8 Khai thác văn bản và phân tích văn bản

Chương này bao gồm

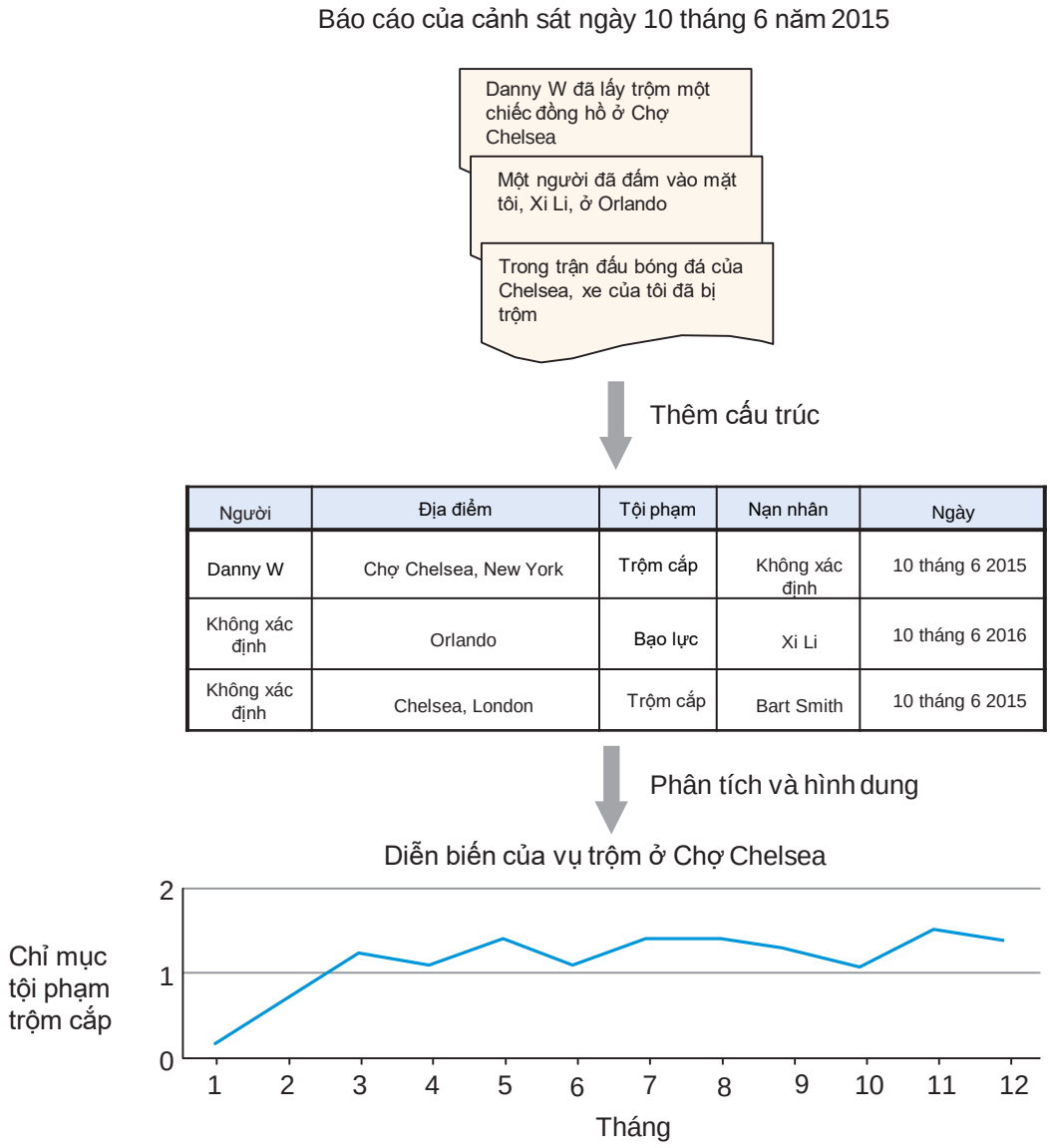
- Hiểu tầm quan trọng của khai thác văn bản
- Giới thiệu các khái niệm quan trọng nhất trong khai thác văn bản
- Làm việc thông qua một dự án khai thác văn bản

Hầu hết các thông tin được con người ghi lại trên thế giới đều ở dạng văn bản. Tất cả chúng ta đều học đọc và viết từ khi còn nhỏ để có thể thể hiện bản thân thông qua việc viết và hiểu những gì người khác biết, suy nghĩ và cảm nhận. Chúng ta sử dụng kỹ năng này mọi lúc khi đọc hoặc viết email, blog, tin nhắn văn bản hoặc cuốn sách này, vì vậy không có gì ngạc nhiên khi ngôn ngữ viết đến với hầu hết chúng ta một cách tự nhiên. Các doanh nghiệp tin rằng nhiều giá trị có thể được tìm thấy trong các văn bản mà mọi người tạo ra, và đúng như vậy bởi vì chúng chứa thông tin về những gì mọi người thích, không thích, những gì họ biết hoặc muốn biết, khao khát và mong muốn, sức khỏe hoặc tâm trạng hiện tại của họ, và nhiều hơn thế nữa. Nhiều thứ trong số này có thể phù hợp với các công ty hoặc nhà nghiên cứu, nhưng không một người nào có thể tự mình đọc và giải thích tài liệu bằng văn bản này. Một lần nữa, chúng ta cần chuyển sang máy tính để làm công việc này.

Tuy nhiên, đáng buồn là ngôn ngữ tự nhiên không đến "tự nhiên" đối với máy tính như đối với con người. Rút ra ý nghĩa và lọc ra những thứ không quan trọng từ những thứ

quan trọng vẫn là thứ con người giỏi hơn máy móc. May mắn thay, các nhà khoa học dữ liệu có thể áp dụng các kỹ thuật khai thác văn bản và phân tích văn bản cụ thể để tìm thông tin liên quan trong hàng đống văn bản mà nếu không thì họ phải mất hàng thế kỷ mới đọc được.

Khai thác văn bản – Text mining hoặc *Phân tích văn bản – Text analytics* là một môn học kết hợp khoa học ngôn ngữ và khoa học máy tính với các kỹ thuật thống kê và học máy. Khai thác văn bản được sử dụng để phân tích văn bản và biến chúng thành một dạng có cấu trúc hơn. Sau đó, nó có dạng cấu trúc này và cố gắng rút ra những hiểu biết sâu sắc từ nó. Ví dụ: khi phân tích tội phạm từ các báo cáo của cảnh sát, khai thác văn bản giúp bạn nhận ra người, địa điểm và loại tội phạm từ các báo cáo. Sau đó, cấu trúc mới này được sử dụng để có được cái nhìn sâu sắc về sự phát triển của tội phạm. Xem hình 8.1.



Hình 8.1 Trong phân tích văn bản, (thường) thách thức đầu tiên là tạo cấu trúc văn bản đầu vào; sau đó nó có thể được phân tích một cách kỹ lưỡng.

Mặc dù ngôn ngữ không giới hạn ở ngôn ngữ tự nhiên, trọng tâm của chương này sẽ là *Xử lý ngôn ngữ tự nhiên – Natural Language Processing* (NLP). Ví dụ về các ngôn ngữ phi tự nhiên sẽ là nhật ký máy, toán học và mã Morse. Về mặt kỹ thuật, ngay cả ngôn ngữ Esperanto, Klingon và Dragon cũng không thuộc lĩnh vực ngôn ngữ tự nhiên vì chúng được phát minh một cách có chủ ý thay vì phát triển theo thời gian; chúng không đến với chúng ta một cách "tự nhiên". Tuy nhiên, những ngôn ngữ cuối cùng này phù hợp với giao tiếp tự nhiên (nói, viết); chúng có ngữ pháp và từ vựng giống như tất cả các ngôn ngữ tự nhiên, và các kỹ thuật khai thác văn bản tương tự có thể áp dụng cho chúng.

8.1 Khai thác văn bản trong thế giới thực

Trong cuộc sống hàng ngày, bạn đã bắt gặp các ứng dụng khai thác văn bản và ngôn ngữ tự nhiên. Tự động điền và sửa lỗi chính tả chính là phân tích liên tục văn bản bạn nhập trước khi gửi email hoặc tin nhắn văn bản. Khi Facebook tự động hoàn thành trạng thái của bạn với tên của một người bạn, nó sẽ thực hiện điều này với sự trợ giúp của một kỹ thuật có tên là *nhận dạng thực thể được đặt tên - named entity recognition*, mặc dù đây sẽ chỉ là một thành phần trong danh mục của họ. Mục tiêu không chỉ để phát hiện bạn đang gõ một danh từ mà còn để đoán bạn đang đề cập đến một người và nhận ra đó có thể là ai. Một ví dụ khác về nhận dạng thực thể được đặt tên được hiển thị trong hình 8.2. Google biết Chelsea là một câu lạc bộ bóng đá nhưng phản hồi khác khi được hỏi về một người.

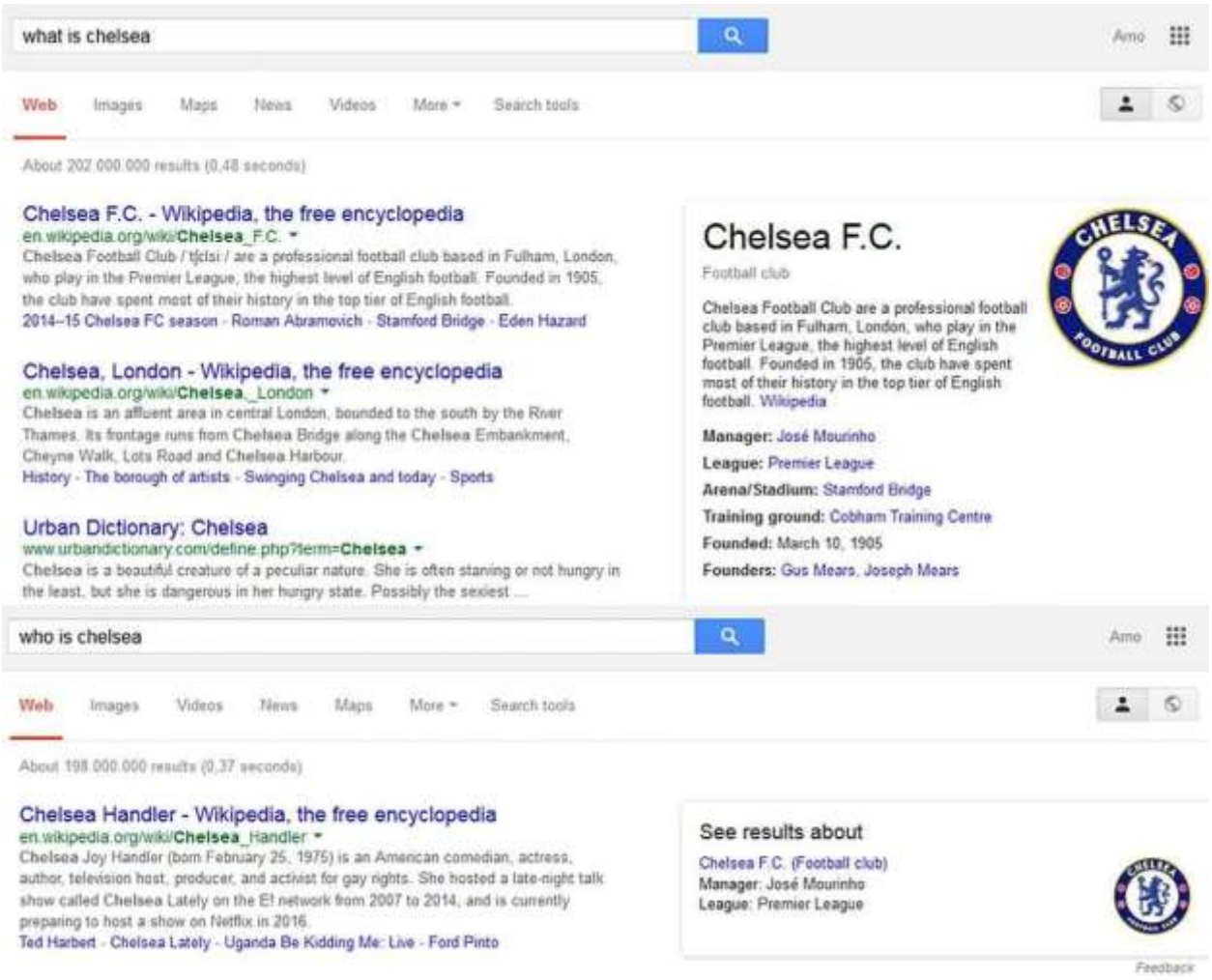
Google sử dụng nhiều loại khai thác văn bản khi trình bày cho bạn kết quả của một truy vấn. Điều gì hiện lên trong đầu bạn khi ai đó nói "Chelsea"? Chelsea có thể là nhiều thứ: một con người; một câu lạc bộ bóng đá; một khu phố ở Manhattan, New York hoặc London; một chợ thực phẩm; một buổi trình diễn hoa; và nhiều hơn nữa. Google biết điều này và trả về các câu trả lời khác nhau cho câu hỏi "Chelsea là ai?" so với "Chelsea là gì?". Để cung cấp câu trả lời phù hợp nhất, Google phải thực hiện (trong số những việc khác) tất cả những điều sau:

- Xử lý trước tất cả các tài liệu mà nó thu thập cho các thực thể được đặt tên
- Thực hiện nhận dạng ngôn ngữ
- Phát hiện loại thực thể bạn đang đề cập đến
- Khớp một truy vấn với một kết quả
- Phát hiện loại nội dung cần trả lại (PDF, dành cho người lớn)

Ví dụ này cho thấy rằng khai thác văn bản không chỉ về ý nghĩa trực tiếp của văn bản mà còn liên quan đến các thuộc tính meta như ngôn ngữ và loại tài liệu.

Google sử dụng khai thác văn bản cho nhiều mục đích hơn là trả lời các truy vấn. Bên cạnh việc bảo vệ người dùng Gmail khỏi thư rác, nó cũng chia email thành các danh mục khác nhau như xã hội, cập nhật và diễn đàn, như thể hiện trong hình 8.3.

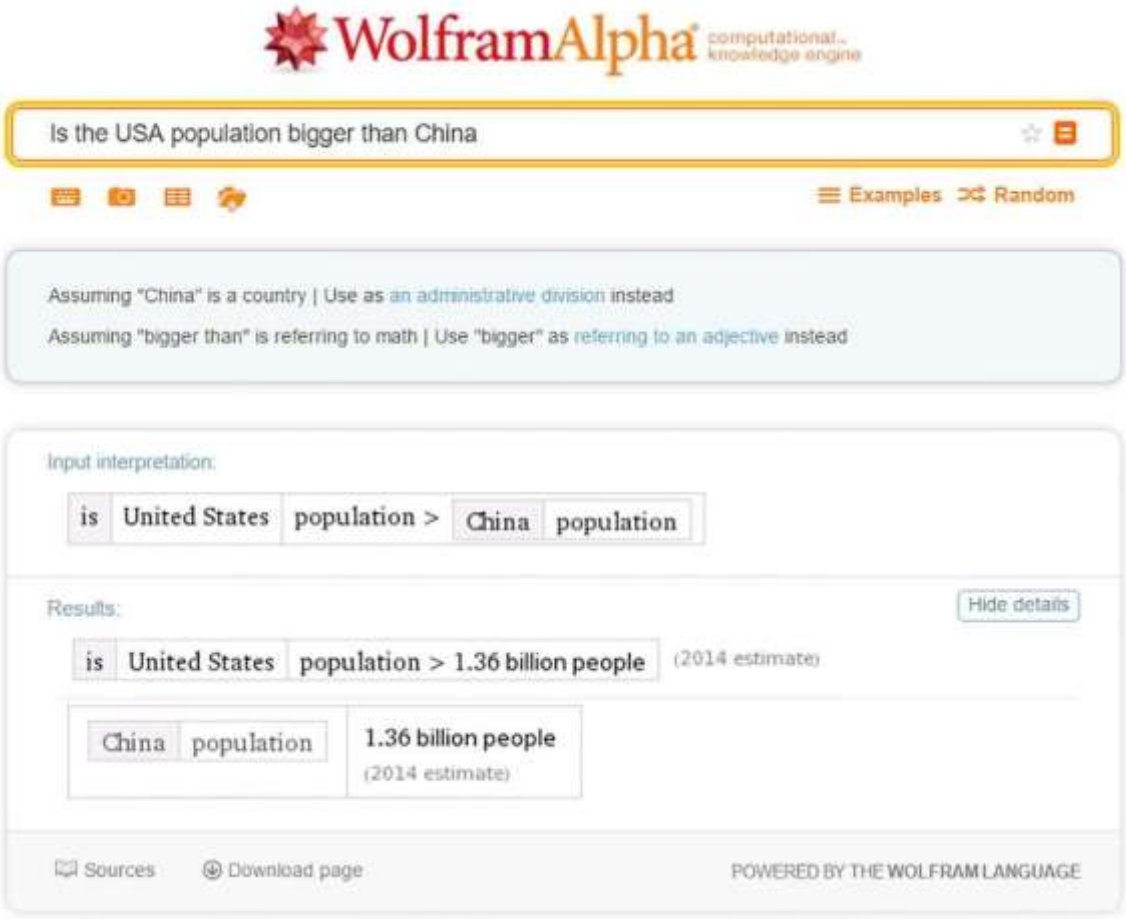
Có thể tiến xa hơn nhiều so với việc trả lời các câu hỏi đơn giản khi bạn kết hợp văn bản với logic và toán học khác.



Hình 8.2 Các câu trả lời khác nhau cho câu hỏi "Chelsea là ai?" và "Chelsea là gì?" ngụ ý rằng Google sử dụng các kỹ thuật khai thác văn bản để trả lời các truy vấn này.

Hình 8.3 Email có thể được tự động chia theo danh mục dựa trên nội dung và nguồn gốc.

Điều này cho phép tạo ra *các công cụ suy luận tự động – automatic reasoning engines* điều khiển bởi các truy vấn ngôn ngữ tự nhiên. Hình 8.4 cho thấy cách "Wolfram Alpha", một công cụ kiến thức tính toán, sử dụng khai thác văn bản và suy luận tự động để trả lời câu hỏi "Dân số Hoa Kỳ có lớn hơn Trung Quốc không?"



Hình 8.4 Công cụ Wolfram Alpha sử dụng khai thác văn bản và lập luận logic để trả lời một câu hỏi.

Nếu điều này vẫn chưa đủ ấn tượng, thì IBM Watson đã khiến nhiều người kinh ngạc vào năm 2011 khi cỗ máy này được thiết lập để chống lại hai người chơi là con người trong một trò chơi *Nguy cơ - Jeopardy*. *Jeopardy* là một chương trình đố vui của Mỹ, nơi mọi người nhận được câu trả lời cho một câu hỏi và điểm được tính khi đoán đúng câu hỏi cho câu trả lời đó. Xem hình 8.5.

Có thể nói vòng này dành cho trí tuệ nhân tạo. IBM Watson là một công cụ nhận thức có thể diễn giải ngôn ngữ tự nhiên và trả lời các câu hỏi dựa trên nền tảng kiến thức sâu rộng.

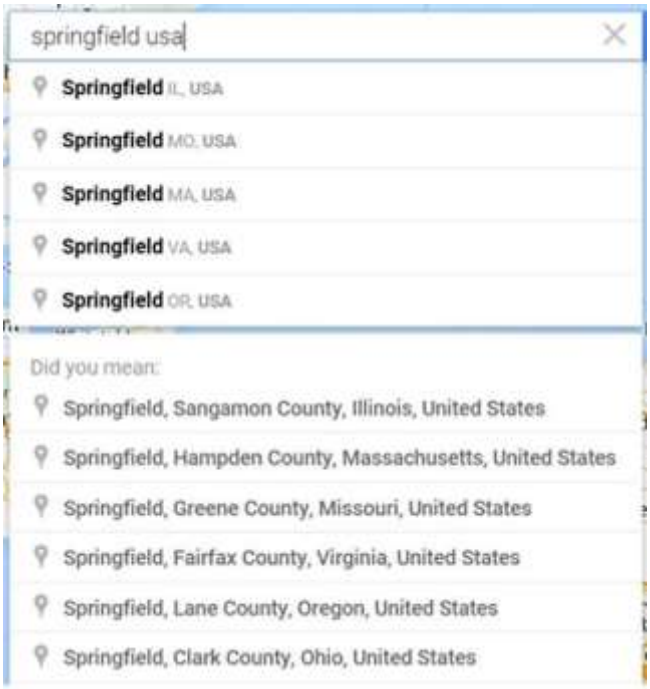
Hình 8.5 IBM Watson thắng trò chơi *Jeopardy* khi chơi với con người.

Khai thác văn bản có nhiều ứng dụng, bao gồm nhưng không giới hạn ở những ứng dụng sau:

- Nhận dạng thực thể
- Phát hiện đạo văn
- Xác định chủ đề
- Phân cụm văn bản
- Dịch
- Tự động tóm tắt văn bản
- Phát hiện gian lận
- Lọc thư rác
- Phân tích ý kiến

Khai thác văn bản là hữu ích, nhưng nó có khó không? Xin lỗi để thất vọng: Vâng, đó là.

Khi xem các ví dụ về Wolfram Alpha và IBM Watson, bạn có thể có ấn tượng rằng việc khai thác văn bản rất dễ dàng. Thật đáng buồn, không. Trong thực tế khai thác văn bản là một công việc phức tạp, thậm chí nhiều việc tưởng chừng như đơn giản cũng không thể thực hiện thỏa đáng. Chẳng hạn, nhận nhiệm vụ đoán đúng địa chỉ. Hình 8.6 cho thấy mức độ khó khăn khi trả về kết quả chính xác với độ chắc chắn và cách Google Maps nhắc bạn cung cấp thêm thông tin khi tìm kiếm “Springfield”. Trong trường hợp này, con người sẽ không làm tốt hơn nếu không có ngữ cảnh bổ sung, nhưng sự mơ hồ này là một trong nhiều vấn đề bạn gặp phải trong ứng dụng khai thác văn bản.



Hình 8.6 Google Maps yêu cầu bạn cung cấp thêm ngữ cảnh do truy vấn "Springfield" không rõ ràng.

Một vấn đề khác là *lỗi đánh vần – spelling mistakes* và *đánh vần (đúng) khác – different (correct) spelling* các dạng của một từ. Lấy ba tham chiếu sau đây đến New York: "NY", "Neww York", và "New York". Đối với một người, thật dễ dàng để thấy tất cả chúng đều đề cập đến thành phố New York. Do cách bộ não của chúng ta diễn giải văn bản, việc hiểu văn bản có lỗi chính tả đến với chúng ta một cách tự nhiên; mọi người thậm chí có thể không nhận thấy chúng. Nhưng đối với máy tính, đây là những chuỗi không liên quan trừ khi chúng ta sử dụng thuật toán để cho máy tính biết rằng chúng đang đề cập đến cùng một thực thể. Các vấn đề liên quan là từ đồng nghĩa và việc sử dụng đại từ. Hãy thử chỉ định đúng người cho đại từ "cô ấy" trong các câu tiếp theo: "John tặng hoa cho bố mẹ Marleen khi anh ấy gặp bố mẹ cô ấy lần đầu tiên. Cô ấy rất hạnh phúc với cử chỉ này". Đủ dễ dàng, phải không? Nhưng không phải cho một máy tính.

Chúng ta có thể giải quyết nhiều vấn đề tương tự một cách dễ dàng, nhưng chúng thường được thấy là khó khăn đối với máy móc. Chúng ta có thể đào tạo các thuật toán hoạt động tốt trên một vấn đề cụ thể trong một phạm vi được xác định rõ, nhưng các thuật toán tổng quát hơn hoạt động trong mọi trường hợp lại hoàn toàn là một điều khác. Chẳng hạn, chúng ta có thể dạy máy tính nhận dạng và truy xuất các số tài khoản của Hoa Kỳ từ văn bản, nhưng điều này không hoạt động tốt cho số tài khoản từ các quốc gia khác.

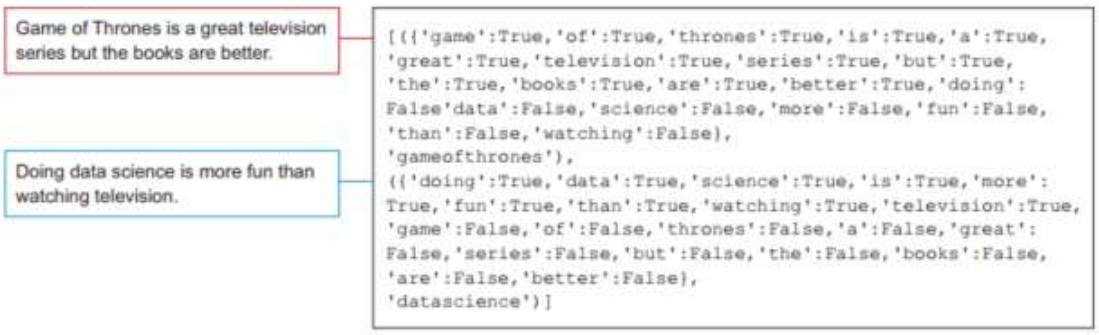
Các thuật toán ngôn ngữ cũng nhạy cảm với ngữ cảnh mà ngôn ngữ được sử dụng, ngay cả khi ngôn ngữ đó không thay đổi. Các mô hình tiếng Anh sẽ không hoạt động đối với tiếng Ả Rập và ngược lại, nhưng ngay cả khi chúng tôi sử dụng tiếng Anh - một thuật toán được đào tạo cho dữ liệu Twitter không có khả năng hoạt động tốt trên các văn bản pháp lý. Hãy ghi nhớ điều này khi chúng ta chuyển sang nghiên cứu tình huống của chương: không có giải pháp hoàn hảo, một giải pháp phù hợp với tất cả trong khai thác văn bản.

8.2 Kĩ thuật khai thác văn bản

Trong nghiên cứu trường hợp sắp tới của chúng tôi, chúng tôi sẽ giải quyết vấn đề *phân loại văn bản – text classification*: tự động phân loại các văn bản chưa được phân loại thành các danh mục cụ thể. Để chuyển từ dữ liệu văn bản dạng thô đến đích cuối cùng, chúng tôi sẽ cần một vài kỹ thuật khai thác dữ liệu yêu cầu thông tin cơ bản để chúng tôi sử dụng chúng một cách hiệu quả. Khái niệm quan trọng đầu tiên trong khai thác văn bản là "túi từ" – "bag of words".

8.2.1 Túi từ

Để xây dựng mô hình phân loại của chúng tôi, chúng tôi sẽ sử dụng cách tiếp cận túi từ. *Túi từ* là cách đơn giản nhất để cấu trúc dữ liệu văn bản: mọi tài liệu được biến thành một vectơ từ. Nếu một từ nào đó xuất hiện trong véc-tơ thì nó được gắn nhãn là "True"; những cái khác được dán nhãn "False". Hình 8.7 cho thấy một ví dụ đơn giản về điều này, trong trường hợp chỉ có hai tài liệu: một về chương trình truyền hình *Trò chơi vương quyền – Game of Thrones* và một về *khoa học dữ liệu – data science*. Hai vectơ từ cùng nhau tạo thành *ma trận thuật ngữ tài liệu – document-term matrix*. Ma trận thuật ngữ tài liệu chứa một cột cho mọi thuật ngữ và một hàng cho mọi tài liệu. Các giá trị là của bạn để quyết định. Trong chương này, chúng ta sẽ sử dụng hệ nhị phân: thuật ngữ có mặt không? Đúng hay sai – True or False.



Hình 8.7 Một văn bản được chuyển đổi thành một túi từ bằng cách gắn nhãn cho mỗi từ (thuật ngữ) là "True" nếu nó có trong tài liệu và "False" nếu không.

Ví dụ từ hình 8.7 cung cấp cho bạn ý tưởng về dữ liệu có cấu trúc mà chúng ta sẽ cần để bắt đầu phân tích văn bản, nhưng nó rất đơn giản: không một từ nào được lọc ra và không áp dụng từ gốc (chúng ta sẽ đi sâu vào vấn đề này sau). Một kho văn bản lớn có thể có hàng ngàn từ độc đáo. Nếu tất cả phải được gắn nhãn như thế này mà không có bất kỳ bộ lọc nào, thì dễ dàng nhận thấy rằng chúng ta có thể thu được một khối lượng lớn dữ liệu. *Túi từ được mã hóa nhị phân – Binary coded bag of words* như trong hình 8.7 chỉ là một cách để cấu trúc dữ liệu; các kỹ thuật khác tồn tại.

Term Frequency – Inverse Document Frequency (TF-IDF)

Tần suất thuật ngữ - Tần suất tài liệu nghịch đảo

Một công thức nổi tiếng để lấp đầy ma trận thuật ngữ tài liệu là TF-IDF hoặc Tần suất Thuật ngữ nhân với Tần suất Tài liệu Nghịch đảo. *Túi từ nhị phân* gán True hoặc False (thuật ngữ là có hay không), trong khi *tần số đơn giản – simple frequencies* đếm số lần thuật ngữ xảy ra. TF-IDF phức tạp hơn một chút và tính đến số lần một thuật ngữ xuất hiện trong tài liệu (TF). TF có thể là số lượng thuật ngữ đơn giản, số lượng nhị phân (True hoặc False) hoặc số lượng thuật ngữ được chia tỷ lệ logarit. Nó phụ thuộc vào những gì làm việc tốt nhất cho bạn. Trong trường hợp TF là tần suất thuật ngữ, công thức TF như sau:

$TF = f_{t,d}$

TF là tần suất (f) của thuật ngữ (t) trong tài liệu (d).

Nhưng TF-IDF cũng tính đến tất cả các tài liệu khác do Tần suất tài liệu nghịch đảo. IDF đưa ra ý tưởng về mức độ phổ biến của từ này trong toàn bộ kho văn bản: tần suất tài liệu càng cao thì càng phổ biến và các từ phổ biến hơn thì ít mang thông tin hơn. Ví dụ: các từ “a” hoặc “the” không có khả năng cung cấp thông tin cụ thể về văn bản. Công thức của IDF với tỷ lệ logarit là dạng IDF được sử dụng phổ biến nhất:

$\text{IDF} = \log(N/|\{d \in D : t \in d\}|)$

với N là tổng số tài liệu trong kho văn bản và $|\{d \in D : t \in d\}|$ là số tài liệu (d) trong đó thuật ngữ (t) xuất hiện.

Điểm TF-IDF nói lên điều này về một thuật ngữ: từ này quan trọng như thế nào để phân biệt tài liệu này với các tài liệu khác trong kho văn bản? Do đó, công thức của TF-IDF là

$\frac{1F}{IDF} = f_{t,d}/\log(N/|\{d \in D : t \in d\}|)$

Chúng tôi sẽ không sử dụng TF-IDF, nhưng khi thiết lập các bước tiếp theo của bạn trong khai thác văn bản, đây sẽ là một trong những điều đầu tiên bạn gặp phải. TF-IDF cũng là thứ đã được Elaticsearch sử dụng ở hậu trường trong chương 6. Đó là một cách hay nếu bạn muốn sử dụng TF- IDF để phân tích văn bản; để việc khai thác văn bản cho phần mềm chuyên dụng như SOLR hoặc Elaticsearch và lấy ma trận tài liệu/thuật ngữ để phân tích văn bản từ đó.

Trước khi đi đến với túi từ thực tế, nhiều bước thao tác dữ liệu khác diễn ra:

- *Token hóa - Tokenization*—Văn bản được cắt thành nhiều phần được gọi là “tokens” hoặc “terms”. Các tokens này là đơn vị thông tin cơ bản nhất mà bạn sẽ sử dụng cho mô hình của mình. Các thuật ngữ thường là từ nhưng đây không phải là điều cần thiết. Toàn bộ các câu có thể được sử dụng để phân tích. Chúng tôi sẽ sử dụng *unigrams*: thuật ngữ bao gồm một từ. Tuy nhiên, thông thường sẽ rất hữu ích nếu bao gồm *bigrams* (hai từ cho mỗi tokens) hoặc *trigrams* (ba từ cho mỗi token) để nắm bắt thêm ý nghĩa và tăng hiệu suất cho các mô hình của bạn.

Tuy nhiên, điều này phải trả giá vì bạn đang xây dựng các véc-tơ thuật ngữ lớn hơn bằng cách đưa bigrams và/hoặc trigrams vào phương trình.

- *Dừng lọc từ - Stop word filtering*—Mọi ngôn ngữ đều có những từ ít có giá trị trong phân tích văn bản vì chúng được sử dụng quá thường xuyên. NLTK đi kèm với một danh sách ngắn các từ để dừng mà chúng ta có thể lọc. Nếu văn bản được mã hóa thành các từ, thì việc loại bỏ vectơ từ của các từ để dừng ít thông tin này là hợp lý.
- *Chữ thường - Lowercasing*—Những từ có chữ in hoa xuất hiện ở đầu câu, khác những từ khác vì chúng là danh từ hoặc tính từ riêng. Chúng ta không nhận được giá trị gia tăng nào khi tạo ra sự khác biệt đó trong ma trận thuật ngữ, vì vậy tất cả các thuật ngữ sẽ được đặt thành chữ thường.

Một kỹ thuật chuẩn bị dữ liệu khác là *stemming*. Điều này đòi hỏi nhiều công phu hơn.

8.2.2 Từ gốc và từ vựng

Stemming là quá trình đưa các từ trở lại dạng gốc của chúng; theo cách này, bạn sẽ có ít phương sai hơn trong dữ liệu. Điều này có ý nghĩa nếu các từ có nghĩa tương tự nhưng được viết khác nhau bởi vì, ví dụ, khi chúng ở dạng số nhiều. Từ gốc cố gắng thống nhất bằng cách cắt bỏ các phần của từ. Ví dụ: “planes” và “plane” đều trở thành “plane”.

Một kỹ thuật khác, được gọi là *lemmatization*, có cùng mục tiêu này nhưng thực hiện theo cách nhạy cảm hơn về mặt ngữ pháp. Ví dụ, trong khi cả stemming và lemmatization sẽ biến “cars” thành “car”, thì lemmatization cũng có thể đưa các động từ liên hợp trở lại dạng không liên hợp của chúng, chẳng hạn như “are” thành “be”. Cái nào bạn sử dụng tùy thuộc vào trường hợp của bạn, và lemmatization thu được nhiều lợi nhuận từ Gắn thẻ POS - POS Tagging (Một phần của Gắn thẻ bài phát biểu – Speech Tagging).

POS Tagging là quá trình gán nhãn ngữ pháp cho mọi phần của câu. Bạn có thể đã làm điều này một cách thủ công ở trường như một bài tập ngôn ngữ. Lấy câu “*Game of Thrones is a television series.*” Nếu chúng tôi áp dụng POS Tagging trên đó, chúng tôi sẽ nhận được

```
({“game”:”NN”},{“of”:”IN},{“thrones”:”NNS},{“is”:”VBZ},{“a”:”DT},{“television”:”NN},
{“series”:”NN})
```

NN là danh từ, IN là giới từ, NNS là danh từ ở dạng số nhiều, VBZ là động từ ngôi thứ ba số ít và DT là từ hạn định. Bảng 8.1 có danh sách đầy đủ.

Bảng 8.1 Danh sách tất cả các thẻ POS

Tag	Meaning	Tag	Meaning
CC	Coordinating conjunction	CD	Cardinal number
DT	Determiner	EX	Existential
FW	Foreign word	IN	Preposition or subordinating conjunction
JJ	Adjective	JJR	Adjective, comparative
JJS	Adjective, superlative	LS	List item marker
MD	Modal	NN	Noun, singular or mass

Bảng 8.1 Danh sách các thẻ POS (tiếp tục)

Tag	Meaning	Tag	Meaning
NNS	Noun, plural	NNP	Proper noun, singular
NNPS	Proper noun, plural	PDT	Predeterminer
POS	Possessive ending	PRP	Personal pronoun
PRP$	Possessive pronoun	RB	Adverb
RBR	Adverb, comparative	RBS	Adverb, superlative
RP	Particle	SYM	Symbol
UH	Interjection	VB	Verb, base form
VBD	Verb, past tense	VBG	Verb, gerund or present participle
VBN	Verb, past participle	VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present	WDT	Wh-determiner
WP	Wh-pronoun	WP$	Possessive wh-pronoun
WRB	Wh-adverb		

POS Tagging là một trường hợp sử dụng mã hóa câu thay vì mã hóa từ. Sau khi POS Tagging hoàn tất, bạn vẫn có thể tiếp tục mã hóa từ, nhưng POS Tagger yêu cầu cả câu. Việc kết hợp POS Tagging và lemmatization có khả năng cung cấp dữ liệu rõ ràng hơn so với việc chỉ sử dụng stemmer. Để đơn giản hóa, chúng tôi sẽ tập trung vào stemming trong nghiên cứu điển hình, nhưng hãy coi đây là cơ hội để giải thích chi tiết về bài tập.

Bây giờ chúng ta đã biết những điều quan trọng nhất mà chúng ta sẽ sử dụng để thực hiện thao tác và làm sạch dữ liệu (khai thác văn bản). Đối với phân tích văn bản của chúng tôi, hãy thêm trình phân loại cây quyết định – decision tree classifier vào danh mục.

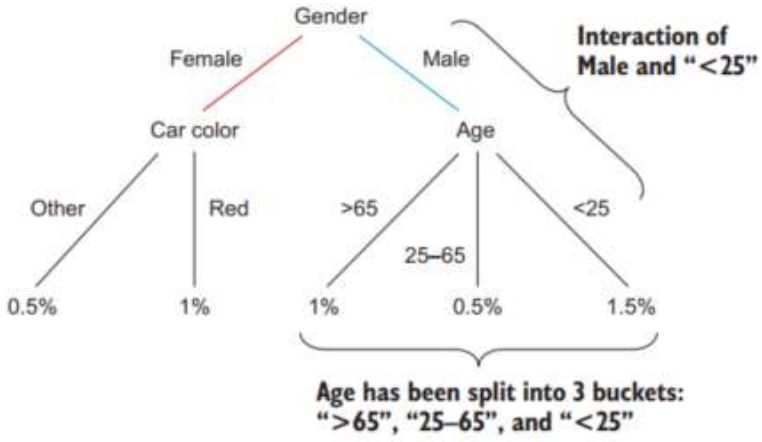
8.2.3 Trình phân loại cây quyết định

Phần phân tích dữ liệu trong nghiên cứu điển hình của chúng tôi cũng sẽ được giữ đơn giản. Chúng tôi sẽ kiểm tra bộ phân loại Naïve Bayes và bộ phân loại cây quyết định. Như đã thấy trong chương 3, bộ phân loại Naïve Bayes được gọi như vậy bởi vì nó coi mỗi biến đầu vào là độc lập với tất cả các biến khác, điều này là ngây thơ, đặc biệt là trong khai thác văn bản. Lấy ví dụ đơn giản về “data science”, “data analysis” hoặc “game of thrones”. Nếu chúng ta cắt dữ liệu của mình theo unigrams, chúng ta sẽ nhận được các biến riêng biệt sau (nếu chúng ta bỏ qua gốc): “data”, “science”, “analysis”, “game”, “of” và “thrones”. Rõ ràng là các liên kết sẽ bị mất. Ngược lại, điều này có thể được khắc phục bằng cách tạo ra bigrams (data science, data analysis) và trigrams (game of thrones).

Tuy nhiên, trình phân loại cây quyết định không coi các biến là độc lập với nhau và tích cực tạo *các biến tương tác – interaction variables* và *các vùng chứa - buckets*. Một biến tương tác

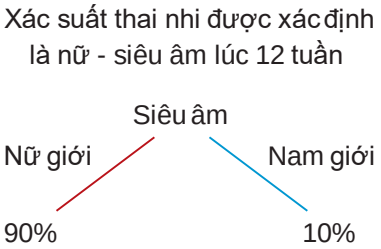
là một biến kết hợp các biến khác. Ví dụ, “data” và “science” có thể là những dự báo tốt theo cách riêng của chúng nhưng hai từ này cùng xuất hiện trong một văn bản có thể có giá trị riêng của nó. Một vùng chứa có phần ngược lại. Thay vì kết hợp hai biến, một biến được chia thành nhiều biến mới. Điều này có ý nghĩa đối với các biến dạng số. Hình 8.8 cho thấy cây quyết định trông như thế nào và nơi bạn có thể tìm thấy sự tương tác và vùng chứa.

Cây quyết định bảo hiểm xe hơi: Xác suất người có bảo hiểm đâm xe trong vòng một năm



Hình 8.8 Mô hình cây quyết định hư cấu. Cây quyết định tự động tạo các vùng chứa và giả sử sự tương tác giữa các biến đầu vào.

Trong khi Naïve Bayes giả định tính độc lập của tất cả các biến đầu vào, cây quyết định được xây dựng dựa trên giả định về sự phụ thuộc lẫn nhau. Nhưng làm thế nào để nó xây dựng cấu trúc này? Cây quyết định có một số tiêu chí khả thi mà nó có thể sử dụng để chia thành các nhánh và quyết định xem biến nào quan trọng hơn (gần với gốc của cây hơn) các biến khác. Cái mà chúng ta sẽ sử dụng trong bộ phân loại cây quyết định NLTK là “thu được thông tin” – “information gain”. Để hiểu information gain, trước tiên chúng ta cần xem xét entropy. *Entropy* là thước đo của sự không thể đoán trước hoặc sự hỗn loạn. Một ví dụ đơn giản là giới tính của một em bé. Khi một người phụ nữ mang thai, giới tính của thai nhi có thể là nam hoặc nữ, nhưng chúng ta không biết đó là giới tính nào. Nếu bạn đoán, bạn có 50% cơ hội đoán đúng (vì phân bổ giới tính không đồng đều 100%). Tuy nhiên, trong thời kỳ mang thai, bạn có cơ hội siêu âm để xác định giới tính của thai nhi. Siêu âm không bao giờ có thể kết luận chính xác 100%, nhưng càng xa trong quá trình phát triển của thai nhi, nó càng trở nên chính xác hơn. Độ chính xác này đạt được, hay *information gain*, là có bởi vì sự không chắc chắn hoặc entropy giảm xuống. Giả sử siêu âm khi thai 12 tuần có độ chính xác 90% trong việc xác định giới tính của em bé. Độ không chắc chắn 10% vẫn tồn tại, nhưng siêu âm đã làm giảm độ không chắc chắn



Hình 8.9 Cây quyết định với một biến: kết luận của bác sĩ khi xem siêu âm lúc mang thai. Xác suất thai là nữ là bao nhiêu?

từ 50% xuống còn 10%. Đó là một sự phân biệt khá tốt. Một cây quyết định cũng tuân theo nguyên tắc này, như thể hiện trong hình 8.9.

Nếu một bài kiểm tra giới tính khác có khả năng dự đoán cao hơn, nó có thể trở thành gốc của cây với bài kiểm tra siêu âm nằm trong các nhánh và điều này có thể tiếp tục cho đến khi chúng ta hết biến hoặc quan sát. Chúng ta có thể hết các quan sát, bởi vì ở mỗi lần chia nhánh, chúng ta cũng chia dữ liệu đầu vào. Đây là một điểm yếu lớn của cây quyết định, bởi vì cấp độ lá của cây sẽ bị phá vỡ nếu còn quá ít quan sát; cây quyết định bắt đầu quá khớp - overfit dữ liệu. *Overfitting* cho phép mô hình nhầm lẫn tính ngẫu nhiên với các tương quan thực. Để chống lại điều này, một cây quyết định *được cắt tỉa - pruned*: các nhánh vô nghĩa bị loại khỏi mô hình cuối cùng.

Bây giờ chúng ta đã xem xét các kỹ thuật mới quan trọng nhất, hãy đi sâu vào nghiên cứu điển hình.

8.3 Nghiên cứu điển hình: Phân loại bài đăng Reddit

Trong khi khai thác văn bản có nhiều ứng dụng, trong nghiên cứu điển hình của chương này, chúng tôi tập trung vào *phân loại tài liệu – document classification*.Như đã chỉ ra trước đó trong chương này, đây chính xác là những gì Google làm khi sắp xếp email của bạn theo danh mục hoặc cố gắng phân biệt thư rác với email thông thường. Nó cũng được sử dụng rộng rãi bởi các trung tâm liên hệ xử lý các câu hỏi hoặc khiếu nại của khách hàng: các khiếu nại bằng văn bản trước tiên sẽ đi qua bộ lọc phát hiện chủ đề để chúng có thể được chỉ định cho đúng người xử lý. Phân loại tài liệu cũng là một trong những tính năng bắt buộc của hệ thống giám sát truyền thông xã hội. Các tweet, bài đăng trên diễn đàn hoặc Facebook, bài báo và nhiều tài nguyên internet khác được giám sát đều được gán nhãn chủ đề. Bằng cách này, chúng có thể được sử dụng lại trong các báo cáo. *Phân tích ý kiến – Sentiment analysis* là một kiểu phân loại văn bản cụ thể: tác giả của bài đăng là tiêu cực, tích cực hay trung lập về điều gì đó? “Điều gì đó” đó có thể được nhận dạng bằng nhận dạng thực thể.

Trong nghiên cứu điển hình này, chúng ta sẽ dựa trên các bài đăng từ Reddit, một trang web tự xưng là “trang nhất của internet”, và cố gắng đào tạo một mô hình có khả năng phân biệt liệu ai đó đang nói về “data science” hay “game of thrones”.

Kết quả cuối cùng có thể là một bản trình bày về mô hình hoặc một ứng dụng tương tác toàn diện. Trong chương 9, chúng ta sẽ tập trung vào việc xây dựng ứng dụng cho người dùng cuối, vì vậy bây giờ chúng ta sẽ tiếp tục trình bày mô hình phân loại của mình.

Để đạt được mục tiêu của mình, chúng tôi sẽ cần tất cả sự trợ giúp và công cụ mà chúng tôi có thể có, đó một lần nữa sẽ là Python, thứ sẵn sàng cung cấp mọi thứ cho chúng tôi.

8.3.1 Làm quen với Bộ công cụ ngôn ngữ tự nhiên NLTK

Python có thể không phải là ngôn ngữ thực thi hiệu quả nhất trên trái đất, nhưng nó có một gói hoàn thiện để khai thác văn bản và xử lý ngôn ngữ: Bộ công cụ ngôn ngữ tự nhiên – *Natural Language Toolkit* (NLTK). NLTK là một tập hợp các thuật toán, hàm và các tác phẩm được chú thích sẽ hướng dẫn bạn thực hiện các bước đầu tiên trong khai thác văn bản và xử lý ngôn ngữ tự nhiên. NLTK cũng được ghi lại một cách xuất sắc trên nltk.org. Tuy nhiên, NLTK thường không được sử dụng cho công việc cấp sản xuất như các thư viện khác, ví dụ là scikit-learn.

Cài đặt NLTK và kho dữ liệu của nó

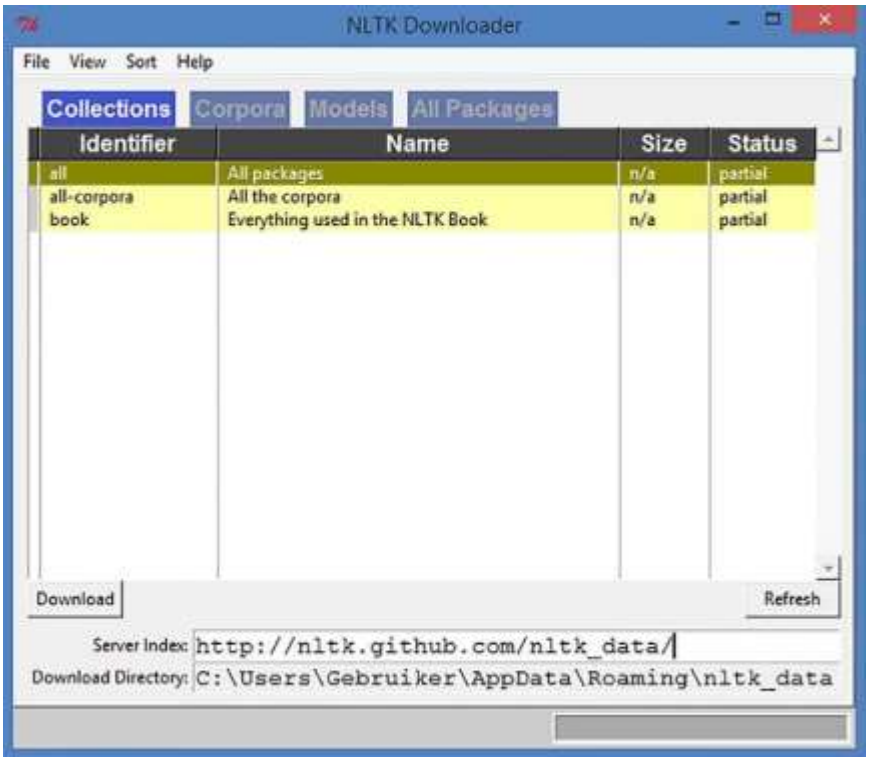
Cài đặt NLTK bằng trình cài đặt gói yêu thích của bạn. Trong trường hợp bạn đang sử dụng Anaconda, nó sẽ được cài đặt với thiết lập Anaconda mặc định. Nếu không, bạn có thể sử dụng “pip” hoặc “easy_install”. Khi điều này được thực hiện, bạn vẫn cần phải cài đặt các mô hình và kho văn bản đi kèm để nó có đầy đủ chức năng. Đối với điều này, hãy chạy mã Python sau:

- `import nltk`
- `nltk.download()`

Tùy thuộc vào cài đặt của bạn, điều này sẽ cung cấp cho bạn một cửa sổ bật lên hoặc nhiều tùy chọn dòng lệnh hơn.

Hình 8.10 hiển thị hộp bật lên mà bạn nhận được khi thực hiện câu lệnh nltk.download().

Bạn có thể tải xuống tất cả kho ngữ liệu nếu muốn, nhưng đối với chương này, chúng tôi sẽ chỉ sử dụng “punkt” và “stopwords”. Việc tải xuống này sẽ được đề cập rõ ràng trong mã đi kèm với cuốn sách này.



Hình 8.10 Chọn Tất cả các Gói để hoàn thành đầy đủ cài đặt NLTK .

Hai tệp sổ tay IPython có sẵn cho chương này:

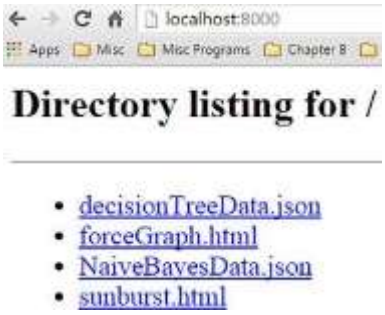
- *Thu thập dữ liệu – Data collection*—Sẽ bao gồm phần thu thập dữ liệu của nghiên cứu điển hình của chương này.
- *Chuẩn bị và phân tích dữ liệu – Data preparation and analysis*—Dữ liệu được lưu trữ được đưa vào quá trình chuẩn bị dữ liệu và sau đó được phân tích.

Tất cả mã trong nghiên cứu điển hình sắp tới có thể được tìm thấy trong hai tệp này theo cùng một trình tự và cũng có thể chạy như vậy. Ngoài ra, có sẵn hai biểu đồ tương tác để tải xuống:

- *forceGraph.html*—Đại diện cho 20 tính năng hàng đầu của mô hình Naïve Bayes của chúng tôi
- *Sunburst.html*—Đại diện cho bốn nhánh hàng đầu của mô hình cây quyết định của chúng tôi

Để mở hai trang HTML này, cần có máy chủ HTTP, bạn có thể sử dụng máy chủ này bằng Python và cửa sổ lệnh:

- Mở một cửa sổ lệnh (Linux, Windows, bất cứ thứ gì bạn thích).
- Di chuyển đến thư mục chứa các tệp HTML và tệp dữ liệu JSON của chúng: decisionTreeData.json cho biểu đồ sunburst và NaiveBayesData.json cho biểu đồ lực. Điều quan trọng là các tệp HTML phải ở cùng vị trí với các tệp dữ liệu của chúng, nếu không bạn sẽ phải thay đổi JavaScript trong tệp HTML.
- Tạo một máy chủ HTTP Python bằng lệnh sau: `python –m Simple-HTTPServer 8000`
- Mở trình duyệt và truy cập localhost:8000; tại đây bạn có thể chọn các tệp HTML, như thể hiện trong hình 8.11.



Hình 8.11 Máy chủ HTTP Python phục vụ đầu ra của chương này

Các gói Python chúng tôi sẽ sử dụng trong chương này:

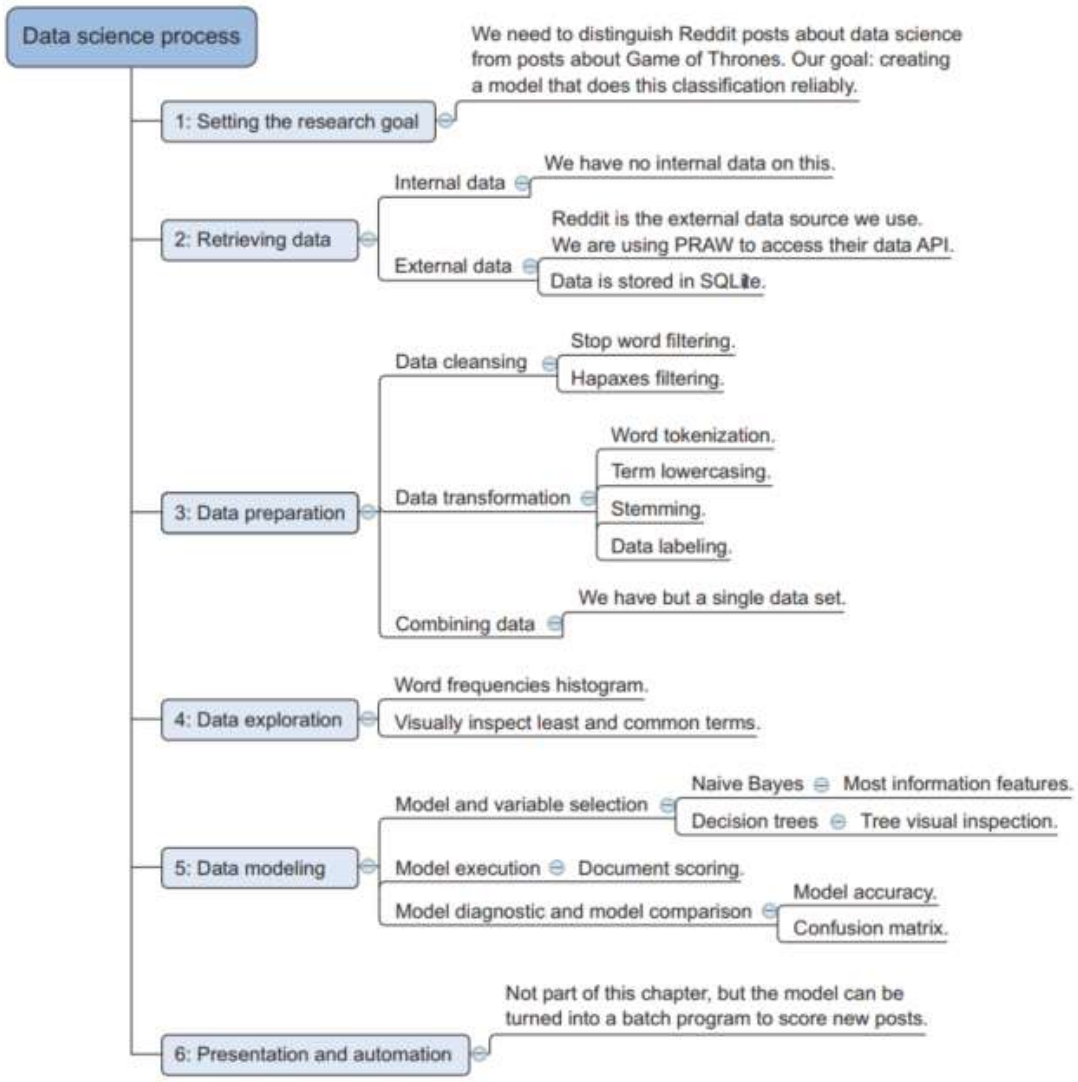
- *NLTK*—Để khai thác văn bản
- *PRAW*—Cho phép tải xuống các bài đăng từ Reddit
- *SQLite3*—Cho phép chúng ta lưu trữ dữ liệu ở định dạng SQLite
- *Matplotlib*—Một thư viện vẽ sơ đồ để trực quan hóa dữ liệu

Đảm bảo cài đặt tất cả các thư viện và kho dữ liệu cần thiết trước khi tiếp tục. Tuy nhiên, trước khi đi sâu vào hành động, hãy xem xét các bước chúng ta sẽ thực hiện để đạt được mục tiêu tạo mô hình phân loại chủ đề.

8.3.2 Tổng quan về quy trình khoa học dữ liệu và bước 1: Mục tiêu nghiên cứu

Để giải bài tập khai thác văn bản này, một lần nữa chúng ta sẽ sử dụng quy trình khoa học dữ liệu. Hình 8.12 cho thấy quy trình khoa học dữ liệu được áp dụng cho phân loại trường hợp Reddit của chúng tôi.

Không phải tất cả các yếu tố được mô tả trong hình 8.12 đều có ý nghĩa vào thời điểm này và phần còn lại của chương được dành để giải quyết vấn đề này trong thực tế khi chúng ta hướng tới mục tiêu nghiên cứu của mình: tạo ra một mô hình phân loại có khả năng phân biệt các bài viết về “data science” từ các bài đăng về “Game of Thrones”. Không chần chừ gì nữa, hãy đi lấy dữ liệu của chúng ta.



Hình 8.12 Tổng quan về quy trình khoa học dữ liệu được áp dụng cho nghiên cứu tình huống phân loại chủ đề trên Reddit

8.3.3 Bước 2: Truy xuất dữ liệu

Chúng tôi sẽ sử dụng dữ liệu Reddit cho trường hợp này và đối với những người chưa quen với Reddit, hãy dành thời gian để tự làm quen với các khái niệm của nó tại www.reddit.com.

Reddit tự gọi mình là "trang nhất của internet" vì người dùng có thể đăng những thứ họ thấy thú vị và/ hoặc tìm thấy ở đâu đó trên internet, và chỉ những thứ được nhiều người cho là thú vị mới được giới thiệu là "phổ biến" trên trang chủ của nó. Bạn có thể nói Reddit đưa ra cái nhìn tổng quan về những thứ thịnh hành trên internet. Bất kỳ người dùng nào cũng có thể đăng trong một danh mục được xác định trước được gọi là "subreddit". Khi một bài đăng được tạo, những người dùng khác có thể nhận xét về bài đăng đó và có thể ủng hộ nếu họ thích hoặc không ủng hộ nếu họ không thích. Bởi vì một bài đăng luôn là một phần của subreddit, chúng tôi có siêu dữ liệu này khi chúng tôi kết nối với API Reddit để lấy dữ liệu của mình. Chúng tôi đang tìm nạp dữ liệu được gắn nhãn một cách hiệu quả vì chúng tôi cho rằng một bài đăng "gameofthrones" trên subreddit sẽ có thứ gì đó để làm với "gameofthrones".

Để truy cập dữ liệu của mình, chúng tôi sử dụng thư viện chuẩn Reddit Python API có tên là PRAW. Khi chúng tôi nhận được dữ liệu mình cần, chúng tôi sẽ lưu trữ dữ liệu đó trong một tệp giống cơ sở dữ liệu nhẹ có tên là SQLite. SQLite là lí tưởng để lưu trữ một lượng nhỏ dữ liệu vì nó không yêu cầu bất kỳ thiết lập nào để sử dụng và sẽ phản hồi các truy vấn SQL giống như bất kỳ cơ sở dữ liệu quan hệ thông thường khác. Bất kỳ phương tiện lưu trữ dữ liệu nào khác cũng được; nếu bạn thích cơ sở dữ liệu Oracle hoặc Postgres, Python có một thư viện tuyệt vời để tương tác với chúng mà không cần phải viết SQL. SQLAlchemy cũng sẽ hoạt động đối với các tệp SQLite. Hình 8.13 cho thấy bước truy xuất dữ liệu trong quy trình khoa học dữ liệu.

Hình 8.13 Bước truy xuất dữ liệu của quy trình khoa học dữ liệu cho trường hợp phân loại chủ đề trên Reddit

Mở trình thông dịch Python yêu thích của bạn; đã đến lúc hành động, như trong danh sách 8.1. Trước tiên, chúng tôi cần thu thập dữ liệu của mình từ trang web Reddit. Nếu bạn chưa có, hãy sử dụng `install praw` or `conda install praw` (Anaconda) trước khi chạy tập lệnh sau.

GHI CHÚ Bạn cũng có thể tìm thấy mã cho bước 2 trong tệp IPython "Chapter 8 data collection". Nó có sẵn trong phần tải xuống của cuốn sách này.

Liệt kê 8.1 Thiết lập cơ sở dữ liệu SQLLite và ứng dụng khách API Reddit

```
import praw
import sqlite3

conn = sqlite3.connect('reddit.db')
c = conn.cursor()

c.execute('''DROP TABLE IF EXISTS topics''')
c.execute('''DROP TABLE IF EXISTS comments''')
c.execute('''CREATE TABLE topics
             (topicTitle text, topicText text, topicID text,
topicCategory text)''')
c.execute('''CREATE TABLE comments
             (commentText text, commentID text ,
topicTitle text, topicText text, topicID text ,
 topicCategory text)''‘’)

user_agent = "Introducing Data Science Book"
r = praw.Reddit(user_agent=user_agent)

subreddits = ['datascience','gameofthrones’]

limit = 1000
```

Nhập thư viện PRAW và SQLite3.

Thiết lập kết nối với cơ sở dữ liệu SQLite.

Thực thi câu lệnh SQL để tạo chủ đề và bảng nhận xét.

Tạo PRAW user agent để chúng ta có thể sử dụng API Reddit.

Danh sách các subreddits của chúng tôi sẽ được đưa vào cơ sở dữ liệu SQLite.

Số bài đăng tối đa chúng tôi sẽ tìm nạp từ Reddit cho mỗi danh mục. Reddit tối đa cho phép tại bất kỳ thời điểm nào cũng là 1,000.

Trước tiên hãy nhập các thư viện cần thiết.

Bây giờ chúng ta có quyền truy cập vào các tính năng của SQLite3 và PRAW, chúng ta cần chuẩn bị cơ sở dữ liệu cục bộ nhỏ của mình cho dữ liệu sắp nhận. Bằng cách xác định kết nối tới tệp SQLite, chúng tôi sẽ tự động tạo nó nếu nó chưa tồn tại. Sau đó, chúng tôi xác định một con trỏ dữ liệu có khả năng thực thi bất kỳ câu lệnh SQL nào, sử dụng nó để xác định trước cấu trúc cơ sở dữ liệu của mình. Cơ sở dữ liệu sẽ chứa hai bảng: bảng chủ đề chứa các chủ đề Reddit, tương tự như việc ai đó bắt đầu một bài đăng mới trên diễn đàn và bảng thứ hai chứa các nhận xét và được liên kết với bảng chủ đề thông qua cột “topicID”. Hai bảng có mối quan hệ một (bảng chủ đề) với nhiều (bảng nhận xét). Đối với nghiên cứu điển hình, chúng tôi sẽ giới hạn bản thân trong việc sử dụng bảng chủ đề, nhưng việc thu thập dữ liệu sẽ kết hợp cả hai vì điều này cho phép bạn thử nghiệm với dữ liệu bổ sung này nếu bạn muốn. Để trau dồi kỹ năng khai thác văn bản của mình, bạn có thể thực hiện phân tích ý kiến đối với các nhận xét về chủ đề và tìm ra chủ đề nào nhận được nhận xét tiêu cực hoặc tích cực. Sau đó, bạn có thể tương quan điều này với các đặc điểm của mô hình mà chúng tôi sẽ tạo ra ở cuối chương này.

Chúng tôi cần tạo ứng dụng khách PRAW để có quyền truy cập vào dữ liệu. Mọi subreddit đều có thể được xác định bằng tên của nó và chúng tôi quan tâm đến “datascience” và “gameofthrones”. Giới hạn thể hiện số lượng chủ đề tối đa (bài đăng, không phải bình luận) mà chúng tôi sẽ rút ra từ Reddit. Một nghìn cũng là số lượng tối đa mà API cho phép chúng tôi tìm nạp theo bất kỳ yêu cầu cụ thể nào, mặc dù chúng tôi có thể yêu cầu nhiều hơn sau này khi mọi

người có đăng những điều mới. Trên thực tế, chúng tôi có thể chạy yêu cầu API theo định kỳ và thu thập dữ liệu theo thời gian. Mặc dù tại bất kỳ thời điểm nào, bạn bị giới hạn ở một nghìn bài đăng, nhưng không có gì ngăn cản bạn phát triển cơ sở dữ liệu của riêng mình trong suốt nhiều tháng. Điều đáng chú ý là tập lệnh sau có thể mất khoảng một giờ để hoàn thành. Nếu bạn không muốn chờ đợi, vui lòng tiếp tục và sử dụng tệp có thể tải xuống SQLite. Ngoài ra, nếu bạn chạy nó bây giờ, bạn sẽ không có khả năng nhận được kết quả đầu ra chính xác như khi nó được chạy lần đầu tiên để tạo kết quả đầu ra được hiển thị trong chương này.

Hãy xem chức năng truy xuất dữ liệu của chúng tôi, như được hiển thị trong danh sách sau.

Liệt kê 8.2 Truy xuất và lưu trữ dữ liệu Reddit trong SQLite

Các lĩnh vực cụ thể của chủ đề được thêm vào danh sách. Chúng tôi chỉ sử dụng tiêu đề và văn bản trong suốt bài tập nhưng ID chủ đề sẽ hữu ích cho việc xây dựng cơ sở dữ liệu chủ đề (lớn hơn) của riêng bạn.

Từ subreddits, lấy 1,000 chủ đề hot nhất (trong trường hợp của chúng ta)

Phần này là một bản in thông tin và không cần thiết để mã hoạt động. Nó chỉ thông báo cho bạn về tiến độ tải xuống.

Nối nhận xét thành một danh sách. Những thứ này không được sử dụng trong bài tập nhưng bạn có thể dùng chúng để thử nghiệm.

Chèn tất cả chủ đề vào cơ sở dữ liệu SQLite.

Chèn tất cả nhận xét vào cơ sở dữ liệu SQLite.

Hàm này được thực thi cho tất cả các subreddits mà chúng tôi đã chỉ định trước đó.

Cam kết thay đổi (chèn dữ liệu) vào cơ sở dữ liệu. Nếu không có cam kết, sẽ không có dữ liệu nào được chèn vào.

```
def prawGetData(limit,subredditName):
    topics = r.get_subreddit(subredditName).get_hot(limit=limit)
    commentInsert = []
    topicInsert = []
    topicNBR = 1
    for topic in topics:
        if (float(topicNBR)/limit)*100 in xrange(1,100):
            print '********** TOPIC:' + str(topic.id)
+ ' *********COMPLETE: ' + str((float(topicNBR)/limit)*100)
+ ' % ****'
        topicNBR += 1
        try:
            topicInsert.append((topic.title,topic.selftext,topic.id,
            subredditName))
        except:
            pass
        try:
            for comment in topic.comments:
                commentInsert.append((comment.body,comment.id,
topic.title,topic.selftext,topic.id,subredditName))
        except:
            pass
    print '********************************'
    print 'INSERTING DATA INTO SQLITE'
    c.executemany('INSERT INTO topics VALUES (?,?,?,?)', topicInsert)
    print 'INSERTED TOPICS'
    c.executemany('INSERT INTO comments VALUES (?,?,?,?,?,?)', commentInsert)
    print 'INSERTED COMMENTS'
    conn.commit()

for subject in subreddits:
    prawGetData(limit=limit,subredditName=subject)
```

Hàm `prawGetData()` truy xuất các chủ đề "hấp dẫn nhất" trong subreddit của nó, nối chủ đề này vào một mảng và sau đó nhận tất cả các nhận xét liên quan. Điều này tiếp tục cho đến khi đạt được một nghìn chủ đề hoặc không còn chủ đề nào để tìm nạp và mọi thứ được lưu trữ trong cơ sở dữ liệu SQLite. Các báo cáo in ở đó để thông báo cho bạn về quá trình thu thập hàng nghìn chủ đề của nó. Tất cả những gì còn lại để chúng tôi làm là thực hiện chức năng cho từng subreddit.

Nếu bạn muốn phân tích này kết hợp nhiều hơn hai subreddit, thì đây là vấn đề khi thêm một danh mục bổ sung vào mảng subreddits.

Với dữ liệu được thu thập, chúng tôi đã sẵn sàng chuyển sang chuẩn bị dữ liệu.

8.3.4 Bước 3: Chuẩn bị dữ liệu

Như mọi khi, chuẩn bị dữ liệu là bước quan trọng nhất để có được kết quả chính xác. Đối với khai thác văn bản, điều này thậm chí còn đúng hơn vì chúng tôi thậm chí không bắt đầu với dữ liệu có cấu trúc.

Mã sắp tới có sẵn trực tuyến dưới dạng tệp IPython "Chapter 8 data preparation and analysis". Hãy bắt đầu bằng cách nhập các thư viện cần thiết và chuẩn bị cơ sở dữ liệu SQLite, như trong danh sách sau.

Liệt kê 8.3 Khai thác văn bản, thư viện, phụ thuộc kho văn bản và kết nối cơ sở dữ liệu SQLite

```
import sqlite3
import nltk
import matplotlib.pyplot as plt
from collections import OrderedDict
import random
```

Nhập tất cả thư viện yêu cầu

```
nltk.download('punkt')
nltk.download('stopwords')
```

Tải về kho văn bản chúng ta sử dụng

```
conn = sqlite3.connect('reddit.db')
c = conn.cursor()
```

Tạo kết nối với cơ sở dữ liệu SQLite chứa dữ liệu Reddit của chúng tôi

Trong trường hợp bạn chưa tải xuống kho dữ liệu NLTK đầy đủ, bây giờ chúng ta sẽ tải xuống phần mà chúng ta sẽ sử dụng. Đừng lo lắng nếu bạn đã tải xuống, tập lệnh sẽ phát hiện xem kho dữ liệu của bạn có được cập nhật hay không.

Dữ liệu của chúng tôi vẫn được lưu trữ trong tệp Reddit SQLite, vì vậy hãy tạo kết nối với nó.

Ngay cả trước khi khám phá dữ liệu của mình, chúng tôi biết ít nhất hai điều chúng tôi phải làm để làm sạch dữ liệu: ngừng lọc từ và viết thường.

Một chức năng lọc từ tổng hợp sẽ giúp chúng ta lọc bỏ những phần chưa sạch. Hãy tạo một cái trong danh sách sau.

Liệt kê 8.4 Chức năng lọc từ và viết thường

```
def wordFilter(excluded,wordrow):
    filtered = [word for word in wordrow if word not in excluded]
    return filtered
stopwords = nltk.corpus.stopwords.words('english')
def lowerCaseArray(wordrow):
    lowercased = [word.lower() for word in wordrow]
    return lowercased
```

Hàm wordFilter() sẽ xóa một thuật ngữ khỏi mảng các thuật ngữ.

Biến stop word chứa các English stop words theo mặc định trong NLTK

Hàm LowerCaseArray() chuyển đổi bất kỳ thuật ngữ nào thành thuật ngữ của nó ở dạng chữ thường.

Các English stop word sẽ là từ đầu tiên rời khỏi dữ liệu của chúng tôi. Đoạn mã sau sẽ cung cấp cho chúng ta những stop word này:

```
stopwords = nltk.corpus.stopwords.words('english')
print stopwords
```

Hình 8.14 hiển thị danh sách English stop words trong NLTK.

```
stopwords = nltk.corpus.stopwords.words('english')
print stopwords
```

```
[u'i', u'me', u'my', u'myself', u'we', u'our', u'ours', u'ourselves', u'you',
u'your', u'yours', u'yourself', u'yourselves', u'he', u'him', u'his', u'himsel
f', u'she', u'her', u'hers', u'herself', u'it', u'its', u'itself', u'they', u'th
em', u'their', u'theirs', u'themselves', u'what', u'which', u'who', u'whom', u't
his', u'that', u'these', u'those', u'am', u'is', u'are', u'was', u'were', u'be',
u'been', u'being', u'have', u'has', u'had', u'having', u'do', u'does', u'did',
u'doing', u'a', u'an', u'the', u'and', u'but', u'if', u'or', u'because', u'as',
u'until', u'while', u'of', u'at', u'by', u'for', u'with', u'about', u'against',
u'between', u'into', u'through', u'during', u'before', u'after', u'above', u'bel
ow', u'to', u'from', u'up', u'down', u'in', u'out', u'on', u'off', u'over', u'un
der', u'again', u'further', u'then', u'once', u'here', u'there', u'when', u'wher
e', u'why', u'how', u'all', u'any', u'both', u'each', u'few', u'more', u'most',
u'other', u'some', u'such', u'no', u'nor', u'not', u'only', u'own', u'same', u's
o', u'than', u'too', u'very', u's', u't', u'can', u'will', u'just', u'don', u'sh
ould', u'now']
```

Hình 8.14 English stop words trong NLTK

Với tất cả các thành phần cần thiết, chúng ta hãy xem chức năng xử lý dữ liệu đầu tiên của chúng tôi trong danh sách sau.

Liệt kê 8.5 Chức năng chuẩn bị dữ liệu đầu tiên và thực hiện

Chúng tôi sẽ sử dụng data['all_words']

để khám phá dữ liệu.

```
def data_processing(sql):
    c.execute(sql)
    data = {'wordMatrix':[],'all_words':[]}
    row = c.fetchone()
    while row is not None:
        wordrow = nltk.tokenize.word_tokenize(row[0]+" "+row[1])
        wordrow_lowercased = lowerCaseArray(wordrow)
        wordrow_nostopwords = wordFilter(stopwords,wordrow_lowercased)
        data['all_words'].extend(wordrow_nostopwords)
        data['wordMatrix'].append(wordrow_nostopwords)
        row = c.fetchone()
    return data
subreddits = ['datascience','gameofthrones']
data = {}
for subject in subreddits:
    data[subject] = data_processing(sql='''SELECT
     topicTitle,topicText,topicCategory FROM topics
WHERE topicCategory = '''+"'"+subject+"'")
```

Tạo con trỏ vào dữ liệu AWLite.

Tìm nạp dữ liệu hàng đến hàng

Row[0] là tiêu đề, row[1] là chủ đề văn bản; chúng tôi biến chúng thành blob.

Nhận tài liệu mới từ cơ sở dữ liệu SQLite.

Các subreddit được xác định trước đó của chúng tôi.

Gọi hàm xử lí dữ liệu cho mọi subreddit.

data['wordMatrix'] là một ma trận bao gồm các véctơ từ; 1 véctơ trên mỗi văn bản.

Hàm `data_processing()` nhận một câu lệnh SQL và trả về ma trận thuật ngữ tài liệu. Nó thực hiện điều này bằng cách lặp qua từng mục nhập dữ liệu (chủ đề Reddit) và kết hợp tiêu đề chủ đề và nội dung chủ đề thành một vectơ từ duy nhất bằng cách sử dụng word tokenization. Một *tokenizer* là một tập lệnh xử lý văn bản để cắt văn bản thành nhiều phần. Bạn có nhiều cách khác nhau để mã hóa văn bản: bạn có thể chia văn bản thành câu hoặc từ, bạn có thể chia theo dấu cách và dấu câu hoặc bạn có thể tính đến các ký tự khác, v.v. Ở đây, chúng tôi đã chọn word tokenizer NLTK tiêu chuẩn. Word tokenizer này rất đơn giản; tất cả những gì nó làm là chia văn bản thành các thuật ngữ nếu có khoảng cách giữa các từ. Sau đó, chúng tôi viết thường vectơ và lọc ra stop word. Lưu ý thứ tự quan trọng như thế nào ở đây; một stop word ở đầu câu sẽ không được lọc nếu trước tiên chúng ta lọc các stop word trước khi viết thường. Ví dụ: trong “I like Game of Thrones”, chữ “I” sẽ không được viết thường và do đó sẽ không bị lọc ra. Sau đó, chúng tôi tạo một ma trận từ (ma trận tài liệu thuật ngữ) và một danh sách chứa tất cả các từ. Lưu ý cách chúng tôi mở rộng danh sách mà không cần lọc gấp đôi; bằng cách này, chúng tôi có thể tạo biểu đồ tần số về các lần xuất hiện từ trong quá trình khám phá dữ liệu. Hãy thực hiện chức năng cho hai loại chủ đề của chúng tôi.

Hình 8.15 cho thấy vectơ từ đầu tiên của danh mục “datascience”.

```
print data['datascience']['wordMatrix'][0]
```

```
print data['datascience']['wordMatrix'][0]
```

```
[u'data', u'science', u'freelancing', u"'m", u'currently', u'master
s', u'program', u'studying', u'business', u'analytics', u"'m", u'try
ing', u'get', u'data', u'freelancing', u'.', u"'m", u'still', u'lear
ning', u'skill', u'set', u'typically', u'see', u'right', u"'m", u'fa
irly', u'proficient', u'sql', u'know', u'bit', u'r.', u'freelancer
s', u'find', u'jobs', u'?']
```

Hình 8.15 Vectơ từ đầu tiên của danh mục "datascience" sau lần thử xử lý dữ liệu đầu tiên

Điều này chắc chắn có vẻ bị ô nhiễm: dấu câu được giữ dưới dạng các thuật ngữ riêng biệt và một số từ thậm chí còn chưa được tách ra. Việc khám phá dữ liệu sâu hơn sẽ làm rõ một số điều cho chúng tôi.

8.3.5 Bước 4: Khám phá dữ liệu

Hiện tại, chúng tôi đã tách biệt tất cả các thuật ngữ của mình, nhưng kích thước khổng lồ của dữ liệu cản trở sự nắm bắt tốt của chúng tôi rằng liệu nó có đủ sạch để sử dụng thực tế hay không. Tuy nhiên, bằng cách xem xét một vectơ đơn lẻ, chúng tôi đã phát hiện ra một số vấn đề: một số từ chưa được phân tách chính xác và vectơ chứa nhiều thuật ngữ có một ký tự. Các thuật ngữ một ký tự có thể là yếu tố phân biệt chủ đề tốt trong một số trường hợp nhất định. Ví dụ: một văn bản kinh tế sẽ chứa nhiều kí hiệu $, £ và € hơn một văn bản y tế. Nhưng trong hầu hết các trường hợp, các thuật ngữ một ký tự này là vô ích. Đầu tiên, chúng ta hãy xem phân phối tần số của các thuật ngữ của chúng tôi.

```
wordfreqs_cat1 = nltk.FreqDist(data['datascience']['all_words'])
plt.hist(wordfreqs_cat1.values(), bins = range(10))
plt.show()
wordfreqs_cat2 = nltk.FreqDist(data['gameofthrones']['all_words'])
plt.hist(wordfreqs_cat2.values(), bins = range(20))
plt.show()
```

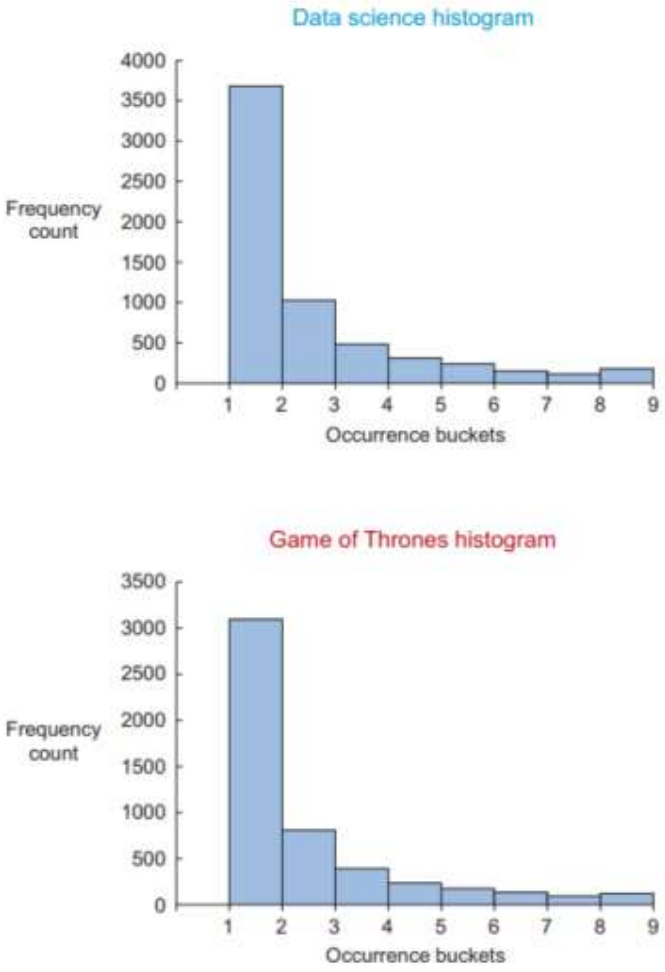
Bằng cách vẽ biểu đồ tần số phân bổ (hình 8.16), chúng tôi nhanh chóng nhận thấy rằng phần lớn các thuật ngữ của chúng tôi chỉ xuất hiện trong một tài liệu.

Các thuật ngữ xảy ra một lần như thế này được gọi là *hapaxes*, và là mô hình khôn ngoan, chúng vô dụng vì một lần xuất hiện của một tính năng không bao giờ đủ để xây dựng một mô hình đáng tin cậy. Đây là tin tốt cho chúng tôi; việc loại bỏ các hapaxes này sẽ thu nhỏ đáng kể dữ liệu của chúng tôi mà không làm hại đến mô hình cuối cùng. Hãy xem xét một số thuật ngữ xuất hiện đơn lẻ này.

```
print wordfreqs_cat1.hapaxes()
print wordfreqs_cat2.hapaxes()
```

Các thuật ngữ chúng ta thấy trong hình 8.17 có ý nghĩa và nếu chúng ta có nhiều dữ liệu hơn thì chúng có thể xuất hiện thường xuyên hơn.

```
print wordfreqs_cat1.hapaxes()
print wordfreqs_cat2.hapaxes()
```



Hình 8.16 Biểu đồ tần số thuật ngữ này cho thấy cả ma trận thuật ngữ "data science" và "game of thrones" có hơn 3000 thuật ngữ xuất hiện một lần.

Least frequent terms within data science posts

```
print wordfreqs_cat1.hapaxes()
```

```
[u'post-grad', u'marching', u'cytoscape', u'wizardry', u"'pure", u'i
mmature', u'socrata', u'filenotfoundexception', u'side-by-side', u'b
ringing', u'non-experienced', u'zestimate', u'formatting*', u'sustai
```

Least frequent terms within Game of Thrones posts

```
print wordfreqs_cat2.hapaxes()
```

```
[u'hordes', u'woods', u'comically', u'pack', u'seventy-seven', u"'co
ntext", u'shaving', u'kennels', u'differently', u'screaming', u'her-
', u'complainers', u'sailed', u'contributed', u'payoff', u'hallucina
```

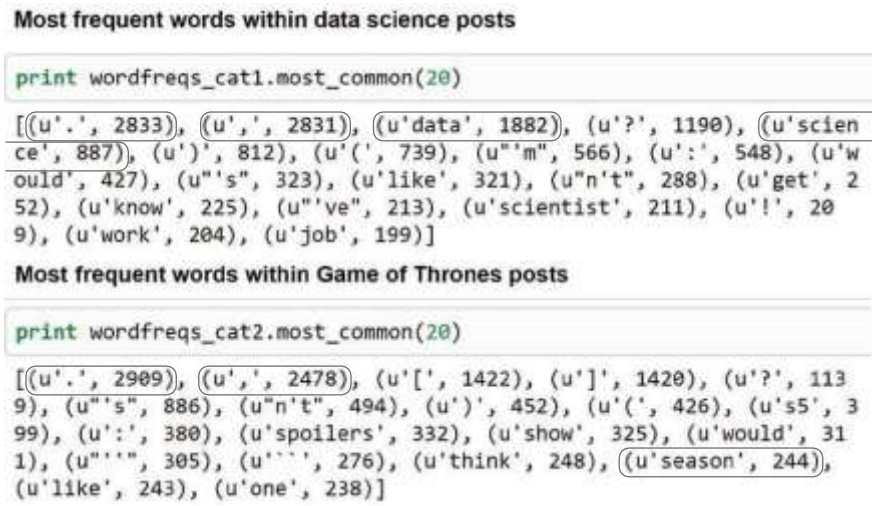
Hình 8.17 Thuật ngữ xuất hiện đơn lẻ (hapaxes) "Data science" và "game of thrones"

Nhiều thuật ngữ trong số này là cách viết sai của những thuật ngữ hữu ích khác, chẳng hạn như: Jaimie là Jaime (Lannister), Milisandre sẽ là Melisandre, v.v. *Game of Thrones* - từ điển đồng nghĩa cụ thể có thể giúp chúng tôi tìm và thay thế các lỗi chính tả này bằng thuật toán tìm kiếm mờ. Điều này chứng tỏ việc làm sạch dữ liệu trong khai thác văn bản có thể tiếp tục vô thời hạn nếu bạn mong muốn; giữ sự nỗ lực và tỉ lệ phần trăm cân bằng là rất quan trọng ở đây.

Bây giờ chúng ta hãy xem những từ thường xuyên nhất.

```
print wordfreqs_cat1.most_common(20)
print wordfreqs_cat2.most_common(20)
```

Hình 8.18 cho thấy kết quả của việc yêu cầu 20 từ thông dụng nhất cho mỗi danh mục.



Hình 8.18 Top 20 từ thường dùng nhất cho các bài đăng "data science" và "game of thrones"

Bây giờ điều này có vẻ đáng khích lệ: một số từ phổ biến có vẻ cụ thể đối với chủ đề của chúng. Những từ như "data", "science" và "season" có khả năng trở thành những yếu tố phân biệt tốt. Một điều quan trọng khác cần lưu ý là sự phong phú của các thuật ngữ ký tự đơn như "." và ","; chúng ta sẽ loại bỏ những thứ này.

Với kiến thức bổ sung này, hãy sửa lại tập lệnh chuẩn bị dữ liệu của chúng ta.

8.3.6 Xem lại bước 3: Chuẩn bị dữ liệu phù hợp

Việc khám phá dữ liệu ngắn này đã thu hút sự chú ý của chúng tôi đến một số chỉnh sửa rõ ràng mà chúng tôi có thể thực hiện để cải thiện văn bản của mình. Một điều quan trọng khác là stemming các thuật ngữ.

Danh sách sau đây cho thấy một thuật toán stemming đơn giản được gọi là “snowball stemming”. Snowball stemmer này có thể là ngôn ngữ cụ thể, vì vậy chúng tôi sẽ sử dụng ngôn ngữ tiếng Anh; tuy nhiên, nó hỗ trợ nhiều ngôn ngữ khác.

Liệt kê 8.6 Quá trình xử lý dữ liệu Reddit được sửa đổi sau khi khám phá dữ liệu

```
stemmer = nltk.SnowballStemmer("english")
def wordStemmer(wordrow):
    stemmed = [stemmer.stem(word) for word in wordrow]
    return stemmed

manual_stopwords = [',','.',')',',','(','m',"'m","n't",'e.g',"'ve",'s','#',’/
    ','```',"'s","'",'!','r',']','=','[','s','&','%','*','...','1','2','3',’
    4','5','6','7','8','9','10','--',"'",';','-',':']
def data_processing(sql,manual_stopwords):
    #create pointer to the sqlite data
    c.execute(sql)
    data = {'wordMatrix':[],'all_words':[]}
    interWordMatrix = []
    interWordList = []

    row = c.fetchone()
    while row is not None:
        tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+|[^\w\s]+’)

        wordrow = tokenizer.tokenize(row[0]+" "+row[1])
        wordrow_lowercased = lowerCaseArray(wordrow)
        wordrow_nostopwords = wordFilter(stopwords,wordrow_lowercased)

        wordrow_nostopwords =
            wordFilter(manual_stopwords,wordrow_nostopwords)
        wordrow_stemmed = wordStemmer(wordrow_nostopwords)

        interWordList.extend(wordrow_stemmed)
        interWordMatrix.append(wordrow_stemmed)

        row = c.fetchone()

    wordfreqs = nltk.FreqDist(interWordList) s
    hapaxes = wordfreqs.hapaxes()
    for wordvector in interWordMatrix:
        wordvector_nohapexes = wordFilter(hapaxes,wordvector)
        data['wordMatrix'].append(wordvector_nohapexes)
        data['all_words'].extend(wordvector_nohapexes)
   return data
for subject in subreddits:
    data[subject] = data_processing(sql='''SELECT
     topicTitle,topicText,topicCategory FROM topics
WHERE topicCategory = '''+"'"+subject+"'",
manual_stopwords=manual_stopwords)
```

Khởi tạo stemmer từ thư viện NLTK

Mảng stop words xác định các thuật ngữ để loại bỏ/bỏ qua

Bây giờ chúng tôi xác định sự chuẩn bị dữ liệu được sửa đổi.

Tìm nạp dữ liệu (các bài đăng trên reddit) từng cái một từ CSDL SQLite.

row[0] và row[1] chứa lần lượt tiêu đề và văn bản của bài đăng. Chúng tôi kết hợp chúng thành một văn bản blob đơn.

Loại bỏ thủ công stop word được thêm vào từ văn bản blob.

Danh sách từ tạm thời được sử dụng để loại bỏ hapaxes sau đó.

Ma trận từ tạm thời; sẽ trở thành ma trận cuối cùng sau khi loại bỏ hapaxes.

Lấy chủ đề mới.

Tạo tần suất phân phối tất cả các thuật ngữ.

Lấy danh sách hapaxes.

Vòng lặp qua ma trận từ tạm thời.

Loại bỏ hapaxes trong mỗi vectơ từ.

Mở rộng danh sách các thuật ngữ với vectơ từ chính xác.

Nối vectơ từ chính xác vào ma trận từ cuối cùng

Chạy hàm xử lí dữ liệu mới cho cả hai subreddit.

Lưu ý những thay đổi kể từ hàm `data_processing()` trước. Tokenizer của chúng tôi hiện là tokenizer biểu thức chính quy. Các biểu thức chính quy không phải là một phần của cuốn sách này và thường được coi là thách thức để thành thạo, nhưng tất cả những điều đơn giản này làm là cắt văn bản thành các từ. Đối với các từ, bất kỳ tổ hợp chữ và số nào cũng được phép (\w), vì vậy không có thêm ký tự đặc biệt hoặc dấu chấm câu nào. Chúng tôi cũng đã áp dụng từ gốc và loại bỏ danh sách các stop word thừa. Và, tất cả các hapax đều được loại bỏ ở cuối vì mọi thứ cần phải được xử lý trước. Hãy chạy lại quá trình chuẩn bị dữ liệu của chúng tôi.

Nếu chúng tôi thực hiện cùng một phân tích thăm dò như trước đây, chúng tôi sẽ thấy nó có ý nghĩa hơn và chúng tôi không có hapax nào nữa.

```
print wordfreqs_cat1.hapaxes()
print wordfreqs_cat2.hapaxes()
```

Hãy lấy lại 20 từ hàng đầu của mỗi loại (xem hình 8.19).

Top 20 most common "Data Science" terms after more intense data cleansing

```
wordfreqs_cat1 = nltk.FreqDist(data['datascience']['all_words'])
print wordfreqs_cat1.most_common(20)
```

```
[(u'data', 1971), (u'scienc', 955), (u'would', 418), (u'work', 368), (u'use', 34
7), (u'program', 343), (u'learn', 342), (u'like', 341), (u'get', 325), (u'scient
ist', 310), (u'job', 268), (u'cours', 265), (u'look', 257), (u'know', 239), (u's
tatist', 228), (u'want', 225), (u've', 223), (u'python', 205), (u'year', 204),
(u'time', 196)]
```

Top 20 most common "Game of Thrones" terms after more intense data cleansing

```
wordfreqs_cat2 = nltk.FreqDist(data['gameofthrones']['all_words'])
print wordfreqs_cat2.most_common(20)
```

```
[(u's5', 426), (u'spoiler', 374), (u'show', 362), (u'episod', 300), (u'think', 2
89), (u'would', 287), (u'season', 286), (u'like', 282), (u'book', 271), (u'one',
249), (u'get', 236), (u'sansa', 232), (u'scene', 216), (u'cersei', 213), (u'kno
w', 192), (u'go', 188), (u'king', 183), (u'throne', 181), (u'see', 177), (u'char
act', 177)]
```

Hình 8.19 Top 20 từ thường gặp nhất trong các bài đăng "data science" và "game of thrones" trên Reddit sau khi chuẩn bị dữ liệu

Chúng ta có thể thấy trong hình 8.19 chất lượng dữ liệu đã được cải thiện đáng kể như thế nào. Ngoài ra, hãy chú ý cách một số từ được rút ngắn do chúng tôi đã áp dụng stemming. Chẳng hạn, "science" và "sciences" đã trở thành "science"; "courses" và "course" đã trở thành "cours", v.v. Các thuật ngữ kết quả không phải là từ thực tế nhưng vẫn có thể hiểu được. Nếu bạn khăng khăng rằng các thuật ngữ của mình vẫn là các từ thực tế, thì lemmatization sẽ là cách tốt nhất.

Với quá trình làm sạch dữ liệu "đã hoàn thành" (nhận xét: bài tập làm sạch khai thác văn bản hầu như không bao giờ có thể được hoàn thành đầy đủ), tất cả những gì còn lại là một vài chuyển đổi dữ liệu để lấy dữ liệu ở định dạng túi từ.

Trước tiên, hãy gắn nhãn cho tất cả dữ liệu của chúng tôi và cũng tạo một mẫu lưu giữ gồm 100 quan sát cho mỗi danh mục, như được hiển thị trong danh sách sau.

Liệt kê 8.7 Chuyển đổi dữ liệu cuối cùng và chia tách dữ liệu trước khi lập mô hình

Mẫu giữ lại bao gồm dữ liệu chưa được gắn nhãn từ hai subreddits: 100 quan sát từ mỗi bộ dữ liệu. Các nhãn được giữ trong một bộ dữ liệu riêng biệt.

Mẫu giữ lại sẽ được sử dụng để xác định lỗi của mô hình bằng cách xây dựng ma trận nhầm lẫn.

```
holdoutLength = 100

labeled_data1 = [(word,'datascience') for word in
    data['datascience']['wordMatrix'][holdoutLength:]]
labeled_data2 = [(word,'gameofthrones') for word in
    data['gameofthrones']['wordMatrix'][holdoutLength:]]
labeled_data = []
labeled_data.extend(labeled_data1)
labeled_data.extend(labeled_data2)

holdout_data = data['datascience']['wordMatrix'][:holdoutLength]
holdout_data.extend(data['gameofthrones']['wordMatrix'][:holdoutLength])
holdout_data_labels = ([('datascience')
for _ in xrange(holdoutLength)] + [('gameofthrones') for _ in
    xrange(holdoutLength)])

data['datascience']['all_words_dedup'] =
list(OrderedDict.fromkeys(
data['datascience']['all_words']))
data['gameofthrones']['all_words_dedup'] =
list(OrderedDict.fromkeys(
data['gameofthrones']['all_words']))
all_words = []
all_words.extend(data['datascience']['all_words_dedup'])
all_words.extend(data['gameofthrones']['all_words_dedup'])
all_words_dedup = list(OrderedDict.fromkeys(all_words))

prepared_data = [({word: (word in x[0]) for word
in all_words_dedup}, x[1]) for x in labeled_data]
prepared_holdout_data = [({word: (word in x[0])
for word in all_words_dedup})
for x in holdout_data]

random.shuffle(prepared_data)
train_size = int(len(prepared_data) * 0.75)
train = prepared_data[:train_size]
test = prepared_data[train_size:]
```

Chúng tôi tạo một bộ dữ liệu duy nhất với mọi vectơ từ được gắn thẻ là 'datascience' hoặc 'gameofthrones'. Chúng tôi giữ một phần dữ liệu sang một bên cho mẫu giữ lại nắm giữ.

Một danh sách tất cả các thuật ngữ duy nhất được tạo để xây dựng túi từ dữ liệu mà chúng tôi cần để đào tạo hoặc chấm điểm một mô hình.

Dữ liệu cho đào tạo và kiểm thử được bố trí lại đầu tiên.

Dữ liệu được chuyển vào một túi từ định dạng nhị phân.

Kích thước của dữ liệu đào tạo sẽ là 75% tổng số và 25% còn lại sẽ được sử dụng để kiểm tra hiệu suất của mô hình.

Mẫu giữ lại sẽ được sử dụng cho thử nghiệm cuối cùng của chúng tôi về mô hình và tạo ma trận nhầm lẫn. Một *ma trận nhầm lẫn – confusion matrix* là một cách để kiểm tra xem một mô hình đã hoạt động tốt như thế nào trên dữ liệu chưa từng thấy trước đây. Ma trận cho biết có bao nhiêu quan sát được phân loại đúng và sai.

Trước khi tạo hoặc đào tạo và kiểm tra dữ liệu, chúng ta cần thực hiện một bước cuối cùng: đổ dữ liệu vào định dạng túi từ trong đó mọi thuật ngữ được gắn nhãn "Đúng" hoặc "Sai" tùy thuộc vào sự hiện diện của nó trong bài đăng cụ thể đó. Chúng tôi cũng cần làm điều này cho mẫu giữ lại chưa được gắn nhãn.

Dữ liệu đã chuẩn bị của chúng ta hiện chứa mọi thuật ngữ cho mỗi vectơ, như thể hiện trong hình 8.20.

```
print prepared_data[0]
```

```
print prepared_data[0]
({u'sunspear': False, u'profici': False, u'pardon': False, u'selye
s': False, u'four': False, u'davo': False, u'sleev': False, u'slee
                                ⋮
u'daeron': False, u'portion': False, u'emerg': False, u'fifti': Fals
e, u'decemb': False, u'defend': False, u'sincer': False}, 'datascien
ce')
```

Hình 8.20 Một túi nhị phân các từ sẵn sàng để lập mô hình là dữ liệu rất thưa thớt.

Chúng tôi đã tạo một ma trận lớn nhưng thưa thớt, cho phép chúng tôi áp dụng các kỹ thuật từ chương 5 nếu nó quá lớn để xử lý trên máy của chúng tôi. Tuy nhiên, với một bảng nhỏ như vậy, bây giờ không cần điều đó và chúng ta có thể tiến hành trộn và chia dữ liệu thành tập đào tạo và tập kiểm thử.

Mặc dù phần lớn nhất trong dữ liệu của bạn phải luôn được chuyển đến quá trình đào tạo mô hình, nhưng vẫn tồn tại một tỷ lệ phân chia tối ưu. Ở đây, chúng tôi đã chọn chia tỷ lệ 3-1, nhưng hãy thoải mái thử với điều này. Bạn càng có nhiều quan sát, bạn càng có nhiều tự do ở đây. Nếu bạn có ít quan sát, bạn sẽ cần phân bổ nhiều hơn tương đối để huấn luyện mô hình. Bây giờ chúng ta đã sẵn sàng chuyển sang phần bổ ích nhất: phân tích dữ liệu.

8.3.7 Bước 5: Phân tích dữ liệu

Đối với phân tích của chúng tôi, chúng tôi sẽ điều chỉnh hai thuật toán phân loại cho dữ liệu của mình: Naïve Bayes và cây quyết định. Naïve Bayes đã được giải thích trong chương 3 và cây quyết định ở trước đó trong chương này.

Trước tiên, hãy kiểm tra hiệu suất của bộ phân loại Naïve Bayes. NLTK đi kèm với bộ phân loại, nhưng bạn có thể thoải mái sử dụng các thuật toán từ gói khác, chẳng hạn như SciPy.

```
classifier = nltk.NaiveBayesClassifier.train(train)
```

Với trình phân loại được đào tạo, chúng tôi có thể sử dụng dữ liệu thử nghiệm để đo lường độ chính xác tổng thể.

```
nltk.classify.accuracy(classifier, test)
```

```
nltk.classify.accuracy(classifier, test)
```

```
0.9681528662420382
```

Hình 8.21 Độ chính xác của phân loại là thước đo thể hiện phần trăm các quan sát được phân loại chính xác trên dữ liệu thử nghiệm.

Độ chính xác của dữ liệu thử nghiệm được ước tính là lớn hơn 90%, như thể hiện trong hình 8.21. *Độ chính xác phân loại – Classification accuracy* là số quan sát được phân loại chính xác theo tỷ lệ phần trăm của tổng số quan sát. Tuy nhiên, xin lưu ý rằng điều này có thể khác trong trường hợp của bạn nếu bạn sử dụng dữ liệu khác.

```
nltk.classify.accuracy(classifier, test)
```

Đó là một con số tốt. Bây giờ chúng ta có thể ngả lưng và thư giãn, phải không? Không thật sự lắm. Hãy kiểm tra lại lần nữa trên mẫu giữ lại 200 quan sát và lần này hãy tạo một ma trận nhầm lẫn.

```
classified_data = classifier.classify_many(prepared_holdout_data)
cm = nltk.ConfusionMatrix(holdout_data_labels, classified_data)
print cm
```

Ma trận nhầm lẫn trong hình 8.22 cho chúng ta thấy 97% có thể cao hơn vì chúng ta có 28 (23 + 5) trường hợp phân loại sai. Một lần nữa, điều này có thể khác với dữ liệu của bạn nếu bạn tự điền vào tệp SQLite.

```
              |     g |
              |     a |
              |  d  m |
              |  a  e |
              |  t  o |
              |  a  f |
              |  s  t |
              |  c  h |
              |  i  r |
              |  e  o |
              |  n  n |
              |  c  e |
              |  e  s |
--------------+-------+
  datascience |<77>23 |
gameofthrones |  5<95>|
--------------+-------+
(row = reference; col = test)
```

Hình 8.22 Ma trận nhầm lẫn mô hình Naïve Bayes cho thấy 28 (23 + 5) quan sát trong số 200 quan sát bị phân loại sai

28 phân loại sai có nghĩa là chúng tôi có độ chính xác 86% trên mẫu giữ lại. Điều này cần được so sánh với việc chỉ định ngẫu nhiên một bài đăng mới cho nhóm “datascience” hoặc “gameofthrones”. Nếu chúng tôi chỉ định chúng một cách ngẫu nhiên, chúng tôi có thể mong đợi

một độ chính xác là 50% và mô hình của chúng tôi dường như hoạt động tốt hơn thế. Hãy xem những gì nó sử dụng để xác định các danh mục bằng cách đào sâu vào các tính năng mô hình có nhiều thông tin nhất.

```
print(classifier.show_most_informative_features(20))
```

Hình 8.23 cho thấy 20 thuật ngữ hàng đầu có khả năng phân biệt giữa hai loại.

```
Most Informative Features
                   data = True           datasc : gameof =    365.1 : 1.0
                  scene = True           gameof : datasc =     63.8 : 1.0
                 season = True           gameof : datasc =     62.4 : 1.0
                   king = True           gameof : datasc =     47.6 : 1.0
                     tv = True           gameof : datasc =     45.1 : 1.0
                   kill = True           gameof : datasc =     31.5 : 1.0
                compani = True           datasc : gameof =     28.5 : 1.0
                analysi = True           datasc : gameof =     27.1 : 1.0
                process = True           datasc : gameof =     25.5 : 1.0
                  appli = True           datasc : gameof =     25.5 : 1.0
               research = True           datasc : gameof =     23.2 : 1.0
                 episod = True           gameof : datasc =     22.2 : 1.0
                 market = True           datasc : gameof =     21.7 : 1.0
                  watch = True           gameof : datasc =     21.6 : 1.0
                    man = True           gameof : datasc =     21.0 : 1.0
                  north = True           gameof : datasc =     20.8 : 1.0
                     hi = True           datasc : gameof =     20.4 : 1.0
                  level = True           datasc : gameof =     19.1 : 1.0
                  learn = True           datasc : gameof =     16.9 : 1.0
                    job = True           datasc : gameof =     16.6 : 1.0
```

Hình 8.23 Các thuật ngữ quan trọng nhất trong mô hình phân loại Naïve Bayes

Thuật ngữ “data” được coi trọng và dường như là chỉ số quan trọng nhất cho biết liệu một chủ đề có thuộc danh mục khoa học dữ liệu hay không. Các thuật ngữ như “scene”, “season”, “king”, “tv” và “kill” là những dấu hiệu tốt cho chủ đề Game of Thrones hơn là khoa học dữ liệu. Tất cả những điều này đều có ý nghĩa hoàn hảo, vì vậy mô hình đã vượt qua cả kiểm tra độ chính xác và sự đúng đắn.

Naïve Bayes hoạt động tốt, vì vậy chúng ta hãy xem cây quyết định trong danh sách sau.

Liệt kê 8.8 Huấn luyện và đánh giá mô hình cây quyết định

Đào tạo phân loại cây quyết định

Bài kiểm tra độ chính xác phân loại

Tạo ma trận nhầm lẫn dựa trên kết quả phân loại và nhãn thực tế

```
classifier2 = nltk.DecisionTreeClassifier.train(train)
nltk.classify.accuracy(classifier2, test)
classified_data2 = classifier2.classify_many(prepared_holdout_data)
cm = nltk.ConfusionMatrix(holdout_data_labels, classified_data2)
print cm
```

Nỗ lực để phân loại dữ liệu giữ lại (ghi điểm)

Hiển thị ma trận nhầm lẫn

```
nltk.classify.accuracy(classifier2, test)
```

```
0.9333333333333333
```

Hình 8.24 Độ chính xác của mô hình cây quyết định

Như thể hiện trong hình 8.24, độ chính xác được hứa hẹn là 93%.

Bây giờ chúng ta biết rõ hơn là chỉ dựa vào thử nghiệm đơn lẻ này, vì vậy, một lần nữa chúng ta chuyển sang ma trận nhầm lẫn trên tập dữ liệu thứ hai, như thể hiện trong hình 8.25.

Hình 8.25 cho thấy một câu chuyện khác. Trên 200 quan sát này của mẫu nắm giữ, mô hình cây quyết định có xu hướng phân loại tốt khi bài viết về Game of Thrones nhưng thất bại thảm hại khi đối mặt với các bài đăng về khoa học dữ liệu. Có vẻ như mô hình có một sự ưu tiên cho Game of Thrones,và bạn có thể đổ lỗi cho nó? Hãy xem mô hình thực tế, mặc dù trong trường hợp này chúng ta sẽ sử dụng Naïve Bayes làm mô hình cuối cùng.

```
                    |      g |
                    |      a |
                    |  d   m |
                    |  a   e |
                    |  t   o |
                    |  a   f |
                    |  s   t |
                    |  c   h |
                    |  i   r |
                    |  e   o |
                    |  n   n |
                    |  c   e |
                    |  e   s |
--------------------+--------+
  datascience       |<26>74  |
gameofthrones       |  2<98>|
--------------------+--------+
(row = reference; col = test)
```

Hình 8.25 Ma trận hỗn loạn trên mô hình cây quyết định

```
print(classifier2.pseudocode(depth=4))
```

Cây quyết định, như tên gợi ý, là một mô hình dạng cây, như thể hiện trong hình 8.26.

Naïve Bayes xem xét tất cả các thuật ngữ và có trọng số được quy cho, nhưng mô hình cây quyết định thông qua chúng một cách tuần tự, theo con đường từ gốc đến các cành và lá bên ngoài. Hình 8.26 chỉ hiển thị bốn lớp trên cùng, bắt đầu với thuật ngữ “data”. Nếu "data" xuất hiện trong bài đăng, thì đó luôn là khoa học dữ liệu. Nếu không tìm thấy “data”, nó sẽ kiểm tra thuật ngữ “learn” và cứ thế tiếp tục. Một lý do có thể khiến cây quyết định này không hoạt động tốt là do thiếu sự cắt tỉa. Khi một cây quyết định được xây dựng, nó có nhiều lá, thường là quá nhiều. Sau đó, một cái cây được cắt tỉa ở một mức độ nhất định để giảm thiểu việc overfitting. Một lợi thế lớn của cây quyết định là hiệu ứng tương tác ngầm giữa các từ mà nó tính đến khi xây dựng các nhánh. Khi nhiều

```
if data == False:
  if learn == False:
    if python == False:
      if tool == False: return 'gameofthrones'
      if tool == True: return 'datascience'
    if python == True: return 'datascience'
  if learn == True:
    if go == False:
      if wrong == False: return 'datascience'
      if wrong == True: return 'gameofthrones'
    if go == True:
      if upload == False: return 'gameofthrones'
      if upload == True: return 'datascience'
if data == True: return 'datascience'
```

Hình 8.26 Biểu diễn cấu trúc cây mô hình cây quyết định

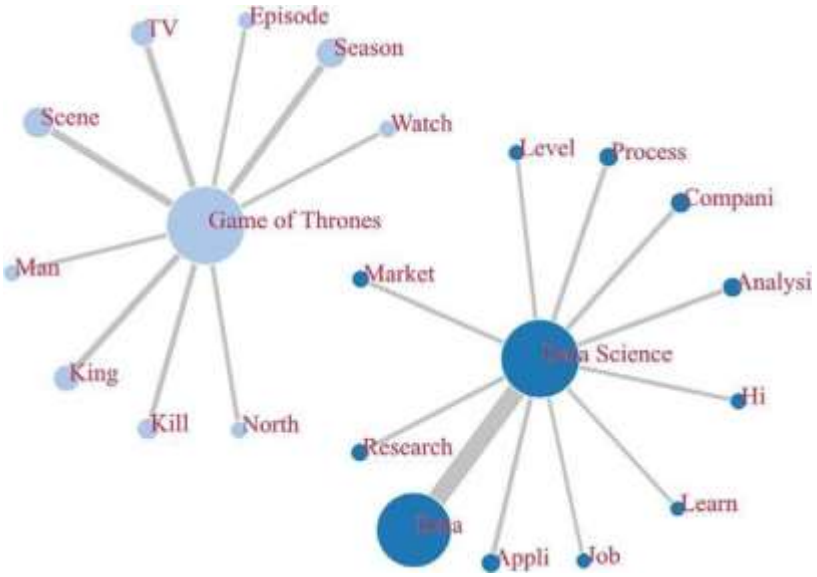
thuật ngữ cùng nhau tạo ra một phân loại mạnh hơn so với các thuật ngữ đơn lẻ, cây quyết định sẽ thực sự vượt trội so với Naïve Bayes. Chúng tôi sẽ không đi sâu vào chi tiết ở đây, nhưng hãy coi đây là một trong những bước tiếp theo mà bạn có thể thực hiện để cải thiện mô hình.

Hiện tại, chúng tôi có hai mô hình phân loại giúp chúng tôi hiểu rõ hơn về sự khác biệt giữa hai nội dung của các subreddits. Bước cuối cùng là chia sẻ thông tin mới tìm thấy này với những người khác.

8.3.8 Bước 6: Trình bày và tự động hóa

Bước cuối cùng, chúng ta cần sử dụng những gì đã học và biến nó thành một ứng dụng hữu ích hoặc trình bày kết quả của mình cho người khác. Chương cuối cùng của cuốn sách này thảo luận về việc xây dựng một ứng dụng tương tác, vì bản thân đây là một dự án. Bây giờ, chúng tôi sẵn lòng với một cách hay để truyền đạt những phát hiện của chúng tôi. Một biểu đồ đẹp hoặc tốt hơn nữa là một biểu đồ tương tác có thể bắt mắt; đó là lớp kem trên chiếc bánh thuyết trình. Mặc dù việc biểu diễn các con số như vậy hoặc biểu đồ thanh là dễ dàng và hấp dẫn nhất, nhưng có thể tốt hơn nếu tiến thêm một bước nữa.

Ví dụ: để biểu thị mô hình Naïve Bayes, chúng ta có thể sử dụng biểu đồ lực lượng (hình 8.27), trong đó bong bóng và kích thước liên kết biểu thị mức độ liên quan chặt chẽ của một từ với các subreddit “game of thrones” hoặc “data science”. Lưu ý cách các từ trên bong bóng thường bị cắt; hãy nhớ rằng điều này là do chúng tôi đã áp dụng stemming.



Hình 8.27 Đồ thị lực tương tác với 20 thuật ngữ Naïve Bayes có nghĩa hàng đầu và trọng lượng của chúng

Trong khi bản thân hình 8.27 là tĩnh, bạn có thể mở tệp HTML “forceGraph.html” để thưởng thức hiệu ứng biểu đồ lực d3.js như đã giải thích trước đó trong chương này. d3.js nằm ngoài phạm vi của cuốn sách này nhưng bạn không cần có kiến thức phức tạp về d3.js để sử dụng nó. Có thể sử dụng một tập hợp ví dụ mở rộng với các điều chỉnh tối thiểu đối với mã được cung cấp tại https://github.com/mbostock/d3/wiki/Gallery. Tất cả những gì bạn cần là ý thức chung và kiến

thức nhỏ về JavaScript. Bạn có thể tìm thấy mã cho ví dụ về biểu đồ lực tại http://bl.ocks.org/mbostock/4062045.

Chúng ta cũng có thể biểu diễn cây quyết định của mình theo một cách khá độc đáo. Chúng ta có thể sử dụng một phiên bản ưa thích của sơ đồ cây thực tế, nhưng sơ đồ sunburst sau đây nguyên bản hơn và không kém phần thú vị khi sử dụng.

Hình 8.28 cho thấy lớp trên cùng của biểu đồ sunburst. Có thể phóng to bằng cách nhấp vào một đoạn vòng tròn. Bạn có thể thu nhỏ lại bằng cách nhấp vào vòng tròn trung tâm. Bạn có thể tìm thấy mã cho ví dụ này tại http://bl.ocks.org/metmajer/5480307.

Hình 8.28 Biểu đồ Sunburst được tạo từ bốn nhánh trên cùng của mô hình cây quyết định

Hiển thị kết quả của bạn theo cách ban đầu có thể là chìa khóa cho một dự án thành công. Mọi người không bao giờ đánh giá cao nỗ lực bạn bỏ ra để đạt được kết quả nếu bạn không thể truyền đạt chúng và chúng có ý nghĩa với họ. Một trực quan hóa dữ liệu ban đầu ở đây và ở đó chắc chắn sẽ giúp ích cho việc này.

8.4 Tóm tắt

- Khai thác văn bản được sử dụng rộng rãi cho những thứ như nhận dạng thực thể, phát hiện đạo văn, nhận dạng chủ đề, dịch thuật, phát hiện gian lận, lọc thư rác, v.v.
- Python có bộ công cụ hoàn thiện để khai thác văn bản được gọi là NLTK hoặc bộ công cụ ngôn ngữ tự nhiên. NLTK rất tốt để thử xung quanh và tìm hiểu các liên kết; tuy nhiên, đối với các ứng dụng thực tế, Scikit-learning thường được coi là "sẵn sàng sản xuất" hơn. Scikit-learning được sử dụng rộng rãi trong các chương trước.
- Việc chuẩn bị dữ liệu của dữ liệu văn bản chuyên sâu hơn chuẩn bị dữ liệu số và bao gồm các kỹ thuật bổ sung, chẳng hạn như:
  - *Stemming*—Cắt phần cuối của một từ một cách thông minh để nó có thể được ghép với một số phiên bản liên hợp hoặc số nhiều của từ này.
  - *Lemmatization*—Giống như stemming, nó có nghĩa là loại bỏ các từ kép, nhưng không giống như stemming, nó xem xét cả nghĩa của từ.
  - *Stop word filtering*—Một số từ xuất hiện quá thường xuyên nên không hữu ích và việc lọc chúng ra có thể cải thiện đáng kể các mô hình. Các stop word thường là ngữ liệu cụ thể.
  - *Tokenization*—Cắt văn bản thành nhiều mảnh. Token có thể là từ đơn lẻ, tổ hợp từ (n-grams) hoặc thậm chí cả câu.
  - *POS Tagging*—Gắn thẻ một phần của bài phát biểu. Đôi khi có thể hữu ích khi biết chức năng của một từ nhất định trong câu là gì để hiểu nó tốt hơn.
- Trong nghiên cứu điển hình của mình, chúng tôi đã cố gắng phân biệt các bài đăng trên Reddit về "Game of Thrones" với các bài đăng về "data science". Trong nỗ lực này, chúng tôi đã thử cả Naïve Bayes và phân loại cây quyết định. Naïve Bayes giả định rằng tất cả các đặc điểm đều độc lập với nhau; bộ phân loại cây quyết định giả định sự phụ thuộc, cho phép các mô hình khác nhau.
- Trong ví dụ của chúng tôi, Naïve Bayes mang lại mô hình tốt hơn, nhưng thường thì trình phân loại cây quyết định thực hiện công việc tốt hơn, thường là khi có nhiều dữ liệu hơn.
- Chúng tôi đã xác định sự khác biệt về hiệu suất bằng cách sử dụng ma trận nhầm lẫn mà chúng tôi đã tính toán sau khi áp dụng cả hai mô hình trên dữ liệu mới (nhưng được gắn nhãn).
- Khi trình bày kết quả cho người khác, có thể hữu ích khi đưa vào hình ảnh trực quan hóa dữ liệu thú vị có khả năng truyền đạt kết quả của bạn theo cách dễ nhớ.