Chapter 9
Topic Detection and Tracking

9.1 History of Topic Detection and Tracking

Traditional TDT technology was established and developed in an evaluation-driven manner. The original motivation for TDT research was proposed by the Defense Advanced Research Projects Agency (DARPA) in 1996. Their aim was to explore a new technology to automatically detect and track topics in news data streams without human intervention.

In 1997, researchers from DARPA, Carnegie Mellon University (CMU), and University of Massachusetts (UMass) initiated preliminary studies of TDT, later called TDT1997 or TDT Pilot. They focused on how to find topic-related information from data streams (text or voice) and included two parts: enabling the system to automatically locate the boundaries of two events by searching for fragments consistent with the intrinsic theme and detecting the emergence of new events and the reproduction of old events. They carried out basic research (Allan et al. 1998a) and established a TDT pilot corpus.[1] This corpus includes nearly 16,000 stories, from July 1, 1994, to June 30, 1995, taken half from Reuters newswire and half from CNN broadcast news transcripts. For the evaluation of TDT performance, they proposed the metrics of miss and false alarm rates and used a detection error tradeoff (DET) plot to visually display the errors in the TDT system.

Starting in 1998, the National Institute of Standards and Technology (NIST), sponsored by DARPA, hosted the annual TDT evaluation conference, which was one of two conferences in the Translingual Information Detection, Extraction and Summarization (TIDES) project (the other is the Text REtrieval Conference, TREC). Many famous universities, companies, and research institutes, such as IBM Watson Research Center, BBN Technologies Company, CMU, UMass, University of Pennsylvania, University of Maryland, and Dragon Systems Company, actively participated in the conference. TDT1998 held the first public TDT evaluation, with

[1] https://catalog.ldc.upenn.edu/LDC98T25.

C. Zong et al., *Text Data Mining*, https://doi.org/10.1007/978-981-16-0100-2_9

the evaluation tasks including news story segmentation, topic detection, and topic tracking, and for the first time, it introduced the Chinese corpus. TDT1999 added two new tasks: first story detection (FSD) and link detection (LD).

TDT2002, the 5th TDT conference, held in autumn 2002, further enriched the corpus by incorporating the Arabic corpus. At the same time, new technologies such as text filtering, speech recognition, machine translation, and text segmentation were added to the research content of TDT.

TDT2004 canceled the task of news story segmentation because most instances of practical application were easily separable. Meanwhile, two new tasks, supervised adaptive topic tracking and hierarchical topic detection, were added. The TDT conference was held for 7 consecutive years, with the last one being TDT2004. The TDT corpus is still open to the public, and researchers can obtain the TDT tasks and corpus through the website of the Linguistic Data Consortium[2] (LDC).

In recent years, social media platforms, such as Twitter, Facebook, Weibo, and WeChat, have developed into important channels for discussions on current affairs, information exchanges, and the expression of opinions. A large number of users participate in discussions of events, persons, products, and other content on these platforms and generate a large amount of text data, which becomes a mirror reflecting society. The research on TDT in social media has even more important practical significance. However, at the same time, texts on social media platforms raise new problems and challenges to TDT research because of their properties, such as short contexts, rich forms, dynamic topics, massive volumes, and a large number of nonstandard language phenomena.

Allan (2012) and Yu et al. (2007) summarized the studies of traditional TDT. In the following sections, we first introduce the terminologies and tasks in TDT and then review the traditional technologies considering four aspects: text representation, text similarity, topic detection, and topic tracking. Finally, following the extension from traditional media to social media, we introduce the research of TDT in social media.

9.2 Terminology and Task Definition

9.2.1 Terminology

The goal of TDT is to automatically discover topics from text data streams and link topic-related content together. It involves concepts such as event, topic, story, and subject.

Event: In a TDT study, an event refers to an activity or a phenomenon that occurs at a specific time and place and is associated with certain

[2] https://www.ldc.upenn.edu/.

actions or conditions. Usually, an event is a story or a series of stories, which consist of detailed descriptions of the cause, time, place, process, and result of the event. For instance, "Trump defeated Hillary in the November 8, 2016, presidential election and became the 45th President of the United States" is an event in TDT. It has specific attributes such as a time, place, and person.

Topic: Topic was defined as an event in the original TDT Pilot study, but since TDT1998, it has been given a broader meaning that includes not only the initial event but also the subsequent events and other events directly related to it. In other words, a topic can be viewed as a core event together with its direct-relevant events, making it a collection of related stories about one event. Assume that the "512 Wenchuan earthquake" is a topic and "a strong earthquake of magnitude 8.2 occurred in Wenchuan, China, on May 12, 2008" is the core event of this topic. Subsequent events, such as earthquake rescue and post-earthquake reconstruction, are also part of the topic because they are directly related to the core event. The research on TDT originated from early event detection and tracking (EDT). However, compared with EDT, the object of TDT extends from events occurring at specific times and places to topics with more relevant extensions.

Subject: The subject in TDT is a summary of one kind of event or topic; it covers a group of similar events but does not involve any specific events. It therefore has a wider meaning than a topic in TDT. For instance, "earthquake disaster" is a subject, and the "512 Wenchuan earthquake" is a specific topic under that subject. Note that the concept of a "topic" in the topic model is different from that in TDT. Specifically, "topic" and "subject" in TDT are concepts describing events, representing a series for a specific event and a group of similar events, respectively, while "topic" in a topic model represents the underlying semantics of words in the text.

Story: A story in TDT denotes an article in a newswire or a piece of broadcast news that is composed of two or more statements of independent events.

9.2.2 Task

NIST divides TDT into the following five basic tasks.

(1) Story Segmentation

The purpose of story segmentation (SS) is to discover all topics and their boundaries in a news story and divide the story into multiple substories with a complete structure and independent topics, as shown in Fig. 9.1. For example, given a piece of broadcast news that includes multiple topics such as politics, sports events, finance, and economics, an SS system needs to divide the broadcast into

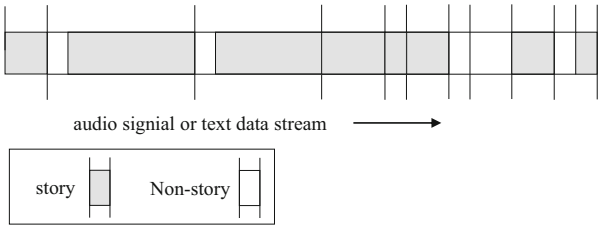

Fig. 9.1 Story segmentation task in TDT

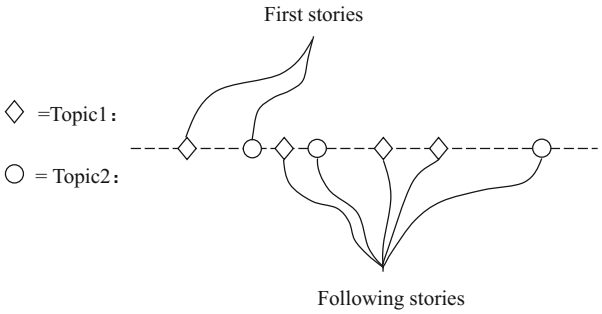

Fig. 9.2 First story detection task in TDT

segments of different topics. SS is designed mainly for news broadcasting, which contains two kinds of data streams: the audio signal and the text data stream transcribed from the audio signal. TDT2004 removed this task because most of the instances can be easily segmented in practice.

(2) First Story Detection

The FSD task aims to automatically detect the first discussion of a given topic from a chronological stream of news, as shown in Fig. 9.2. This task requires judging whether a new topic is discussed in each story. It is therefore considered to be the basis for topic detection, and it is called transparent testing of topic detection. TDT2004 renamed FSD to new event detection (NED).

(3) Topic Detection

The goal of the TD task is to detect topics in the news data streams without providing prior knowledge about any topics, as shown in Fig. 9.3. The output of FSD is one story, while the output of TD is a collection of stories that discuss the same topic. The difficulty of TD is the absence of prior knowledge of the topic; it means that the TD system must be independent of a certain topic but apply to any topic.

Although most of the stories refer to only one topic, there are also some stories that involve multiple topics organized in a hierarchical structure. In response to this problem, TDT2004 defined a hierarchical topic detection (HTD) task that

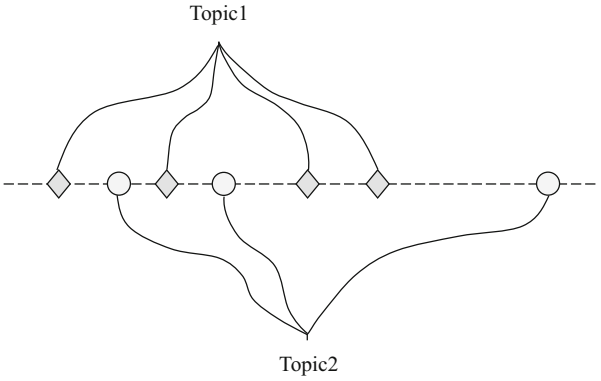

Fig. 9.3 Topic detection task in TDT

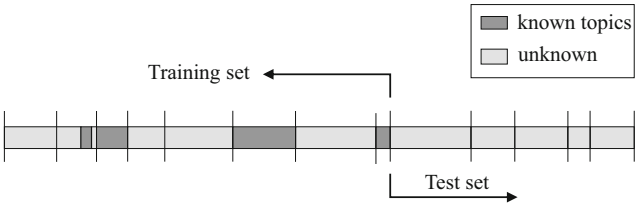

Fig. 9.4 Topic tracking task in TDT

transformed the organization of topics from a parallel relationship in FSD and TD to a hierarchical structure.

(4) Topic Tracking

The goal of topic tracking (TT) is to track subsequent stories of known topics, that is, to detect the related follow-up stories in the data streams given one or more stories associated with a topic, as shown in Fig. 9.4. The topic is denoted by several related stories rather than by a query (the NIST evaluation usually provided one to four stories for each topic).

(5) Link Detection

The goal of link detection (LD) is to judge whether two stories belong to the same topic, as shown in Fig. 9.5. Similar to TD, no prior knowledge is provided; the LD system establishes a topic relevance detection model that does not depend on stories from one topic as the reference. LD is often considered a core module in other TDT tasks (e.g., topic detection and topic tracking) rather than an independent task. A good link detection system can improve the performance of other TDT tasks.

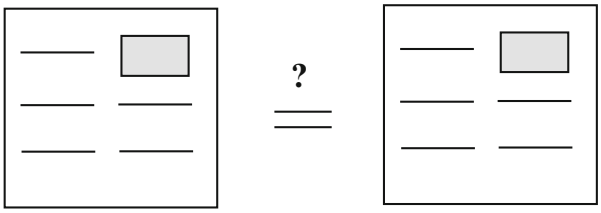

Fig. 9.5 Link detection in TDT

Table 9.1 Basic text mining techniques involved in TDT

Main task	Basic technique
Representation of topic/story	Text representation
Link detection	Text similarity computation
Topic detection	Text clustering
Topic tracking	Text classification

In general, TDT studies the relationship between stories and topics. It mainly solves the following technical problems: (a) representation of topics and stories; (b) similarity between topics and stories; (c) clustering of topics and stories; and (d) classification of topics and stories. Table 9.1 shows the text mining techniques involved in different TDT tasks.

9.3 Story/Topic Representation and Similarity Computation

In Chaps. 3, 5, and 6, we introduced the standard text representation and text similarity computation methods in detail. This section will briefly introduce the methods of text representation and similarity computation in TDT. Text preprocessing techniques, including stemming, lemmatization, and stop word filtering, are usually employed first. Then, a vector space model (VSM) or language model (LM) is often applied for text representation.

VSM is one of the most commonly used text representation models in TDT. It regards a story as a document, ignores the order of terms, and uses a vector to represent this document. TF-IDF and its variants are usually used as the term-weighting scheme. Allan et al. (2000) pointed out the limit of text similarity computation in VSM. Many researchers have proposed improving the representation ability of VSMs based on information extraction and feature engineering. For example, the information of the name entity (Yang and Liu 1999; Kumaran and Allan 2004, 2005), the 4Ws (who, what, when, where) (Kumaran and Allan 2004), and semantic concepts (Kumaran and Allan 2004) have been added to vector space to improve the performance.

There are three kinds of similarity measures between stories or topics: the similarity between two stories, the similarity between a story and a topic, and the similarity between two topics; these correspond to the content in Sect. 6.2.

Identifying the similarity between two stories is also called link detection in TDT. The goal of link detection is to determine whether two randomly selected stories discuss the same topic. A basic approach is as follows: first, each story is represented as a vector based on VSM, and then the similarity is calculated by the cosine distance of the two vectors. Finally, a preset threshold is used to determine whether the corresponding stories are relevant to each other. If the cosine similarity is greater than the threshold, the two stories are relevant; otherwise, they are irrelevant. The similarities between stories can also be measured by traditional Euclidean distance, the Pierson correlation coefficient, and other similarity measures.

The correlation between a story and a topic can be transformed into the problem of computing the similarities between the story and all the stories that constitute the topic, in which the key problem is link detection for a pair of stories. In some work, a topic is represented as a single model (e.g., using the centroid vector of all stories under the topic to represent the topic), thereby converting the similarity between a story and a topic into that between the story and the centroid vector, where the key technique is still link detection.

Researchers at the University of Massachusetts (UMass) studied a variety of similarity computation methods, including cosine distance, weighted sum, language models, and Kullback-Leibler divergence. Experiments on the TDT3 corpus showed that cosine distance performs best in link detection (Allan et al. 2000).

Another line of methods applies language models to story representation and link detection. The language model has been widely used in text mining as a generative probability model for representing natural language. Let the random variables C and S represent a topic and a story, respectively. According to Bayes' theorem, the posterior probability $p(C|S)$ of the topic C conditioned on the story S is proportional to the product of the prior probability $p(C)$ and the conditional probability $p(S|C)$, that is

$$p\left(C|S\right) = \frac{p(C)p(S|C)}{p(S)} \propto p\left(C\right) p\left(S|C\right) \tag{9.1}$$

Assuming that the terms in story S are independent of each other given the topic, we obtain

$$p\left(C|S\right) \propto p\left(C\right) \prod_i p\left(t_i|C\right) \tag{9.2}$$

where $p(t_i|C)$ is the probability that term t_i appears in topic C.

Language modeling furthermore provides a method for computing the similarity between a story and a topic (or two stories). In the unigram language model, a subset

C_j with respect to the j-th topic can be represented as a multinomial distribution as follows:

$$p\left(S|C_j\right) = \prod_i p(t_i|C_j) \tag{9.3}$$

where t_i denotes the ith term in the vocabulary. Based on maximum likelihood estimation, we can estimate $p(t_i|C_j)$ as the term frequency of t_i in C_j divided by the total number of terms in C_j.

In practice, the data sparsity problem may cause $p(t_i|C_j)$ to equal zero. To avoid this problem, we can use the smoothing technique to estimate $p(t_i|C_j)$

$$p_{\text{smooth}}\left(t_i|C_j\right) = \lambda p\left(t_i|C_j\right) + (1-\lambda)\, p\left(t_i|G\right) \tag{9.4}$$

where $p(t_i|G)$ is an estimated probability of word t_i in a general corpus G. Since the texts in TDT appear in a time series, and new texts may have words that did not appear in previous documents, as a kind of prior knowledge, estimation based on a general corpus is reasonable.

The problem determining which topic is most likely to generate the given story S can be described as

$$\arg\max_j \frac{p(S|C_j)}{p(S)} = \arg\max_j \prod_i \frac{p(t_i|C_j)}{p(t_i)} = \arg\max_j \log \prod_i \frac{p(t_i|C_j)}{p(t_i)} \tag{9.5}$$

Therefore, $D\left(S, C_j\right) = \sum_i \log \frac{p(t_i|C_j)}{p(t_i)}$ can be defined as the similarity between story S and topic C_j.

If a story is regarded as a distribution of words, then the similarity between a story S and a topic C can be measured by the similarity between two distributions, e.g., Kullback-Leibler divergence:

$$D_{\text{KL}}\left(C\|S\right) = -\sum_i p\left(t_i|C\right) \log \frac{p\left(t_i|S\right)}{p\left(t_i|C\right)} \tag{9.6}$$

Moreover, if two stories S_a and S_b to be compared are regarded as two multinomial distributions of terms, the Kullback-Leibler (KL) divergence can also be used for LD. Similarly, KL divergence can also measure the similarity between two topics. These techniques have been applied in Lavrenko and Croft (2001) and Leek et al. (2002).

On the basis of story/topic representation and similarity computation, most TDT tasks, such as topic detection and topic tracking, can be formulized as clustering or classification problems.

9.4 Topic Detection

The purpose of topic detection is to capture new (i.e., previously undefined) topics from a continuous stream of stories. The topic information, such as time, content, and number of stories, is unknown in advance, and there are also no annotated data for supervised learning. Therefore, topic detection is an unsupervised learning task and usually considered to be a clustering problem. Therefore, most topic detection algorithms can be regarded as a kind of modification or extension to standard text clustering algorithms. The standard clustering algorithms take the whole dataset as the input, while the input of topic detection is a continuous data stream of stories with a clear temporal relationship. The topics in the data stream also tend to change dynamically. These issues need to be addressed when using traditional clustering methods for topic detection.

Topic detection can be divided into two main types: online topic detection and retrospective topic detection. The input of online topic detection is a real-time story data stream, and thus subsequent stories do not yet exist. When a new story appears, the system is required to make a real-time decision on whether the story is a new topic. The input of retrospective topic detection is the whole corpus, containing all stories over time. Retrospective topic detection requires the system to decide for each story which topic it belongs to in an offline manner and to divide the whole corpus into several topic clusters accordingly. In comparison, the focus of online topic detection is to detect new topics from real-time data streams, while the purpose of retrospective topic detection is to discover previously unmarked news topics from existing stories.

In the following, we will describe the two topic detection tasks separately.

9.4.1 Online Topic Detection

Online topic detection aims to detect new topics from real-time stories. Since the information for new topics is unknown beforehand, it cannot be retrieved by a certain query. In addition, the task requires that the system make real-time decisions as soon as each story appears. For these reasons, incremental clustering algorithms are usually employed for online topic detection.

One simple method is based on single-pass clustering. The algorithm processes the input stories sequentially and represents each story based on a VSM. The model uses words (or phrases) as terms and TF-IDF (or its variants) as the term-weighting scheme to represent each story. Then, the similarities between the new story and all existing topics are computed. The similarity between a story and a topic is usually transformed into the similarity between the story and the centroid vector of the topic. If the similarity is higher than a preset merge-split threshold, the story will be classified into the most similar cluster (a cluster represents a topic); otherwise, the story will establish a new cluster. The above process is repeated until all the stories

in the data stream have been processed. This algorithm ultimately forms a set of flat clusters, where the number of clusters depends on the merge-split threshold. More details of the single-pass clustering algorithm can be found in Sect. 6.3.2.

In the early research into TDT, researchers at UMass and CMU adopted the single-pass clustering method (Allan et al. 1998b; Yang et al. 1998). To make the algorithm better suited to real-time data streams, they made some modifications to the text representation and similarity computations.

Specifically, Allan et al. (1998b) represented the content of a story as a query and compared it to all previous queries. If a new story triggers an existing query, it is assumed that the story discusses the topic corresponding to the triggered query. Otherwise, the story is considered to contain a new topic.

Assume that q is a query and denoted as a vector over a set of terms. Based on the term set, a document is represented as a representation vector $\boldsymbol{d}$. The correlation between a query $\boldsymbol{q}$ and a story $\boldsymbol{d}$ is defined as

$$\text{eval}\,(\boldsymbol{q}, \boldsymbol{d}) = \frac{\sum_{i=1}^{N} w_i \cdot d_i}{\sum\limits_{i=1}^{N} w_i} \tag{9.7}$$

where w_i represents the relative weight of a query term q_i and d_i is the appearance of term q_i in the story.

Because the future documents (i.e., stories) are unknown, the inverse document frequency (IDF) is estimated based on an auxiliary corpus c (which should belong to the same domain):

$$\text{idf}_i = \frac{\log \frac{|c|+0.5}{\text{df}_i}}{|c| + 1} \tag{9.8}$$

where df_i represents the document frequency of q_i in corpus c and $|c|$ is the number of documents contained in corpus c. Meanwhile, the average term frequency is calculated as

$$\text{tf}_i = \frac{t_i}{t_i + 0.5 + 1.5 \cdot \frac{\text{dl}}{\text{avg\_dl}}} \tag{9.9}$$

where t_i denotes the frequency of q_i in $\boldsymbol{d}$, dl is the length of $\boldsymbol{d}$, and avg_dl is the average length of all documents in c. On this basis, they set the weight of q_i as

$$\text{tw}_i = 0.4 + 0.6 \cdot \text{tf}_i \cdot \text{idf}_i \tag{9.10}$$

In addition, the features in query $\boldsymbol{q}$ are dynamic. Each time a new story appears, the top n high-frequency words of all existing documents in the data stream are selected to construct the new term set. Thus, all query representations in the past

need to be updated. The corresponding weight of q_i is the average value of tf_i in all existing stories.

Many studies have observed that documents that appear more closely in time in the data stream are more likely to discuss the same topic; therefore, using the timing of news stories may improve NED performance. Based on this idea, a time penalty was added to the threshold model. When the jth document in the data stream is compared with the ith query ($i < j$), $j - i$ is introduced to the threshold as a time penalty:

$$\theta\left(\boldsymbol{q}^{(i)}, \boldsymbol{d}^{(j)}\right) = 0.4 + p \cdot \left(\mathrm{eval}\left(\boldsymbol{q}^{(i)}, \boldsymbol{d}^{(j)}\right) - 0.4\right) + \mathrm{tp} \cdot (j - i) \tag{9.11}$$

where $\mathrm{eval}(\boldsymbol{q}^{(i)}, \boldsymbol{d}^{(i)})$ is the initial threshold of query $\boldsymbol{q}^{(i)}$, p is the weight of the initial threshold, and tp is the weight of the time penalty.

As mentioned in Sect. 6.3.2, the single-pass clustering algorithm is very sensitive to the order of the input sequence. Once the order changes, the clustering results may vary greatly. However, in TDT, the order of the input stories is fixed, which makes single-pass clustering highly suitable for TDT. Meanwhile, single-pass clustering has its advantage for real-time large-scale topic detection because it is simple and fast and supports online operations. The aforementioned work mainly involves three aspects of improvement upon standard single-pass clustering: (1) establish a better story representation, (2) find a more reasonable similarity computation method, and (3) make full use of the time information in the data stream.

9.4.2 Retrospective Topic Detection

The main goal of retrospective topic detection (GTD) is to review all news stories that have happened in the past and detect topics from them.

To address this task, the researchers at CMU proposed a hierarchical clustering algorithm based on group average clustering (Allan et al. 1998a; Yang et al. 1998), which has since been widely used in retrospective detection. This method adopts a divide-and-conquer strategy to hierarchical clustering: it divides the ordered story stream into several averaged buckets, adopts a bottom-up hierarchical clustering in each bucket, and then aggregates the more proximate clusters into a new cluster. Through repeated iterations, a topic cluster structure with a hierarchical relationship can ultimately be obtained.

Subsection 6.2.3 has already introduced bottom-up hierarchical clustering in detail. The basic idea is to initially treat each example as a separate cluster and then repeatedly merge the two most similar clusters until all examples have been merged into one cluster.

Finally, the algorithm constructs a hierarchical clustering dendrogram. The top level of the dendrogram represents a coarse-grained topic partition, and the lower level represents a more fine-grained topic partition. The time complexity of the

algorithm is $O(mn)$, where n is the number of stories in the corpus and m is the size of the bucket. The disadvantage of the algorithm is that it is only suitable for retrospective topic detection and cannot be applied to online topic detection.

9.5 Topic Tracking

The goal of topic tracking is to detect follow-up related stories from the news data stream given a small number of stories related to the topic as a priori knowledge.

On the one hand, topic tracking is related to information filtering in the information retrieval field. We can thereby perform topic tracking based on the information filtering techniques. The basic approach in topic tracking is to establish a query filter that takes a small number of stories to be tracked as positive examples, where the other stories are the negative examples. We then compute the similarity between the query and each subsequent story and finally determine whether the story matches the tracking topic by comparing the similarity score to a preset threshold. There are normally two ways to build a query filter in practice. The first focuses on how to better represent the topics to be tracked based on VSM, including establishing queries based on relevance feedback, extracting features based on shallow parsing, and attempting different feature weighting methods. The other is based on language modeling, which usually requires a large-scale background corpus.

On the other hand, topic tracking can also be viewed as two kinds of text classification tasks. Stories are categorized into two classes: the positive class denotes the relevance to the topic, and the negative class denotes irrelevance to the topic. A training set is constructed based on a small number of positive stories and a large number of negative stories, and a linear classifier is trained to predict the category of new stories.

CMU is the representative for research institutes using the k-NN classifier for topic tracking. Their algorithm incrementally builds a training set comprising positive and negative stories. When a new story appears, the similarities between it and each example in the training set are calculated. After comparing the similarity with a preset threshold, the new story is first classified as positive or negative. Then, the nearest k training examples are assessed to determine which topic the story belongs to. Although the k-NN method is simple and straightforward, the class imbalance problem (i.e., the number of negative samples is much higher than that of positive samples) makes it difficult to find a reasonable threshold for the algorithm. One improvement is a k-NN model based on positive and negative examples: the former k-NN is used to compute the similarity between a new story and positive examples S^+, while the latter is used to compute the similarity between a new story and the negative examples S^-. Last, a linear weighted combination of the two K-NN predictions is used for the final prediction.

Researchers from UMass used the Rocchio algorithm for topic tracking. They used three different term-weighting schemes for story representation and similarity

calculation. They also tried to dynamically adjust the topic vector during the tracking process.

Some researchers have employed decision trees for topic tracking. The major drawback of this method is that it can only give prediction results such as "yes" or "no" and cannot output a continuous prediction score, which is needed to produce a valid DET curve. Subsequent research includes introducing more information on news stories (such as "when," "where," and "who") into story and topic representation and constructing a strong topic tracker with an ensemble of multiple weak trackers.

Since the initial training data used to construct a topic model is normally very sparse, and there is also insufficient prior knowledge about the tracking topics, a topic tracking model that is trained based only on initial training data is often insufficient and inaccurate. Furthermore, because the topics are dynamic in topic tracking, the model cannot always track effectively after a period of time. To address this problem, some researchers proposed a new subtask called adaptive topic tracking (ATT), with the goal of adjusting the topic tracking model dynamically during the tracking process.

The work on ATT mainly focused on modifying the topic tracking model based on the system's pseudolabels. Most approaches established a dynamic term vector, adjusted the weight of terms dynamically, and trained the model in an incremental learning manner. The systems developed by the Dragon company (Yamron et al. 2000) and UMass (Connell et al. 2004) were the first to attempt unsupervised learning for ATT. The former added relevant stories into the training corpus and learned a new language model for topic tracking. The latter took the centroid of all prior stories as the representation of a topic and used the average correlation between prior stories and the centroid topic as the threshold. Each time a relevant story is detected during the follow-up process, it is added into the corpus, and the centroid and threshold are re-estimated correspondingly. By self-learning, ATT gradually adds pseudolabeled examples for model learning and modification, which reduces the limitation created by training only on the initial training corpus. However, the self-learning module in ATT is totally based on pseudolabeled examples. When the pseudolabels are not correct, this method can easily lead to the incorporation of irrelevant information, subsequently cause concept drift, and ultimately affect the performance of follow-up topic tracking.

9.6 Evaluation

TDT is an evaluation-driven technology. The TDT conferences have released five TDT corpora, including the TDT pilot corpus, TDT2, TDT3, TDT4, and TDT5. These corpora are provided by the Linguistic Data Consortium (LDC).

The corpora contain both broadcasting and text data except TDT5. The initial TDT corpus contained only English languages and subsequently added the Chinese and Arabic languages. Three annotations ("yes," "brief," and "no") were employed

in TDT2 and TDT3, and two annotations ("yes" and "no") were employed in TDT4 and TDT5, where "yes" means that the story and the topic are highly correlated, "brief" means that the correlation score is less than 10%, and "no" means that the two evaluate as uncorrelated. The broadcasting corpus includes not only news stories but also non-news stories such as commercial trade and financial stories, for which LDC provided three additional annotations: "news," "miscellaneous," and "untranscribed."

The TDT task can be essentially considered as a binary classification problem. Similar to the method of evaluating text classification described in Sect. 5.6, we can categorize the prediction results for TDT into four different cases, as shown in Table 9.2. By using the missed detection rate (MDR) and false alarm rate (FAR) as the basis, a DET curve can be plotted to observe the mistakes of a TDT system. Figure 9.6 is an example of the DET curve, where the x-axis is FAR and the y-axis is MDR. The closer the DET curve is to the lower-left corner of the coordinate, the better the TDT system performance is.

The performance of a TDT system can be quantified by a C_{Det} indicator defined as

$$C_{\text{Det}} = C_{\text{MD}} \cdot p_{\text{MD}} \cdot p_{\text{target}} + C_{\text{FA}} \cdot p_{\text{FA}} \cdot p_{\text{non\_target}} \tag{9.12}$$

where p_{MD} and p_{FA} are the conditional probabilities of missed detections (MD) and false alarms (FA), respectively, C_{MD} and C_{FA} are preset coefficients of MD and FA, p_{target} represents the prior probability of a target topic, and $p_{\text{non\_target}} = 1 - p_{\text{target}}$. C_{MD}, C_{FA}, and p_{target} are all preset parameters. The formulations of p_{MD} and p_{FA} are as follows:

$$p_{\text{MD}} = \frac{\#\text{missed\_detections}}{\#\text{targets}} \times 100\% \tag{9.13}$$

$$p_{\text{FA}} = \frac{\#\text{false\_alarms}}{\#\text{non\_targets}} \times 100\% \tag{9.14}$$

Generally, the normalized C_{Det} is used as the final performance of a TDT system:

$$C_{\text{Det-Norm}} = \frac{C_{\text{Det}}}{\min\left\{C_{\text{MD}} \cdot p_{\text{target}}, C_{\text{FA}} \cdot p_{\text{non\_target}}\right\}} \tag{9.15}$$

Table 9.2 Four kinds of prediction results from the TDT tasks

		Reference	
		Target	Non-target
Prediction	Yes	Correct	False alarm
	No	Missed detections	Correct

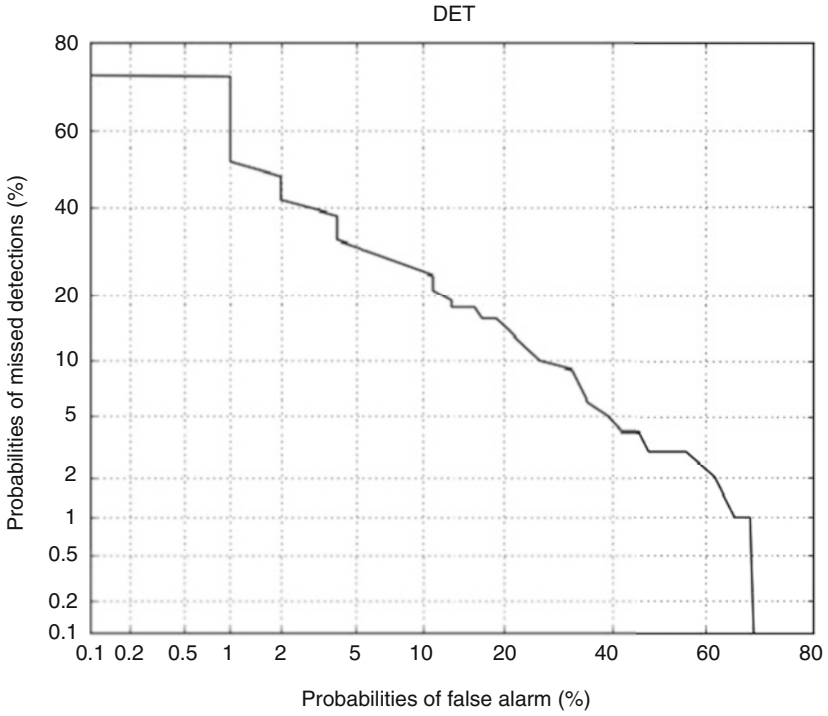

Fig. 9.6 TDT detection error tradeoff (DET) diagram

9.7 Social Media Topic Detection and Tracking

In recent years, the ways in which information is shared and disseminated on the Internet have gradually moved from the Web 1.0 era, which is represented by mainstream media websites, to the Web 2.0 era, which is represented by social media websites and applications. Traditional TDT mainly focuses on the content of traditional media, while social media TDT confront the following challenges: (1) the characteristics of user-generated context (UGC) in social media (e.g., short text, dynamic topic, irregular gramma, and diversified modals) increase the difficulty of text representation and TDT modeling; (2) the huge amount of data shared and propagated through social media brings great difficulty to real-time TDT; and (3) due to wide participation and openness, social media platforms are often the first site people use to report many emergencies. Therefore, bursty/breaking topic detection has attracted much attention in social media TDT.

In the following, we first introduce the differences between TDT in social media and traditional TDT and then introduce the main tasks and approaches of social media TDT. Lastly, we emphasize bursty topic detection in social media.

9.7.1 Social Media Topic Detection

The main goal of social media topic detection is to detect hot topics in the social media data stream. Similar to traditional topic detection, the social media topic detection task can also be divided into online topic detection and retrospective topic detection. However, due to the real-time nature of social media, more attention is being paid to online topic detection.

From the perspective of event types, social media topic detection can be categorized into specific and nonspecific topic detection. Specific topic detection aims at discovering historical topics that have already happened or detecting planned topics such as upcoming meetings or festival celebrations. Related information, such as the time, place, and main content of the known events, can be used to construct a topic detection model. Nonspecific topic detection focuses on detecting new topics from real-time data streams without any knowledge of the topic in advance (e.g., earthquakes) and collecting relevant follow-up stories. Nonspecific topic detection is the emphasis of social media topic detection.

(1) Specific Topic Detection

Specific topic detection methods can be divided into unsupervised and supervised machine learning methods. Similar to traditional topic detection, unsupervised topic detection methods in social media are mainly based on clustering or dynamic query expansion. The difference between traditional and unsupervised topic detection is that in addition to text content, the latter normally incorporates more social media-related information for topic representation and similarity calculation. For example, Lee and Sumiya (2010) proposed a local festival detection task from Twitter data streams. They found that the number of users and tweets will significantly increase when there are local festivals. They first collected Twitter data with geographical tags and then used the k-means algorithm to cluster these data and find topics in specific areas to detect local festivals. Massoudi et al. (2011) proposed a topic detection model for microblogs based on dynamic query expansion, in which they integrated text content and social media attributes such as emoji, hyperlink, number of fans, and number of retweets and replies for topic representation.

When the topic information is known in advance, such information can be used to compare with a labeled dataset. Then, supervised machine learning algorithms can be applied for topic detection. For example, Popescu and Pennacchiotti (2010) first collected a Twitter corpus and labeled it manually according to known topics. A supervised gradient boosted decision tree was then trained to detect controversial topics. They emphasized the importance of a rich and diverse feature set including hashtags, linguistic structure, and emotion features. Popescu et al. (2011) subsequently tried more features such as location and the number of replies. Supervised topic detection performs more effectively than unsupervised methods.

(2) Nonspecific Topic Detection

Information on nonspecific topics is unknown in advance. Traditional methods mainly use clustering to detect nonspecific topics, but the character of social media content makes these methods less effective.

On the one hand, some studies added social media-related features as new features for topic representation. For example, based on the classical incremental clustering algorithm (Allan et al. 1998a; Becker et al. 2011) explored the usage of retweets, replies, and mentions as features to detect social media topics. Feng et al. (2015) aggregated Twitter data in two dimensions (time and space) and designed a hashtag-based single-pass clustering method. Phuvipadawat and Murata (2010) concluded that accurate recognition of the proper name of an entity could help in the accurate calculation of text similarity and ultimately improve topic detection performance. The topics were then sorted by the number of fans and retweets to identify breaking news in the Twitter data stream.

On the other hand, some research tried to modify existing clustering algorithms or design new clustering algorithms to meet the requirements of social media applications. For example, Petrović et al. (2010) attempted to improve performance when applying traditional topic detection approaches to large-scale real-time data streams from social media. They further proposed an online NED method with constant time and space based on locality sensitive hashing. This method can effectively reduce the search space and significantly improve the efficiency of the system without decreasing the topic detection performance.

9.7.2 Social Media Topic Tracking

The main task of social media topic tracking is detecting microblogs related to existing topics from the social media data stream.

Similar to social media topic detection, existing studies mainly focus on how to use the special attributes of social media for topic representation and how to improve the sparseness of that representation. Phuvipadawat and Murata (2010) used rich social attributes such as URLs, hashtags, number of retweets, and user portraits to calculate the popularity of tweets and successfully tracked unexpected topics in social media. Lin et al. (2011) viewed the hashtag as a kind of topic indicator and used them to train a pretopic language model. Perplexity-based classifiers were then applied to filter the tweet stream to detect topics of interest.

9.8 Bursty Topic Detection

Bursty topic detection, also known as bursty/breaking event detection, refers to the detection of unexpected topics that develop rapidly in microblog data streams.

Bursty topic detection is different from traditional topic detection. Traditional topic detection emphasizes the detection of new topics without judging whether the detected topic is bursty or not. However, bursty topic detection focuses on the detection of topics' bursty features and bursty periods. Fung et al. (2005) divided bursty topic detection methods into document-pivot methods and feature-pivot methods. The former first detects topics through document clustering and then evaluates the burst of topics; the latter first extracts bursty features and then clusters these features to generate bursty topics.

The traditional topic detection approaches are mainly document-pivot methods. However, because the number of topics and stories is huge and hot topics change rapidly in social media, traditional document-pivot detection methods are often inefficient for social media bursty topic detection, and feature-pivot methods have attracted more attention.

Both document-pivot and feature-pivot methods need to identify the bursty status. The former usually recognizes burst states based on clustered topics, while the latter usually recognizes burst states based on feature discovery. In the following, we first introduce the classical burst status recognition algorithms and then review the representative document-pivot and feature-pivot bursty topic detection methods.

9.8.1 Burst State Detection

Kleinberg (2003) proposed a burst state detection algorithm for text data streams. It was later called the Kleinberg algorithm, and it has been widely used in bursty topic detection. The core idea of the algorithm is to simulate the time intervals between adjacent texts or sets of features in a data stream with an automation model to discover the optimal hidden state of the text at different time points. The states consist of a normal state and a burst state, which are denoted by different distributions of features, and the transition between states indicates the emergence or disappearance of a "burst."

In the Kleinberg algorithm, a text stream is organized into a sequence of messages, where each message has a corresponding arrival time. For a given term w, the algorithm records the arrival time of w and accordingly obtains a sequence of arrival times $\boldsymbol{t}^w = (t_0, t_1, \ldots, t_n)$. This determines a sequence of time intervals (called interarrival gaps) $\boldsymbol{x}^w = (x_1, \ldots, x_n)$ where $x_i = t_i - t_{i-1}$. If $\boldsymbol{x}^w$ is assumed to be generated by a binary state automaton, the problem will be transformed into a hidden Markov problem with the goal of solving the hidden state sequence with a known observation sequence. Finally, the bursty period is determined based on the obtained dynamic hidden state of the feature from the bursty and normal periods.

In detail, an exponential distribution is used to simulate the interarrival gaps. Suppose the interval x is distributed according to the density function as follows:

$$f(x) = \alpha e^{-\alpha x}, \quad \alpha > 0, x > 0 \tag{9.16}$$

and the corresponding cumulative distribution function is

$$F(x) = 1 - e^{-\alpha x}, \quad \alpha > 0, x > 0 \tag{9.17}$$

The expectation of x is α^{-1}, where α represents the arrival rate of the documents.

For a two-state model, a normal state q_0 (low state) and a burst state q_1 (high state) are defined. At each arrival time for w, the automaton must be in one of the states, which potentially affects the next arrival time of the w. The state will switch to another state or remain unchanged with a certain probability. Bursty topics are recognized as transitioning from a low state to a high state in a period of time.

As shown in Fig. 9.7, when a term is in a low state q_0, the interval x has a density function $f_0(x) = \alpha_0 e^{-\alpha_0 x}$. When a term is in a high state q_1, the interval x has a different density function $f_1(x) = \alpha_1 e^{-\alpha_1 x}$. Obviously, the arrival rate $\alpha_1 > \alpha_0$.

Suppose that the corresponding state sequence of $\boldsymbol{x}$ is $\boldsymbol{q} = (q_{i_1}, q_{i_2}, \ldots, q_{i_n})$, where $i_n \in \{0, 1\}$, the probability of state transition is p, and the number of state transitions in the sequence is b. Then, the density function for interval sequence $\boldsymbol{x}$ is

$$f_{\boldsymbol{q}}(\boldsymbol{x}) = \prod_{t=1}^{n} f_{i_t}(x_t) \tag{9.18}$$

and the prior probability of $\boldsymbol{q}$ is

$$p(\boldsymbol{q}) = p^b (1-p)^{n-b} \tag{9.19}$$

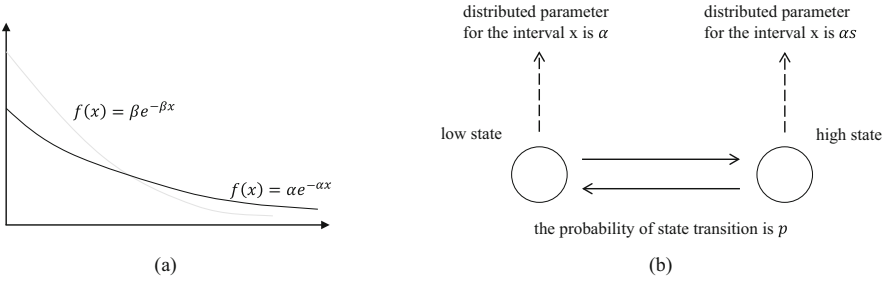

Fig. 9.7 (**a**) The distribution of the interval time for a normal and a burst state. (**b**) State transition model

According to Bayes' theorem, the posterior probability of $\boldsymbol{q}$ under $\boldsymbol{x}$ can be written as

$$p(\boldsymbol{q}|\boldsymbol{x}) = \frac{p(\boldsymbol{q}) f_q(\boldsymbol{x})}{\sum\limits_{\boldsymbol{q}'} p(\boldsymbol{q}') f_{\boldsymbol{q}'}(\boldsymbol{x})}$$
$$= \frac{1}{Z}\left(\frac{p}{1-p}\right)^b (1-p)^n \prod_{t=1}^{n} f_{i_t}(x_t) \tag{9.20}$$

where $Z = \frac{p(\boldsymbol{q}) f_q(\boldsymbol{x})}{\sum_{\boldsymbol{q}'} p(\boldsymbol{q}') f_{\boldsymbol{q}'}(\boldsymbol{x})}$.

The negative log-likelihood of the posterior distribution is

$$-\ln p(\boldsymbol{q}|\boldsymbol{x}) = b \ln\left(\frac{1-p}{p}\right) + \left(\sum_{t=1}^{n} -\ln f_{i_t}(x_t)\right) - n \ln(1-p) + \ln Z \tag{9.21}$$

where the third and fourth terms in the above formula are independent of $\boldsymbol{q}$. According to the maximum likelihood estimation, the following loss function can be defined:

$$c(\boldsymbol{q}|\boldsymbol{x}) = b \ln\left(\frac{1-p}{p}\right) + \left(\sum_{t=1}^{n} -\ln f_{i_t}(x_t)\right) \tag{9.22}$$

Determining the optimal hidden state sequence is equivalent to finding a state sequence that minimizes $c(\boldsymbol{q}|\boldsymbol{x})$. The first term in $c(\boldsymbol{q}|\boldsymbol{x})$ favors a sequence with a small number of state transitions, while the second term favors state sequences that conform well to the sequence $\boldsymbol{x}$ (i.e., making the value of the density function corresponding to each x_t as large as possible).

If each state in the state sequence $\boldsymbol{q}$ belongs to several continuous state levels $(q_0, q_1, \ldots, q_i, \ldots)$, the Kleinberg algorithm can be further extended from two states to an infinite number. The function $\tau(i, j)$ is defined to capture the loss of the transition from state s_i to state s_j. The transition loss from the low state to the high state is proportional to the number of intervening states, and the transition loss from the high state to the low state is 0:

$$\tau(i, j) = \begin{cases} (j-i)\gamma \ln n, & j > i \\ 0, & j \le i \end{cases} \tag{9.23}$$

where γ is the state transition control parameter (usually set to 1). Given the parameters s and γ, this automaton can be represented by $A^*_{s,\gamma}$ (asterisk denotes the infinite states). For a given interval sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, the algorithm's goal is to solve a state sequence $\boldsymbol{q} = (q_{i_1}, q_{i_2}, \ldots, q_{i_n})$ to minimize the cost function. Let $\delta(\boldsymbol{x}) = \min\limits_{i=1,\ldots,n} \{x_i\}$, and the maximum state level can be obtained

by $k = \lceil 1 + \log_s T + \log_s \delta(\boldsymbol{x})^{-1} \rceil$, where $\lceil \cdot \rceil$ is the ceiling function. It can be proven that if $\boldsymbol{q}^*$ is the optimal state sequence of automaton $A^k_{s,\gamma}$, it is also the optimal sequence of $A^*_{s,\gamma}$. Thus, the infinite state sequence optimization problem is transformed into the finite state optimization problem.

In the last step, a standard forward dynamic programming algorithm (such as the Viterbi algorithm) can be used to solve the above problem. Given an interval sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_t)$, the minimum loss sequence $C_j(t)$ can be expressed as follows:

$$C_j(t) = -\ln f_j(x_t) + \min_l (C_l(t-1) + \tau(l, j)) \tag{9.24}$$

$C_j(t)$ can be solved iteratively according to time t, where the initial state value is $C_0(0) = 0, C_j(0) = +\infty$. Finally, the optimal state sequence corresponding to $\boldsymbol{x}$ is obtained.

It is worth mentioning that the Kleinberg algorithm can detect bursts at the feature level (detecting the burst state of features/terms), as well as at the topic level (detecting the burst state of clustered topics). Therefore, it can be applied not only to feature-pivot bursty topic detection but also to document-pivot topic detection.

9.8.2 Document-Pivot Methods

Document-pivot methods first detect new topics from text data streams and then determine their burstiness. A traditional method is to first divide the text data stream into different windows according to the time of their appearance and perform clustering on the text in each window. Each cluster represents one topic, and features are extracted from the cluster to represent that topic. Finally, a bursty state recognition algorithm is applied to determine whether the topic is bursty or not.

Chen et al. (2013) first designed a strategy to obtain a real-time microblog data stream related to a given entity (such as a person or company name). For each time step t, a single-pass clustering algorithm is applied to the messages within the time window $[t - T, t]$ (T is the length of a unit window). The similarity between each message and the clustering centers is calculated. If the similarity is larger than the preset threshold, the message will be merged into the existing cluster; otherwise, the message constitutes a new cluster. Finally, each cluster is treated as a topic. The algorithm runs continuously to detect new topics in real-time data streams. They further established a semi-supervised classifier based on cotraining to detect whether the topics are burst or not. Figure 9.8 denotes a bursty topic evolution curve, where t_s denotes the time of one topic's occurrence, t_{hot} denotes the time the topic becomes hot, and the period $[t_s, t_{\text{hot}}]$ was defined as the bursty period. They labeled t_s and t_{hot} for each bursty topic in an offline training dataset. An SVM classifier was trained based on six features, including user growth rate, message growth rate, and response

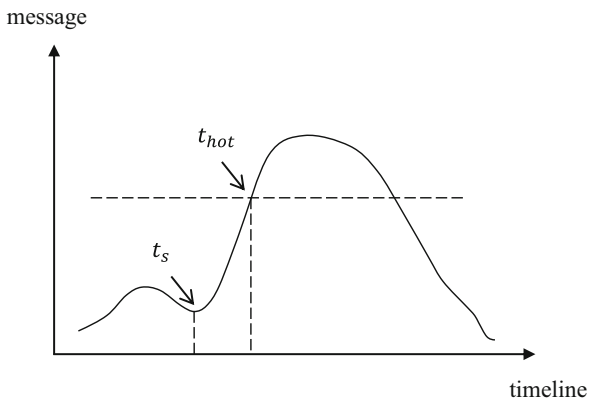

Fig. 9.8 The bursty period of one bursty topic

growth rate, and then it offered predictions for new topics detected from an online data stream as to whether they were bursty topics.

Diao et al. (2012) proposed a topic model called TimeUserLDA to detect bursty topics in social media data streams. The model was motivated by the finding that messages published at the same time are more likely to have the same topic and that messages published by the same author are also more likely to describe the same topic. Based on this, they incorporated the time and author information into a traditional LDA to model the messages. They mined a set of potential concepts C from a large-scale Twitter dataset, with each concept representing a topic in social media. For each topic $c \in C$, they calculated its occurrence frequency $(m_1^c, m_2^c, \ldots, m_T^c)$ along the time axis. Finally, they used an automaton similar to Kleinberg (2003) to identify the bursty topics.

Document-pivot methods are more suitable for topic detection in traditional media. As we have mentioned, the characteristics of social media, such as the short length, high volume, and broad topics, make document-pivot methods less efficient. Therefore, most of the applications for the bursty topic detection of social media are based on feature-pivot detection methods.

9.8.3 Feature-Pivot Methods

As shown in Fig. 9.9, the feature-pivot methods first discover a set of bursty features and then generate bursty topics. Here, "features" usually denote words or terms in texts. Text data streams are generally divided into equal-length and nonoverlapping time windows (such as "hours" or "days") in advance. Then, different kinds of methods, including feature selection methods, probabilistic methods, and the Kleinberg algorithm, are used to identify the bursty features.

One type of simple method uses the absolute or relative number of features and their changing speed as indicators for bursty feature selection. For example, for each

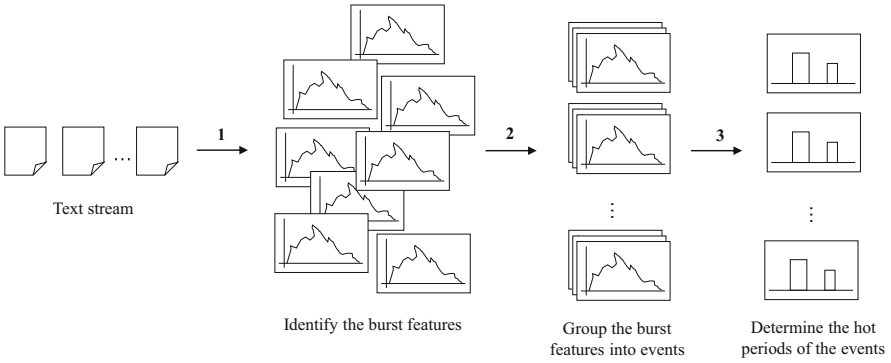

Fig. 9.9 Feature-pivot bursty topic detection methods

feature (e.g., words) in each time window, the relative word frequency $A_{ij} = \frac{F_{ij}}{F_{\max}}$ and the word frequency growth rate $B_{ij} = \frac{F_{ij} - F_{i(j-1)}}{1 + F_{i(j-1)}}$ are calculated. The features are ordered according to these indices, and a set of burst features is finally selected.

Fung et al. (2005) proposed a feature-pivot emergency detection method based on a probabilistic model. They first divided the text stream into $D = \{d_1, d_2, \ldots\}$, where d_i represents the text published on day i. Based on a binomial distribution, they identified a set of bursty features by comparing a feature's daily occurrence probability with its global occurrence probability. Subsequently, these bursty features were grouped into several bursty events, each of which comprised a subset of bursty features. Finally, the probability of the hot bursty event was calculated by computing the expected probability of the bursty event based on the subset of bursty features and comparing it with the expected value to determine the hot periods of each bursty event.

Other studies used spectrum analysis to detect bursty features and topics. For example, He et al. (2007a) employed discrete Fourier transform (DFT) to transform the time-series text data stream from the time domain to the frequency domain. The bursty topics in the time domain were supposed to correspond to the peaks in the frequency domain. The frequency domain attributes were then used to identify bursty features and their related periods.

He et al. (2007b) employed the Kleinberg algorithm to first recognize bursty features. For each bursty feature $f_j(t)$ in time window t, they calculated a bursty weight, which they combined with the static weight (e.g., TF-IDF weight) as a dynamic term-weighting scheme for bursty feature representation $\text{tf-idf}_{ij} + \delta w_j(t)$, where i is the index of documents and $\delta > 0$ is the bursty coefficient. Based on such a dynamic weight, topic clustering and classification experiments carried out on the TDT3 corpus achieved better performance than the traditional methods.

Cataldi et al. (2010) proposed a bursty feature extraction and bursty topic detection method based on content aging theory. First, nutrition is defined for each feature k under each time window TW^t taking into account the factors of word

frequency and user authority. The energy of feature k in time window TW^t is then defined as the mean square difference between the nutrient value of feature k in current time window TW^t and that in the previous s time windows. Energy is also used as an indicator to measure the burstiness of feature k in time window TW^t: the feature with a larger energy value has higher burstiness. By using all the features in window TW^t as candidates, the bursty feature set EK^t is obtained by sorting the candidates according to their energy value. Finally, a feature relation graph TG^t is constructed with features in a window as nodes and correlation coefficients between features as edge weights. The burst topics are then sorted and annotated based on strongly connected subgraphs containing bursty features.

9.9 Further Reading

At the beginning of this century, TDT was an active research direction in text mining. Recent development in this direction, on the one hand, is reflected in the change in its application (i.e., from traditional media to social media), which we have described in Sects. 9.7 and 9.8. In addition, several studies have attempted to apply the latest machine learning theory to this development. For example, Fang et al. (2016) improved the traditional feature space via word embedding to improve the performance of story and topic representation and similarity computation. However, there are few works of this type. Moreover, since clustering is the most frequent task in TDT and there are few deep learning-based clustering algorithms, research on TDT based on deep learning is also scarce.

Meanwhile, TDT is closely related to several hot areas of text mining, such as information retrieval, sentiment analysis, and event extraction. In comparison with information retrieval, extraction, and summarization, TDT emphasizes the abilities to detect, track, and integrate information. In addition, TDT usually addresses text data streams with temporal relationships rather than static texts. TDT can be used to monitor several kinds of information sources to capture new topics in time and to carry out historical research on the origin and development of topics. It has broad application prospects in many fields, such as information security, public opinion mining, and social media analysis. The joint technique of TDT and sentiment analysis can effectively detect not only hot topics but also people's views and opinions about that topic.

TDT also has a strong correlation with event extraction. The former emphasizes the automatic organization of macrolevel events in text data, while the latter emphasizes fine-grained event recognition and element extraction in a piece of text. The studies of TDT were driven by the TDT conferences, while research into event extraction focused on the evaluations of ACE (automatic content extraction) and KBP (knowledge base population).

Exercises

9.1 Please point out the similarities and differences between topic detection and topic tracking.

9.2 Compared with traditional news texts, what are the characteristics of text representation under social media?

9.3 What are the disadvantages of applying the standard single-pass clustering algorithm to the online topic detection of news stories? Do you have any solutions to these issues?

9.4 What is the difference between feature-pivot methods and document-pivot methods for bursty event detection?

9.5 Please derive the Kleinberg algorithm if there are five states (representing five-level burstiness).

9.6 How can the Kleinberg algorithm be used in the case of feature-pivot methods and document-pivot methods for bursty event detection, respectively?