# Filtering Entity Centric Documents using Numerics and Temporals features within RF Classifier

**Vincent Bouvier**
Kware / Aix-Marseille Université
CNRS, LSIS UMR 7296
Domaine universitaire de Saint Jérôme
Avenue Escadrille Normandie Niemen
13397 MARSEILLE Cedex 20
vincent.bouvier@kware.fr

**Patrice Bellot**
Aix-Marseille Université
CNRS, LSIS UMR 7296
Domaine universitaire de Saint Jérôme
Avenue Escadrille Normandie Niemen
13397 MARSEILLE Cedex 20
patrice.bellot@lsis.org

## 1 Introduction

This article introduces the system designed to work for the Knowledge Base Acceleration (KBA) track at Text REtrieval Conference (TREC). This is the second time this track is run with yet this year there is even more challenge.

KBA is focused on keeping up to date knowledge bases (KB) such as Wikipedia[1] (WP) where each KB node is considered as an entity (WP page for wikipedia example). It has been shown in (Frank et al., 2012) that the time lag between the publication date of cited news articles and the date of an edit to wikipedia article creating the citation can be really big (median 356 days) for non-popular entities.

KBA is to give a chance to non-popular entities information to be updated as soon as a useful information is published on the internet. The KBA organizers have built up a stream-corpus which is a huge corpus of timestamped web documents that can be processed chronologically. Hence it is possible to simulate a real time system. The documents come from newswires, blogs, forums, review, memetracker... . In addition, a set of target entities, coming from wikipedia or from twitter, has been selected for their ambiguity or unpopularity. And last but not least, more than 60,000 documents have been annotated so that systems can train on it. The train period starts on documents published from october 2011 until february 2012, and the test period starts from february 2012 to february 2013.

The KBA track is divided in two tasks: CCR (Cumulative Citation Recommendation) and SSF (Streaming Slot Filling). CCR task is to filter out documents worth citing in a profile of an entity (e.g., wikipedia or freebase article). SSF task is to detect changes on given slots for each of the target entities. This article is focused only on CCR task.

In CCR task, the system is to filter out documents relative to target entities out from the stream-corpus. The system must also be able to give the usefulness of a document ranked using one of those 4 relevance classes:

**garbage :** no information about target entity;

**neutral :** informative but not citable;

**useful :** bio, primary or secondary source useful when creating a profile from scratch;

**vital** : timely info about the entity's current state, actions, or situation.

The remaining of this article is organized as follows. The next section give details on how we designed our system. Then the different runs we sent are analyzed to eventually conclude and give a brief outline on perspectives.

## 2 KBA System

Before starting designing our system, we first had to think about how to deal with the corpus. For the KBA track, a 6.5 terabytes (compressed) corpus of documents is given to participants where each documents are organized so it can be read chronologically. It is really important to read documents after documents in chronological order to make sure that the system is not considering future information when processing a document. Therefore it is not possible to build a big index on the whole corpus since the retrieval process would be influenced by all the documents. It is yet possible to build an incremental index however, this can be really a time consuming task.

Our last year system we indexed each hour of stream to make sure we were not messing up with the "look ahead" behavior. This year we have modified our system since the corpus is way much bigger, it would have been too much time consuming to index every hour of stream. In order not to have to process all the corpus each time we want to test our system we first rebuild a new corpus with only documents containing a mention of an entity in a preprocessing phase. Then we can perform the ranking

[1]wikipedia: http://wikipedia.org

phase which consists in giving a class to each document having a mention of its target entity.

## 3 Preprocessing Data

To extract documents, we first pre-process the target entities. For the CCR task, the entities come either from wikipedia; so the url page is used as a "`target_id`"; or from twitter; so only the user starting with a '@' is given. For twitter entities, we also were allowed to get real name from the twitter profile.

The remaining of this section explains first how we build entities profiles we use for both extracting documents from the corpus and for document's classification.

### 3.1 Building Entity Profile

From now on, we are going to use the word "Profile" to define the structure that holds different data that must define as much as possible the entity. We identified three types of data :

- variant collection : the variant collection contains different ways to refer to an entity;
- relation collection: relation collection contains the different entities that have a relation with the target entity;
- language model : is to contain a textual representation of the entity as a bag of n-grams. This model is used to evaluate how similar/divergent a document is from it.

We describe in the remaining of this sub-section how we gathered those data.

#### 3.1.1 Variant and Relation Collections

An entity may be cited in a document using different manners. For instance, the homonymic entities Boris Berezovsky (Pianist and Business man), has different middle names (cf., table 1, 2) that can be helpful to identify which one is referred within a document. In a recall oriented system, the variant collection is mandatory.

| **target_id** | Boris_Berezovsky_the_Pianist |
|---|---|
| **variants** | Boris *Vadimovitch* Berezovsky |

Table 1: Variants for Boris Berezovsky the pianist

Variant collections can be filled either supervised or unsupervised manner. We use in our system only the unsupervised way. We build a system that goes

| **target_id** | Boris_Berezovsky_Businessman |
|---|---|
| **variants** | Boris *Abramovich* Berezovsky |

Table 2: Variants for Boris Berezovsky businessman

through a wikipedia dump[2] and fills variant collections using (Cucerzan, 2007) methods:

- bold words in the first paragraph of the target entity's WP page;
- legend of anchors in any pages that points to the target entity's WP page.

While searching for variants we also look for the different kind of relations the target entity may have with other entities and remember each one as : *outgoing relation* (an anchor in the target entity WP page that points to another WP page), *incoming relation* (an anchor in any WP page that points to the target entity WP page) or *mutual relation* (anchors are found in both a WP page and the target entity WP page).

This year we also have Twitter entities. For those entity we simply use the name displayed on the profile of the entity using the Twitter API (cf. figure 1).

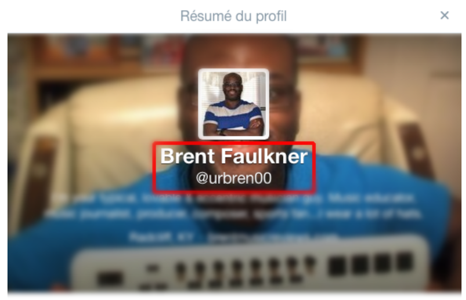

Figure 1: Variant's name for Twitter entity `@urben00`: Brent Faulkner

#### 3.1.2 The language model

In the profile, we also save as a Bag Of Word (BOW with n-gram) we call the reference BOW. It contains a representation of the WP Page of the target entity when it is a WP entity. For Twitter entities it is empty otherwise. The last point is quite important since some profiles have a lack of information. In addition, the entities may evolve as the time goes by. So we added to the profil another BOW that we call the dynamic BOW. This dynamic BOW is filled with

[2]from before the 1$^{rst}$ of January 2012

documents discovered on the stream when they fulfill the prerequisites: mention the entity, document is ranked as vital (V_UPDT), document is scored as useful (VU_UPDT). The two last conditions are tested seperately in two different runs.

In addition, we add another parameter to fill the dynamic BOW. When having a document ranked even vital for an entity, it may not be relevant to consider the whole document since it may not only contain information about the target entity. So we tested both approaches : add whole document (_DOC); add snippet (_SNPT). To build the snippet we simply use every paragraph that contains a mention of the target entity.

The process of updating the dynamic BOW is performed during the ranking phase of our system since it does require the rank our system assigns to the document. To summarize, our system is giving 5 different outputs: NO_UPDT, V_UPDT_DOC, V_UPDT_SNPT, VU_UPDT_DOC, VU_UPDT_SNPT.

### 3.2 Extracting documents from the Corpus

Using variant collection from target entities profile, we go through every single document in the corpus and search for a mention of any variant. A single document can be extracted more than once if multiple target entities appear in it. We use a hash of the "target_id" as well as the document unique identifier (stream_id) to name the xml file that represents the document so it is easier to find which document contains which entity while keeping track on what time the document appears on the stream. It is therefore possible to read all documents still chronologically.

In order to find a match in a document we implemented an algorithm inspired of the "Backward Nondeterministic Dawg Matching" (BNDM) to which we add the possibility to ignore space and some punctuation such as dash "-" (BNDM_IR). An entity name can indeed be composed of several names not always segmented the same way. For instance KBA12 entity "lovebug starsky" was also found as "love bug starski".

To have an idea of the gain of this method, we compute the recall using the formula in equation 1 on the training and test collection:

$$recall = \frac{\#documentsfound \in corpus}{\#documentsfound \in train \cup test} \quad (1)$$

## 4 The Ranking System

The ranking system is to identify according to a profile how useful a document is. This system can only consider prior and current documents to decide the rank of a document in : garbage, neutral, useful and vital. In addition, the system must issue a confidence score $\in ]0, 1000] \in \mathbb{Z}$ where 1000 is very confident.

| Method | Recall |
|---|---|
| BNDM | .7396 |
| BNDM_IR | **.8226** |

Table 3: Recall computed using training and test set

Our system uses Random Forest (RF) classifiers with a set of features to determine the rank. The RF is composed of a multitude of decision trees, where each one uses a subset of features. The final decision is made by averaging scores of the trees. We use three different kind of features : documents related features, entity related features and time related features. We design our features so adaptive learning can be used. Therefore our system could be used to rank documents of an entity that has not been part of the training. The remaining of this section discuss the features we use for classification.

**Document related features** are used to depict a layout from documents independently of the entity. We use three features here apply to a document $D$ as shown in table 4.

| | |
|---|---|
| $has\_title(D)$ | $\in \{0, 1\}$ |
| $voc\_size(D)$ | $\|D\|$ |
| $entropy(D)$ | $\sum_{i=0}^{D} p(w_i, D) log_2(p(w_i, D))$ |

Table 4: Document related Features

**Entity related features** are used to determine how a document concerns the target entity it has been extract for. Here, the previously build profile is used to compute the different features(cf., table 5).

Some entity features require term frequency (TF) to be computed. To compute a TF of an entity $e$, we sum up the frequency of all mentions of variant names $v_i$ from the collection $V_e$ in a document $D$. We eventually normalize by the number of words$|D|$ in $D$ (cf., equation 2).

$$tf(e, D) = \frac{\sum_{i=1}^{V_e} f(v_i, D)}{|D|} \quad (2)$$

The features that compute the coverage is computed as in equation 3

$$cov.(D_{snippet}, D)) = \frac{|D_{snippet}|}{|D|} \quad (3)$$

| | |
|---|---|
| $tf_{title}$ | $tf(e, D_{title})$ |
| $tf_{document}$ | $tf(e, D)$ |
| $voc\_size_{document}$ | $\lvert D \rvert$ |
| $cov_{snippet}$ | equation 3 |
| $tf_{relationType}$ | $tf(rel_{type}, D)$ |

Table 5: Entity related features

When building profile, we said that we extract relation an entity may have with another from WP using three different kind of relations: incoming, outgoing and mutual. For each kind of relation and for each entity in this relation group, we compute the average tf on the whole document.

**Time related Features:** the corpus offers the pros to be able to work with time information. We designed the time related features so the classifiers are able to work with information concerning previous documents. Such information may help detecting that may be something is going on about an entity using different clues such as burst effect. As shown on the figure 2, the burst does not always depicts vital documents, although it still might be a relevant information for classification.

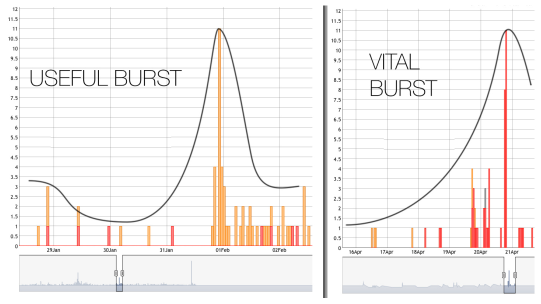

Figure 2: Burst on different entities does not always imply vital documents.

To depict the burst effect we used an implementation of the Kleinberg Algorithm (Kleinberg, 2003). Given a time series, it captures burst and measure the strength of it as well as the direction (up or down). We decided to scale the time series on an hour basis. In order not to mess the classifiers with too many information we decided not to use the direction as a feature but to merge the direction with the strength by applying a coefficient of -1 when direction is down and 1 otherwise.

In addition to burst detection, we also consider the number of documents having a mention the last 24hours.

We noticed from our last year experiments on KBA12 that time features were actually degrading final results since when ignoring them our scores was better. So we decided to focus only on features (cf table 6) that can really bring useful time information.

| | |
|---|---|
| $kleinberg_{1h}$ | burst strength and direction |
| $match_{24h}$ | # documents found last 24h |

Table 6: Time related features used for classification

### 4.1 Classification

As a reminder of section 3.1.2, we implemented different ways to update (or not) a dynamic language model:

- No Update: NO_UPDT
- Update with Snippet: UPDT_SNPT
- Update with Document: UPDT_DOC

When we update the dynamic model, we can choose to update either Vital or Vital and Useful documents which adds 2 different outputs. In total 5 outputs are computed.

To classify documents based on computed features, we designed several ways to handle it. The first method "TwoStep" we use, considers the problem as a binary classification problem where we use two classifiers in cascade. The first one $C_{GN/UV}$ is to classify between two classes: "Garbage/Neutral" and "Useful/Vital". For documents being classified as "Useful/Vital" the second classifier $C_{U/V}$ is used to determine the final output class between "Useful" and "Vital".

The second method "Single" performs directly a classification between the four classes.

The third method "VitalVSOthers" trains a classifier on recognizing vital documents amongst all others classes. When this classifier gives a non-vital class, the "Single" method is used to determine another class from "Garbage" to "Useful".

The last but not least method "CombineScores" uses scores emitted by all previous classifier and try to learn the best output class considering all classifiers scores for every classes.

### 4.2 System Outputs

To summarize, we have 5 different outputs possible with 4 different methods which makes 20 different runs. For the official run submission, we had issues with our system making our runs not consistent enough. In addition, we also had issues for extracting documents from stream-corpus which makes our system miss a lot of documents. The result of those experiments have been performed after the TREC conference held in november 2013.

## 5 Result Analysis

All results have been computed using the official kba-scorer with the following command line provided in documentation. Moreover, the official metrics is the $f_1$ score which is an harmonic mean of precision and recall. Precision and Recall are both micro and macro average before computing $f_{micro}$ and $f_{macro}$.

Since there are many results to discuss, I'll summary the more interesting ones. First of all the system which, uses no updates, performs better than any other system on micro-average point of view. The table 7 shows that most ranking methods perform equally besides VitalVsOther which has a score a bit below.

| **Ranking Methods** | $f_{micro}$ | $f_{macro}$ |
|---|---|---|
| Single | .400 | .341 |
| TwoStep | .418 | .340 |
| VitalVsOther | .383 | 289 |
| Combine | .415 | .327 |

Table 7: Scores from system that do not update a dynamic model.

The system that performs best on macro-average point of view is the one where only vital document are use to update dynamic model and the model is updated with the snippet (UPDT_SNIPPET_VOnly) (cf. table 8).

| **Ranking Methods** | $f_{micro}$ | $f_{macro}$ |
|---|---|---|
| Single | .458 | .316 |
| TwoStep | .452 | .303 |
| VitalVsOther | .428 | .273 |
| Combine | .456 | .305 |

Table 8: Scores from system that uses update using snippet of vital documents only.

When observing the different curves given by the official scorer, we noticed that our system has not a good recall which penalize a lot the scores. In addition, the official scorer is considering that documents appearing in test collection and not in the evaluate run as false negative. Let's consider the document not being found by our system and therefore not classified, it has been considered as if we classified it wrong.

We implemented a similar scorer that consider only what has been classified using different cutoff. When the confidence score is below the cutoff, the document is considered as false negative. We run the scorer on the both better results above and obtained the following results (we select the best trade off between precision and recall for each methods):

**Scores for No Update:**

| **Methods** | **Precision** | **Recall** | $f_1$ |
|---|---|---|---|
| Single | .652 | .423 | .513 |
| TwoStep | .627 | **.453** | .526 |
| VitalVsOther | **.716** | .391 | .506 |
| **Combine** | .669 | .434 | **.527** |

Table 9: Scores from alternative scorer to measure classifier performance on system without update.

**Scores for Update Snippet with Vital:**

| **Methods** | **Precision** | **Recall** | $f_1$ |
|---|---|---|---|
| **Single** | .771 | .422 | **.545** |
| TwoStep | .726 | **.427** | .537 |
| VitalVsOther | **.791** | .386 | .519 |
| Combine | .750 | .425 | .542 |

Table 10: Scores from alternative scorer to measure classifier performance on system without update.

Those two tables (9, 10) show that using a dynamic model really helps for both precision and recall. Therefore, the f-measure is also better.

## 6 Conclusion and Perspectives

We present an approach to filter out entity centric documents from a stream. This approach has the pros to be adaptive and can therefore be used on entities which no training data have been provided for. In our last year system (Bonnefoy et al., 2013) we showed that even though the system has no training data for an entity, it is still able to find out some vital documents for the entity. We also show that time features as well as a dynamic model is valuable for the classification purposes although we have to improve our IR system which lake of performances.

We are currently working on improving our system to find out whether it may have any other way to update profile to improve even more precision. We also plan to investigate on a method to know when is the best moment to update a profile, whether some information must be forgotten inside the profile. Eventually, we will look at methods to improve the recall.

## References

Ludovic Bonnefoy, Vincent Bouvier, and Patrice Bellot. 2013. A Weakly-Supervised Detection of Entity Central Documents in a Stream. *SIGIR 2013*, pages 1–4, February.

S Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *EMNLP-CoNLL*.

J Frank, M Kleiman-Weiner, D Roberts, F Niu, and C Zhang. 2012. Building an Entity-Centric Stream Filtering Test Collection for TREC 2012. *. . . of The 21th TREC*.

J Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*.