



Hướng dẫn tiền xử lý dữ liệu tiếng Việt cho SVM phân loại cảm xúc

Phân loại cảm xúc (sentiment analysis) trên các đánh giá sản phẩm tiếng Việt đòi hỏi việc tiền xử lý kỹ lưỡng để mô hình SVM có thể học tốt nhất. Dữ liệu review sản phẩm thường ngắn gọn và đơn giản, nhưng tiếng Việt có đặc thù riêng (dấu câu, từ ghép, tiếng lóng, v.v.). Dưới đây là hướng dẫn chi tiết từng bước tiền xử lý văn bản tiếng Việt, giúp tối ưu hóa quá trình huấn luyện mô hình SVM phân loại cảm xúc.

1. Chuẩn hóa văn bản (Normalization & làm sạch dữ liệu)

Trước tiên, cần làm sạch và chuẩn hóa văn bản thô để giảm độ nhiễu và độ phức tạp của dữ liệu đầu vào. Các bước bao gồm:

- **Chuyển thành chữ thường (lowercase):** Chuyển tất cả ký tự chữ cái về dạng chữ thường để thống nhất định dạng và tránh phân biệt từ chỉ do khác hoa/thường `ptithcm.edu.vn` . (Ví dụ: "Tốt" và "tốt" được xem là một từ duy nhất). Việc này quan trọng vì chữ in hoa và chữ thường có mã khác nhau nhưng ý nghĩa như nhau, nếu không thống nhất sẽ gây khó khăn cho mô hình `ptithcm.edu.vn` .
- **Xóa ký tự đặc biệt, dấu câu không cần thiết:** Loại bỏ các dấu câu thừa và ký tự không thuộc bảng chữ cái (như `!@#%&*. . .`) trừ khi chúng mang ý nghĩa đặc biệt. Thông thường trong phân tích cảm xúc, dấu câu như `!` hoặc `?` có thể biểu thị cảm xúc mạnh, nhưng với mô hình bag-of-words đơn giản, ta thường bỏ chúng để giảm nhiễu `ptithcm.edu.vn` . Bạn cũng nên loại bỏ các ký tự lặp vô nghĩa (vd: `"!!!!!"` hoặc `"^^"`, `"~"`) do người dùng hay chèn để nhấn mạnh, vì chúng không đóng góp ý nghĩa rõ rệt.

- **Loại bỏ emoji, icon:** Các biểu tượng cảm xúc (emoji) và icon đặc biệt thường không được mô hình văn bản xử lý trực tiếp. Bạn nên xóa hoặc thay thế chúng bằng từ tương ứng. Ví dụ có thể thay 😊 bằng từ khóa `positive_emoticon` nếu muốn giữ thông tin, nhưng cách đơn giản nhất là loại bỏ hoàn toàn các icon/emoji để tránh gây nhiễu `ptithcm.edu.vn` .
- **Xóa liên kết (URL) và thẻ HTML:** Đường dẫn website, địa chỉ email hoặc đoạn mã HTML rút lại không mang ý nghĩa cảm xúc về sản phẩm. Ta nên loại bỏ toàn bộ chuỗi URL (vd: `http://...`) hoặc các thẻ `<tag>` nếu dữ liệu được thu thập từ web.
- **Loại bỏ khoảng trắng thừa:** Sau khi xóa các thành phần trên, thay thế nhiều khoảng trắng liên tiếp bằng một khoảng trắng, và loại bỏ khoảng trắng ở đầu/cuối câu. Điều này giúp văn bản gọn gàng và các bước xử lý sau hoạt động chính xác (vd: "Sản phẩm tốt " thành "sản phẩm tốt").

Thực hiện các bước trên giúp dữ liệu "sạch" và thống nhất hơn, giảm số lượng biến thể của từ và tập trung vào nội dung chính của review. Theo một nghiên cứu, việc làm sạch và chuẩn hóa đúng cách giúp cải thiện độ tin cậy của việc trích xuất đặc trưng và hiệu suất mô hình `ptithcm.edu.vn` `ptithcm.edu.vn` .

2. Xử lý dấu tiếng Việt (Giữ hay bỏ dấu?)

Dấu chữ (dấu tiếng Việt) là yếu tố phân biệt nghĩa quan trọng trong tiếng Việt. Ví dụ, "gan", "gân" và "gắn" là ba từ khác nhau hoàn toàn nhờ dấu. Do đó, thường **nên giữ lại dấu** trong văn bản để bảo toàn ngữ nghĩa. Việc **bỏ dấu** có thể làm nhiều từ trở nên nhập nhằng, gây khó khăn cho cả người và máy trong việc hiểu đúng ý nghĩa [arxiv.org](#) . Theo thống kê, 95% từ tiếng Việt có dấu và có đến 80% những từ này trở nên mơ hồ nếu bị loại bỏ dấu [arxiv.org](#) . Chẳng hạn, câu "**Co ay rat dam dang**" khi không dấu có thể hiểu thành "**Cô ấy rất đảm đang**" (ý nghĩa tích cực) hoặc "**Cô ấy rất dâm dăng**" (ý nghĩa tiêu cực) – một sự khác biệt rất lớn về cảm xúc [arxiv.org](#) .

Tuy nhiên, trong một số trường hợp cụ thể, người dùng có thể nhập **không dấu** (vd: "san pham tot" thay vì "sản phẩm tốt"). Nếu tập dữ liệu của bạn *phần lớn* không có dấu hoặc không nhất quán (lúc có dấu, lúc không), bạn **có thể cân nhắc bỏ dấu toàn bộ** để thống nhất định dạng. Cách làm này giúp mô hình không bị rối bởi hai phiên bản của cùng một từ (có dấu vs. không dấu). **Lưu ý:** Nếu chọn bỏ dấu, cần ý thức rằng thông tin ngữ nghĩa sẽ mất mát đáng kể. Mô hình SVM vẫn có thể hoạt động trên dữ liệu không dấu, nhưng bạn có thể cần bổ sung các kỹ thuật khác (như mô hình khôi phục dấu tự động) để giảm thiểu ảnh hưởng tiêu cực. Nhìn chung, **lời khuyên** cho bài toán phân loại cảm xúc là **giữ nguyên dấu tiếng Việt** để đảm bảo độ chính xác cao nhất, trừ phi dữ liệu của bạn buộc phải xử lý khác.

3. Chuẩn hóa chính tả và từ viết tắt

Ngôn ngữ dùng trong các review thực tế thường không theo chuẩn chính tả 100%. Người dùng có xu hướng **viết tắt, dùng tiếng lóng hoặc sai chính tả** khi đánh giá sản phẩm. Ví dụ phổ biến: "ko" hoặc "khong" thay cho "**không**", "bt" hoặc "bth" thay cho "**bình thường**", "sp" cho "**sản phẩm**", "mn" cho "**mọi người**", "sdT" cho "**số điện thoại**", v.v. Việc chuẩn hóa những trường hợp này là **rất cần thiết** để giảm độ biến thiên của dữ liệu và giúp mô hình hiểu được ý nghĩa thực sự:

- **Thay thế từ viết tắt bằng từ đầy đủ:** Xác định các từ/cụm từ viết tắt thông dụng trong tiếng Việt và thay bằng từ gốc. Ví dụ: "ko", "k", "khong" → "**không**"; "j" → "**gi**"; "bt", "bth" → "**bình thường**"; "sp" → "**sản phẩm**"; "dc" → *****được*****; "ae" → *****anh em*****; "mk" → *****mình** (tùy ngữ cảnh); v.v. Sau bước này, tất cả văn bản sẽ ở dạng tiếng Việt chuẩn, có dấu, giúp bước tách từ và phân tích kế tiếp hoạt động hiệu quả hơn.

- **Sửa lỗi chính tả thường gặp:** Nếu người dùng hay gõ sai chính tả (ví dụ: "sản phẩm" thay vì "sản phẩm", "tôt" thay vì "tốt"), ta nên có bước sửa lại cho đúng (dựa trên từ điển hoặc luật). Một số lỗi phổ biến gồm gõ thiếu dấu, sai dấu hoặc nhầm lẫn phụ âm (vd: "nj" thay "nh", "w" thay "qu"...). Có thể sử dụng thư viện kiểm tra chính tả hoặc tự xây dựng bảng quy tắc cho những lỗi phổ biến trong tập dữ liệu.
- **Thống nhất cách viết:** Đối với những trường hợp một khái niệm có nhiều cách viết, hãy chọn một cách viết nhất quán. Ví dụ: số **"100 nghìn"** có thể viết là "100k", "100K", "100.000" – nên quy đổi hết về một dạng (chẳng hạn "100000") hoặc một token thống nhất (chẳng hạn `<num>` nếu muốn đặc biệt xử lý số).

Việc chuẩn hóa chính tả và từ viết tắt giúp giảm đáng kể số lượng **từ vựng độc nhất** (unique tokens) trong tập dữ liệu, từ đó mô hình học máy sẽ không bị phân tán bởi các biến thể của cùng một từ. Nghiên cứu đã chỉ ra các bình luận của người dùng thường *viết tắt, sai chính tả là chuyện hiển nhiên*, nếu dữ liệu không được chuẩn hóa thì kết quả phân tích sẽ bị ảnh hưởng rất nhiều ptithcm.edu.vn. Ví dụ trong dữ liệu đánh giá: "ko đúng" sẽ được thay thành **"không đúng"** để mô hình nhận ra từ **"không"** (phủ định) và **"đúng"** (tích cực) thay vì gặp token lạ "ko" ptithcm.edu.vn. Hãy xây dựng một bảng các từ viết tắt/cách nói lóng thường gặp trong tập dữ liệu của bạn và thay thế chúng trong giai đoạn tiền xử lý này.

4. Loại bỏ từ dừng (Stopwords)

Từ dừng (stopwords) là các từ phổ biến ít mang nội dung ý nghĩa, ví dụ như "là", "thì", "và", "nhưng", "các", "một", "những", "rằng",... Trong tiếng Việt, đặc biệt văn phong bình luận, còn có các từ thừa như "ạ", "ạ?", "nhé", "thôi", v.v. Loại bỏ stopwords giúp giảm **nhều** và **giảm số chiều của vector đặc trưng**, tập trung mô hình vào các từ ngữ mang thông tin cảm xúc rõ rệt hơn [slideshare.net](https://www.slideshare.net). Thông thường, ta sẽ có sẵn một danh sách stopwords tiếng Việt (có thể tự xây dựng hoặc sử dụng các bộ có sẵn từ cộng đồng) và loại bỏ mọi từ trong danh sách đó khỏi văn bản sau khi đã tách từ. Ví dụ: câu **"Sản phẩm này rất là tốt và cực kỳ hữu ích"** sau khi loại bỏ stopwords không cần thiết có thể còn **"sản phẩm rất tốt cực kỳ hữu ích"**.

Tuy nhiên, cần hết sức lưu ý trong bài toán phân tích cảm xúc: **một số từ dừng mang tính phủ định hoặc cảm thán có thể ảnh hưởng đến ý nghĩa câu**. Đặc biệt là từ **"không"** (và các từ phủ định tương tự như "chẳng", "chưa", "chả", "đâu", ...). Nếu ta mù quáng loại bỏ tất cả stopwords, câu **"không tốt"** sẽ mất từ "không" và chỉ còn "tốt", hoàn toàn trái ngược nghĩa gốc. Do đó, đối với các **từ phủ định**, ta **không nên loại bỏ** như stopwords thông thường [researchgate.net](https://www.researchgate.net). Có hai hướng xử lý: (1) giữ lại từ "không" trong dữ liệu đã token hóa, hoặc (2) xử lý đặc biệt bằng cách gắn nó với từ phía sau để tạo thành một token mới mang nghĩa phủ định (xem bước mẹo nâng cao bên dưới về xử lý phủ định).

Tóm lại, hãy **loại bỏ stopwords một cách có chọn lọc**. Giữ lại những từ chức năng có vai trò trong diễn đạt cảm xúc (phủ định, mức độ, liên kết quan trọng như "nhưng"). Các từ dừng thông thường khác có thể loại bỏ an toàn để tập trung vào nội dung chính. Thực nghiệm cho thấy việc tách từ và loại bỏ hư từ (stopwords) trong tiếng Việt giúp cải thiện độ chính xác phân tích cảm xúc [slideshare.net](https://www.slideshare.net). Hãy đảm bảo thực hiện bước này **sau khi đã chuẩn hóa và tách từ** (xem bước tiếp theo), vì khi đó bạn mới xác định đúng ranh giới các từ để loại bỏ.

5. Vector hóa văn bản bằng TF-IDF (tối ưu cho tiếng Việt)

Sau khi đã có văn bản sạch và chuẩn hóa, bước tiếp theo là **biến đổi văn bản thành vector số** để mô hình SVM có thể xử lý. Kỹ thuật thông dụng cho bài toán này là sử dụng **TF-IDF** (Term Frequency - Inverse Document Frequency) để biểu diễn tài liệu. Đối với tiếng Việt, ta cần lưu ý một số điểm để việc vector hóa TF-IDF hiệu quả:

5.1 Tách từ (Tokenization) cho tiếng Việt

Trước khi áp dụng TF-IDF, cần **tách văn bản thành các từ (token)** phù hợp. Khác với tiếng Anh, tiếng Việt dùng khoảng trắng để ngăn cách **âm tiết** chứ không phải từ hoàn chỉnh. Một từ ghép trong tiếng Việt thường gồm nhiều âm tiết, ví dụ "**đất nước**" gồm hai âm tiết "đất" + "nước" nhưng ghép lại mới tạo thành nghĩa "**nation**" (đất nước)

ptithcm.edu.vn . Nếu không tách từ đúng, mô hình sẽ coi "đất" và "nước" là hai từ độc lập, mất đi ý nghĩa thực sự. Vì vậy, **word segmentation** là bước rất quan trọng trong xử lý ngôn ngữ tiếng Việt ptithcm.edu.vn .

Bạn có thể sử dụng các công cụ tách từ tiếng Việt phổ biến như **VnTokenizer (thuộc thư viện `pyvi` hoặc `VietIZA`)**, **Underthesea**, **VnCoreNLP**,... Các công cụ này sẽ gộp các âm tiết thành từ hoàn chỉnh (thường bằng cách nối dấu gạch dưới, ví dụ "đất_nước"). Sau khi tách từ, câu ví dụ: "Sản phẩm này không tốt lắm" có thể được chuyển thành "sản_phẩm này không tốt_lắm" . Lúc này, mỗi token (như "sản_phẩm", "tốt_lắm") đại diện cho một ý nghĩa cụ thể.

5.2 Tính toán TF-IDF cho văn bản

Sau khi có danh sách các token (từ) của mỗi câu, ta tiến hành vector hóa bằng TF-IDF:

- Ý tưởng TF-IDF:** TF-IDF tạo một ma trận đặc trưng trong đó mỗi cột là một từ (token) trong từ vựng, mỗi hàng tương ứng với một tài liệu (review). Giá trị của mỗi ô là mức độ quan trọng của từ đó trong tài liệu, tính bằng công thức: **TF * IDF**. TF (term frequency) là tần suất xuất hiện của từ trong tài liệu; IDF (inverse document frequency) giảm trọng số của những từ xuất hiện quá phổ biến trong toàn bộ tập tài liệu. Nhờ IDF, TF-IDF làm giảm ảnh hưởng của các từ rất thông dụng (như stopwords hoặc từ chung chung xuất hiện ở hầu hết các review) thorphan.github.io . Ví dụ, trong phân tích review quán ăn, cụm từ "**quán ăn**" có thể xuất hiện ở hầu hết các đánh giá nên thông tin phân biệt cảm xúc không cao, TF-IDF sẽ gán trọng số thấp cho nó thorphan.github.io . Ngược lại, từ hiếm hơn như "**ngon**", "**kinh khủng**" sẽ được tăng trọng số nếu chúng xuất hiện, vì chúng mang nhiều ý nghĩa phân biệt cảm xúc.
- Sử dụng thư viện:** Có thể dùng trực tiếp `TfidfVectorizer` của `scikit-learn` để chuyển danh sách câu thành ma trận TF-IDF. Trước tiên cần đảm bảo input đã được tách từ (có thể nối lại thành chuỗi với dấu cách hoặc dấu gạch dưới giữa các token). Ví dụ:

```
python
```

 Copy

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vec = TfidfVectorizer(max_features=3000, min_df=5, max_df=0.8, sublinear_tf=True)
X = tfidf_vec.fit_transform(list_cac_cau_da_xu_ly)
```

Tham số `max_features=3000` giới hạn kích thước từ vựng còn 3000 từ quan trọng nhất (giúp giảm chiều dữ liệu), `min_df=5` bỏ qua các từ xuất hiện ít hơn 5 tài liệu, `max_df=0.8` bỏ qua các từ xuất hiện quá nhiều (>80% tài liệu) thorphan.github.io, còn `sublinear_tf=True` áp dụng thang đo logarit cho TF (giảm ảnh hưởng của việc một từ lặp lại quá nhiều lần trong cùng một văn bản dài) thorphan.github.io. Những lựa chọn này sẽ **loại bỏ từ quá hiếm hoặc quá phổ biến** khỏi vector hóa, giúp giảm nhiễu và tăng tốc độ tính toán thorphan.github.io. Ngoài ra, vì ta đã tự làm sạch stopwords trước đó, có thể không cần dùng tham số `stop_words` của `TfidfVectorizer` thorphan.github.io.

- **Sử dụng n-gram:** Đối với dữ liệu tiếng Việt, bạn có thể cân nhắc sử dụng **bi-gram** (cặp từ liên tiếp) bên cạnh unigram để nắm bắt ngữ cảnh tốt hơn. Ví dụ, bi-gram cho phép mô hình biết "**không tốt**" khác với "**tốt**", hay "**rất tốt**" mang sắc thái mạnh hơn "**tốt**". Thiết lập `ngram_range=(1,2)` trong `TfidfVectorizer` sẽ tạo đặc trưng cho cả đơn từ và cặp từ. Tuy nhiên, lưu ý rằng thêm bi-gram sẽ làm tăng số lượng đặc trưng đáng kể; nên kết hợp với `max_df`, `min_df` để kiểm soát.
- **Chuẩn hóa và trọng số:** `TfidfVectorizer` mặc định sẽ chuẩn hóa vector của mỗi tài liệu về độ dài 1 (`norm="l2"`), điều này phù hợp cho SVM vì đảm bảo các tài liệu dài ngắn không ảnh hưởng quá mức bởi độ dài. Bạn nên giữ việc chuẩn hóa này. Ngoài ra, IDF trong sklearn được tính với công thức log có điều chỉnh (`smooth idf`) để tránh chia cho 0. Những thiết lập mặc định này thường đã tối ưu cho bài toán chung.

Kết quả của bước TF-IDF là một **ma trận thưa (sparse matrix)** kích thước (số review) x (kích thước từ vựng). Ma trận này có rất nhiều giá trị 0 (vì mỗi review ngắn chỉ chứa một tập nhỏ từ vựng) – ví dụ có thể 99% phần tử là 0 với dữ liệu review ngắn viblo.asia. Điều này bình thường đối với bài toán text. Lúc này, mỗi review đã được biến thành một vector số, sẵn sàng để đưa vào mô hình SVM huấn luyện.

6. Mẹo nhỏ cải thiện hiệu quả mô hình SVM trên tiếng Việt

Sau khi thực hiện các bước tiền xử lý và vector hóa, bạn đã có dữ liệu sẵn sàng cho mô hình. Dưới đây là một số mẹo bổ sung giúp SVM phân loại cảm xúc tiếng Việt hiệu quả hơn:

- Xử lý từ phủ định (negation):** Như đã đề cập, các từ phủ định như "không", "chẳng", "chưa" có thể đảo ngược polarities của từ cảm xúc đi sau. Một kỹ thuật hữu ích là thêm dấu hiệu negation vào từ phía sau. Ví dụ: biến cụm "**không tốt**" thành "**không_NOT_tốt**" hoặc thay "tốt" bằng "**NOT_tốt**" như một token mới [researchgate.net](https://www.researchgate.net/publication/351841414) . Cách này giúp mô hình nhận biết "tốt" có "**NOT_**" phía trước mang nghĩa tiêu cực. Sau khi gắn nhãn negation, bạn thậm chí có thể loại bỏ token "không" riêng lẻ (để tránh nó xuất hiện như một từ độc lập gây nhiễu) vì thông tin phủ định đã được gắn vào từ kế tiếp. Xử lý phủ định đúng cách sẽ cải thiện độ chính xác khi gặp những câu như "không hài lòng", "chưa tốt lắm".
- Cân bằng dữ liệu huấn luyện:** Kiểm tra tỷ lệ các nhãn (positive/negative) trong tập dữ liệu. Dữ liệu review sản phẩm thường lệch về tích cực (vì người hài lòng có xu hướng đánh giá hơn). Nếu tập huấn luyện **mất cân bằng** (ví dụ 90% tích cực, 10% tiêu cực), mô hình SVM có thể bị bias về phía nhãn đa số, dẫn đến việc dự đoán kém nhãn thiểu số (như tiêu cực) [thorapham.github.io](https://github.com/thorapham) . Trong trường hợp này, hãy cân nhắc **tăng trọng số** cho lớp ít (tham số `class_weight='balanced'` trong SVM) hoặc **làm tăng dữ liệu** (bằng cách nhân bản dữ liệu tiêu cực, hoặc tạo dữ liệu mới từ dữ liệu hiện có) để mô hình học được đầy đủ cả hai lớp.
- Điều chỉnh tham số SVM:** SVM có tham số quan trọng là C (điều chỉnh mức độ regularization). Hãy dùng cross-validation để tìm giá trị C tối ưu cho dữ liệu của bạn. C nhỏ làm mô hình regular hóa mạnh (giảm overfit nhưng có thể underfit), C lớn cho phép mô hình fit dữ liệu chặt hơn. Vì dữ liệu text thường nhiều chiều nhưng mỗi điểm dữ liệu (một review) khá thưa, SVM tuyến tính với C phù hợp thường cho kết quả tốt.

- **Ưu tiên SVM tuyến tính:** Đối với bài toán nhiều đặc trưng (số chiều từ vựng có thể lên đến hàng nghìn hoặc hơn) và dữ liệu thưa, **mô hình tuyến tính** (như SVM kernel linear hoặc Logistic Regression) thường hiệu quả và nhanh hơn so với các mô hình phi tuyến. Linear SVM có thể xử lý hàng triệu đặc trưng một cách nhanh chóng và cho độ chính xác cao trên dữ liệu dạng bag-of-words viblo.asia . Ngược lại, các kernel phi tuyến (RBF, poly) ít được sử dụng cho dữ liệu text vì dữ liệu đã rất phi tuyến nhưng lại thừa chiều; dùng kernel phi tuyến sẽ cực kỳ chậm và dễ overfit trừ khi dữ liệu rất nhỏ. Vì vậy, hãy sử dụng `kernel='linear'` cho SVM (hoặc dùng `LinearSVC`), bạn sẽ tận dụng được tính hiệu quả trên dữ liệu thưa.
- **Các thủ thuật khác:** Bạn có thể thử một số cách cải tiến nhỏ khác như:
 - **Loại đặc trưng ít thông tin:** Sau khi TF-IDF, có thể loại bỏ thêm những từ có trọng số IDF quá thấp (xuất hiện hầu như khắp mọi nơi) vì chúng không đóng góp phân loại. Cũng có thể loại bỏ từ quá dài hoặc quá ngắn nếu chúng không liên quan (ví dụ từ 1-2 ký tự thường là từ rất chung).
 - **Thêm đặc trưng thủ công:** Trong một số trường hợp, thêm các đặc trưng như **độ dài review**, **số lượng từ cảm xúc tích cực/tiêu cực** (dựa trên một lexicon) hoặc **số dấu chấm than**, **emoji tích cực/tiêu cực** trong câu có thể giúp mô hình. Tuy nhiên, với dữ liệu ngắn, đóng góp này thường không lớn.
 - **Ensemble nhiều mô hình:** Nếu có điều kiện, có thể kết hợp SVM với các mô hình khác (như Naive Bayes, Logistic) theo kiểu ensemble để tận dụng ưu điểm của từng mô hình, nhưng điều này phức tạp hơn và nằm ngoài phạm vi tiền xử lý.

Cuối cùng, sau khi hoàn tất các bước trên, bạn sẽ tiến hành huấn luyện mô hình SVM trên tập train đã được vector hóa và đánh giá trên tập test. Hãy đảm bảo quy trình **tiền xử lý áp dụng nhất quán** cho cả dữ liệu huấn luyện và dữ liệu mới khi dự đoán. Với việc tiền xử lý đúng cách và các mẹo tối ưu cho tiếng Việt như trên, mô hình SVM của bạn sẽ có nền tảng dữ liệu tốt để đạt kết quả phân loại cảm xúc chính xác và tin cậy.

Tài liệu tham khảo:

1. Huy, N.T. (2020). *Nhận diện cảm xúc trong văn bản tiếng Việt bằng mô hình học máy*. (Trích các bước tiền xử lý dữ liệu) ptithcm.edu.vn ptithcm.edu.vn ptithcm.edu.vn ptithcm.edu.vn .

2. Pham, T.H. et al. (2017). *A Study on Diacritic Restoration Problem in Vietnamese Text*. (Tầm quan trọng của dấu trong tiếng Việt) [arxiv.org](#) .
3. Thor Pham (2019). *Sentiment Analysis sử dụng TF-IDF áp dụng cho ngôn ngữ Tiếng Việt*. (Code mẫu tiền xử lý và vector hóa TF-IDF) [thorphan.github.io](#) [thorphan.github.io](#) .
4. Ho H.T. (2023). *Kỹ thuật làm tăng dữ liệu trong phân tích cảm xúc tiếng Việt*. (Xử lý từ phủ định trong tiền xử lý) [researchgate.net](#) .
5. Viblo AI blog (2020). *Linear models for sentiment analysis*. (Ưu tiên mô hình tuyến tính cho dữ liệu nhiều chiều thưa) [viblo.asia](#) .
6. Tài liệu AWS & Streetcode (2019). *Tổng quan phân tích cảm xúc đa ngôn ngữ*. (Khái niệm stopwords và tách từ trong NLP) [slideshare.net](#) .
7. Wikipedia Tiếng Việt. *Danh sách từ viết tắt tiếng lóng của giới trẻ*. (Tham khảo các từ viết tắt và tiếng lóng thông dụng trong tiếng Việt).

Citations



https://ptithcm.edu.vn/wp-content/uploads/2023/07/2020_HTTT_NguyenThanhHuy_LV.pdf



https://ptithcm.edu.vn/wp-content/uploads/2023/07/2020_HTTT_NguyenThanhHuy_TTLV.pdf



<https://arxiv.org/pdf/1709.07104>



https://ptithcm.edu.vn/wp-content/uploads/2023/07/2020_HTTT_NguyenThanhHuy_LV.pdf

Phân tích cảm xúc trong tiếng việt bằng phương pháp máy học.pdf


<https://www.slideshare.net/man2017/phn-tch-cm-xc-trong-ting-vit-bng-phng-php-my-hcpdf>

R⁶


https://www.researchgate.net/profile/Thien-Ho-Huong/publication/367706663_Ky_thuat_lam_tang_du_lieu_trong_phan_tich_cam_xuc_tren_ngon_n...




https://ptithcm.edu.vn/wp-content/uploads/2023/07/2020_HTTT_NguyenThanhHuy_LV.pdf

 **Sentiment Analysis sử dụng Tf-Idf áp dụng cho ngôn ngữ tiếng việt | ThorPham**


<https://thorpham.github.io/blog/Sentiment-Analysis-s%E1%BB%AD-d%E1%BB%A5ng-Tf-Idf>

 **Sentiment Analysis sử dụng Tf-Idf áp dụng cho ngôn ngữ tiếng việt | ThorPham**

<https://thorpham.github.io/blog/Sentiment-Analysis-s%E1%BB%AD-d%E1%BB%A5ng-Tf-Idf>

 **Sentiment Analysis sử dụng Tf-Idf áp dụng cho ngôn ngữ tiếng việt | ThorPham**

<https://thorpham.github.io/blog/Sentiment-Analysis-s%E1%BB%AD-d%E1%BB%A5ng-Tf-Idf>

 **Sentiment Analysis sử dụng Tf-Idf áp dụng cho ngôn ngữ tiếng việt | ThorPham**

<https://thorpham.github.io/blog/Sentiment-Analysis-s%E1%BB%AD-d%E1%BB%A5ng-Tf-Idf>

 **Bài 3 - Linear models for sentiment analysis**

<https://viblo.asia/p/bai-3-linear-models-for-sentiment-analysis-Eb85oxv8K2G>

 **Sentiment Analysis sử dụng Tf-Idf áp dụng cho ngôn ngữ tiếng việt | ThorPham**


<https://thorpham.github.io/blog/Sentiment-Analysis-s%E1%BB%AD-d%E1%BB%A5ng-Tf-Idf>

 **Bài 3 - Linear models for sentiment analysis**

<https://viblo.asia/p/bai-3-linear-models-for-sentiment-analysis-Eb85oxv8K2G>



https://ptithcm.edu.vn/wp-content/uploads/2023/07/2020_HTTT_NguyenThanhHuy_TTLV.pdf

 **Sentiment Analysis sử dụng Tf-Idf áp dụng cho ngôn ngữ tiếng việt | ThorPham**

<https://thorpham.github.io/blog/Sentiment-Analysis-s%E1%BB%AD-d%E1%BB%A5ng-Tf-Idf>

All Sources



ptithcm.edu



arxiv



slideshare



researchgate



thorpham.github



viblo