Multimodal document model

MQ

April 24, 2017

Abstract

1 Introduction

1.1 Definitions

Tweet A *tweet* can be seen as a struct with the following fields:

- *text*: the textual content of the tweet.
- *creation_date*: a timestamp when the tweet was published.
- *no_retweets*: the amount of times that tweet was shared or forwarded by other users.
- *no_likes*: the amount of times users "liked" the tweet.
- *user*: an identifier of the user who published the tweet.

Summary Given a set of tweets $E = \{t_1, t_2, \ldots, t_N\}$, called an *event*, we want to select a subset $S \subseteq E$ of tweets, called a *summary*. The summary must fulfill the following criteria:

- **Topical coverage**: the tweets in S must cover the same topics as E.
- **Redundancy**: the content of tweets in S must not be redundant with each other.

- **Importance**: the tweets in S must be the top $|S|$, with respect to E, according to a pre-defined ranking function, considering into account the previous two criteria. For example, if two tweets have the same value according to the ranking, but the two of them are equal in terms of content, then only one of them should be in S.
- **Human-manageable size**: the size of S must be of much less size than E, only if E is large. **(TODO: define what is "large" and how less is "less.")**

Replies and retweets We denote by $URL(t) = \{u_1, u_2, \ldots, u_m\}$ the URLs shared by the tweet t. $URL(t)$ is empty if t does not share any URL.

We also denote by $replies(t)$ the set of all tweets t' such that t' is a *reply* of t, or t' is a reply of another tweet in $replies(t)$. The same applies to $retweets(t)$, but by considering *retweets* instead of replies.

Document We now define a *document* d_u as a set of tweets, such that those tweets share the same URL u, plus their replies and retweets, that is,

$$d_u = \{t \ : \ u \in URL(t) \vee \exists t'(t \in replies(t') \vee t \in retweets(t')) \wedge u \in URL(t')\}.$$

Note that a tweet t can be a member of different documents, if t shares more than one URL.

1.2 Methodology

We make use of the context of multiple tweets in order to arrange them into topically similar groups. When a tweet shares an URL u, the content of the tweet can be seen as a description (or *anchor text*) or a comment on the content of u. This also applies when a tweet is a *reply* of another tweet: those two tweets (the reply and the replied) are topically similar, because both of them refers to the same subject of discussion. We use this context to group tweets into documents.

Given an event E, let U be the set of all the URLs shared across tweets in E. The documents induced by U are all the subsets of tweets that share at least one URL, $D = \{d_u \ : \ u \in U\}$. Our goal is to select representative tweets to create S, by using D as a proxy by grouping similar tweets into documents.

One main task is to compare documents. Two documents whose content is topically similar should be similar according to the features of its constituents. Note that the documents may have very different sizes, and it is possible that two different documents are topically similar. This makes comparing documents a difficult task.

By comparing documents, our goal is to discover sub-topics inside E, in order to achieve coverage in the resulting summary. Possible implementations for sub-topic identification are clustering (e.g. K-Means, K-Medoids, hierarchical, or online/incremental), or process an induced graph from the documents (e.g. community detection, connected components, or centrality measures). Another alternative, which does not require computing a similarity measure between documents, is the use of topic modelling (LDA or Dynamic Topic Models if the time dimension is considered).

Similarity between documents There are two alternatives when computing similarity between documents:

1. *Consider all the elements of a document as a single element.* For example, compute a vector space model (e.g. tf-idf) over the concatenation of tweets in a document and then compare the representations using standard cosine similarity. A problem with this approach is that diverse content inside a document can be shadowed by the most popular content inside a document. Or, on the other hand, if there is a lot of diverse content, then the focus of the document can be inaccurate, compared by a smaller, focused document.

2. *Consider each element of a document as a single element.* For example, compare individual pairs of tweets of different documents, and then compute an average of similarities to assess the similarity of the two documents. This could share a similar problem with the other alternative, as diverse content may diffuse the main focus of a document (or be shadowed by the popular content). Another alternative is to derive high similarity between unequally-sized documents if *some* of the tweets in the larger document are *very* similar to the tweets in the smaller one.

   Another problem is the sparsity of vocabulary of tweets, if we compare them one by one. But this can be settled by using more information about the tweets (for example, by using a word embedding to compute similarity between words).

The second approach may be better adjusted to the specific domain, where certain topics of an event are way more popular than the rest.