CoBAn: A Context Based Approach for Text Classification

Gilad Katz, Bracha Shapira, Nir Ofek, Yedidya Bar-Zev

Department of Information Systems Engineering, Ben Gurion University of the Negev, Israel

1. The Proposed Approach

1.1. The Concept

The proposed approach first identifies key terms that serve as initial indication for the presence of sentiment and then analyzes the context in which they appear. This enables us to detect even small amounts of relevant text hidden in a much larger section. The original rule-based model, presented in [1], was evaluated in the field of data leakage detection. It used predefined formulae to determine the "confidentiality score" of the analyzed text. The model was able to detect small amounts of rephrased confidential text hidden in larger non-confidential documents, a task which proved difficult both for fingerprinting algorithms [2, 3] as well as BOW classifiers.

Despite its effectiveness for the security domain, the rule-based method proved difficult to adapt to the field of sentiment analysis. This had several reasons:

a) Sentiment analysis deals with highly nuanced text that requires consideration of multiple factors. This complexity is difficult to model using a rule-based approach.

b) We were dealing with transcribed text (generated using speech-to-text software) and wanted to utilize metadata information (duration, speed of speech etc.) together with our algorithm. The rule-based approach makes the addition of such features difficult, because of the multiple scenarios that need to be considered.

c) The rule based approach should be adapted to every new domain: the parameters of the formulae should be changed for each new datasets. This stands in sharp contrast to learning-based solutions that can automatically assign weights to parameters.

These reasons led us to the development of the method presented in this work. Like its predecessor in [1], the first step of this method is the identification of indicative key terms and the relevant contexts in which they appear. However, instead of rules we utilize the detected terms to generate a set of features that are far more capable of representing the various aspects of highly nuanced text.

NOTATIONAL CONVENTIONS. $T$ denotes the training set used to create the model while $E$ denotes the evaluation set. We refer to the documents that are the target of detection as the relevant documents and denote them by $T_{rel}$ while the non-relevant documents are denoted by $T_{non-rel}$. (in our case relevant documents are documents with negative sentiment that we want to detect). A key term is denoted as $t_{key}$ and a context term as $t_{context}$. The language model generated from a corpus of documents $C$ is denoted as $lm_C$. The probability value of a term $t$ in a language model is denoted as $lm(t)$.

1.2. The learning process
The learning process consists of two steps. In the first, we identify key terms indicating negative sentiment and the context in which they appear. During the second step these terms are used to create features that are later used for supervised learning.

1.2.1. Identifying key terms and their contexts
In order to capture the semantic meaning of text segments we chose to detect specific terms whose existence in a relevant context is indicative for the existence of sentiment. We denote these terms as key terms and the terms that make up the relevant context as context terms. We next describe the process of detecting these terms.

Key terms detection:
The key terms have two purposes: a) they serve as initial indicators of relevant content; and b) they serve of the axis around which the context terms are generated. Without a relevant and robust set of key terms, the method presented in this paper is not likely to succeed.

The key terms detection process is based on a technique from the field of information retrieval called language modeling [4-6]. This technique denotes the value of a term (or a sequence of terms) as the probability of selecting it when randomly sampling a term from the text represented by the language model (the documents corpus). This approach can be mathematically represented as $lm(w_1, w_2, w_3 ...) = P(w_1, w_2, w_3 ... | C)$, where $C$ denotes the documents corpus. This approach is often used for documents ranking in response to a query [7, 8] and to related tasks such as query expansion [9, 10] and query performance prediction [11, 12].

The detection of the key terms is done as follows: we create a language model separately for $T_{rel}$ and $T_{non\_rel}$, which we denote $lm_{rel}$ and $lm_{non\_rel}$ accordingly. Then, for each term $t$ in $lm_{rel}$ we calculate its score using the following formula $score(t) = \frac{lm_{rel}(t)}{lm_{non\_rel}(t)}$. In other words, the score assigned to terms reflects how much more likely they are to appear in a relevant document than in a non-relevant one. All terms whose score is above a predefined threshold are selected as key terms.

In order to correct for inaccuracies that may arise due to data sparseness, we apply the Dirichlet smoothing technique [13] when calculating the scores of the possible key terms. Smoothing is considered a key element in improving the performance of language model and its importance is evident by the large body of work dedicated to the subject [8, 14, 15].

Context terms detection:
The context terms serve three purposes: a) they serve as validators, enabling us to determine whether or not the detected key terms are truly indicative of relevant content; b) they allow us to quantify the degree of relevance (many relevant context terms = higher relevance for a key term) and; c) they enable us to determine whether the detected key terms are connected by analyzing their shared contexts.

The context terms detection process is similar to that of the key terms, with a few minor modifications. Language models are again used to calculate the scores of the terms, but instead of using whole documents we only analyze document segments. In addition, the
context terms are generated for each key term individually (since each key term possibly has its own unique context).

The process of detecting the context terms for a single key term (denoted as $t_{key}$) is as follows:

a) Find all the instances of $t_{key}$, both in $T_{rel}$ and in $T_{non,rel}$. Please note that there can be several instances in a document.

b) For each instance, extract the terms around $t_{key}$, using a sliding window of size $X$ ($\frac{X}{2}$ terms before the position of the term and $\frac{X}{2}$ terms after it). Every term in this text excerpt (aside from the key term itself) will be considered as a possible context term. We denote the size of this sliding window as the context span.

c) Each such excerpt will now be considered as a document $d$. Excerpts from relevant and non-relevant documents will be denoted as $d_{rel}$ and $d_{non,rel}$, accordingly. The set of all $d_{rel}$ will be denoted as $D_{rel}$ and $D_{non,rel}$ is similarly defined.

d) Next, we calculate the score of each possible context using the following formula:

\[
\text{score}(t_{context}) = \text{imp}_{rel}(t_{context}) - \text{imp}_{non,rel}(t_{context}).
\]

e) If the score of possible context term exceeds a predefined threshold, it will be defined as a context term of the key term $t_{key}$.

We use subtraction when calculating the scores of the context terms (instead of division, as was done for the key terms) because we want to take into account not only the relative probability of a context term to appear next to a key term in a relevant document but also the absolute probability of the two appearing together at all. Subtraction enables us to achieve these two goals.

The result of the process described above is a set of key terms, each affiliated with a set of context terms. The easiest method of visualizing the results is using a graph. In the following section we show how this representation can be used to generate the features that enable us to transform this text-classification problem into one of machine learning.

### 1.2.2. Generating context-based features for supervised learning

Following the creation of the key and context terms, we now need to train a classifier by generating a set of features for each document in the training set $T$. Since these features all rely on the key and context terms contained in each document, a preliminary step must be the detection of these terms. We next describe the key and context terms detection process, and then present the features generation process.

**Detecting key and context terms in a document:**

The detection process of key terms in a document is straightforward: for every key term, we check whether or not it appears in the document. Then, for each detected key term, we scan the text surrounding every one of its instances (appearances) in the document in search of the context terms assigned to it in the previous phase. The size of the area in which we search for the context terms is defined by the context span parameter, which was defined in the previous section.

**Generating the features for the analyzed document:**
Following the detection of the key terms and their corresponding context terms in the document, we now generate our proposed features. With our features, our aim was to represent multiple aspects of the detected terms, taking into account their assigned scores, distances from one another in the document and interconnectedness. Our features can be categorized into three groups, based on the data they utilize: key term-based features, context term-based features and co-occurrence and statistical features. We next describe each of these groups.

**Key term-based features:**

The key terms serve as the initial indicators of relevant content. The following features provide various "perspectives" on the detected key terms, their scores, their proximity in the text etc.

- **KT_Num** – the number of key terms detected in the document.
- **KTS\text{max}, KTS\text{avg} \& KTS\text{stddev}** – the maximal, average and standard deviation of the scores of the key terms detected in the document.
- **KTNCT\text{cnt} \& KTNCT\text{perc}** – the number of key terms without context terms, and their percentage of the total number of detected key terms.
- **KTF\text{max}, KTF\text{avg} \& KTF\text{stddev}** – the max, average and standard deviation of times each key term has appeared in the analyzed text. Our study of transcribed text has shown us that when users repeat the same phrases multiple times, it is often an indication of anger or frustration. This set of features is used to capture this phenomenon.
- **KTMD\text{max}, KTMD\text{avg} \& KTMD\text{stddev}** – for each key term, we calculate its minimal distance (in number of terms) from another key term. If the term appears more than once in the text, we check all instances and take the minimal value. We then extract the minimal and maximal distances between terms and calculate the average and the standard deviation over all the terms. We hypothesize that key terms that appear in close proximity are more indicative than those scattered throughout the text. These features are used to model this aspect.
- **KTIW_{10}, KTIW_{20}, KTIW_{30}** – The maximal number of key terms that can be found in the document within a sliding window of X terms. Three "window sizes" were used – 10, 20 and 30 words.
- **KTSL\text{max}, KTSL\text{avg} \& KTSL\text{stddev}** – for each key term, we extract the maximal number of times it has appeared in a sequence (without any other key term appearing between the consecutive appearances of the said key term). From these values we then extract the maximum, average and standard deviation of these values. As mentioned above, the repeated use of certain terms is sometimes an indication of negative sentiment. These features model this repetition for all detected key terms.

**Context based-terms features:**

The context-based features attempt to provide multiple perspectives of the context of the detected key terms. Using this information, we are able to better deal with cases where key terms appear in an unrelated context or have a different meaning than those relevant to the algorithm. These features, to a large degree, are what separates our proposed approach from rule-based and fingerprint-based [2, 16, 17] systems which simply scan for key terms.

- **CTN\text{max}, CTN\text{avg} \& CTN\text{stddev}** – the maximal, average and standard deviation of the number of the context terms detected per key term in the text.
- $\text{CTP}_{\text{max}}, \text{CTP}_{\text{avg}} \& \text{CTP}_{\text{stdev}}$ – the maximal, average and standard deviation of the percentage of the context terms detected in the text out of all possible context terms detected for this key term (calculated per key term). These features are an attempt to quantify the percentage of all possible relevant context detected in the document.

- $\text{CTS}_{\text{max}}, \text{CTS}_{\text{avg}} \& \text{CTS}_{\text{stdev}}$ – the maximal, average and standard deviation of the average scores of the detected context nodes per key term.

- $\text{CTSR}_{\text{max}}, \text{CTSR}_{\text{avg}} \& \text{CTSR}_{\text{stdev}}$ – for each key term, we calculate the ratio of the score of its detected context terms' to that of all its context terms. We then extract the maximal value, the average and the standard deviation of these values. These features attempt to measure, for each key term, whether its "stronger" or "weaker" contexts were detected in the analyzed text.

- $\text{CTF}_{\text{max}}, \text{CTF}_{\text{avg}} \& \text{CTF}_{\text{stdev}}$ – the max, average and standard deviation of times each context term has appeared in the analyzed text (the values are first calculated on average for each key term).

- $\text{CKTD} \& \text{CKTT}$ – the number of key terms which shared at least one context term in the analyzed document and in the entire training set, respectively.

- $\text{SCTD}_{\text{max}}, \text{SCTD}_{\text{min}}, \text{SCTD}_{\text{avg}} \& \text{SCTD}_{\text{stdev}}$ – for every pair of detected key terms, we count the number of their shared context terms detected in the analyzed document. We then extract the min and max values and calculate the average and standard deviation over all values.

- $\text{SCTT}_{\text{max}}, \text{SCTT}_{\text{min}}, \text{SCTT}_{\text{avg}} \& \text{SCTT}_{\text{stdev}}$ – for every pair of detected key terms, we count the number of their shared context terms detected in the entire terms graph. We then extract the min and max values and calculate the average and standard deviation over all values. These features model how connected the detected key terms are in the dataset, not the document, in order to obtain an additional perspective on the detected terms.

**Co-Occurrence and statistical features:**

The features of this group are derived from a multitude of sources – the analyzed text itself, the language models that were used to generate the key terms and co-occurrence calculations performed over the entire training set. These features can be thought of as providing "complementary" information.

- $\text{KTLM}_{\text{avg}} \& \text{KTLM}_{\text{stdev}}$ – the average and standard deviation of the language model values of the key terms detected in the document.

- $\text{CTLM}_{\text{avg}} \& \text{CTLM}_{\text{stdev}}$ – the average and standard deviation of the language model values of the context terms detected in the document.

- $\text{KTCO}_{\text{max}}, \text{KTCO}_{\text{avg}} \& \text{KTCO}_{\text{stdev}}$ – for each pair of key terms detected in the document, we count the number of documents in the training set $T$ in which they both appeared. We then generate three features from these values: the maximum, average and standard deviation of these values. These features are used to measure how connected the detected key terms are

- $\text{NKTL}_{\text{max}}, \text{NKTL}_{\text{avg}} \& \text{NKTL}_{\text{stdev}}$ – the maximum, average and standard deviation of the lengths of text (counted by terms) in the document in which no key terms were detected.

- $\text{NKTLR}_{\text{max}}, \text{NKTLR}_{\text{avg}}$ – these features are identical to the previous set of attributes (NKTL), but their values are divided by the total length of the document.
• **NCKT** – the number of detected key terms which are contained within another key term. For example, the key term “know” is contained in the key term “don’t know”.
• **POKT** – the number of detected key terms which have at least one shared word. For example, the key terms “don't know” and “know what” share the word “know” and are therefore overlapping.
• **TL** – the overall length (in characters) of the analyzed text.

### 1.2.3. Enhancing the model with additional and external features

A major drawback of BOW-based solutions is the difficulty in integrating additional information into the model. The main reason for this difficulty is the well-known *curse of dimensionality* [18-20]: the number of features is measured in the thousands (one for each unique term in the text) and classifiers find it difficult to utilize them in an efficient manner. The large number of features in BOW approaches also narrows the selection of possible classifiers, ruling out popular algorithms such as the C4.5 decision trees algorithm [21], which are incapable of efficiently dealing with a feature set of this magnitude.

Our proposed model is affected to a much lesser degree by the curse of dimensionality, as it employs a small set of advanced features, which are created through the process described in Section 3. This enables us to easily and effectively add additional (and not necessarily textual) features to the model. This trait enabled us to improve the performance of our model in two important ways:

a) **The incorporation of documents metadata** - usually, the metadata consists of a small set of useful non-textual features, providing additional information about the document. The curse of dimensionality can sometimes prevent BOW-based approached from using the full potential of this information (as shown in our experiments). Our proposed model, on the other hand, shows considerable improvement when these features are added.

b) **Analyzing several text segments simultaneously** – as shown by our experiments on transcribed call-center conversations, calculating two sets of features – one for the analyzed paragraph and one for the paragraph and its surroundings – boosted the performance of our model. The same approach, however, damages the performance of the BOW classifier. We suspect that the reason for this was that while doubling the number of features on our model required adding 50-something additional features, doing the same for the BOW classifier meant adding thousands of additional features, exacerbating the curse of dimensionality.

The ability to easily integrate additional features into our model, as well as the ability to employ a large number of learning methods (including those not usually applicable for text classification) is one of the major strengths of our method. This strength is especially important in domains where the text itself is merely one of several facets of the data (as in sentiment analysis).

### 2. References

