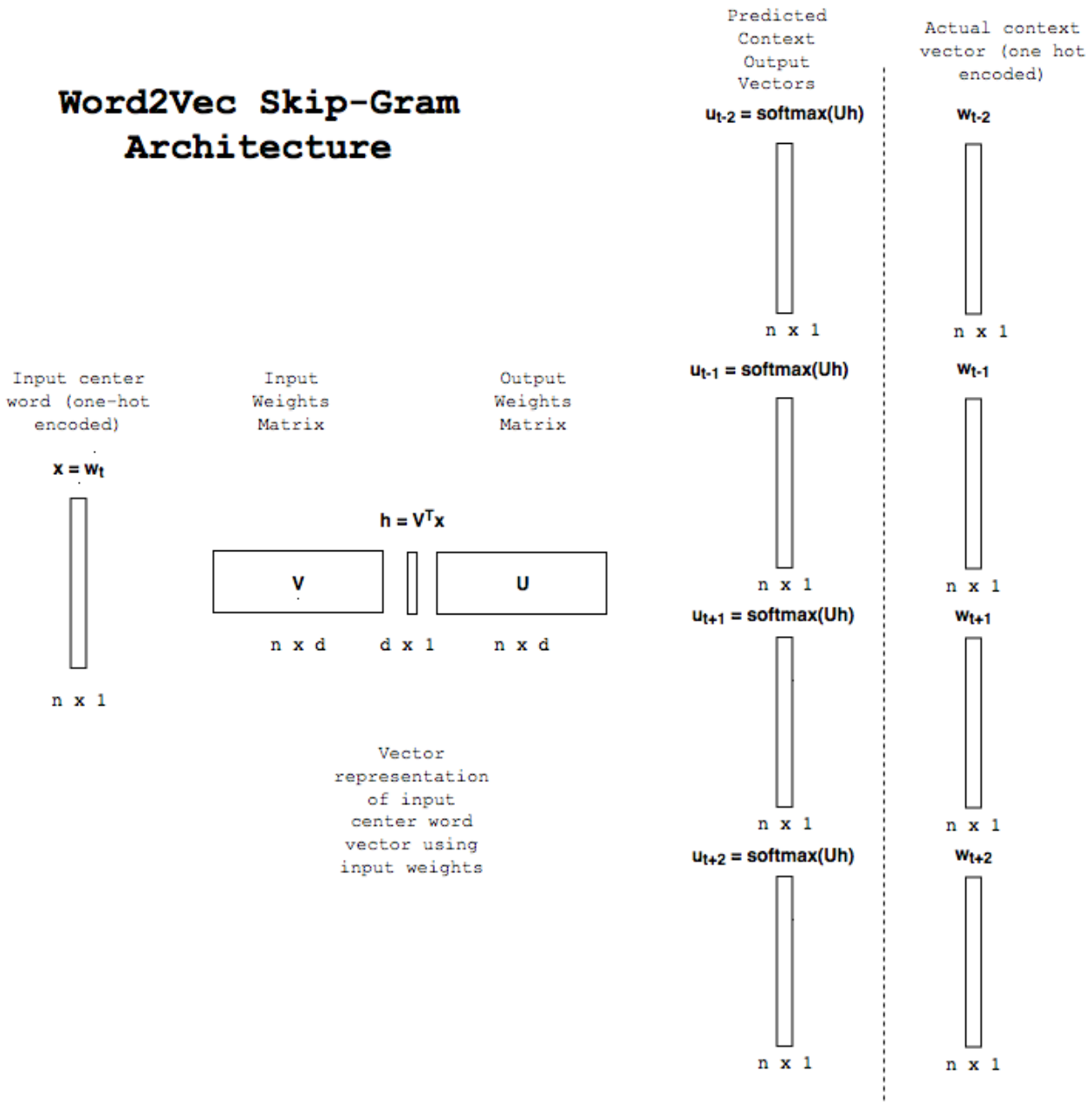



Hình 6: Mô hình Skip-gram

**2.1.Đầu vào và Biểu diễn từ:**

- Mỗi từ trong tập dữ liệu được biểu diễn dưới dạng một vector one-hot encoding hoặc bằng các kỹ thuật nhúng từ trước.

- Vector one-hot của một từ có chiều dài bằng số lượng từ trong từ điển, và chỉ có một phần tử bằng 1, còn lại là 0. Phần tử bằng 1 này ứng với chỉ số của từ đó trong từ điển.




Hình 7 : Biểu diễn input vector one-hot encoding

### 2.2.Lớp nhúng (Embedding Layer):

- Một lớp nhúng (embedding layer) chuyển đổi vector one-hot của mỗi từ thành một vector nhúng (embedding vector) có chiều thấp hơn, thường là hàng trăm hoặc hàng nghìn chiều.

- Vector nhúng này biểu diễn ngữ nghĩa của từ trong không gian số học.

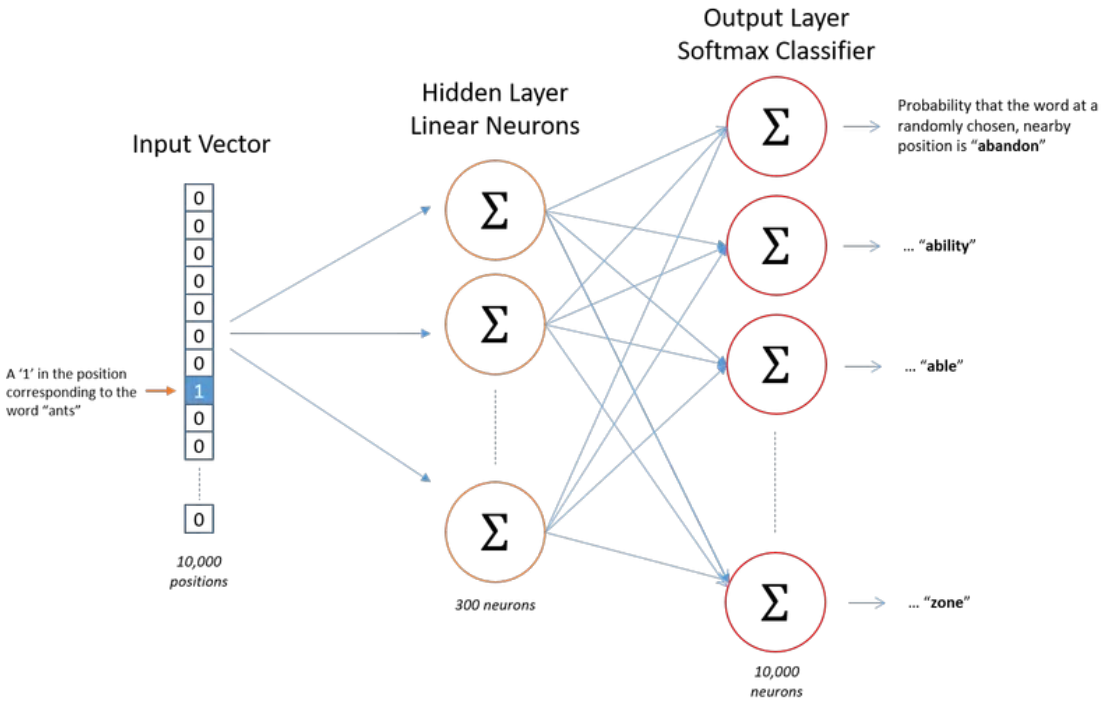
### 2.3.Mô hình Skip-gram:

- Mô hình Skip-gram cố gắng dự đoán các từ xung quanh từ mục tiêu trong một cửa sổ trượt.

- Mỗi từ mục tiêu được biểu diễn bằng một vector nhúng.

- Các vector nhúng của từ mục tiêu được đưa vào một lớp ẩn (hidden layer) với kích thước nhỏ để tính toán mối quan hệ giữa từ mục tiêu và từ xung quanh nó.

- Đầu ra của mô hình là một tập hợp các vector xác suất, mỗi vector đại diện cho xác suất của một từ xung quanh được dự đoán.




Hình 8 : Mô hình kiến trúc Skip-gram và training data

**2.4.Huấn luyện:**

- Mô hình Skip-gram được huấn luyện thông qua các cặp từ mục tiêu và từ xung quanh.

- Mục tiêu của mô hình là tối thiểu hóa sai số giữa xác suất dự đoán và xác suất thực tế của các từ xung quanh từ mục tiêu.

**2.5.Đầu ra:**

- Khi huấn luyện hoàn thành, các vector nhúng từ mô hình Skip-gram có thể được sử dụng để đo lường sự tương đồng ngữ nghĩa giữa các từ, tìm kiếm thông tin hoặc sử dụng trong các ứng dụng khác trong xử lý ngôn ngữ tự nhiên.

## 3. Huấn luyện mô hình Skip-gram

### 3.1 Chuẩn bị dữ liệu huấn luyện:

- Tạo một tập dữ liệu văn bản lớn.
- Chia văn bản thành các từ riêng lẻ (tokenization).
- Tạo các cặp từ (từ đích và từ xung quanh) từ dữ liệu văn bản.

```
# Dữ liệu giả định
corpus = [
    "I like playing football with my friends",
    "She enjoys reading books in her free time",
    "We love listening to music while driving"]
     vocab_size = len(word_to_idx)
```

### 3.2 Xây dựng mô hình Skip-gram:

- Khởi tạo các vector biểu diễn (embedding) cho từng từ, có thể là ngẫu nhiên hoặc sử dụng các vector đã được huấn luyện trước.
- Xây dựng kiến trúc mạng thần kinh với lớp đầu vào (context words), lớp ẩn và lớp đầu ra (từ đích).
- Lớp ẩn biểu diễn các vector từ.

```
# Hàm tạo dữ liệu đào tạo cho Skip-gram
def create_training_data(corpus, window_size):
    data = []
    for sentence in corpus:
        words = sentence.split()
        for i, target_word in enumerate(words):
            for j in range(i - window_size, i + window_size + 1):
                if j != i and j >= 0 and j < len(words):
                    context_word = words[j]
                    data.append((target_word, context_word))
    return data
```

```
# Mô hình Skip-gram
class SkipGram(nn.Module):
    def __init__(self, vocab_size, embedding_dim):
        super(SkipGram, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.linear = nn.Linear(embedding_dim, vocab_size)

    def forward(self, target_word):
        embedded = self.embeddings(target_word)
        out = self.linear(embedded)
        return out

# Tham số
embedding_dim = 10
window_size = 2
learning_rate = 0.001
epochs = 100

# Dữ liệu đào tạo
training_data = create_training_data(corpus, window_size)

# Khởi tạo mô hình và bộ tối ưu
model = SkipGram(vocab_size, embedding_dim)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=learning_rate)
```

**3.3 Huấn luyện mô hình**:

- Sử dụng thuật toán tối ưu hóa như stochastic gradient descent (SGD) để tối ưu hóa mô hình.
- Đối với mỗi cặp từ (từ đích, từ xung quanh):
  + Tính toán từ xung quanh dự đoán dựa trên các vector từ hiện tại.
  + Tính toán hàm mất mát (sự khác biệt giữa từ xung quanh dự đoán và thực tế).

+ Cập nhật các vector từ để giảm thiểu hàm mất mát.

- Lặp lại quá trình huấn luyện qua nhiều epochs

```
# Huấn luyện mô hình
for epoch in range(epochs):
    total_loss = 0
    for target_word, context_word in training_data:
        target_idx = torch.tensor([word_to_idx[target_word]], dtype=torch.long)
        model.zero_grad()
        output = model(target_idx)
        loss = criterion(output, torch.tensor([word_to_idx[context_word]],
dtype=torch.long))
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f'Epoch {epoch+1}, Loss: {total_loss/len(training_data):.4f}')
```

.

**3.4 Vector từ**:

- Sau khi huấn luyện, các vector từ biểu diễn ý nghĩa ngữ nghĩa của từ.
- Các từ tương tự sẽ có các vector tương tự (dựa trên độ tương đồng cosine).

```
# Lấy vectơ biểu diễn từ
word_embeddings = model.embeddings.weight.detach().numpy()
for i in range(len(idx_to_word)):
    print(f'Word: {idx_to_word[i]}, Embedding: {word_embeddings[i]}')
```

## 4 .Tìm hiểu về FastText

- FastText là một thư viện mã nguồn mở, miễn phí, nhẹ cho phép người dùng tìm hiểu cách trình bày văn bản và phân loại văn bản. Nó được phát triển bởi phòng thí nghiệm AI Research (FAIR) của Facebook. Thư viện có thể được sử dụng để đạt được việc học cách biểu diễn và phân loại văn bản hiệu quả bằng cách đào tạo các mô hình trên kho văn bản lớn. fastText nổi bật về tốc độ và hiệu quả, giúp đào tạo các mô hình với hàng tỷ từ chỉ trong vài phút trên CPU đa lõi tiêu chuẩn.
- Một nhược điểm lớn của word2vec là nó chỉ sử dụng được những từ có trong dataset, để khắc phục được điều này chúng ta có FastText là mở rộng của Word2Vec, được xây dựng bởi facebook năm 2016. Thay vì training cho đơn vị word, nó chia text ra làm nhiều đoạn nhỏ được gọi là n-gram cho từ, ví dụ apple sẽ thành app, ppl, and ple, vector của từ apple sẽ bằng tổng của tất cả cái này. Do vậy, nó xử lý rất tốt cho những trường hợp từ hiếm gặp.
- Chức năng cốt lõi của fastText nằm ở việc sử dụng túi từ (bag of words) và túi các n-gram để biểu diễn văn bản, cho phép nó xem xét cả bản chất phi cấu trúc của văn bản và thứ tự cục bộ của các từ. Cách tiếp cận này giúp nắm bắt được nhiều thông tin ngữ cảnh và ngữ nghĩa hơn so với các mô hình túi từ đơn giản. fastText cũng sử dụng hàm softmax phân cấp dựa trên cây mã hóa Huffman giúp tăng tốc đáng kể thời gian đào tạo và dự đoán cho các tập dữ liệu lớn có nhiều nhãn đầu ra.
- Dưới đây là mã giả để train fasttext embedding:

```
from gensim.models import FastText
pathdata = './datatrain.txt'
def read_data(path):from gensim.models import FastText
pathdata = './datatrain.txt'

def read_data(path):
    traindata = []
    with open(path, 'r', encoding='utf-8') as file:  # Sử dụng with để tự động đóng file
        for line in file:
            words = line.strip().split()  # Loại bỏ ký tự xuống dòng và tách từ
            traindata.append(words)
    return traindata

if _name_ == '_main_':
    train_data = read_data(pathdata)
```

```
    model_fasttext = FastText(vector_size=150, window=10, min_count=2,
workers=4, sg=1)
    model_fasttext.build_vocab(corpus_iterable=train_data)
    model_fasttext.train(corpus_iterable=train_data, total_examples=len(train_data),
epochs=10)  # Giả sử train với 10 epochs

    model_fasttext.save("./fasttext_gensim.model")  # Lưu mô hình vào thư mục hiện
tại hoặc chỉ định đường dẫn đến thư mục bạn muốn lưu
```

Khởi tạo một mô hình FastText trong Gensim với các tham số cụ thể:

vector_size=150: Kích thước của vector từ, tức là số chiều của vector từ.
window=10: Kích thước cửa sổ ngữ cảnh tối đa là 10 từ xung quanh từ đích.
min_count=2: Loại bỏ tất cả các từ xuất hiện ít hơn 2 lần.
workers=4: Sử dụng 4 luồng để tăng tốc độ training.
sg=1: Sử dụng mô hình Skip-gram (ngược lại với sg=0 là CBOW - Continuous Bag of Words).

## III.Ứng dụng FastText

Để giải quyết vấn đề này, chúng ta cần thực hiện các bước sau:

1. Tải và sử dụng mô hình FastText tiếng Việt (cc.vi.300.vec) để nhúng các câu trong tập S và câu X vào không gian vector.
2. Tính toán độ tương tự giữa vector của câu X và các vector của câu trong tập S, sử dụng một metric như cosine similarity.
3. Tìm câu trong S có độ tương tự cao nhất với câu X.

Code:

```
import numpy as np
from gensim.models import KeyedVectors

# Bước 1: Tải mô hình FastText
model_path = '/content/drive/MyDrive/midtermdata/cc.vi.300.vec'  # Đường dẫn
tới file vector FastText
model = KeyedVectors.load_word2vec_format(model_path, limit=50000)  # Giới
hạn số lượng từ tải vào để giảm thời gian tải

# Hàm để tính vector trung bình cho một câu
def sentence_to_vector(sentence, model):
```

```
    words = sentence.split()
    word_vectors = [model[word] for word in words if word in model]
    if len(word_vectors) == 0:
        return np.zeros(model.vector_size)
    return np.mean(word_vectors, axis=0)

# Bước 2: Tính độ tương tự giữa các câu
def find_most_similar_sentence(input_sentence, sentence_set, model):
    input_vector = sentence_to_vector(input_sentence, model)
    max_similarity = -1
    most_similar_sentence = None
    for sentence in sentence_set:
        sentence_vector = sentence_to_vector(sentence, model)
        similarity = np.dot(input_vector, sentence_vector) /
(np.linalg.norm(input_vector) * np.linalg.norm(sentence_vector))
        if similarity > max_similarity:
            max_similarity = similarity
            most_similar_sentence = sentence
    return most_similar_sentence
```

Thử nghiệm 1 và kết quả:

```
# Giả sử có tập câu S và câu X như sau
S = ['Câu này rất hay', 'Hôm nay trời đẹp quá', 'Tôi yêu lập trình',
     'Ngày mai trời có mưa', 'Con chó có bộ lông vàng', 'Chị tôi làm nghề
kế toán']
X = 'Thời tiết nóng quá'

# Bước 3: Tìm câu tương tự nhất
most_similar_sentence = find_most_similar_sentence(X, S, model)
print("Câu tương tự nhất:", most_similar_sentence)
```

```
Câu tương tự nhất: Hôm nay trời đẹp quá
```

Thử nghiệm 2:

```
most_similar_sentence = find_most_similar_sentence("Tôi thích code", S, model)
print("Câu tương tự nhất:", most_similar_sentence)

Câu tương tự nhất: Tôi yêu lập trình
```

Thử nghiệm 3:

```
most_similar_sentence = find_most_similar_sentence("Con mèo nhà tôi có đôi mắt xanh", S, model)
print("Câu tương tự nhất:", most_similar_sentence)

Câu tương tự nhất: Con chó có bộ lông vàng
```

# CHƯƠNG 2 – HATE SPEECH DETECTION MODEL

*Phát hiện lời nói thù ghét trên mạng xã hội là nhiệm vụ phân loại tự động phát hiện những bình luận có hại từ người dùng và ngăn chặn sự xuất hiện của những bình luận độc hại đó trên các trang xã hội. Lợi ích của nhiệm vụ phát hiện lời nói thù ghét là ngăn chặn nội dung quấy rối và độc hại trên trang mạng xã hội để bảo vệ người dùng tham gia mạng truyền thông xã hội. Tuy nhiên, việc phát hiện lời nói thù ghét là một thách thức, đặc biệt đối với các ngôn ngữ có nguồn tài nguyên thấp như tiếng Việt.*

## 1. Giới thiệu

- Số lượng người dùng internet ở Việt Nam đã tăng lên rất nhiều trong những năm gần đây. Tính đến tháng 1 năm 2024, Việt Nam có 78,44 triệu người dùng Internet, với tỷ lệ sử dụng Internet đạt 79,1% so với tổng dân số. Mỗi người dùng Internet tại Việt Nam dành khoảng 2 giờ 32 phút mỗi ngày cho mạng xã hội và các ứng dụng nhắn tin, một con số cao hơn một chút so với mức trung bình toàn cầu. Sự thâm nhập của mạng xã hội làm thay đổi thói quen hàng ngày của con người về nhiều mặt, đặc biệt là trong kinh doanh và giải trí. Mạng xã hội mở ra một trang nơi mọi người tự do bày tỏ ý kiến và thảo luận về các sự kiện, vấn đề và chính sách. Tuy nhiên, nhiều người có hành vi thô lỗ trên mạng.Lời nói căm thù là ngôn ngữ tấn công và kích động bạo lực chống lại các cá nhân hoặc nhóm dựa trên đặc điểm, tôn giáo, quốc tịch, nguồn

gốc dân tộc, bản dạng giới và khuynh hướng tình dục của họ bằng các hình thức ngôn ngữ khác nhau. Theo định nghĩa, văn bản có lời nói căm thù có những đặc điểm sau:

- Chứa đựng mục tiêu cụ thể. Các mục tiêu cụ thể có thể là các cá nhân, nhóm người, tổ chức, cộng đồng hoặc thậm chí là một quốc gia.
- Kích động bạo lực và hận thù thông qua ngôn ngữ, ngay cả dưới hình thức hài hước tinh tế. Các ngôn ngữ tấn công có thể xuất hiện ở dạng rõ ràng hoặc hàm ý

- Hậu quả của lời nói căm thù ảnh hưởng đến tâm trí con người và cuộc sống của họ. Những người trẻ tuổi dễ bị tổn thương hơn và trở thành nạn nhân của bắt nạt trên mạng, có thể dẫn đến tự tử. Vì vậy, mạng xã hội thường ẩn những bình luận căm thù và cấm những người dùng đăng nhiều bình luận có hại để giảm sự lan truyền của lời nói thù ghét. Tuy nhiên, không ẩn hết mọi bình luận ác ý của người dùng vì số lượng bình luận rất lớn. Vì vậy, mạng xã hội cần sự hỗ trợ để loại bỏ những bình luận độc hại, thù hận. Đây chính là động lực cho việc nghiên cứu tự động phân loại và phát hiện các phát ngôn thù trên các trang mạng xã hội nhằm giữ cho môi trường trực tuyến trong sạch và đảm bảo an toàn cho người dùng trực tuyến.
- Phát hiện lời nói căm thù được phân loại là nhiệm vụ học tập có giám sát. Bên cạnh đó, nhiệm vụ phát hiện lời nói căm thù có liên quan chặt chẽ đến nhiệm vụ phân tích tình cảm (sentiment analysis) vì việc phát hiện lời nói căm thù tương tự như việc xác định tình cảm tiêu cực trong văn bản.Ngoài ra, tập dữ liệu còn đóng vai trò quan trọng trong nhiệm vụ phát hiện lời nói căm thù, đặc biệt là tập dữ liệu có chú thích của con người
- Cuối cùng, sự mất cân bằng của tập dữ liệu là một vấn đề trong nhiệm vụ phân loại. Khi tập dữ liệu mất cân bằng, bộ phân loại được huấn luyện từ tập dữ liệu có xu hướng phân loại dữ liệu mới cho lớp đa số, điều này làm giảm hiệu suất của mô hình phân loại. Để khắc phục tình trạng mất cân bằng của tập dữ liệu, có hai chiến lược: cấp độ dữ liệu và cấp độ thuật toán, nhằm mục đích giảm tác động do tập dữ liệu bị sai lệch gây ra và tăng hiệu suất của mô hình phân loại.

- Trong bài báo cáo này, chúng em tiến hành một số thử nghiệm để nghiên cứu khả năng của các mô hình deep neural network và transformer-based tiên tiến nhất cho bộ dữ liệu tiếng việt trong việc phát hiện lời nói thù ghét.

## 2. Nghiên cứu liên quan

- Các phương pháp chủ yếu để phát hiện lời nói thù ghét là các mô hình học tập có giám sát, phân loại xem các nhận xét hoặc bài đăng có mang tính chất thù địch hay không. Các bộ phân loại truyền thống được áp dụng trong việc phát hiện lời nói căm thù, chẳng hạn như Hồi quy logistic [a], SVM [b] và Random Forest Classifier [c].
- Tuy nhiên, các mô hình deep learning chứng minh sức mạnh của việc phát hiện lời nói căm thù so với các mô hình truyền thống. Các mô hình học sâu được sử dụng cho các tác vụ phát hiện lời nói căm thù, bao gồm Text-CNN , LSTM, Bi-LSTM, GRU và các mô hình kết hợp các mô hình như CNN+GRU và Bi-GRU-LSTM-CNN. Sự xuất hiện của BERT tạo ra một cách tiếp cận nổi bật cho nhiều tác vụ ngôn ngữ tự nhiên như phân loại văn bản, trả lời câu hỏi, nhận dạng thực thể được đặt tên và dịch máy.Theo [d], mô hình được huấn luyện trước dựa trên kiến trúc BERT có thể cải thiện khả năng phát hiện lời nói căm thù do khả năng nắm bắt thông tin ngữ cảnh lớn.
  - Ngoài ra,cũng có mô hình PhoBERT-CNN với một ứng dụng phát trực tuyến để phát hiện các văn bản lời nói căm thù và độc hại trực tuyến. Tóm lại, nhiệm vụ phát hiện lời nói thù ghét được phân loại là nhiệm vụ học tập có giám sát và gần giống với nhiệm vụ phân tích cảm xúc. Do đó, các mô hình SOTA áp dụng cho nhiệm vụ phân loại văn bản và cảm xúc khác có thể được triển khai cho nhiệm vụ phát hiện lời nói căm thù.

## 3. Dữ liệu

- Tập dữ liệu đóng một vai trò thiết yếu trong các nhiệm vụ phát hiện lời nói căm thù. Trong bài này, chúng em sử dụng hai bộ dữ liệu phát hiện lời nói căm thù bằng tiếng Việt: facebook_small_comments và ViHSD.

- Bộ dữ liệu facebook_comment_small được thu thập từ các bình luận trên internet. Tập dữ liệu chứa 2700 bình luận, mỗi bình luận có một hoặc nhiều trong 5 nhãn: normal, hate, harassment, Sexually explicit, Dangerous.
- Chúng em xây dựng mô hình phát hiện lời nói thù ghét nên chỉ giữ lại 3 nhãn: normal, hate, harassment. Các nhãn còn lại mô hình không cần phát hiện nên sẽ được sửa lại thành normal. Ngoài ra, mỗi dữ liệu cũng chứa nhiều nhãn nên các nhãn nào có chứa harassment, chúng em chuyển sang 1 nhãn là harassment, nhãn có chứa hatespeech sẽ chuyển thành 1 nhãn hatespeech.
- Các nhãn được định nghĩa như sau:

| Categories | Descriptions |
|---|---|
| Harassment | Negative or harmful comments targeting identity and/or protected attributes. |
| Hate speech | Content that is rude, disrespectful, or profane. |
| Sexually explicit | Contains references to sexual acts or other lewd content. |
| Dangerous | Promotes, facilitates, or encourages harmful acts. |

Hình 9: Định nghĩa nhãn cho bộ facebook_small_comments

- Tập dữ liệu hiện có chỉ bao gồm 2.700 mẫu, một lượng dữ liệu khá hạn chế để đạt được hiệu suất tối ưu trong việc huấn luyện các mô hình học sâu. Các mô hình học sâu thường yêu cầu một lượng lớn dữ liệu để có thể hiểu được các xu hướng phức tạp và tinh tế từ dữ liệu, đặc biệt là trong lĩnh vực phát hiện ngôn từ thù địch, nơi sự tinh tế và ngữ cảnh của ngôn ngữ đóng một vai trò quan trọng. Do đó, việc chỉ sử dụng 2.700 mẫu là không đủ để đảm bảo mô hình có thể tự học và phát hiện chính xác ngôn từ thù địch trong một phạm vi rộng lớn của dữ liệu thực tế. Để giải quyết vấn đề này, chúng tôi quyết định tích hợp thêm bộ dữ liệu VIHSD vào nghiên cứu của mình do sự tương đồng và bổ sung giá trị cho bộ dữ liệu ban đầu. VIHSD là một tập dữ liệu lớn, được thiết kế đặc biệt cho nhiệm vụ phát hiện ngôn từ thù địch trong ngữ

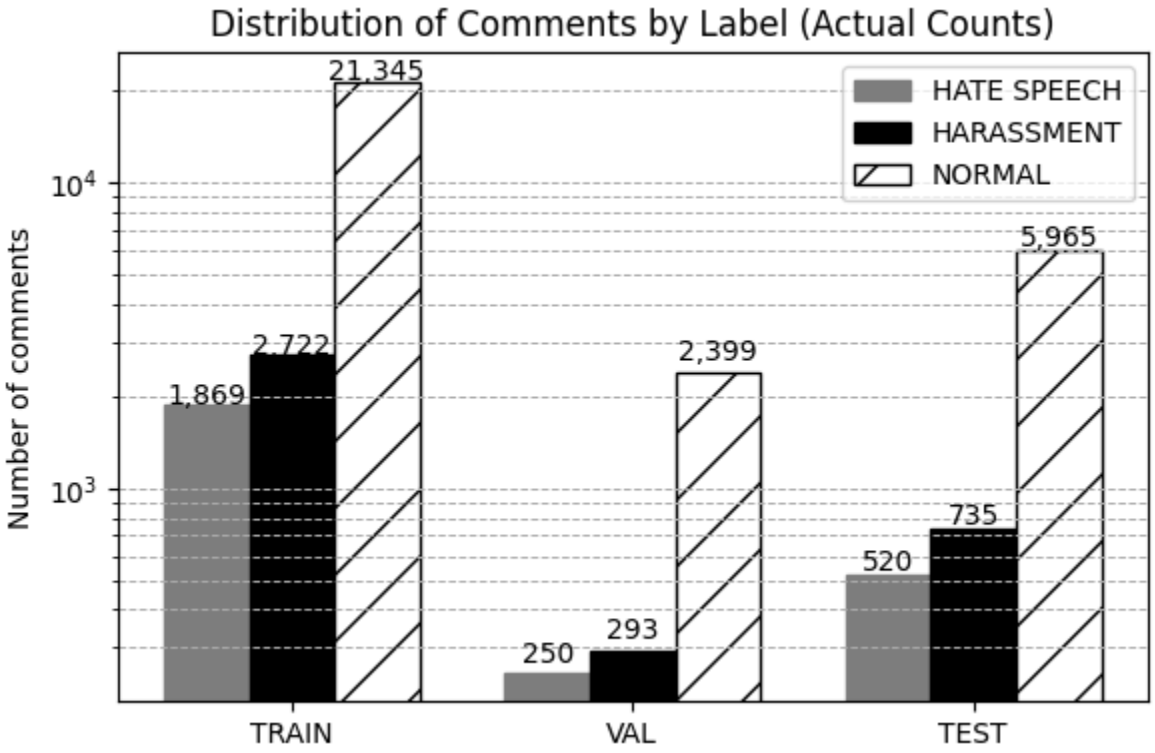
cảnh của ngôn ngữ Việt Nam, điều này giúp tăng cường khả năng đại diện và đa dạng của dữ liệu huấn luyện. Sự kết hợp này nhằm mục đích không chỉ mở rộng kích thước của tập dữ liệu mà còn cải thiện độ phong phú và tính toàn diện của dữ liệu, giúp mô hình học sâu của chúng tôi có cơ hội học hỏi từ một lượng dữ liệu lớn hơn và đa dạng hơn, từ đó tăng khả năng tổng quát hóa và hiệu suất phát hiện ngôn từ thù địch.

- Bộ dữ liệu ViHSD được thu thập từ Facebook và Youtube, hai nền tảng truyền thông xã hội phổ biến nhất ở Việt Nam. Tập dữ liệu chứa 33.400 bình luận, mỗi bình luận có một trong ba nhãn: CLEAN, OFFENSIVE, và HATE. Ngoài ra, tập dữ liệu được chia thành ba phần: tập train, tập dev và tập test với tỷ lệ 7:1:2.

Bảng 1: Định nghĩa nhãn cho bộ ViHSD

| Label | Description | Example |
|---|---|---|
| CLEAN | The comments have no harassment at all. | Comment 1: M.n ơi cho mik hỏi mik theo dõi cô mà mik hk pít cô là con gái thiệt hả mn. |
| OFFENSIVE | The comments contain harassment contents, even profanity words, but do not attack any specific object. | Comment 2: Đồ khùng |
| HATE | The comments have harassment and abusive contents and directly aim at an individual or a group of people based on their characteristics, religion, and nationality | Comment 4: Dm Có a mới không ổn. Mày rình mày chịch riết ổn cái lol |

- Có sự tương đồng giữa các loại nhãn của 2 bộ dữ liệu, cụ thể là hatespeech trong facebook_small_comments sẽ ứng với OFFENSIVE trong ViHSD, nhãn đề cập tới những bị luận thô tục nhưng không ám chỉ 1 cá nhân, tổ chức cụ thể nào. Tương tự harrassment trong facebook_small_comments sẽ ứng với HATE trong ViHSD. Bộ dữ liệu sau khi kết hợp sẽ có các nhãn giống như tập facebook_small_comments ban đầu.
- Các bước kết hợp 2 bộ dữ liệu:
  - Chia tập facebook_small_comments thành 3 tập train, val, test tỉ lệ 7:2:1 bằng phương pháp stratified random sampling.
  - Kết hợp các tập dữ liệu train, val, test tương ứng của 2 bộ dữ liệu
  - Xáo trộn toàn bộ dữ liệu



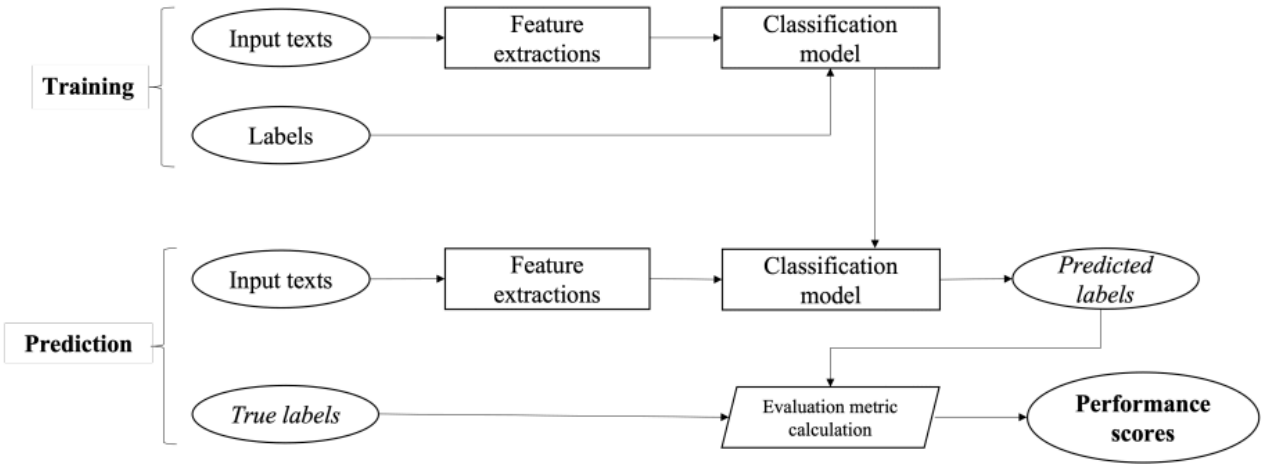

Hình 10: Sự phân bố của ba nhãn trên tập train, val, test

- Biểu đồ cho thấy sự mất cân bằng đáng kể trong tập dữ liệu, với số lượng nhận xét 'normal' vượt xa số lượng nhận xét về 'hate_speech' và 'harassment'.
    - Mất cân bằng: Các mô hình có xu hướng thiên về class đa số, có khả năng dẫn đến việc xác định kém các class thiểu số.
    - Overfitting: Dữ liệu hạn chế về 'hate_speech' và 'harassment' có thể khiến mô hình overfitting dữ liệu đào tạo, cản trở khả năng khái quát hóa của mô hình.
    - Có thể không có đủ sự đa dạng trong các ví dụ về hate_speech và harassment để nắm bắt được các sắc thái của nội dung đó.
    - Metric: Accuracy không phải là số liệu đáng tin cậy do tính mất cân bằng nên chúng em sẽ dùng điểm F1 để mang lại nhiều thông tin hơn.

## 4. Phương pháp

### 4.1 Định nghĩa công việc

- Nhiệm vụ phát hiện lời nói căm thù được xác định như sau:
    - Input: Bình luận của người dùng dưới dạng văn bản.
    - Output: Một nhãn để xác định xem nhận xét đưa ra có phải là lời nói căm thù hay không.




Hình 11: Nhiệm vụ phân loại văn bản

- Phát hiện lời nói căm thù là nhiệm vụ phân loại văn bản. Nhiệm vụ này bao gồm 2 giai đoạn: giai đoạn huấn luyện và dự đoán. Trong giai đoạn huấn luyện, tập dữ liệu huấn luyện, bao gồm văn bản đầu vào và nhãn, được sử dụng để xây dựng mô hình phân loại. Trước khi đưa vào mô hình phân loại, văn bản đầu vào được chuyển đổi thành các vectơ đặc trưng. Các mô hình phân loại được sử dụng để dự đoán dữ liệu mới trong các giai đoạn dự đoán. Các văn bản đầu vào trong giai đoạn này trước tiên được chuyển đổi thành các vectơ đặc trưng. Các vectơ đặc trưng này được mô hình phân loại dự đoán và trả về các nhãn dự đoán. Cuối cùng, để đánh giá hiệu suất của mô hình phân loại, các nhãn dự đoán được so sánh với các nhãn thực (ground-truth) và được định lượng bằng các số liệu đánh giá như accuracy, precision, recall, and F1-score. Điểm F1 với mức trung bình vĩ mô xử lý tất cả các lớp như nhau trong khi accuracy được ưu tiên ở các lớp chính, điều này tạo ra sai lệch cho các tập dữ liệu không cân bằng. Do đó, đối với lớp phân loại văn bản, F1 trung bình vĩ mô được khuyến nghị để đánh giá hiệu suất của các mô hình phân loại.

**4.2 Biểu diễn vectơ và nhúng từ**

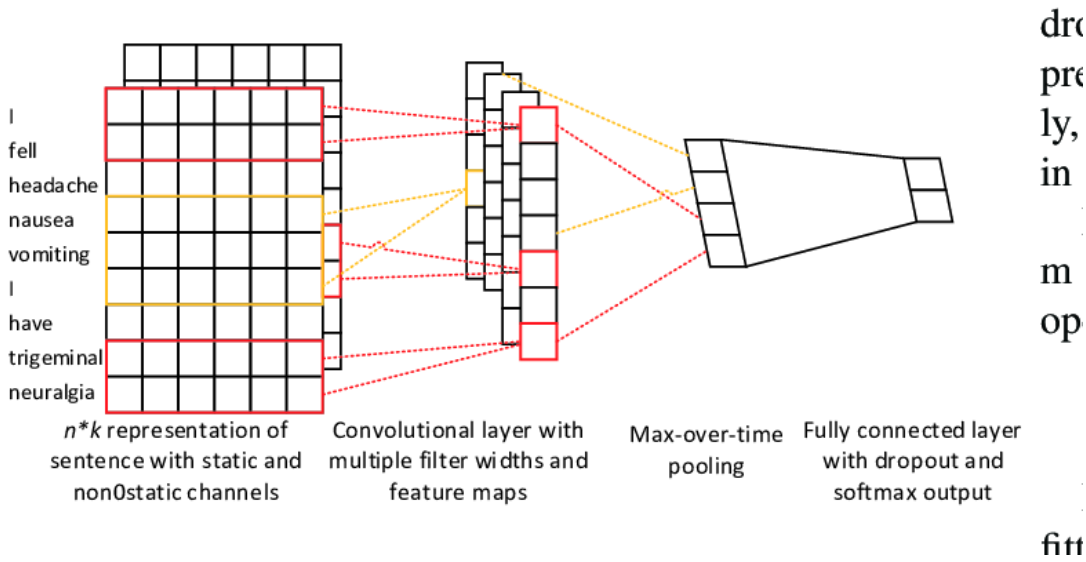
- Biểu diễn vectơ là một phương pháp thiết yếu để hiểu ý nghĩa ngữ nghĩa của văn bản trong xử lý ngôn ngữ tự nhiên. Theo Hình , các văn bản đầu vào được chuyển đổi thành các vectơ đặc trưng trước khi đưa vào mô hình phân loại. Nhúng từ là một trong những cách phổ biến để thể hiện văn bản dưới dạng vectơ.
- Nhúng từ là các kỹ thuật trong NLP biến các từ thành vectơ dựa trên ngữ cảnh của chúng, như các từ xung quanh hoặc toàn bộ bài viết, để giúp máy tính hiểu nghĩa của từ. Quá trình này bắt đầu bằng việc phân đoạn văn bản thành các từ bằng cách sử dụng các mã thông báo, chẳng hạn như pyvi, Underthesea, RRDSegmenter và VnCoreNLP cho tiếng Việt. Ban đầu, các kỹ thuật như TF-IDF và độ tương tự cosine được sử dụng để biểu diễn vectơ, dẫn đến các vectơ thưa thớt. Tuy nhiên, nhu cầu

tính toán ngữ nghĩa hiệu quả hơn đã dẫn đến sự phát triển của các mô hình vectơ dày đặc như Word2Vec, sử dụng phương pháp Skip-gram và CBOW để biểu diễn từ tốt hơn. Quá trình phát triển tiếp tục với việc giới thiệu các mô hình được đào tạo trước như fastText, ELMO, GloVe và BERT, mang lại những cải tiến đáng kể trong các tác vụ NLP khác nhau bằng cách sử dụng các bộ dữ liệu lớn để tránh các vấn đề trang bị quá mức trong các bộ dữ liệu nhỏ. Những mô hình này vượt trội trong việc nắm bắt ý nghĩa ngữ cảnh sâu sắc, khiến chúng trở nên lý tưởng cho nhiều ứng dụng.

- Trong tiếng Việt, có một số từ nhúng được huấn luyện trước như ETNLP7, PhoW2V và fastText. Theo [e], cách nhúng được đào tạo trước fastText có hiệu suất cao hơn các mô hình được đào tạo trước khác của Việt Nam về văn bản trên mạng xã hội. Do đó, chúng em sử dụng từ nhúng này (được gọi là cc.vi.300.vec) cho các thử nghiệm của mình.

### 4.3 Mô hình Deep neural và transformer

- TextCNN, một biến thể của mạng nơ ron tích chập (CNN) được điều chỉnh để xử lý văn bản, được giới thiệu như một cách tiếp cận sáng tạo cho các nhiệm vụ phân loại văn bản. Không giống như các CNN truyền thống được sử dụng rộng rãi để nhận dạng và xử lý hình ảnh, TextCNN áp dụng các nguyên tắc của lớp tích chập cho văn bản, nắm bắt các phần phụ thuộc cục bộ trong văn bản thông qua các filter hoặc kernel.TextCNN đã được áp dụng thành công cho các nhiệm vụ NLP khác nhau như phân tích tình cảm, phân loại chủ đề và phát hiện thư rác. Khả năng nắm bắt các mẫu cục bộ trong văn bản khiến nó đặc biệt hữu ích cho các nhiệm vụ mà ý nghĩa ngữ nghĩa phụ thuộc nhiều vào các chuỗi từ ngắn (n-gram).



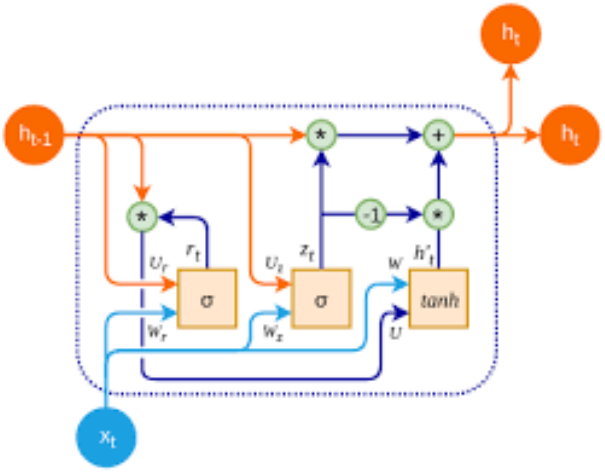

Hình 12: Kiến trúc mô hình TextCNN

- Embedding Layer: Lớp đầu tiên là lớp nhúng giúp chuyển đổi văn bản đầu vào thành vectơ. Mỗi từ trong văn bản được chuyển đổi thành biểu diễn vectơ bằng cách sử dụng các phần nhúng từ được đào tạo trước như Word2Vec, GloVe hoặc các phần nhúng đã học được trong quá trình đào tạo. Sự chuyển đổi này cho phép mô hình hiểu được sự tương đồng về ngữ nghĩa giữa các từ.
- Convolutional Layer: Lớp nhúng được theo sau bởi một hoặc nhiều lớp chập. Các lớp này áp dụng các bộ lọc khác nhau cho các vectơ từ được nhúng. Mỗi bộ lọc trượt trên các vectơ văn bản (thường bao gồm nhiều từ cùng một lúc) và thực hiện các thao tác tích chập, thu thập các mẫu hoặc tính năng như n-gram (nhóm từ). Các bộ lọc có kích thước khác nhau, cho phép mô hình thu được nhiều tính năng n-gram khác nhau cùng một lúc.
- Activation Function: Sau thao tác tích chập, một chức năng kích hoạt như ReLU (Đơn vị tuyến tính chỉnh lưu) sẽ được áp dụng. Bước này giới thiệu tính phi tuyến tính cho mô hình, cho phép mô hình tìm hiểu các mẫu phức tạp trong văn bản.
- Pooling Layer: Các lớp tích chập được theo sau bởi các lớp gộp, thường là các lớp gộp tối đa, làm giảm tính chiều của bản đồ đối tượng bằng cách chỉ giữ lại các đối

tượng quan trọng nhất. Bước này giúp giảm bớt việc tính toán và cũng giúp trích xuất các tính năng quan trọng nhất (như n-gram quan trọng nhất) từ toàn bộ văn bản.

- Fully Connected Layer: Sau khi gộp, đầu ra được làm phẳng và chuyển qua các lớp được kết nối đầy đủ. Các lớp này là các lớp mạng thần kinh truyền thống trong đó mọi đầu vào được kết nối với mọi đầu ra bằng trọng số đã học. Các lớp được kết nối đầy đủ kết hợp các tính năng được học bởi các bộ lọc tích chập khác nhau để hiểu các mẫu cấp cao hơn trong văn bản.
- Output Layer: Lớp cuối cùng là lớp đầu ra, thường là lớp softmax cho các nhiệm vụ phân loại. Nó cung cấp xác suất của từng lớp/danh mục mà văn bản đầu vào có thể thuộc về, dựa trên các tính năng được mạng trích xuất và học.

- GRU (Gated recurrent units) là một loại kiến trúc mạng thần kinh tái phát (RNN), được giới thiệu bởi Cho et al. vào năm 2014, nhằm giải quyết vấn đề biến mất độ dốc phổ biến trong RNN truyền thống. GRU được thiết kế để nắm bắt một cách hiệu quả các phần phụ thuộc trong dữ liệu tuần tự cho các tác vụ như xử lý ngôn ngữ tự nhiên (NLP), phân tích chuỗi thời gian, v.v. Chúng tương tự như một biến thể RNN phổ biến khác, mạng Bộ nhớ ngắn hạn dài (LSTM), nhưng có cấu trúc đơn giản hơn thường dẫn đến thời gian đào tạo nhanh hơn và chi phí tính toán ít hơn.

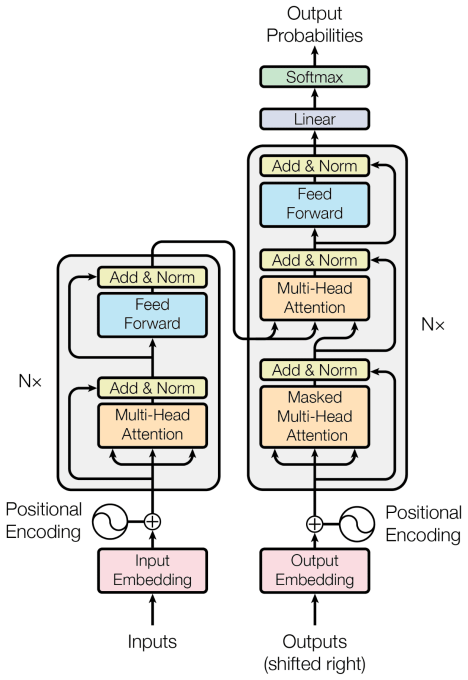


Hình 13: Kiến trúc đơn vị GRU

- GRU có hai cổng:

- Update Gate: Xác định lượng thông tin trong quá khứ (từ các bước thời gian trước đó) cần được chuyển đến tương lai. Điều này rất quan trọng để mô hình quyết định nên giữ hay quên bao nhiêu thông tin trong quá khứ. Cổng cập nhật giúp GRU nắm bắt được sự phụ thuộc lâu dài bằng cách giữ lại thông tin liên quan qua các chuỗi dài.
- Reset Gate: Quyết định lượng thông tin trong quá khứ cần quên. Cổng này được sử dụng để làm cho mô hình quên đi những thông tin không liên quan, về cơ bản cho phép GRU tìm hiểu dữ liệu nào trong quá khứ là quan trọng cần giữ lại và dữ liệu nào có thể loại bỏ. Cổng đặt lại có thể loại bỏ thông tin không liên quan một cách hiệu quả trước khi nội dung bộ nhớ hiện tại được cập nhật.

- PhoBERT là mô hình ngôn ngữ được đào tạo sẵn được thiết kế dành riêng cho văn bản tiếng Việt. Nó tận dụng kiến trúc Transformer, tương tự như BERT (Bidirectional Encoding Regressions from Transformers), được Google giới thiệu vào năm 2018. PhoBERT thể hiện sự tiến bộ đáng kể về khả năng xử lý ngôn ngữ tự nhiên (NLP) cho tiếng Việt, giải quyết các đặc điểm cú pháp và ngữ nghĩa độc đáo của ngôn ngữ.




Hình 14: Kiến trúc Transformer

- Đào tạo trước theo ngôn ngữ cụ thể: PhoBERT được đào tạo trước trên một kho văn bản tiếng Việt lớn. Quá trình đào tạo trước này giúp mô hình hiểu được các sắc thái của cú pháp và ngữ nghĩa tiếng Việt ở mức độ sâu, mang lại hiệu quả cao cho nhiều nhiệm vụ NLP bằng tiếng Việt.
- Ngữ cảnh hai chiều: Giống như BERT, PhoBERT được thiết kế để hiểu ngữ cảnh của một từ dựa trên tất cả môi trường xung quanh nó (trái và phải của từ đó). Sự hiểu biết hai chiều này cho phép thể hiện chính xác hơn ý nghĩa của từ trong các ngữ cảnh khác nhau, dẫn đến những cải tiến trong các nhiệm vụ NLP xuôi dòng.
- Kiến trúc Transformer : PhoBERT sử dụng kiến trúc Transformer, cho phép xử lý các từ song song và chú ý đến những phần có liên quan nhất của văn bản đầu vào khi đưa ra dự đoán. Điều này làm cho PhoBERT hoạt động hiệu quả trong việc nắm bắt sự phức tạp của tiếng Việt.

- PhoBERT đã được áp dụng thành công cho nhiều nhiệm vụ NLP khác nhau dành cho tiếng Việt, bao gồm: Phân loại văn bản, Nhận dạng thực thể được đặt tên (NER), Question Answering, Text Summarization,...
- Ưu điểm chính của PhoBERT so với các mô hình đa ngôn ngữ thông thường hoặc các mô hình được đào tạo về ngôn ngữ có đặc điểm tương tự tiếng Việt là đào tạo chuyên sâu. Bằng cách tập trung vào văn bản tiếng Việt, PhoBERT cung cấp khả năng hiểu và xử lý vượt trội cho ngôn ngữ tiếng Việt, dẫn đến kết quả chính xác hơn và phù hợp với ngữ cảnh hơn trên một loạt nhiệm vụ NLP.

## 5. Kết quả thực nghiệm

### 5.1 Chuẩn bị

- Chúng em mô tả các bước tiền xử lý được áp dụng cho tập dữ liệu trước khi đưa chúng vào mô hình. Các bước chi tiết được mô tả dưới đây:
  - Phân đoạn văn bản thành word bằng cách sử dụng công cụ pyvi. Đối với PhoBERT, chúng tôi sử dụng VnCoreNLP để mã hóa văn bản thành từ.
  - Loại bỏ stopwords khỏi văn bản bình luận bằng danh sách stopwords tiếng Việt.
  - Chuyển văn bản sang dạng chữ thường.
  - Xóa các biểu tượng cảm xúc, thẻ bắt đầu bằng # và liên kết url khỏi nhận xét.

- Đối với các mô hình TextCNN và GRU, đặt sequence_length bằng 100, 40 kỷ nguyên và tỷ lệ drop_out bằng 0,5. Đối với TextCNN, số lượng bộ lọc là 32. Chúng em sử dụng thư viện Keras để triển khai TextCNN và GRU. Đối với các mô hình transformer, chúng em đặt sequence_length bằng 100 và 4 kỷ nguyên. Sử dụng thư viện Huggingface để triển khai PhoBERT.

### 5.2 Hiệu suất các mô hình phân loại

- Theo Bảng 2, mô hình PhoBERT (m-bert-cased) thu được kết quả tốt nhất trên tập dữ liệu tổng hợp cũng như tập facebook_small_comments, đạt 65,14% theo điểm F1 và 87,44% theo điểm Accuracy.

Bảng 2: Kết quả thực nghiệm của các mô hình phân loại trên tập dữ liệu kết hợp

| **Pre-train model** | **Accuracy (%)** | **F1-macro (%)** |
|---|---|---|
| Text CNN + fastText | 86.01 | 61.38 |
| GRU + fastText | 84.37 | 60.14 |
| PhoBERT | **87.44** | **65.14** |

Bảng 3: Kết quả thực nghiệm của các mô hình phân loại trên tập dữ liệu facebook_small_comments

| **Pre-train model** | **Accuracy (%)** | **F1-macro (%)** |
|---|---|---|
| Text CNN + fastText | 82.78 | 60.59 |
| GRU + fastText | 81.30 | 59.01 |
| PhoBERT | **86.50** | **67.91** |

- Mô hình transformer như PhoBERT cho kết quả tốt hơn các mô hình Deep Neural.
- Có thể thấy từ Bảng rằng có sự chênh lệch đáng kể giữa điểm Accuracy và điểm macro F1 do sự mất cân bằng giữa các lớp trong bộ dữ liệu tổng hợp và facebook_small_comments đã được đề cập trước đó.

Bảng4 : Hiệu suất của các mô hình phân loại trên mỗi lớp của tập dữ liệu tổng hợp theo F1 score

| **Models** | **normal** | **hate** | **harassment** |
|---|---|---|---|
| Text CNN + fastText | 0.93 | 0.39 | 0.51 |
| GRU + fastText | 0.92 | 0.38 | 0.49 |
| PhoBERT | 0.94 | 0.44 | 0.58 |

- Có thể thấy từ Bảng  rằng mô hình PhoBERT vượt trội hơn các mô hình còn lại ở cả ba loại nhãn normal, hate, harassment. Do đó, không cần sử dụng mô hình ensemble trong trường hợp này.

**6. Phân tích lỗi**

- Như được hiển thị trong Hình 13, mô hình trên tập dữ liệu tổng hợp có vấn đề về dự đoán giữa nhãn hate với normal. Số lượng bình luận hate được dự đoán là bình luận normal nhiều hơn số bình luận dự đoán đúng.


Hình 15: Confusion matrices của mô hình tốt nhất trên tập dữ liệu tổng hợp

- Thử thách đầu tiên trong việc phát hiện ý nghĩa xúc phạm từ các bình luận bằng văn bản là các bình luận được viết bằng từ lóng như: "l" trong Bình luận số 6, "vl". Ngoài tiếng lóng, các bình luận còn chứa teen code và chữ viết tắt. Ví dụ chữ "K" ở đầu Comment 2 và 3 là viết tắt của "Không". Thử thách cuối cùng là người dùng viết từ trong bình luận sai cú pháp. Ví dụ: từ "í thức" ở bình luận số 3 là đúng nếu viết là "ý thức".

Bảng 5: Một số bình luận dự đoán sai trên tập dữ liệu tổng hợp

| **No** | **Comments** | **True Label** | **Predicted Label** |
|---|---|---|---|
| 1 | từ dịch này thấy ý thức người nhật xăng đổ nửa bình, nhưng giấy vệ sinh không thể thiếu. không hy sinh bản thân để cứu người khác, nhưng dính dịch thì kéo nhiều | harassment | normal |

| | | | |
|---|---|---|---|
| | người dính theo. | | |
| 2 | K có nghìn nào giúp dân thì nín đi! | harassment | normal |
| 3 | K nói thì tự kiêu,nói thì lại tự ái chứ í thức thế ai mà chịu nổi😡 | hate | normal |
| 4 | Nguyễn Đức Ánh khôn hơn cả họ nhà m. | harassment | normal |
| 5 | Người ta đã xin lỗi rồi sao cứ vào nick người ta xem rồi đăng lên chửi hoài,nó chửi thề thì kệ nó,nó chửi mấy người chửi nó chứ đau có dám đem người VN ra chửi nữa đau mà nhào vào chửi nó cả vô duyên thế | hate | normal |
| 6 | Bú l ngọc Trinh sẽ tinh | harassment | normal |

- Bên cạnh thách thức ở cấp độ từ, ẩn ý trong các bình luận cũng khiến mô hình phân loại khó có thể dự đoán nhãn chính xác. Ví dụ Bình luận số 1. Tuy không chứa từ ngữ xúc phạm nào nhưng nó có ý lên án về đối tượng 'người nhật'. Comments #4 còn thể hiện sự mỉa mai đối với người dùng khác. Bình luận này cũng khó được xếp vào loại thù ghét vì sự mỉa mai không được thể hiện một cách trực tiếp, từ "khôn" ở đây có nghĩa mỉa mai.
- Nhìn chung, có 2 lý do dẫn đến dự đoán sai khi phát hiện lời nói căm thù:
  - Các bình luận được viết bằng teen code và dạng viết tắt.
  - Các bình luận được viết dưới dạng ẩn ý, khó xác định được hàm ý thù ghét, độc hại tiềm ẩn.

- Mô hình trên tập dữ liệu facebook_small_comments cũng gặp vấn đề trong việc dự đoán giữa lớp hate và harassment. Ngược lại, số lượng bình luận hate được dự đoán là harassment gần bằng số lượng bình luận dự đoán đúng. Sự sai lệch là do sai sót trong việc đánh nhãn với bộ dự liệu này, được chỉ ra dưới đây:


Hình 16: Confusion matrices của mô hình tốt nhất trên tập dữ liệu facebook_small_comments

Bảng 6: Một số bình luận dự đoán sai trên tập dữ liệu facebook_small_comments

| **No** | **Comments** | **True Label** | **Predicted Label** |
|---|---|---|---|
| 1 | Phạm Thái Hoàng Long ồ loại m đ cãi dc là lại địt mẹ mày à, loạn luân vcl | hate | harassment |
| 2 | Joseph Tuấn Anh T mà ăn hiếp m à thằng lồn | hate | harassment |
| 3 | Má sử cho đẹp con đĩ đó nha má nghe nó ns mà tức thiệt chứ | hate | harassment |
| 4 | huyhy nguyễn thôi bạn ơi. Đừng có lên mạng xl về trải sự đời với thất bại và thành công với người khác.\nKhông lại thành người hay nói đạo lý thì thường sống như l*n đấy. | hate | harassment |

- Như bảng trên, rõ ràng các bình luận phải được xem là harassment vì đều ám chỉ một đối tượng cụ thể, mô hình được kết hợp đào tạo trên bộ dữ liệu ViHSD có chất lượng tốt hơn nên cho chất lượng tốt. Bộ dữ liệu facebook_small_comments vẫn còn sai lệch giữa hatespeech và harassment.

**7. Hướng giải quyết trong tương lai**

- Trong bài lần thử nghiệm này có sự chênh lệch đáng kể giữa điểm Accuracy và điểm macro F1 do sự mất cân bằng giữa các lớp trong bộ dữ liệu. Để giải quyết vấn đề này, có thể sử dụng các giải pháp sau:

- Data Augmentation: Làm phong phú dữ liệu bằng cách sử dụng các phương pháp như synonym replacement (thay thế từ đồng nghĩa), back-translation (dịch ngược), hoặc tạo câu mới bằng cách sử dụng các mô hình ngôn ngữ để tăng số lượng mẫu cho lớp thiểu số. Cách tiếp cận này giúp tăng số lượng mẫu dữ liệu mà không làm thay đổi phân phối của dữ liệu ban đầu.
- Under-sampling và Over-sampling: Under-sampling loại bỏ một số mẫu từ lớp đa số, trong khi over-sampling tăng cường số lượng mẫu trong lớp thiểu số. SMOTE (Synthetic Minority Over-sampling Technique) là một kỹ thuật over-sampling phổ biến, tạo ra các mẫu dữ liệu mới trong lớp thiểu số bằng cách kết hợp các đặc điểm của các mẫu hiện tại.
- Thiết kế Mô Hình Có Trọng Số: Áp dụng trọng số cho các lớp khác nhau trong hàm mất mát của mô hình để tăng độ quan trọng của lớp thiểu số. Kỹ thuật này giúp giảm thiểu ảnh hưởng của sự thiên vị về lớp đa số bằng cách phạt nặng hơn đối với những sai lầm trên lớp thiểu số.
- Sử Dụng Các Thuật Toán Đặc Biệt Cho Dữ Liệu Mất Cân Bằng: Một số thuật toán đã được thiết kế hoặc tối ưu hóa để làm việc tốt với dữ liệu mất cân bằng, như Cost-sensitive Learning hoặc Ensemble Methods (ví dụ: Bagging, Boosting).

Ensemble Methods kết hợp nhiều mô hình nhỏ để cải thiện hiệu suất và độ chính xác trên dữ liệu mất cân bằng.

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**


1. Mô hình CBOW (Continuous Bag of Words) – The blog of Nguyen Truong Long
2. Word2vec-Machine Learning cho dữ liệu dạng bảng (machinelearningcoban.com)
3. Feature Engineering (Phần 5): Phương pháp nâng cao để xử lý dữ liệu dạng văn bản, phi cấu trúc (2/2) (viblo.asia)
4. NLP 101: Word2Vec - Skip-gram và CBOW (ichi.pro)
5. Waseem, Z., Hovy, D.: Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In: Proceedings of the NAACL Student Research Workshop, pp. 88–93. Association for Computational Linguistics, San Diego, California (2016). https://doi.org/10.18653/v1/ N16-2013. https://www.aclweb.org/anthology/N16-2013
6. Luu, S.T., Nguyen, H.P., Van Nguyen, K., Nguyen, N.L.-T.: Comparison between traditional machine learning models and neural network models for vietnamese hate speech detection. In: 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), pp. 1–6 (2020). IEEE
7. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: Proceedings of the 26th International Conference on World Wide Web Companion. WWW '17 Companion, pp. 759–760. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2017). https://doi.org/10. 1145/3041021.3054223. https://doi.org/10.1145/3041021.3054223
8. Ali, R., Farooq, U., Arshad, U., Shahzad, W., Beg, M.O.: Hate speech detection on twitter using transfer learning. Computer Speech & Language 74, 101365 (2022)
9. Huynh, H.D., Do, H.T.-T., Nguyen, K.V., Nguyen, N.T.-L.: A simple and efficient ensemble classifier combining multiple neural network models on social media datasets in Vietnamese. In: Proceedings of the 34th Pacific Asia Conference on

Language, Information and Computation, pp. 420– 429. Association for Computational Linguistics, Hanoi, Vietnam (2020). https://aclanthology.org/2020.paclic-1.48

**Tiếng Anh**

1. Fortuna, P., Nunes, S.: A survey on automatic detection of hate speech in text. ACM Comput. Surv. 51(4) (2018)..
2. https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa

# PHỤ LỤC

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn