Comparison of Short-Text Embeddings for Unsupervised Event Detection in a Stream of Tweets

Béatrice Mazoyer, Nicolas Hervé, Céline Hudelot and Julia Cagé

Abstract In this article, we apply recent short-text embeddings techniques to the automatic detection of events in a stream of tweets. We model this task as a dynamic clustering problem. Our experiments are conducted on a publicly available English corpus of tweets and on a French similar dataset annotated by our team. We show that recent techniques based on deep neural networks (ELMo, Universal Sentence Encoder, BERT, SBERT), although promising on many applications, are not well adapted for this task, even on the English corpus. We also experiment with different types of fine-tuning in order to improve the results of these models on French data. We propose a detailed analysis of the results obtained, showing the superiority of traditional tf-idf approaches for this type of task and corpus.

1 Introduction

In the field of language processing, recent works have made it possible to reach a performance level close to human capacity in several tasks, particularly when evaluating the semantic similarity between two sentences. However, these advances, based on the training of neural networks on very large corpora of texts, may not always be applicable to specific tasks.

Béatrice Mazoyer
Université Paris-Saclay, CentraleSupélec, Mathématiques et Informatique pour la Complexité et les Systèmes, 91190, Gif-sur-Yvette, France. e-mail: beatrice.mazoyer@centralesupelec.fr

Nicolas Hervé
Institut National de l'Audiovisuel, Bry-sur-Marne, France. e-mail: nherve@ina.fr

Céline Hudelot
Université Paris-Saclay, CentraleSupélec, Mathématiques et Informatique pour la Complexité et les Systèmes, 91190, Gif-sur-Yvette, France. e-mail: celine.hudelot@centralesupelec.fr

Julia Cagé
Department of Economics, Sciences Po, Paris, France. e-mail: julia.cage@sciencespo.fr

Indeed, despite rapid progress in recent years in the adaptability of language processing models (the GLUE benchmark[1] [Wang et al., 2018] consists of 9 different tasks, and the models are evaluated according to their average performance on all these tasks), it remains difficult to adapt these models to new tasks. In this work we focus on sentence topic similarity[2] (evaluate whether two sentences address the same subject), which in some cases differs from semantic similarity (evaluate whether two sentences mean the same thing). Even recent models, such as BERT [Devlin et al., 2019], which are intended to be easily adapted to new corpora, require some fine-tuning on (at least) a few thousand sentences, which involves hours of manual annotation to create a suitable dataset. Besides, even on a strictly identical task, the performances announced in the literature are perfectly reproducible only on English language corpora.

Finally, most of these models are designed to be used as input to end-to-end systems. These architectures do not apply well to information retrieval systems that involve comparing hundreds of thousands of sentences.[3] For clustering or information retrieval tasks, it is more efficient to represent each sentence in a vector space where similar sentences are close (so-called *embeddings*), and then apply conventional distance measurements (cosine, euclidean distance, etc.). With recent index structures, finding the most similar pair among 10,000 sentences can be done in milliseconds [Johnson et al., 2019].

In this paper, we test several text embeddings for the task of topical clustering of tweets: the objective is to group together tweets dealing with the same subject, in annotated corpora of tweets about media events. We compare a standard representation of documents in the form of tf-idf vectors [Sparck Jones, 1972] with more recent embeddings: Word2Vec [Mikolov et al., 2013], *Univsersal Sentence Encoder* [Cer et al., 2018], SBERT [Reimers and Gurevych, 2019]. In order to obtain text embeddings from deep neural networks, one can use the output of an intermediary layer, which we test for two models: ELMo [Peters et al., 2018] and BERT [Devlin et al., 2019].

Our experiments are conducted on a dataset of tweets in English [McMinn et al., 2013], as well as on a similar dataset of tweets in French [Mazoyer et al., 2020]. The code of our experiments is available online.[4] We show that text embeddings built using recent techniques based on deep neural networks do not improve the performance of our clustering algorithm, even on the dataset in English. We also perform several fine-tuning experiments in order to improve BERT and SBERT for the

[1] See the results on the GLUE benchmark: https://gluebenchmark.com/leaderboard

[2] More specifically, we are interested in the topic similarity of short texts, here tweets, which have the particularity of not always being grammatically correct. To simplify, we consider tweets as sentences in this article.

[3] For example, to calculate a similarity score between sentences with BERT [Devlin et al., 2019], it is necessary to process sentences in pair instead of individually. Using the example proposed by Reimers and Gurevych [2019], to find the two most similar sentences in a corpus of $n = 10,000$ sentences, the number of treatments to be performed would be $n\frac{(n-1)}{2} = 49,995,000$ operations, which represents approximately 65 hours of processing with BERT on a V100 GPU.

[4] https://github.com/ina-foss/twembeddings

French corpus. In this context, we show that using a dataset automatically translated from English to French is a potential way to obtain fine-tuning datasets. Finally, we propose a detailed analysis of the obtained results and show the superiority of classical tf-idf approaches for tweet clustering.

2 State of the Art

In this section, we first describe previous works on tweet clustering. We devote particular attention to the type of vector representations used. Then, we detail existing techniques in the field of sentence embeddings.

2.1 Tweet Clustering

Tweet clustering is the task of grouping together tweets dealing with the same subject. Typically, a tweets clustering algorithm is expected (i) to automatically infer the number of topics, (ii) to handle the inherent brevity of tweets, and (iii) to efficiently process very large volumes of data. We divide existing works on this subject into two families of approaches: topic models and incremental clustering.

2.1.1 Topic Models

A number of articles have specifically tackled the issue of applying topic models to short texts. The recent survey by Likhitha et al. [2019] summarizes the topic modeling techniques used to find topics within short text documents.

Based on the assumption that in the case of short texts, each document is generated from a single topic, Yin and Wang [2014] propose the Dirichlet Multinomial Mixture (DMM) model. In this model, the words within a document are assumed to be all sampled from the same multinomial distribution. The authors provide a collapsed Gibbs Sampling algorithm (GSDMM) to approximate the hidden variables in this generative process.

This method has been adapted and improved by several authors: Nguyen et al. [2015] integrate word embeddings into DMM by replacing the topic-word Dirichlet multinomial component with a draw from a Bernoulli distribution to determine whether the Dirichlet multinomial component or a word embedding component will be used to draw each term. Both Word2Vec [Mikolov et al., 2013] and GloVe [Pennington et al., 2014] are tested. Li et al. [2017] propose the Poisson-based Dirichlet Multinomial Mixture Model (PDMM), an extension of the DMM model allowing short texts to be generated from one to three topics and drawn from a Poisson distribution, and GPU-PDMM, which exploit word-embeddings to enrich the latent topic learning with external semantic relations. However, these advances seem to

increase calculation times of DMM by several orders of magnitude, while only increasing performance by a few percent (see the comparative table provided by Li et al. [2017]).

2.1.2 Incremental Clustering

Incremental clustering approaches do not require a fixed number of clusters. These methods are well fitted for discovering clusters of textual documents dynamically as new documents are added to the collection. For this type of clustering, the algorithms that are often used take into account both the thematic similarity of the documents and their temporal proximity, in order to avoid grouping in the same cluster tweets posted at very distant times. This type of approach generally uses the state-of-the-art First Story Detection (FSD) [Allan et al., 2000] algorithm as a reference method. In this method (see Algorithm 1), documents are represented using tf-idf and their distance is evaluated using cosine distance. The cosine similarity of two vectors u and v is computed as follows:

$$cos(u,v) = \frac{u \cdot v}{\|u\|\|v\|} \tag{1}$$

We denote $\delta(u,v)$ the cosine distance between u and v, which is defined as:

$$\delta(u,v) = 1 - cos(u,v) \tag{2}$$

Each new document joins the cluster of its nearest neighbor in the collection. If the cosine distance to the nearest neighbor is higher than a pre-defined threshold t, a new cluster is created containing the new document. The oldest documents are dropped from the collection after w tweets. This insures that each new document is only compared to the most recent documents. This algorithm is detailed below (see Algorithm 1) since this is the one we also use in this paper.

This technique has been applied to tweets in several works [Hasan et al., 2016; Petrović et al., 2010; Repp and Ramampiaro, 2018]. Petrović et al. [2010] propose a method based on Locality Sensitive Hashing to speed up the search for a neighbor, while Repp and Ramampiaro [2018] introduce mini-batches.

Sankaranarayanan et al. [2009] and Becker et al. [2011] use another algorithm based on the time-weighted distance between the average of the vectors of each cluster and each new tweet. Hasan et al. [2016] use the FSD algorithm in a first step, in order to determine whether a tweet is "unique" (i.e. whether if it is distant enough from the previous tweets), but the distance to the mean is used in a second step in order to assign a tweet to a cluster, in a way similar to Sankaranarayanan et al. [2009].

In this paper, we use the FSD algorithm because it is based on a simple distance metric, which is optimal for testing the quality of embeddings for clustering tasks. We introduce a "mini-batch" variant similarly to Repp and Ramampiaro [2018] in order to reduce processing time.

Algorithm 1: First Story Detection

input: threshold t, window size w, corpus C of documents in chronological order
output: a list T of cluster ids for each document

$T \leftarrow [] ; i \leftarrow 0$;
while *document d in C* **do**
  if *T is empty* **then**
    $cluster\_id(d) \leftarrow i$;
    $i \leftarrow i+1$;
  else
    $d_{nearest} \leftarrow$ nearest neighbor of d in T;
    if $\delta(d, d_{nearest}) < t$ **then**
      $cluster\_id(d) \leftarrow cluster\_id(d_{nearest})$;
    else
      $cluster\_id(d) \leftarrow i$;
      $i \leftarrow i+1$;
  if $|T| \geq w$;
  then
    remove first document from T
  add d to T;

In these previous papers, tweets are mostly represented as tf-idf vectors [Becker et al., 2011; Hasan et al., 2016; Petrović et al., 2010; Sankaranarayanan et al., 2009]. Repp and Ramampiaro [2018] test several embeddings (average Word2Vec, average GloVe, Doc2Vec, average Word2Vec weighted by tf-idf). However, these embeddings are tested on a classification task, and the embedding which allows the best classification results (average Word2Vec) is then used as input to the clustering algorithm. It therefore seems important to update these works by testing embeddings directly for clustering, especially those developed for the representation of sentences.

2.2 Text embeddings

The most commonly used "vectorization" method until the 2010s was the tf-idf, introduced by Sparck Jones [1972]. This is an improvement on the principle "bag of words" vectors [Harris, 1954], where each document is described by the number of occurrences of the words it contains ("term frequency"). The tf-idf representation uses the same vectors, but weights each of the words in inverse proportion to the number of documents in which it appears.

2.2.1 Word embeddings

The publication of Word2Vec [Mikolov et al., 2013] and GloVe [Pennington et al., 2014], two methods based on the prediction of the context of each word (or the prediction of each word depending on its context), made it possible to create word vectors carrying semantics other than the word frequency in the corpus. However, these representations lost the ability to describe each document by a single vector. To get around this problem, each document is often represented by an average of word vectors.

With ELMo [Peters et al., 2018] a new generation of models appears, allowing words to be represented not only depending on their usual context (the words with which they are frequently used in the training corpus), but also according to their local context: the word representation is specific to a given sentence. This constitutes an important advance in language processing, since a word no longer has a single representation that aggregates its different meanings, but several representations for each of the contexts in which it is used. ELMo is based on a bi-directional LSTM neural network trained to predict the next word in a sequence in both directions (i.e. to predict the next word in a sentence, but also, given the end of a sentence, to predict the word just before it). However, ELMo is not designed to produce sentence embeddings, but rather to be used as input to task-specific neural models. Nevertheless, the authors test the performance of word vectors directly from the first or second layer of their model (which contains three layers) for a disambiguation task by searching for the first nearest neighbor. The results obtained are close to the state of the art.

BERT [Devlin et al., 2019] is even more generic than ELMo, because this model does not require a specific architecture for each type of task: it can be fine-tuned to a new dataset by simply adding an output layer. BERT is built with a Transformer-type architecture [Vaswani et al., 2017], and pre-trained on two types of pretext tasks: predicting hidden words in a sentence and predicting the next sentence in a text. As in ELMo, the authors of BERT do not intend to create sentence embeddings from their model, but rather to provide an architecture that should be trained differently for each specific task. However, they demonstrate that a simple transfer learning (extraction of word vectors used at the input of a new model without fine-tuning) can match the state of the art for a named entities detection task.

2.2.2 Sentence embeddings

There is a large number of works that attempt to represent sentences by generic fixed size vectors. An important method is the Paragraph Vector algorithm [Le and Mikolov, 2014], frequently referred to as Doc2Vec. It is an extension of the Word2Vec model to the representation of text segments, where the model is trained to predict the next word in a context composed of a paragraph vector concatenated with word vectors.

Another approach is Skip-Thought [Kiros et al., 2015], an embedding model based on an encoder-decoder architecture trained to generate the sentences framing a given sentence in a text.

Conneau et al. [2017] show with InferSent, a bi-directional Siamese LSTM network (Siamese means that it takes two sentences as input, but applies the same weights in both parts of the network), that supervised learning provides better results for the creation of generic sentence vectors. InferSent is trained on a classification task using the SNLI dataset [Bowman et al., 2015], which contains $570,000$ pairs of English sentences manually annotated in three categories: the first sentence implies the second, the first sentence contradicts the second, or the first sentence and the second sentence are mutually neutral.

With Universal Sentence Encoder, Cer et al. [2018] apply the results of Kiros et al. [2015] and Conneau et al. [2017] by training a Transformer architecture both on unsupervised tasks, as was done for Skip-Thought, and on the SNLI dataset, like InferSent.

Sentence-BERT [Reimers and Gurevych, 2019] does not provide universal vectors, but a fine-tuning architecture of the BERT model specifically adapted to produce sentence embeddings adapted to certain types of tasks. This model modifies BERT into a Siamese network, with a final layer depending on the type of task on which the network is trained. The authors test their representations on the STS dataset [Cer et al., 2017] ($8,628$ pairs of sentences with a similarity score between 0 and 5) by computing a simple cosine similarity score between the vectors associated with each sentence. They show that the best performances on the STS dataset are obtained by a first fine-tuning on SNLI and then a second fine-tuning on the STS training set.

All these embedding methods are potential ways of representing the text of tweets. In the next Section, we detail the methods used to compare different types of text vectors for the task of tweet clustering.

3 Evaluation Methods

Evaluations of the "quality" of tweet representations can be made according to different approaches: first, test if the representation allows a good separability of the different classes (events). Second, make sure that the vectors produced are suitable for distance measurements, which are an important component in clustering algorithms. To evaluate the separability of classes, we initially reduce the task of unsupervised event detection to a classification task.

3.1 Classification

We use SVM classifiers with two types of kernel: a triangular kernel [Fleuret and Sahbi, 2003] and a Radial Basis Function (RBF) kernel.

The triangular kernel function is the following:

$$k(x,y) = 1 - \|x-y\| \tag{3}$$

Our experiments with this type of kernel show that, in addition to being invariant to scale changes [Fleuret and Sahbi, 2003], it can be applied effectively to both dense and sparse vectors. It achieves performance similar to parametric kernels on text classification, with the advantage of not requiring a lengthy grid search process to select the right parameters.

However, as this is not a very well-known form, we also conduct classification experiments with an RBF kernel:

$$k(x,y) = \exp(-\gamma\|x-y\|^2) \tag{4}$$

Both classifiers are trained on a random sample of 50% of the corpus. The classification is evaluated by the macro average of the F1 score of each class.

3.2 Clustering

Second, we model the event detection problem more realistically as dynamic clustering, using a modified FSD algorithm. Our change consists in introducing "mini-batches" of tweets in order to parallel the search for the nearest neighbor of each tweet in the batch. Taking documents in batches rather than one by one allows performance gains in the multiplication of matrices, which is the central step to compute cosine distance. This is particularly true in the case of sparse matrices, where we use the Sparse Matrix Multiplication Package (SMMP) algorithm [Bank and Douglas, 1993].[5] This modification may lower the performance of the algorithm, since the nearest neighbor of a given tweet may be in the same batch and therefore not found. In practice, our experiments show that for small batches, the performance remains unchanged (see Figure 1). Our version of the algorithm also deals with "empty" documents, that is documents that contain no words or only stop-words. This is frequently the case in datasets of tweets. Empty documents are not clustered, i.e. labeled as -1. Our version of the algorithm is described in Algorithm 2 below.

The parameters of this algorithm are w (number of past tweets among which we search for a nearest neighbor), t, the distance threshold above which a tweet is considered sufficiently distant from past tweets to form a new cluster, and b, the batch size. The value of b is set to 8 tweets for all our experiments. The value of w

[5] This algorithm is implemented by the Python Scipy library.

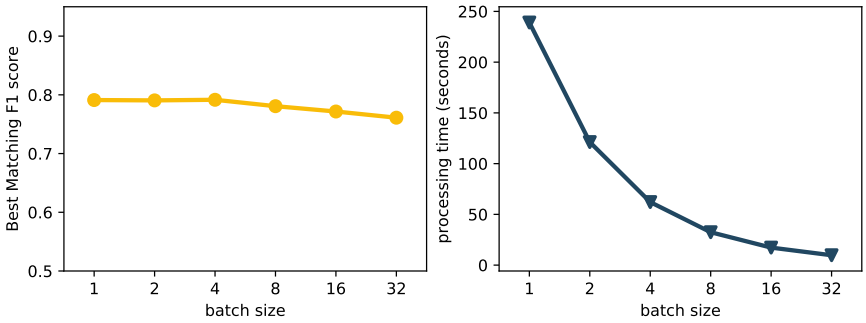


Fig. 1: Impact of batch size (in number of tweets) of the "mini-batch" FSD algorithm on performance (left) and on processing time (right). The tests are conducted on the French corpus (see Section 4.1.2) of 95,000 tweets, with tf-idf vectors.

Algorithm 2: "mini-batch" First Story Detection

input: threshold t, window size w, batch size b, corpus $C = \{d_0 \dots d_n\}$ of documents in chronological order
output: a list T of cluster ids for each document

$T \leftarrow [\,]$; $i \leftarrow 0$; $j \leftarrow 0$;
while $i < n$ **do**
  $batch = \{d_i, \dots d_{i+b-1}\}$;
  do in parallel
    for *document* d *in batch* **do**
      if T *is empty* **then**
        $cluster\_id(d) \leftarrow j$;
        $j \leftarrow j + 1$;
      else
        $d_{nearest} \leftarrow$ nearest neighbor of d in T;
        if $\delta(d, d_{nearest}) < t$ **then**
          $cluster\_id(d) \leftarrow cluster\_id(d_{nearest})$;
        else
          $cluster\_id(d) \leftarrow j$;
          $j \leftarrow j + 1$;
      if $|T| \geq w$; **then**
        remove first document from T
  add d to T;
  $i \leftarrow i + b$;

has been set differently for each corpus: it is set to approximately one day of tweets history, based on the average number of tweets per day in each corpus. We then tested different values of t for each type of representation. Generally speaking, lower t values lead to more frequent clustering, and thus better intra-cluster homogeneity (better precision), but may lead to over-clustering (lower recall).

Clustering performance is evaluated by a measure we call "best matching F1". It is defined by Yang et al. [1998]: we evaluate the F1 score of each pair between clusters (detected) and events (annotated). Each event is then matched to the cluster for which the F1 score is the best. Each event can be associated to only one cluster. The best matching F1 thus corresponds to the average of the F1s of the cluster/event pairs, once the matching is done.

4 Experiments

In this part we first describe the evaluation datasets, then the type of embeddings tested. Finally we detail the fine-tuning tests aimed at improving the performance of BERT and S-BERT on the French corpus.

4.1 Datasets

4.1.1 Corpus in English

The McMinn et al. [2013] dataset is the largest available corpus on Twitter event detection. The corpus consists of more than 100,000 tweets covering 505 events, within a larger corpus of 120 million tweets collected between October and November 2012. The annotation was performed with Amazon Mechanical Turk. Because of tweets being removed and Twitter accounts being closed, a large part of this dataset can no longer be recovered[6]. In August 2018, we could only retrieve 66.5 million tweets from the original collection (55%) and 72,790 tweets from the annotated corpus (72%).

4.1.2 Corpus in French

The French corpus [Mazoyer et al., 2020] was annotated by our team. It consists of 38 million tweets in French (retweets excluded) including more than 130,000 tweets manually annotated by three annotators as related or unrelated to a given event. The 257 events were selected both from press articles and from subjects trending

[6] In accordance with Twitter's terms of use, researchers sharing datasets of tweets do not share the content of the tweets, but only their ids. Using these identifiers, one can query the Twitter API to retrieve the full content of the tweets - but only if they are still online.

on Twitter during the annotation period (July to August 2018). In total, more than 95,000 tweets were annotated as related to one of the selected events.

4.2 Tested Embeddings

We chose models that have both French and English versions. This sub-section details the models used.

1. *Tf-idf.* Due to the inherent brevity of tweets, we simplified the calculation of tf-idf to a simple calculation of idf, since it is rare for a term to be used several times in the same tweet. The form used to calculate the weight of a term t in a tweet is therefore $idf(t) = 1 + log(n+1/df(t)+1)$, where n is the total number of documents in the corpus and $df(t)$ is the number of documents in the corpus that contain t.
   We have distinguished two calculation modes for n and $df(t)$: **tfidf-dataset** denotes the method that counts only annotated tweets, and **tfidf-all-tweets** denotes the calculation method that takes into account all tweets in the corpus (38 million tweets) to obtain n and $df(t)$. For each method, we restrict the vocabulary with a list of *stop-words* and a threshold df_{min}, the minimum number of tweets that must contain t for it to be included in the vocabulary. In all our experiments, $df_{min} = 10$. We thus obtain a vocabulary of nearly 330,000 words in English and 92,000 words in French for **tfidf-all-tweets**, and 5,000 words in English and 9,000 words in French for **tfidf-dataset**.
2. *Word2Vec.* We used pre-trained models for English, and trained our own French models. For each corpus, we distinguish between **w2v-twitter**, models trained on tweets, and **w2v-news**, models trained on press articles. For English, w2v-twitter is a pre-trained model published by Godin et al. [2015][7] (400 dimensions) and w2v-news is a model trained on Google News and published by Google[8] (300 dimensions). In French, w2v-twitter was trained with the CBOW algorithm on 370 million tweets collected between 2018 and 2019, and w2v-news was similarly trained on 1.9 million AFP dispatches collected between 2011 and 2019. Both models have 300 dimensions. As Word2Vec provides word embeddings and not sentence embeddings, the representation of tweets is obtained by averaging the word vectors of each word. Two methods were used for averaging: a simple average, and an idf-weighted average using the **tfidf-all-tweets** calculation method.
3. *ELMo.* For English, we used the model published on TensorFlow Hub[9]. For French, a model trained on French published by Che et al. [2018][10]. In each case, we use the average of the three layers of the network as a representation of

[7] github.com/loretoparisi/word2vec-twitter

[8] code.google.com/archive/p/word2vec/

[9] tfhub.dev/google/elmo/2

[10] github.com/HIT-SCIR/ELMoForManyLangs

each word. The representation of a tweet is produced by averaging these vectors (of dimension $1,024$).

4. *BERT.* Google provides an English model and a multilingual model[11]. In order to improve the performance of the multilingual model on French tweets, we continued training for 150,000 steps on tweets collected in June 2018. We refer to the simple multilingual model as **bert** and the model trained on tweets as **bert-tweets**. In each case, we used the penultimate layer of the network (of dimension 768) as embedding, by averaging the tokens to obtain a tweet representation.
5. *Universal Sentence Encoder.* The provided models (English[12] and multilingual[13]) are designed to provide sentence embeddings, so we were able to use them as is, without averaging vectors as in the previous representations. The vectors are of dimension 512.
6. *Sentence-BERT.* The authors of SBERT provide pre-trained models for English[14]. For French, we had to perform a fine-tuning of the multilingual BERT model, which we present in the next Section. The vectors obtained are of dimension 768.

4.3 Fine-tuning Sentence-BERT for French

The SBERT model is specifically trained to provide cosine similarity scores. Its architecture is presented in Section 2.2.2. The objective function is a mean-square-error loss between $cos(u,v)$ and the similarity score evaluated manually in the dataset. This model seems particularly suitable for a clustering algorithm based on cosine similarity, and indeed, among the sentence embeddings (*Universal Sentence Embedding* and SBERT), it is the one that provides the best clustering results in English.

However, the English pre-trained model is based on fine-tuning on supervised tasks (see 2.2.2 for details of SNLI and STS tasks), which cannot be done in French without an annotated corpus. We have therefore implemented two strategies to perform a fine-tuning[15] of the **bert-tweets** model on French data: first we have used *Cloud Translation API*[16] within the free use limit to translate part of the STS dataset (we obtained $2,984$ pairs of sentences in French). Second, we manually annotated 500 pairs of headlines of selected newspaper articles (we selected headlines that contained common terms). The annotation was done on a scale of 0 to 5, in the same way as for STS. However, instead of indicating the degree of semantic similarity be-

[11] github.com/google-research/bert models: bert-large, uncased and bert-base, multilingual cased

[12] tfhub.dev/google/universal-sentence-encoder-large/3

[13] tfhub.dev/google/universal-sentence-encoder-multilingual-large/1

[14] github.com/UKPLab/sentence-transformers. Model: bert-large-nli-stsb-mean-tokens

[15] We used the fine-tuning architecture provided by the authors of Sentence-BERT (sbert.net/docs/training/overview.html#sentence_transformers.SentenceTransformer.fit) with a batch size of 16, and 4 epochs. We have kept the default configuration for all other parameters.

[16] cloud.google.com/translate/docs/reference/rest/

tween the sentences, we sought to assess whether the two headlines described the same event. The two types of fine-tuning (translated corpus, or translated corpus + annotated corpus) are designated by **sbert-tweets-sts-short** and **sbert-tweets-sts-long**. The performances of the different representations are described in Section 5 below.

5 Results

Whether for the classification task or for the clustering task, none of the tested models manage to outperform the tf-idf model calculated on the whole corpus (tfidf-all-tweets). However, the relative performance of the models varies by language, and by task.

5.1 Classification results

Table 1: SVM classification results on the English corpus.

	Triangular Kernel		**RBF Kernel**
Model	*F1*	γ	*F1*
bert	74.49 $\pm$ 0.41	0.01	75.31 $\pm$ 0.51
elmo	59.81 $\pm$ 0.41	0.10	57.64 $\pm$ 0.36
sbert-nli-sts	80.55 $\pm$ 0.33	0.01	80.37 $\pm$ 0.36
tfidf-all-tweets	**83.50** $\pm$ 0.78	1.00	**81.86** $\pm$ 0.77
tfidf-dataset	**83.46** $\pm$ 0.72	1.00	**82.70** $\pm$ 0.69
use	80.26 $\pm$ 0.38	1.00	79.92 $\pm$ 0.46
w2v-news	81.35 $\pm$ 0.53	1.00	80.42 $\pm$ 0.55
w2v-news tfidf	**82.39** $\pm$ 0.64	0.01	**81.57** $\pm$ 0.73
w2v-twitter	76.68 $\pm$ 0.53	10.0	77.62 $\pm$ 0.62
w2v-twitter tfidf	81.20 $\pm$ 0.48	0.10	81.07 $\pm$ 0.49

Performance is assessed using the macro-average F1 score. Each cell indicates the mean and standard deviation of 5 runs (with different train/test splits), in percentages.

For English, SVM classification results are consistent and robust to kernel change (see Table 1). They show that BERT and ELMo do not provide easily separable short-text vectors. Models intended to be used as embeddings (Word2Vec, Universal Sentence Encoder, SBERT) obtain better results. The tf-idf vectors remain the best adapted, on a par with weighted w2v-news vectors.

On the French corpus (see Table 2), we can make the same conclusions for BERT and ELMo, even for the BERT model further trained on French tweets (bert-tweets): these models do not provide adequate embeddings for sentences or short texts. How-

Table 2: SVM classification results on the French corpus.

	Triangular Kernel	**RBF Kernel**	
Model	*F1*	γ	*F1*
bert	78.46 ± 0.68	0.01	79.08 ± 0.61
bert-tweets	81.77 ± 0.7	0.01	82.68 ± 0.72
elmo	73.59 ± 0.64	0.10	74.40 ± 0.70
sbert-tw-sts-l	86.08 ± 0.86	0.01	86.43 ± 0.81
tfidf-all-tweets	**87.79** ± 0.58	1.00	86.57 ± 0.55
tfidf-dataset	**87.66** ± 0.69	1.00	86.42 ± 0.53
use	**87.45** ± 0.60	1.00	**88.40** ± 0.75
w2v-news	**86.59** ± 0.80	1.00	86.28 ± 0.88
w2v-news tfidf	**87.51** ± 0.71	0.01	86.32 ± 0.77
w2v-twitter	**87.01** ± 0.56	1.00	86.60 ± 0.60
w2v-twitter tfidf	**87.73** ± 0.56	0.01	86.71 ± 0.60

Performance is assessed using the macro-average F1 score. Each cell indicates the mean and standard deviation of 5 runs (with different train/test splits), in percentages.

ever, the two kernels do not provide exactly the same results concerning the Universal Sentence Encoder model (use). With the RBF kernel, this model outperforms the other representations while, with the triangular kernel, tf-idf, Word2Vec and Universal Sentence Encoder perform equally well. In both cases, the difference between these latter models is rather small.

5.1.1 Clustering results

The tfidf-all-tweets vectors give the best results for the clustering task (see Table 3), even more clearly than for classification. This is due to the shape of the tf-idf vectors, which are particularly well suited for cosine similarity calculations, as well as to the event-specific characteristics of the two datasets: the same terms are obviously widely used among tweets of the same event. These results are consistent with those of Cagé et al. [2020], who came to similar conclusions regarding event detection in press articles.

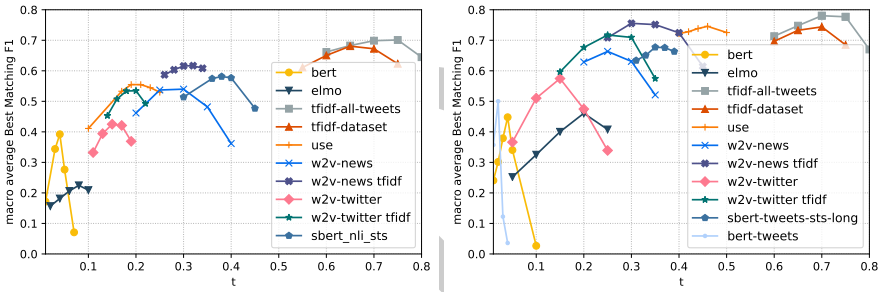
Concerning neural models adapted to sentence embeddings (SBERT, *Universal Sentence Encoder*), they do not perform better than the tf-idf weighted w2v-news models. On the English corpus, we note that the fine-tuning of *Sentence-BERT* on semantic similarity corpora (sbert-nli-sts) allows slightly better results than the generic vectors of *Universal Sentence Encoder*.

Our own fine-tuning of BERT (sbert-tweets-sts-short and sbert-tweets-sts-long) does not outperform *Universal Sentence Encoder* on the French corpus. However, we note that the thematic similarity corpus (which contains only 500 pairs of sentences) allows to increase the clustering performance by 2 points. Nevertheless, our fine-tuning does not achieve as good results as the English model, due to the fact that it could not be trained on a corpus of similar size to SNLI.

Table 3: FSD clustering results for each corpus.

	English		French	
Model	*t*	*F1*	*t*	*F1*
bert	0.04	39.22	0.04	44.79
bert-tweets	-	-	0.02	50.02
elmo	0.08	22.48	0.2	46.08
sbert-nli-sts	0.39	58.24	-	-
sbert-tweets-sts-long	-	-	0.36	67.89
sbert-tweets-sts-short	-	-	0.38	65.71
tfidf-all-tweets	0.75	**70.1**	0.7	**78.05**
tfidf-dataset	0.65	68.07	0.7	74.39
use	0.22	55.71	0.46	74.57
w2v-news	0.3	53.99	0.25	66.34
w2v-news tfidf-weights	0.31	61.81	0.3	75.55
w2v-twitter	0.16	43.2	0.15	57.53
w2v-twitter tfidf-weights	0.2	53.45	0.25	71.73

Performance is assessed using the macro-average "Best Matching F1" score. For each model, the best t threshold value was selected by successive tests. The batch size parameter b is fixed to 8. The window-size parameter w is fixed to the average number of tweets per day in each corpus.



(a) English (b) French

Fig. 2: Macro-average Best Matching F1 score for FSD clustering depending on the threshold parameter t for each corpus. The batch size parameter b is fixed to 8. The window-size parameter w is fixed to the average number of tweets per day in each corpus.

When used with the FSD algorithm, the best embeddings (tf-idf, Word2Vec with tf-idf weights, Universal Sentence Encoder) also have the property of being less sensitive to variations of the threshold t, as shown in Figure 2. Moreover, the optimal value of t for a given embedding seems to be approximately the same for each corpus (0.7 for tf-idf). This result may indicate that the First Story Detection algorithm could be applied to other (not annotated) tweet datasets without adapting the threshold value.

It is surprising that Word2Vec models trained on tweets are no better than models trained on news. However, this can be explained by two factors: the content of the datasets, which are made up of tweets referring to news, and whose vocabulary is therefore probably closer to the corpus from the AFP or from Google News than to the average Twitter. The second factor is the great variability in the vocabulary of tweets: it is possible that the w2v_twitter_en model, trained on tweets from 2015, does not correspond to the vocabulary used in the McMinn et al. [2013] corpus, collected in 2012.

The better performance on the French corpus than on the English corpus is probably due to the fact that classification and clustering on the English corpus are more difficult tasks, for several reasons: first, the English corpus is from 2012, and therefore a substantial share of the tweets have been deleted (our last download in November 2019 allowed us to retrieve 72,484 tweets i.e. 72% of the original annotated dataset). This can have important consequences, especially for the FSD algorithm which over-segments clusters if a tweet is missing to link them. Second, it seems that many events in the English corpus could be considered "sub-events" of the same macro event: "Hurricane Sandy in the Bahamas", "Tweets for Praying for people affected by hurricane Sandy", "Superstorm Sandy hits the east coast of the USA", "They all discuss about Sandy Storm" are for example four different events in the corpus by McMinn et al. [2013]. These events with very close topical similarity are probably more difficult to separate for the algorithm.

6 Conclusion

In this paper, we show that, out of a large panel of embedding methods, the tf-idf weighting remains the most suitable representation of documents for an algorithm such as FSD, which is based on nearest neighbor search with cosine similarity. In addition, tf-idf is not only the representation that provides the best results, but also the most stable to changes in the algorithm threshold parameter. These are important results for many researchers who are looking to use vector representations of short texts for Information Retrieval tasks.

References

Allan, J., Lavrenko, V., Malin, D., and Swan, R. (2000). Detections, bounds, and timelines: Umass and tdt-3. In *Proc. of Topic Detection and Tracking workshop*, pages 167–174.

Bank, R. E. and Douglas, C. C. (1993). Sparse matrix multiplication package (SMMP). *Adv. Comput. Math.*, 1(1):127–137.

Becker, H., Naaman, M., and Gravano, L. (2011). Beyond trending topics: Real-world event identification on twitter. In *Fifth international AAAI conference on*

weblogs and social media.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642. The Association for Computational Linguistics.

Cagé, J., Hervé, N., and Viaud, M.-L. (2020). The production of information in an online world. *The Review of Economic Studies*.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Che, W., Liu, Y., Wang, Y., Zheng, B., and Liu, T. (2018). Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Fleuret, F. and Sahbi, H. (2003). Scale-invariance of support vector machines based on the triangular kernel. In *3rd International Workshop on Statistical and Computational Theories of Vision*, pages 1–13.

Godin, F., Vandersmissen, B., De Neve, W., and Van de Walle, R. (2015). Multimedia lab @ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proc. of Workshop on Noisy User-generated Text*, pages 146–153.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hasan, M., Orgun, M. A., and Schwitter, R. (2016). TwitterNews: Real time event detection from the Twitter data stream. *PeerJ PrePrints*.

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Li, C., Duan, Y., Wang, H., Zhang, Z., Sun, A., and Ma, Z. (2017). Enhancing topic modeling for short texts with auxiliary word embeddings. *ACM Trans. Inf. Syst.*, 36(2):11:1–11:30.

Likhitha, S., Harish, B., and Kumar, H. K. (2019). A detailed survey on topic modeling for document and short text data. *International Journal of Computer Applications*, 975:8887.

Mazoyer, B., Cagé, J., Hervé, N., and Hudelot, C. (2020). A french corpus for event detection on twitter. In *Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020)*, pages 6220–6227.

McMinn, A. J., Moshfeghi, Y., and Jose, J. M. (2013). Building a large-scale corpus for evaluating event detection on twitter. In *Proc. of ACM-CIKM*, pages 409–418. ACM.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nguyen, D. Q., Billingsley, R., Du, L., and Johnson, M. (2015). Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Petrović, S., Osborne, M., and Lavrenko, V. (2010). Streaming first story detection with application to Twitter. In *Proc. of NAACL*, pages 181–189.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Repp, Ø. and Ramampiaro, H. (2018). Extracting news events from microblogs. *Journal of Statistics and Management Systems*, 21(4):695–723.

Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., and Sperling, J. (2009). Twitterstand: news in tweets. In *Proc. of ACM-GIS*, pages 42–51.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355. Association for Computational Linguistics.

Yang, Y., Pierce, T., and Carbonell, J. G. (1998). A study of retrospective and on-line event detection. In *Proc. of ACM-SIGIR*, pages 28–36.

Yin, J. and Wang, J. (2014). A dirichlet multinomial mixture model-based approach for short text clustering. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA*, pages 233–242.