BÁO CÁO THỰC TẬP

Sentiment Analysis Vietnamese

LƯƠNG KHẮC MẠNH

Viện Toán ứng dụng và Tin học
Lớp Hệ thống thông tin quản lý K62

HÀ NỘI, 12/2020

Lời cảm ơn

Em xin chân thành cảm ơn các tới tất cả mọi người đã giúp đỡ em trong quá trình thực tập, nghiên cứu và học hỏi trong để hoàn thành đợt thực tập kỳ 2020.1 này. Đặc biệt em xin gửi lời cảm ơn sâu sắc nhất đến anh Nguyễn Thế Hùng - Phó phòng R&D tại công ty iCOMM đã trực tiếp hướng dẫn và giúp đỡ em trong quá trình thực tập tại công ty.

Em cũng xin chân thành cảm ơn sự giúp đỡ của ban lãnh đạo, anh chị và toàn thể thành viên của Công ty Cổ phần Truyền thông và Công nghệ iCOMM Việt Nam đã tạo điều kiện cho em thực tập và cung cấp tài liệu và dữ liệu cho em trong quá trình thực tập.

Tóm tắt nội dung thực tập

Thực tập tại bộ phần R&D của Công ty Cổ phần Truyền thông và Công nghệ iCOMM Việt Nam và công việc được giao là nghiên cứu đề tài Sentiment Analysis Vietnamese. Bằng cách sử dụng TF-IDF kết hợp với SVM kernel = 'linear' Model để xây dựng mô hình phân loại cảm xúc văn bản. Kết quả đạt được là hiểu bài toán và chạy được mô hình. Tuy nhiên vẫn còn hạn chế ở phần dữ liệu phân tán chưa tập chung vào một lĩnh vực, chưa thử nhiều model để cải thiện độ chính xác cho bài toán.

Mục lục

Lời nói đầu

Chúng ta đang sống trong kỷ nguyên số, đặc biệt những năm gần đây nổi lên mạng xã hội, với hàng triệu người dùng trên thế giới, lượng thông tin nội dung được người dùng tạo ra hằng ngày cực kỳ lớn, với đa dạng các hình thức như dòng trạng thái, hình ảnh, video. Mạng xã hội có những đặc điểm là: thông tin do người dùng tạo ra, mang tính cá nhân cho nên chất lượng nội dung hay tính đúng đắn, xác thực là tương đối; một thông tin mới được tạo lại có sức lan tỏa nhanh đến đông đảo các người dùng khác, so với các kênh thông tin truyền thống như truyền hình, truyền thanh, báo chí, diễn đàn, blog...

Điều này đặt ra cho các doanh nghiệp lớn giải quyết bài toán quản trị thương hiệu doanh nghiệp, quản trị thương hiệu sản phẩm trước các dư luận không tốt trên mạng xã hội rất khó khăn, cả về nguồn xuất phát thông tin, cả về khối lượng thông tin cần xử lý. Chưa kể việc các đối thủ cạnh tranh trên thương trường lợi dụng mạng xã hội để cố ý tạo các thông tin bất lợi cho nhau. Điều này đòi hỏi phải có một công cụ hỗ trợ đắc lực, mà chỉ có áp dụng công nghệ thông tin mới giải quyết được, chứ không lực lượng con người nào có thể làm xuể.

Các doanh nghiệp lớn của Việt Nam như: Vinamilk, các hãng hàng không như VietNam AirLines, Vietjest Art, Jetstar Pacific Airlines... hiện nay đã đặt hàng các doanh nghiệp công nghệ thông tin Việt Nam giải quyết vấn đề này. Giải pháp công nghệ hiện nay được gọi là "lắng nghe mạng xã hội", tức là các doanh nghiệp CNTT mua các dữ liệu thời gian thực (real time) từ các công ty mạng xã hội về để xử lý các thông tin liên quan đến doanh nghiệp hay các sản phẩm mà doanh nghiệp đó kinh doanh, nhằm phát hiện và ngăn chặn sớm sự lan rộng các thông tin bất lợi trên mạng xã hội, có hình thức đính chính phản hồi đến các khách hàng của mình, đồng thời thương lượng, ngăn chặn tận gốc những người tạo ra các nội dung đó.

Điều cốt yếu của giải pháp này chính là phân tích tình cảm của các dòng trạng thái trên mạng xã hội nhằm lọc ra các thông tin bất lợi để xử lý.

Phân tích tình cảm (Sentiment analysis) là nhằm phát hiện ra thái độ mang tính lâu dài, màu sắc tình cảm, khuynh hướng niềm tin vào các đối tượng hay người nào đó. Phát biểu theo góc nhìn của học máy thì nó là bài toán phân lớp cảm xúc dựa trên văn bản ngôn ngữ tự nhiên. Đầu vào của bài toán là một câu hay một đoạn văn bản, còn đầu ra là các giá trị xác suất của N lớp cảm xúc mà ta cần xác định. Nó được ứng dụng trong hàng loạt các vấn đề như: phân tích thị trường, dự đoán biến động giá cổ phiếu, quản trị thương hiệu doanh nghiệp, thương hiệu sản phẩm, khảo sát ý kiến xã hội học, khảo sát sự hài lòng của khách hàng, phân tích trạng thái tâm lý con người...

Chương 1

Giới thiệu về công ty iCOMM

1.1 Giới thiệu chung

Tên công ty: Công ty Cổ phần Truyền thông và Công nghệ iCOMM Việt Nam

Địa chỉ công ty: Tầng 5 CT1, Tòa Nhà C14 Bắc Hà, Tố Hữu, Trung Văn, Nam Từ Liêm, Hà Nội

Website: http://icomm.vn.

Giới thiệu về công ty: Được thành lập vào năm 2014 với nền tảng là các chuyên gia hệ thống xử lý dữ liệu lớn từ các công ty hàng đầu về công nghệ thông tin như: FPT, VTC, VC Corp, Naiscorp,... Công Ty Cổ phần Truyền thông và Công nghệ iCOMM Việt Nam ra đời với mục tiêu tập trung xây dựng các hệ thống thu thập và phân tích dữ liệu quy mô lớn trên Internet và di động nhằm phục vụ nhu cầu ngày càng tăng của các đơn vị quản lý nhà nước về thông tin truyền thông cũng như các doanh nghiệp có nhu cầu tiếp cận và hiểu người dùng thông qua các kênh truyền thông trực tuyến. Đến nay, bằng nỗ lực và quyết tâm không ngừng của công ty cùng với sự tin tưởng của quý khách hàng, iCOMM đã có cơ hội triển khai thành công nhiều dự án CNTT lớn tại Hà Nội, TP.HCM và một số tỉnh thành trong cả nước. Các dự án iCOMM đảm nhiệm đều đòi hỏi năng lực kỹ thuật và tài chính nhất định, cùng sự đồng bộ hạ tầng và giải pháp trong toàn hệ thống.

1.2 Công nghệ và giải pháp

Sản phẩm công nghệ và phải pháp được cung cấp bởi iCOMM là:

- Hệ thống phân tích xử lý dữ liệu lớn: Splunk
- Hệ thống chăm sóc khách hàng hàng ngày qua Internet: iCOMM SocialCare
- Giải pháp quản trị cho Cơ quan & Doanh nghiệp
- Giải pháp quản lý và giám sát tang tin điện tử: mNews
- Giải pháp phát hiện và giám sát game không phép: mGame
- Hệ thống xử lý khủng hoảng Mạng xã hội
- Giải pháp cổng thanh toán, ví điện tử tuw vấn và cung cấp giải pháp bảo mật
- Bảo mật website và hệ thống thông tin: VNCS

- Web Monitoring quảng cáo và truyền thông, Quảng cáo trực tuyến trên di động và Internet
- Xuất bản báo điện tử: Congnghe.vn
- Nghiên cứu thị trường và đối thủ: SocialBeat.vn
- Cung cấp dịch vụ và giải pháp trên nền tảng mobile
- Hệ thống quảng cáo theo ngữ cảnh trên điện thoại thông minh FlexDistro
- Ứng dụng và trò chơi di động: trên 20 ứng dụng với hàng triệu lượt tải trên Google Play
- Dịch vụ VAS.

Chương 2

Nội dung thực tập

2.1 Giới thiệu chung đề tài

Đề tài: Sentiment analysis Vietnamese

Phân tích cảm xúc (Sentiment analysis) là nhằm phát hiện ra thái độ mang tính lâu dài, màu sắc tình cảm, khuynh hướng niềm tin vào các đối tượng hay người nào đó.

Các vấn đề xung quanh việc phân tích cảm xúc:

- Nguồn gốc của cảm xúc
- Mục tiêu của cảm xúc
- Các loại cảm xúc: thích, yêu, ghét, đánh giá, mong mỏi.
- Về mức độ cảm xúc: tích cực, tiêu cực, trung tính.
- Văn bản hàm chứa cảm xúc: một câu hoặc một đoạn văn bản.

Bài toán phân tích cảm xúc thuộc dạng bài toán phân tích ngữ nghĩa văn bản. Vì vậy, ta cần phải xây dựng một mô hình để hiểu được ý nghĩa của câu văn, đoạn văn để quyết định xem câu văn đó hoặc đoạn văn đó mang màu sắc cảm xúc chủ đạo nào.

Phát biểu theo góc nhìn của máy học (Machine Learning) thì phân tích cảm xúc là bài toán phân lớp cảm xúc dựa trên văn bản ngôn ngữ tự nhiên. Đầu vào của bài toán là một câu hay một đoạn văn bản, còn đầu ra là các giá trị xác suất (điểm số) của N lớp cảm xúc mà ta cần xác định.

Trong loại bài toán phân tích cảm xúc được phân thành các bài toán có độ khó khác nhau như sau:

- Đơn giản: Phân tích cảm xúc (thái độ) trong văn bản thành hai lớp: tích cực (positive) và tiêu cực (negative)
- Phức tạp hơn: Xếp hạng cảm xúc (thái độ) trong văn bản từ 1 đến 5.
- Khó: Phát hiện mục tiêu, nguồn gốc của cảm xúc (thái độ) hoặc các loại cảm xúc (thái độ) phức tạp.

Hiện tại thì cộng đồng khoa học mới chỉ giải quyết tốt bài toán phân tích cảm xúc ở cấp độ đơn giản, tức là phân tích cảm xúc với lớn hơn hoặc bằng 2 lớp cảm xúc tiêu cực và tích cực. Vì vậy, bài toán phân tích cảm xúc trong Tiếng Việt trình bày trong bài viết này là kết quả

của nghiên cứu phân tích cảm xúc văn bản Tiếng Việt với 3 lớp cảm xúc là: tiêu cực (negative), trung tính (neutral) và tích cực (positive).

Đầu vào của mô hình xử lý Sentiment Analysis Vietnamese (SAV) là một đoạn văn Tiếng Việt, đầu ra là 2 giá trị mà đoạn văn đầu vào thuộc về lớp cảm xúc: tiêu cực (negative) hay tích cực (positive).

2.2 Cơ sở lý thuyết

2.2.1 Naive Bayes Classifier

Xét bài toán classification với **C** classes **1,2,...,C**. Giải sử có một điểm dữ liệu $\mathbf{x} \in \mathbb{R}^d$. Hãy tính xác suất để điểm dữ liệu rơi vào class **c**. Nói cách khác, hãy tính:

$$p(y = c|\mathbf{x}) \tag{2.1}$$

hoặc viết gọn thành $p(c|\mathbf{x})$

Tức tính xác suất để đầu ra là class **c** biết rằng đầu vào là vector **x**.

Biểu thức này, nếu tính được, sẽ giúp chúng ta xác định được xác suất để điểm dữ liệu rơi vào mỗi class. Từ đó có thể giúp xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất:

$$c = \arg \max_{c \in \{1,\dots,C\}} p(c \mid \mathbf{x}) \tag{2.2}$$

Biểu thức (2.2) thường khó được tính trực tiếp. Thay vào đó, quy tắc Bayes thường được sử dụng:

$$c = \arg \max_c p(c \mid \mathbf{x}) \tag{2.3}$$

$$= \arg \max_c \frac{p(\mathbf{x} \mid c)p(c)}{p(\mathbf{x})} \tag{2.4}$$

$$= \arg \max_c p(\mathbf{x} \mid c)p(c) \tag{2.5}$$

Từ biểu thức (2.3) sang (2.4) vì quy tắc Bayes. Từ biểu thức (2.4) sang biểu thức (2.5) vì mẫu số $p(\mathbf{x})$ không phụ thuộc vào c.

Tiếp tục xét biểu thức (2.5), $p(\mathbf{c})$ có thể được hiểu là xác suất để một điểm rơi vào class **c**. Giá trị này có thể được tính bằng MLE, tức tỉ lệ số điểm dữ liệu trong tập training rơi vào class này chia cho tổng số lượng dữ liệu trong tập training; hoặc cũng có thể được đánh giá bằng MAP estimation. Trường hợp thứ nhất thường được sử dụng nhiều hơn.

Thành phần còn lại $p(\mathbf{x}|c)$, tức phân phối của các điểm dữ liệu trong class **c**, thường rất khó tính toán vì **x** là một biến ngẫu nhiên nhiều chiều, cần rất rất nhiều dữ liệu training để có thể xây dựng được phân phối đó. Để giúp cho việc tính toán được đơn giản, người ta thường giả sử một cách đơn giản nhất rằng các thành phần của biến ngẫu nhiên **x** là độc lập với nhau, nếu biết **c** (given c). Tức là:

$$p(\mathbf{x} \mid c) = p\left(x_1, x_2, \dots, x_d \mid c\right) = \prod_{i=1}^{d} p\left(x_i \mid c\right) \tag{2.6}$$

Giả thiết các chiều của dữ liệu độc lập với nhau, nếu biết **c**, là quá chặt và ít khi tìm được dữ liệu mà các thành phần hoàn toàn độc lập với nhau. Tuy nhiên, giả thiết ngây ngô này lại mang lại những kết quả tốt bất ngờ. Giả thiết về sự độc lập của các chiều dữ liệu này được gọi là Naive Bayes. Cách xác định class của dữ liệu dựa trên giả thiết này có tên là Naive Bayes Classifier (NBC).

NBC, nhờ vào tính đơn giản một cách *ngây thơ*, có tốc độ training và test rất nhanh. Việc này giúp nó mang lại hiệu quả cao trong các bài toán large-scale.

Ở bước training, các phân phối $p(c)$ và $p(x_i|c), i = 1, 2, 3, ..., d$ sẽ được xác định dựa vào training data. Việc xác định các giá trị này có thể dựa vào Maximum Likelihood Estimation hoặc Maximum A Posteriori.

Ở bước test, với một điểm dữ liệu mới **x**, class của nó sẽ được xác đinh bởi:

$$c = \arg \max_{c \in \{1,...,C\}} p(c) \prod_{i=1}^{d} p\left(x_i \mid c\right) \tag{2.7}$$

Khi d lớn và các xác suất nhỏ, biểu thức ở vế phải của (2.7) sẽ là một số rất nhỏ, khi tính toán có thể gặp sai số. Để giải quyết việc này, (2.7) thường được viết lại dưới dạng tương đương bằng cách lấy log của vế phải:

$$c = \arg \max_{c \in \{1,...,C\}} = \log(p(c)) + \sum_{i=1}^{d} \log\left(p\left(x_i \mid c\right)\right) \tag{2.8}$$

Việc này không ảnh hưởng tới kết quả vì log là một hàm đồng biến trên tập các số dương.

Mặc dù giả thiết mà Naive Bayes Classifiers sử dụng là quá phi thực tế, chúng vẫn hoạt động khá hiệu quả trong nhiều bài toán thực tế, đặc biệt là trong các bài toán phân loại văn bản, ví dụ như lọc tin nhắn rác hay lọc email spam. Trong phần sau của bài viết, chúng ta cùng xây dựng một bộ lọc email spam tiếng Anh đơn giản.

Cả việc training và test của NBC là cực kỳ nhanh khi so với các phương pháp classification phức tạp khác. Việc giả sử các thành phần trong dữ liệu là độc lập với nhau, nếu biết class, khiến cho việc tính toán mỗi phân phối $p(x_i|c)$ trở nên cực kỳ nhanh.

Mỗi giá trị $p(c), c = 1, 2, \ldots, C$ có thể được xác định như là tần suất xuất hiện của class c trong training data.

Việc tính toán $p(x_i|c)$ phụ thuộc vào loại dữ liệu. Có ba loại được sử dụng phổ biến là: Gaussian Naive Bayes, Multinomial Naive Bayes, và Bernoulli Naive.

Multinomial Naive Bayes

Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó.

Khi đó, $p(x_i|c)$ tỉ lệ với tần suất từ thứ i (hay feature thứ i cho trường hợp tổng quát) xuất

hiện trong các văn bản của class c. Giá trị này có thể được tính bằng cách:

$$\lambda_{ci} = p\left(x_i \mid c\right) = \frac{N_{ci}}{N_c} \tag{2.9}$$

Trong đó:

- N_{ci} là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c.
- N_c là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c. Có thể suy ra rằng $\sum_{i=1}^{d} N_{ci}$ từ đó $\sum_{i=1}^{d} \lambda_{ci} = 1$.

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức (2.9) sẽ bằng 0, điều này dẫn đến vế phải của (2.7) bằng 0 bất kể các giá trị còn lại có lớn thế nào. Việc này sẽ dẫn đến kết quả không chính xác.

Để giải quyết việc này, một kỹ thuật được gọi là Laplace smoothing được áp dụng:

$$\hat{\lambda}_{ci} = \frac{N_{ci} + \alpha}{N_c + d\alpha} \tag{2.10}$$

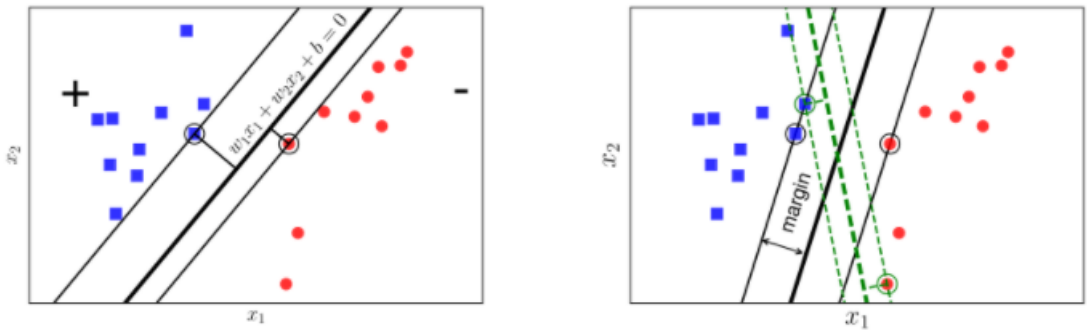
Với α là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với $d\alpha$ để đảm bảo tổng xác suất $\sum_{i=1}^{d} \hat{\lambda}_{ci} = 1$.

Như vậy, mỗi class c sẽ được mô tả bởi bộ các số dương có tổng bằng 1: $\hat{\lambda}_c = \left\{\hat{\lambda}_{c1}, \ldots, \hat{\lambda}_{cd}\right\}$.

2.2.2 Support Vector Machine

a) Support Vector Machine

Giới thiệu

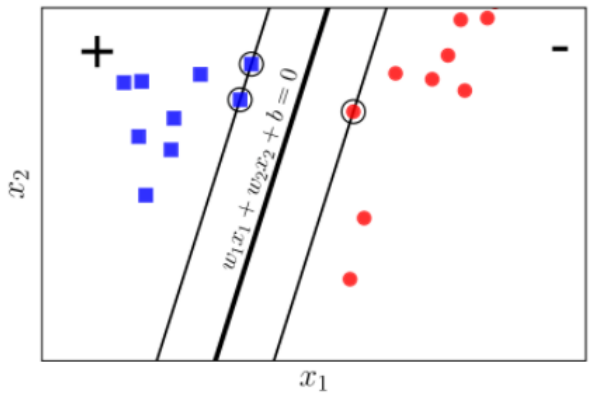


Hình 1: Margin của hai classes là bằng nhau và lớn nhất có thể

Bài toán tối ưu trong Support Vector Machine (SVM) chính là bài toán đi tìm đường phân chia sao cho margin là lớn nhất. Đây cũng là lý do vì sao SVM còn được gọi là Maximum Margin Classifier.

Xây dựng bài toán tối ưu cho SVM

Giả sử rằng các cặp dữ liệu của training set là $(x_1, y_1), (x_2, y_3), \ldots, (x_N, y_N)$ với vector $x_i \in \mathbb{R}^d$ 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).



Hình 2: Phân tích bài toán SVM

Giả sử rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộc class -1 và mặt $\mathbf{w^T x} + b = w_1x_1 + w_2x_2 + b = 0$ là mặt phân chia giữa hai classes (Hình 2). Hơn nữa, class 1 nằm về phía dương, class -1 nằm về phái âm của mặt phân chia. Nếu ngược lại, ta chỉ cần đổi dấu của $\mathbf{w}$ và b. Chú ý rằng chúng ta cần đi tìm các hệ số $\mathbf{w}$ 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:

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

Điều này có thể dễ nhận thấy vì theo giả sử ở trên, y_n luôn cùng dấu với phía của x_n. Từ đó suy ra y_n cùng dấu với $(\mathbf{w^T x_n} + b)$, và tử số luôn là 1 số không âm.

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

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

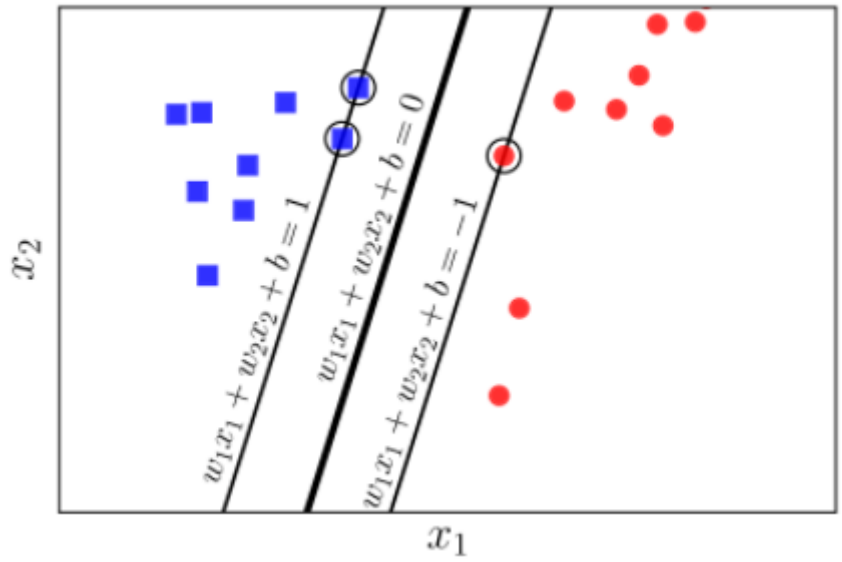
Bài toán tối ưu trong SVM chính là bài toán tìm $\mathbf{w}$ và b sao cho margin này đạt giá trị lớn nhất:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right)}{\|\mathbf{w}\|_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|_2} \min_n y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \right\} \quad (2.11)$$

Nhận xét quan trọng nhất là nếu ta thay vector hệ số $\mathbf{w}$ bởi $k\mathbf{w}$ 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:

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

với những điểm gần mặt phân chia nhất như Hình 3 dưới đây:



Hình 3: Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn

Như vậy, với mọi n, ta có:

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

Vậy bài toán (2.11) có thể đưa về bài toán tối ưu có ràng buộc sau đây:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2} \tag{2.12}$$

$$\text{thỏa mãn: } y_n(\mathbf{w^T x_n} + b) \geq 1,\ \forall n = 1, 2, \ldots, N$$

Bằng 1 biến đổi đơn giản, ta có thể đưa bài toán này về dạng dưới đây:

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 \tag{2.13}$$

$$\text{thỏa mãn: } 1 - y_n(\mathbf{w^T x_n} + b) \leq 0,\ \forall n = 1, 2, \ldots, N$$

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.

Người ta thường giải bài toán đối ngẫu của bài toán này. Thứ nhất, bài toán đối ngẫu có những tính chất thú vị hơn khiến nó được giải hiệu quả hơn. Thứ hai, trong quá trình xây dựng bài toán đối ngẫu, người ta thấy rằng SVM có thể được áp dụng cho những bài toán mà dữ liệu không linearly separable, tức các đường phân chia không phải là một mặt phẳng mà có thể là các mặt có hình thù phức tạp hơn.

Xác định class cho một điểm dữ liệu mới: Sau khi tìm được mặt phân cách $\mathbf{w^T x} + b = 0$, class của bất kỳ một điểm nào sẽ được xác định đơn giản bằng cách:

$$\text{class}(\mathbf{x}) = \text{sgn}\left(\mathbf{w}^T\mathbf{x} + b\right)$$

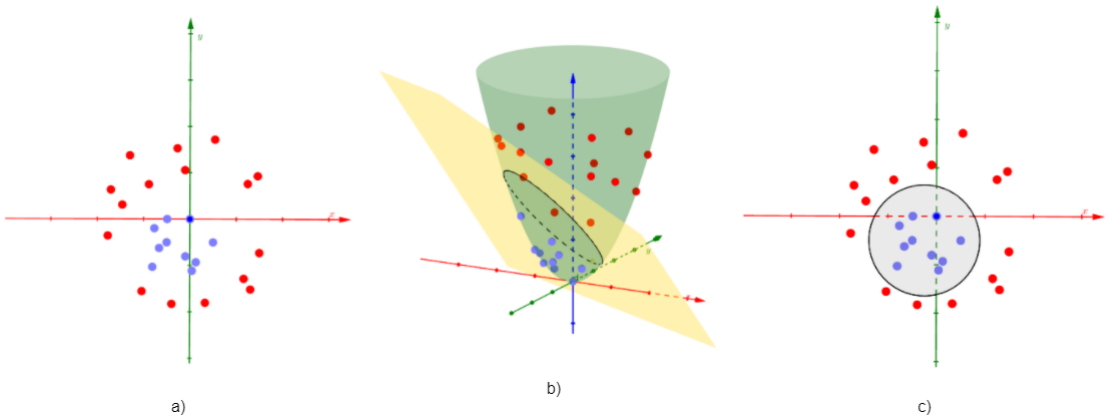
Trong đó hàm **sgn** là hàm xác định dấu, nhận giá trị 1 nếu đối số không âm và -1 nếu ngược lại.

b) Kernel Support Vector Machine

Giới thiệu

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 phân biệt tuyến tính được biến sang không gian mới. 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:



Ví dụ về Kernel SVM. a) Dữ liệu của hai classes là không phân biệt tuyến tính trong không gian hai chiều. b) Nếu coi thêm chiều thứ ba là một hàm số của hai 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 . Mặt phẳng màu vàng là mặt phân chia, có thể tìm được bởi Hard/Soft Margin SVM. c) Giao điểm của mặt phẳng 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 đã tìm được đường phân chia hai classses.

Nói một cách ngắn gọn, Kernel SVM là việc đi tìm một hàm số biến đổi dữ liệu x từ không gian feature ban đầu thành dữ liệu trong một không gian mới bằng hàm số $\Phi(x)$. Trong ví dụ này, hàm $\Phi()$ đơn giản là giới thiệu thêm một chiều dữ liệu mới (một feature mới) là một hàm số của các features đã biết. Hàm số này cần thỏa mãn mục đích của chúng ta: trong không gian mới, dữ liệu giữa hai classes là phân biệt tuyến tính hoặc gần như phần biệt tuyến tính. Khi đó, ta có thể dùng các bộ phân lớp tuyến tính thông thường như PLA, Logistic Regression, hay Hard/Soft Margin SVM.

Các hàm $\Phi()$ thường tạo ra dữ liệu mới có số chiều cao hơn số chiều của dữ liệu ban đầu, thậm chí là vô hạn chiều. Nếu tính toán các hàm này trực tiếp, chắc chắn chúng ta sẽ gặp các vấn đề về bộ nhớ và hiệu năng tính toán. Có một cách tiếp cận là sử dụng các kernel functions mô tả quan hệ giữa hai điểm dữ liệu bất kỳ trong không gian mới, thay vì đi tính toán trực tiếp từng điểm dữ liệu trong không gian mới. Kỹ thuật này được xây dựng dựa trên quan sát về bài toán đối ngẫu của SVM.

Cơ sở toán học

Nhắc lại bài toán đối ngẫu trong Soft SVM (SVM lề mềm) cho dữ liệu gần tách biệt

tuyến tính:

$$\lambda = \arg\max_{\lambda} \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m \tag{2.14}$$

$$\text{Thỏa mãn: } \sum_{n=1}^{N} \lambda_n y_n = 0$$
$$0 \leq \lambda_n \leq C, \forall n = 1, 2, \ldots, N$$

Trong đó:

- N: số cặp điểm dữ liệu trong tập training.
- X_n: feature vector của dữ liệu thứ n trong tập training.
- y_n: nhãn của dữ liệu thứ n, bằng 1 hoặc -1.
- λ_n: nhân tử Lagrange ứng với điểm dữ liệu thứ n.
- C: hằng số dương giúp cân đối độ lớn của margin và sự hy sinh của các điểm nằm trong vùng không an toàn. Khi $C = \infty$ hoặc rất lớn, Soft Margin SVM trở thành Hard Margin SVM.

Sau khi giải được λ cho bài toán (2.14), nhãn của một điểm dữ liệu mới sẽ được xác định bởi dấu của biểu thức:

$$\sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m^T \mathbf{x} + \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m \mathbf{x}_m^T \mathbf{x}_n \right) \tag{2.15}$$

Trong đó:

- $\mathcal{M} = \{n : 0 < \lambda_n < C\}$ là tập hợp những điểm nằm trên margin.
- $\mathcal{S} = n : 0 < \lambda_n$ là tập hợp các điểm support.
- $N_{\mathcal{M}}$ là số phần tử của $\mathcal{M}$.

Với dữ liệu thực tế, rất khó để có dữ liệu gần phân biệt tuyến tính, vì vậy nghiệm của bài toán (2.14) có thể không thực sự tạo ra một bộ phân lớp tốt. Giả sử rằng ta có thể tìm được hàm số $\Phi()$ sao cho sau khi được biến đổi sang không gian mới, mỗi điểm dữ liệu x trở thành $\Phi(x)$, và trong không gian mới này, dữ liệu trở nên gần phân biệt tuyến tính. Lúc này, hy vọng rằng nghiệm của bài toán Soft Margin SVM sẽ cho chúng ta một bộ phân lớp tốt hơn.

Trong không gian mới, bài toán (2.14) trở thành:

$$\lambda = \arg\max_{\lambda} \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m \Phi\left(\mathbf{x}_n\right)^T \Phi\left(\mathbf{x}_m\right) \tag{2.16}$$

$$\text{Thỏa mãn: } \sum_{n}^{N} \lambda_n y_n = 0$$
$$0 \leq \lambda_n \leq C, \forall n = 1, 2, \ldots, N$$

và nhãn của một điểm dữ liệu mới được xác định bởi các dấu của biểu thức:

$$\mathbf{w}^T \Phi(\mathbf{x}) + b = \sum_{m \in \mathcal{S}} \lambda_m y_m \Phi\left(\mathbf{x}_m\right)^T \Phi(\mathbf{x}) + \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m \Phi\left(\mathbf{x}_m\right)^T \Phi\left(\mathbf{x}_n\right) \right) \tag{2.17}$$

Như đã nói ở trên, việc tính toán trực tiếp $\Phi(x)$ cho mỗi điểm dữ liệu có thể sẽ tốn rất nhiều bộ nhớ và thời gian vì số chiều của $\Phi(x)$ thường là rất lớn, có thể là vô hạn! Thêm nữa, để tìm nhãn của một điểm dữ liệu mới x, ta lại phải tìm biến đổi của nó $\Phi(x)$ trong không gian mới rồi lấy tích vô hướng của nó với tất cả các $\Phi(x_m)$ với m trong tập hợp support. Để tránh việc này, ta quan sát thấy một điều thú vị sau đây.

Trong bài toán (2.16) và biểu thức (2.17), chúng ta không cần tính trực tiếp $\Phi(x)$ cho mọi điểm dữ liệu. Chúng ta chỉ cần tính được $\Phi(x)^T\Phi(z)$ dựa trên hai điểm dữ liệu x, z bất kỳ! Kỹ thuật này còn được gọi là **kernel trick**. Những phương pháp dựa trên kỹ thuật này, tức thay vì trực tiếp tính tọa độ của một điểm trong không gian mới, ta đi tính tích vô hướng giữa hai điểm trong không gian mới, được gọi chung là **kernel method**.

Lúc này, bằng cách định nghĩa hàm kernel $k(x, z) = \Phi(x)^T\Phi(z)$, ta có thể viết lại bài toán (2.16) và biểu thức (2.17) như sau:

$$\lambda = \arg\max_{\lambda} \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m k\left(\mathbf{x}_n, \mathbf{x}_m\right) \tag{2.18}$$

$$\text{Thỏa mãn: } \textstyle\sum_n^N \lambda_n y_n = 0$$
$$0 \leq \lambda_n \leq C, \forall n = 1, 2, \ldots, N$$

và:

$$\sum_{m \in \mathcal{S}} \lambda_m y_m k\left(\mathbf{x}_m, \mathbf{x}\right) + \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m k\left(\mathbf{x}_m, \mathbf{x}_n\right) \right) \tag{2.19}$$

Tính chất của các hàm kernel

Không phải hàm $k()$ bất kỳ nào cũng được sử dụng. Các hàm kerrnel cần có các tính chất:

- Đối xứng: $k(x, z) = k(z, x)$. Điều này dễ nhận ra vì tích vô hướng của hai vector có tính đối xứng.
- Về lý thuyết, hàm kerrnel cần thỏa mãn điều kiện Mercer:

$$\sum_{n=1}^{N} \sum_{m=1}^{N} k\left(\mathbf{x}_m, \mathbf{x}_n\right) c_n c_m \geq 0, \forall c_i \in \mathbb{R}, i = 1, 2, \ldots, N \tag{2.20}$$

  Tính chất này để đảm bảo cho việc hàm mục tiêu của bài toán đối ngẫu (2.18) là lồi.
- Trong thực hành, có một vài hàm số $k()$ không thỏa mãn điều kiện Merrcer nhưng vẫn cho kết quả chấp nhận được. Những hàm số này vẫn được gọi là kernel.

Nếu một hàm kernel thỏa mãn điều kiện (2.20), xét $c_n = y_n\lambda_n$, ta sẽ có:

$$\lambda^T \mathbf{K} \lambda = \sum_{n=1}^{N} \sum_{m=1}^{N} k\left(\mathbf{x}_m, \mathbf{x}_n\right) y_n y_m \lambda_n \lambda_m \geq 0, \forall \lambda_n \tag{2.21}$$

với K là một ma trận đối xứng mà phần tử ở hàng thứ n cột thứ m của nó được định nghĩa bởi: $k_{nm} = y_n y_m k(\mathbf{x}_n, \mathbf{x}_m)$

Từ (2.21) ta suy ra K là một ma trận nửa xác định dương. Vì vậy, bài toán tối ưu (2.18) có ràng buộc là lồi và hàm mục tiêu là một hàm lồi (một quadratic form). Vì vậy chúng ta có thể giải quyết bài toán này một cách hiệu quả.

Một số hàm kernel thông dụng

1) Linear

   Đây là trường hợp đơn giản với kernel chính tích vô hướng của hai vector:

$$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$$

   Hàm số này thỏa mãn điều kiện (2.20).

2) Polynomial

$$k(\mathbf{x}, \mathbf{z}) = (r + \gamma\mathbf{x}^T\mathbf{z})^d$$

   Với d là một số dương để chỉ bậc của đa thức. d có thể không là số tự nhiên vì mục đích chính của ta không phải là bậc của đa thức mà là cách tính kernel. Polynomial kernel có thể dùng để mô tả hầu hết các đa thức có bậc không vượt quá d nếu d là một số tự nhiên.

3) Radial Basic Function

   Radial Basic Function (RBF) kernel hay Gaussian kernel được sử dụng nhiều nhất trong thực tế, và là lựa chọn mặc định trong sklearn. Nó được định nghĩa bởi:

$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma||\mathbf{x} - \mathbf{z}||_2^2), \quad \gamma > 0$$

4) Sigmoid

   Sigmoid function cũng được sử dụng làm kernel:

$$k(\mathbf{x}, \mathbf{z}) = \tanh(\gamma\mathbf{x}^T\mathbf{z} + r)$$

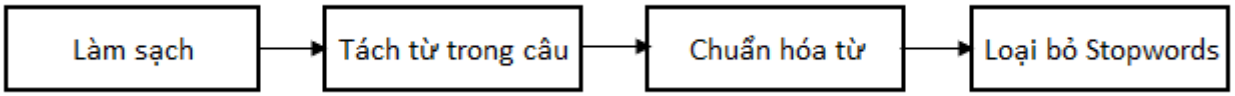
2.3 Phân tích và giải quyết bài toán

2.3.1 Thu thập và phân tích đặc điểm dữ liệu

Bộ dữ liệu gồm 4930 bản ghi gồm hai trường là Label_true và Sentence được cung cung cấp bởi anh Nguyễn Thế Hùng - Phó phòng R&D tại công ty iCOMM. Dữ liệu được cung cấp có 3 loại nhãn tại trường Label_true là -1, 0, 1 tương ứng với tiêu cực, trung tính và tích cực. Cụ thể đối với nhãn tiêu cực có 1515 bản ghi, trung tính có 2346 bản ghi, tích cực có 1069 bản ghi. Nội dung ở trường sentence được tổng hợp từ nhiều nguồn khác nhau.

2.3.2 Tiền xử lý dữ liệu

Quy trình tiền xử lý dữ liệu:



Làm sạch text
Loại bỏ các noise trong data. Đa phần noise là các thẻ HTML, JavaScript, và đương nhiên nếu cứ để noise để tiến hành xử lý sẽ dẫn đến kết quả xử lý không tốt.
Cụ thể đối với dữ liệu được cung cấp đã được làm sạch các thẻ, ta chỉ cần loại thêm các icon trong trường sentence. Cách đơn giản và dễ sử dụng nhất là sử dụng filter theo Regular Expression.

Tách từ
Trong tiếng Việt, dấu cách (space) không được sử dụng như 1 kí hiệu phân tách từ mà nó chỉ có ý nghĩa phân tách các âm tiết với nhau. Vì thế, để xử lý tiếng Việt, công đoạn phân tách từ (Word segmentation) là một trong những bài toán cơ bản và quan trọng bậc nhất.

Ví dụ: từ "cảm xúc" được tạo ra từ 2 âm tiết là "cảm" và "xúc", cả 2 âm tiết này đề có nghĩa riêng khi đứng độc lập, nhưng khi ghép lại sẽ mang một nghĩa khác. Vì đặc điểm này, bài toán tách từ trở thành một bài toán tiền đề cho các ứng dụng xử lý ngôn ngữ tự nhiên như phân loại văn bản, tóm tắt văn bản, máy dịch tự động...

Tách từ chính xác hay không là công việc rất quan trọng, nếu không chính xác rất có thể dẫn đến việc ý nghĩa của câu sai, ảnh hưởng đến tính chính xác của chương trình.

Cụ thể đối với dữ liệu được cung cấp đã được tách từ nên bước này có thể bỏ qua.

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

Ngoài ra trong một vài trường hợp, nếu ký tự số không mang lại lợi ích gì thì cũng sẽ tiến hành loại bỏ các ký tự số đó, nếu cứ để nguyên rất có thể các ký tự số sẽ trở thành noise, ảnh hưởng đển tính chính xác của model sau này.

Áp dụng vào bài toán, em đã chuyển dữ liệu tất cả cùng về chữ thường.

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

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

- Dùng từ điển
- Dựa theo tần suất xuất hiện của từ

Dùng từ điển
Cách đơn giản nhất, chúng ta tiến hành filter văn bản, loại bỏ những từ xuất hiện trong từ điển StopWords.

Ví dụ: "dù anh cứ đi, em cũng kệ" –(Filter StopWords)–> "anh đi, em kệ"

Dựa theo tần suất xuất hiện của từ
Với cách này, chúng ta tiến hành đếm số lần xuất hiện của từng từ trong data sau đó sẽ loại

bỏ những từ xuất hiện nhiều lần (cũng có thể là ít lần). Khoa học đã chứng minh những từ xuất hiện nhiều nhất thường là những từ không mang lại nhiều ý nghĩa chính vì thế chúng ta loại bỏ những từ như vậy.

Trong bài toán đang làm, em sử dụng cách thứ nhất là dùng từ điển, duyệt tất cả các từ trong sentence nếu thuộc trong từ điển StopWords sẽ bị loại bỏ.

2.3.3 Vector hóa dữ liệu

a) **Bag of Words**

Bag of Words (BoW) là một thuật toán hỗ trợ xử lý ngôn ngữ tự nhiên và mục đích của BoW là phân loại text hay văn bản. Ý tưởng của BoW là phân tích và phân nhóm dựa theo "Bag of Words"(corpus). Với test data mới, tiến hành tìm ra số lần xuất hiện từng từ của test data xuất hiện trong "bag". Tuy nhiên BoW vẫn tồn tại khuyết điểm, nên TF-IDF là phương pháp khắc phục. Ta ứng dụng BoW + TF-IDF vào bài toán đang xét.

Ví dụ cách hoạt động của Bag of Words, ta có hai câu sau:

"Hôm nay trời nắng"
"Hôm kia trời mưa"

Từ hai câu trên, tiến hành tạo từ điển các từ xuất hiện trong từng câu: {'hôm': 2, 'nay': 1, 'kia': 1, 'mưa': 1, 'nắng':1, 'trời': 2 } Dựa vào từ điển vừa tạo, ta tiến hành tạo vector lưu trữ số lần xuất hiện của các từ trong từ điển ứng với mỗi câu. Và do từ điển đang có 6 từ nên mỗi vector sẽ có 5 phần tử như sau:

	hôm	kia	mưa	nay	nắng	trời
Câu 1	1	0	0	1	1	1
Câu 2	1	1	1	0	0	1

b) **TF-IDF**

Trong hầu hết các ngôn ngữ, có một số từ có xu hướng xuất hiện thường xuyên như trong tiếng anh có "is", "the",... tương tự của tiếng việt có các từ như "là", "của", "cứ",... Chính vì vậy nếu chỉ xét theo tần số xuất hiện của từng từ thì việc phân loại văn bản có thể cho kết quả sai dẫn tỷ lệ chính xác sẽ thấp. Vậy giải pháp là gì?

Phương pháp phổ biến là sử dụng một phương pháp thống kê có tên là TF-IDF, giá trị của TF-IDF của một từ là một con số thu được qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập các văn bản.

Mà tại sao phương pháp lại có tên là TF-IDF.

Đầu tiên, TF(Term Frequency) là tần số xuất hiện của 1 từ trong 1 văn bản có cách

tính như sau:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Trong đó:

- $f_{t,d}$: số lần xuất hiện từ t trong văn bản d.
- $\sum_{t' \in d} f_{t',d}$: tổng số từ trong văn bản d

Tiếp theo là IDF (Inverse Document Frequency): Tần số nghịch của 1 từ trong tập văn bản (corpus).

Mục đích của việc tính IDF là giảm giá trị của các từ thường xuyên xuất hiện như "ís", "the",... Do các từ này không mang nhiều ý nghĩa trong việc phân loại văn bản.

$$idf(t,D) = \log \frac{|D|}{|d \in D : t \in d|}$$

Trong đó:

- $|D|$: tổng số văn bản trong tập D
- $|d \in D : t \in d|$: số văn bản có chứa từ t. Nếu từ đó không xuất hiện ở bất cứ 1 văn bản nào trong tập thì mẫu số sẽ bằng 0 -> phép chia không hợp lệ, vì thế với trường hợp này thường cộng thêm 1 vào mẫu số.

Và cuối cùng TF-IDF được tính bằng:

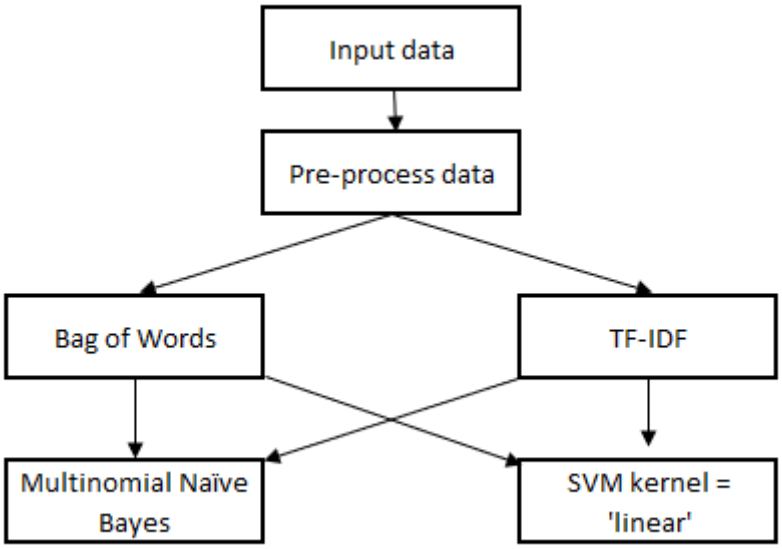
$$tfidf(t,d,D) = tf(t,d) * idf(t,D)$$

Những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khóa của văn bản đó).

2.3.4 Xây dựng và huấn luyện mô hình

- Dữ liệu huấn luyện cho mô hình bao gồm: 2930 bản ghi. Dữ liệu được chia theo tỷ lệ 80/20 bao gồm dữ liệu huấn luyện (train) và tập đánh giá (validation).
- Toàn bộ dữ liệu được tiền xử lý như mục phía trên.
- Vector hóa dữ liệu sử dụng cả Bag of Words và IF-IDF.
- Mô hình huấn luyện sử dụng Multinomial Naive Bayes và Kernel Support Vector Machine

Quá trình huyến luyện:



Kết quả thu được:

Accuracy / Vectorizer	Multinomial Naïve Bayes		Kernel SVM	
	Train	Test	Train	Test
Bag of Words	75.79%	65.42%	94.50%	73.90%
TF-IDF	78.25%	66.63%	85.09%	72.62%

Nhận xét: Sử dụng TF-IDF để vector hóa dữ liệu kết hợp với mô hình SVM kernel='linear' cho kết quả tốt hơn.

Chương 3

Kết luận

3.1 Kết luận

Đề tài này đã nghiên cứu lĩnh vực xử lý ngôn ngữ tự nhiên và áp cụng cụ thể cho tiếng việt. Kết quả đạt được là:

- Hiểu được các bước tiền xứ lý ngôn ngữ tự nhiên
- Hiểu và có sự so sánh giữa hai model huấn luận và Multinomial Naive Bayes và SVM mặc dù cả hai mô hình cho độ chính xác chưa cao.
- Áp dụng được model để phân lớp cảm xúc tuy nhiên dữ liệu đang bị phân tán chưa tập chung vào một lĩnh vực cụ thể.

3.2 Định hướng phát triển

- Thu thập dữ liệu nhiều và đa dạng hơn.
- Xử lý dữ liệu thô hiệu quả hơn.
- Training thêm với các mô hình khác để có sự so sánh và cải thiện độ chính xác.
- Tạo giao diện cho dự án.

Danh mục tài liệu tham khảo

[1] Vũ Hữu Tiệp, "Machine learning cơ bản", 2007

[2] Pouria Kaviani, "Naive Bayes Algorithm", 22/03/2018

[3] Kieu B.T., Pham S.B., "Sentiment analysis for Vietnames", 2010

[4] codetudau.com

[5] sklearn.org

[6] viblo.asia