Functional Summarization of Non-Text Data

Steven J Simske, Marie Vans, Margaret Sturgill; HP Labs; Fort Collins, CO USA 80525

Abstract

Summarization techniques can be applied to non-text data in order to perform classification and clustering of important imaging, video and other document-associated but non-text content. The advantage to this approach is that there is a multiplicity of inexpensive (even free) summarization engines, and so a robust solution can be crafted with relatively modest effort. In this paper, we present the applicability of this approach to video and imaging data, in addition to broader binary and genetic data.

Keywords

Meta-Algorithmics, Compression, Image, Video

Introduction

Summarization can be extractive or abstractive [1]: extractive techniques simply replicate the original text that is determined to be the most germane as a "shorthand" for the document, while abstraction techniques paraphrase sections of the original content, therein condensing information and providing semantic or meaning-based "shorthand".

Extractive summarization sentence weighting approaches reported in the literature are based on the keywords and key phrases extracted; capitalization of text; grammatical case of nouns; word co-occurrences; font formats; sentence position in a paragraph; cue phrases such as "in summary" and "importantly"; correlation of a sentence or phrase with the title, author reported keywords, etc.; sentence length; and sentence centrality or redundancy with other sentences [1].

Summarization is not just useful for the aforementioned shorthand representations. Summarization can also be used for functional purposes in text processing—this includes indexing and tagging, search, classification, and document sequencing for topical understanding. We have shown how multiple summarization engines are used not only to improve summarization accuracy (primary summarization) but also to select text from a document that is optimal for use in a secondary text processing task (functional summarization)[1].

A summarization engine is a computer-based application that receives a text document and provides a summary of the text document. A meta-algorithmic pattern is a computer-based application that can be applied to combine two or more summarizers, analysis algorithms, systems, and/or engines to yield meta-summaries.

Non-textual content may also be summarized. For example, images, audio and/or video content, binary data, genetic data, and/or healthcare data can be summarized using traditional summarization approaches. Video content may include one video, portions of a video, a plurality of videos, and so forth. Likewise, genetic data may include genetic data from an individual, a group of individuals, portions of genetic data, genetic data of one or several organisms, and so forth. Binary data includes any data that may be represented by a sequence of 0's and 1's.

Meta-summaries are summarizations created by the intelligent combination of two or more standard or primary summaries. The intelligent combination of multiple intelligent algorithms, systems, or engines is termed "meta-algorithmics", and first-order, second-order, and third-order patterns for meta-algorithmics may be defined.

The output of a meta-algorithmic pattern may be used as input (in the same way as the output of individual summarization engines) for classification of the non-textual content into a plurality of classes. Each class may include non-textual content, including images, audio, video, binary data, genetic data, healthcare data, and so forth. Summarization terms may be extracted from the meta-summary, where the summarization terms include key tokens that are representative of the meta-summary. Class terms representative of a given class of non-textual content may be generated from the non-textual content in each class. The summarization terms may be compared to the class terms for each class to determine similarity values of the non-textual content over each class. A class of the plurality of classes may be selected based on the similarity values, and the non-textual content may be associated with the selected class.

Combined, these technologies create an accurate system for data mining, classification and processing of non-text data, the high-level architecture for which is given in Figure 1.

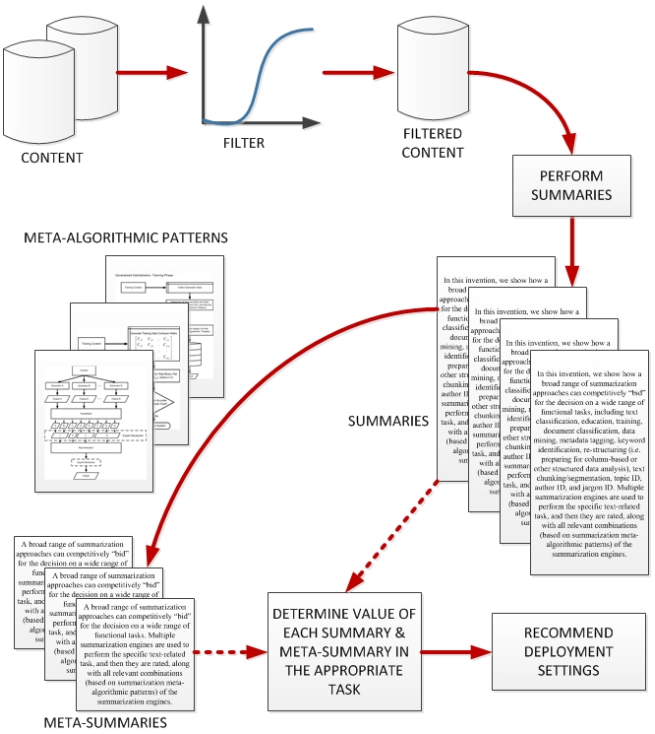

Figure 1*. General diagram of the system. Content is filtered as necessary to create the "Filtered Content" from which the "Summaries" and "Meta-Summaries" are derived. The outputs of this collective "Summarization Set" are evaluated for their utility in the*

functional task, and the optimal "Summarization Architecture" will be chosen for the Deployment.

Video Data

Summarization techniques are often based on identifying the relatively rare terms in a document. Traditional methods multiply the term frequency (TF) in a given document by the inverse of its overall document frequency (IDF). This measurement, TF*IDF, or other variants on this approach, can just as readily be applied to any data sets.

As such, summarization techniques are useful for the compressed representation of other content, such as image libraries and video (herein termed "image data"). Summarization of image data proceeds after the conversion of each image into a set of descriptors. These descriptors comprise a vocabulary which can be very concrete, as is the case for a dictionary of segmented object types, as in for example [2]. Alternatively, the descriptors can be more abstract, as in visual descriptors (shape, color, texture, bounding boxes [regions], motion, etc.). Regardless, once the image data has been associated with such a vocabulary, image summaries for each image or video clip can be processed just as if the set of descriptors were a language. The primary applications are image/video classification and image/video tagging for indexing and search. Search can be performed based on similarity (find "like images") which can use both the concrete and abstract vocabularies as described above, and based on object search (the object type/name being the query).

Binary Data Summarization

Binary data can be tokenized to create a vocabulary based on salient binary substrings. One of the key functional data sets we may wish to extract is the randomness of data. Even when data is truly random, there are still "patterns" in the data that can be used to identify the data. Think about a "random" patch of the night sky. Truly random data features constellations, or groupings of stars that appear to have structure. Small children tend to draw night skies with the stars equally spaced, but in reality this "equally spaced" distribution is highly ordered (its two-dimensional entropy value is rather low). Thus, binary data summarization consists of finding the "constellations", which we will more formally describe as "anomalously high-frequency" binary substrings.

In addition, bit stream entropy, statistics on uniqueness collisions encountered during mass serialization database collection, and differential byte statistics to test for reduced entropy (indicative of cryptographic threats) may be utilized to tokenize binary data. Functional summarizations may be performed on such tokenized data.

Regardless, summarization techniques can be used to summarize binary data, including compressed image or video data. Binary data summarization proceeds after the binary streams of interest are "tokenized", as mentioned above. The following code shows how to detect the most anomalous substring in a specific 100-bit string. We need only two settings:

```
int MinSubstringLength = 5;
int MaxSubstringLength = 15;
```

These two simple settings define a reasonable range of lengths for substrings. Suppose, for example, we wish to find "constellations" in the binary data string: "10110110101001101010111111010111010100010011101110101111010100011001100001101110011". We invoke the code in Figure 2 simply using:

```
FindConstellationsInBinaryData("101
10110101001101010111111010111010100
01001110111010111101010001100101000
01111100001100000110111001 1");
```

```
public int FindConstellationsInBinaryData(string
BinaryString)
{
 int i, ii, j, jj = 0;
 int num = num_matched = 0;
 double current_ratio = 0.0;
 double max_ratio = 0.0;
 String current_string = "";
 String compare_string = "";
 String max_string = "";
 int max_num, index_max_num = 0;
 int max_num_matched = 0;

 if (BinaryString == null)
    return (-1);
 else if (BinaryString.Length <MinSubstringLength)
    return (-2);
 else if (BinaryString.Length < MaxSubstringLength)
    return (-2);

 for (i = MinSubstringLength; i < MaxSubstringLength;
      i++)
 {
    for (j = i; j < BinaryString.Length; j++)
    {
      current_string = "";
      for (ii = j - i; ii < j; ii++)
        current_string += BinaryString[ii];
        num = 0;
        num_matched = 0;
        for (jj = i; jj < BinaryString.Length; jj++)
        {
          num++;
          compare_string = "";
          for (ii = jj - i; ii < jj; ii++)
            compare_string += BinaryString[ii];
            if(current_string.CompareTo(compare_string)
               == 0)
                num_matched++;
        }
        current_ratio = (double)num_matched/
                        (double)num;
        for (ii = 0; ii < i; ii++)
           // normalizes for substring length
          current_ratio /= 2.0;
          if (current_ratio > max_ratio)
          {
            max_ratio = current_ratio;
            max_string = current_string;
            max_num = num;
            index_max_num = j-i;
            max_num_matched = num_matched;
          }
    }
 }
 return (0);
}
```

***Figure 2**. Program to find the longest repeated binary string between `MinSubstringLength` and `MaxSubstringLength` in length in a binary stream.*

We may define constellations to comprise substrings of length 5 or more. The anomalously high-frequency binary substring in this string is "10101", occurring with the most anomalous frequency (7 times out of a possible 96, well above its expected value of 3 occurrences), as highlighted here (note that two occurrences

overlap):
"10110110101001101010111111010111010100010011101110101
11101010001100101000011111000011000001101110011".

Depending on the length of the binary information in individual files and the number of files, the values of minimum and maximum anomalously high-frequency binary substring length will vary. In practice, we wish to optimally define a large set of substrings, and let the individual summarization engines decide how they wish to represent the binary substrings. We then proceed to use these summaries as input into the meta-algorithmic classification patterns and determine which pattern of tokenized-binary summarizers works best to classify the binary strings.

The advantage of this method is obvious: existing text summarizers can be used for binary summarization. The binary summarization can then be used for classification of binary data (even clustering of truly random, e.g. encrypted, binary data). In the case of related binary data, this provides good classification; in the case of encrypted or otherwise random data, this provides a good hash table.

Other Data Types

Genetic data summarization is used to identify patients without surrendering potentially compromising clinical and/or prognostic information. Here, the data converter or "mormalizer" converts genetic data into binary data for the purposes of functional summarization. Genetic data may be utilized to identify patients without surrendering potentially compromising clinical and/or prognostic information. In the case of genetic information, the patient's DNA is assigned to two logical sequences: the introns, which are associated with the transcription of genes, and the exons, which are not-fully understood DNA sequences not directly associated with genes. These exons, since they cannot be used to "snoop" an individual's risk for genetically-associated disease, are summarized to form the look-up for patient participation in clinical trials and for other situations in which a dual-access security/privacy mechanism is preferred. Generally, the exons may be treated as quaternary sequences (pairs of bits), where for example adenosine is "00", cytosine is "01", guanine is "10", and thymine is "11". A similar mapping may be used for RNA, protein sequences, and so forth. Once genetic data is converted into binary data, functional summarization of genetic data may proceed as for binary data. Similarly, de-identified or "anonymized" patient healthcare data into binary data. Publicly available healthcare data may be converted to genetic and/or binary data.

Classification Approach

For the Weighted Voting classification pattern described above, once tokenization has been performed, meta-algorithmic approaches are used to find the optimum combination of summarizers for the classification. The optimum combination of summarization engines to use on the "tokenized non-text" data was determined from the set of meta-algorithmic patterns in [1].

The summarizations are used to reduce the document to the key terms and/or phrases that can then be used as the means to classify the document. The best class is the one for which the cosine between the summarization terms and the class terms is maximized.

The vector space model (VSM) is used extensively to compute the similarity of documents, and in this case the similarity of the summarization and the class descriptors (terms, phrases or summary of the representative or "training" documents of the distinct classes). The vector space itself is an N-dimensional space in which the occurrences of each of N terms (e.g. terms in a query) are the values plotted along each axis for each of D documents. The vector $\vec{d}$ is the line from origin to the term set for the summarization of document d, while the vector $\vec{c}$ is the line from origin to the term set for class c. The dot product of $\vec{d}$ and $\vec{c}$, or $\vec{d} \bullet \vec{c}$, is given by Equation 1:

$$\vec{d} \bullet \vec{c} = \sum_{w=1}^{N} d_w c_w \tag{1}$$

From this, the cosine between the given class and the document is given by Equation 2:

$$\cos(\vec{d}, \vec{c}) = \frac{\vec{d} \bullet \vec{c}}{\left|\vec{d}\right| \left\|\vec{c}\right|} = \frac{\sum_{w=1}^{N} d_w c_w}{\sqrt{\sum_{w=1}^{N} d_w^2} \sqrt{\sum_{w=1}^{N} c_w^2}} \tag{2}$$

The cosine measure, or *normalized correlation coefficient*, is used for document categorization: the maximum cosine measure over all classes {c} is the class selected. This is employed for each of the meta-algorithmic algorithms described next.

(1) The Sequential Try pattern is employed to attempt classification of the documents (using the summarization words for classification) until one class is selected with a given confidence relative to the other classes. If no classification is obvious after the sequential set of tries is exhausted, the next pattern is selected. If, however, the following holds:

$$\cos(\vec{d}, \vec{c}_i) - \max\{\cos(\vec{d}, \vec{c}_i); \\ j = 1....N_{classes}; j \neq i\} > T_{STC} \tag{3}$$

where T_{STC} is the threshold for Sequential Try Classification (assignment) and $N_{classes}$ is the number of document classes, then the Sequential Try meta-algorithmic pattern terminates and the document is assigned to class i.

Importantly, T_{STC} can be adjusted based on the confidence in the individual summarizer (higher confidence generally lowers T_{STC} for a classifier), the size of the ground truth set (larger ground truth sets allow greater specificity of T_{STC}), and the number of summarizers to be used in sequence (more summarization engines generally increase T_{STC} for all classifiers).

(2) If the Sequential Try pattern results in no clear classification, the Weighted Voting pattern is used. Each of the multiple summarizers is tested against a ground truth (training) set of classes, and weighted by one of six methods described next.

The first concern for this meta-algorithmic pattern is how to weight the individual classifiers. We are concerned primarily with using the error rate on the training set for weight determination.

In a previous article [3], we showed that for a simplistic classification problem – wherein there are $N_{classes}$ number of classes, to which the *a priori* probability of assigning a sample is equal, and wherein there are $N_{classifiers}$ number of classifiers, each with its own accuracy in classification of p_j, where j=1… $N_{classifiers}$ – the following classifier weights are expected, in Equation 4:

$$W_j = \ln\left(\frac{1}{N_{classes}}\right) + \ln\left(\frac{p_j}{e_j}\right) \tag{4}$$

Here, the weight of classifier j is W_j and where the term e_j is given by Equation 5:

$$e_j = \frac{1 - p_j}{N_{classifiers} - 1} \tag{5}$$

Five other weighting schemes are also relevant. When the weights are proportional to the inverse of the error, then the weight for classifier j is given by Equation 6:

$$W_j = \frac{1.0 \Big/ (1.0 - p_j)}{\sum_{i=1}^{N_{classifiers}} 1.0 \Big/ (1.0 - p_i)} \tag{6}$$

The weights derived from the inverse-error proportionality approach are already normalized – that is, sum to 1.0 – by design.

The next weighting scheme is one based on proportionality to accuracy squared. The associated weights are described by the following Equation 7:

$$W_j = \frac{p_j^{\,2}}{\sum_{i=1}^{N_{classifiers}} p_i^{\,2}} \tag{7}$$

The inverse error based method heavily favors the more accurate classifiers in comparison to the "optimal" weighting of [3], while the accuracy-squared based method favors the less accurate classifiers in comparison to the "optimal" weighting. This implies that a hybrid method, taking the mean weighting of these two methods, may provide performance closer to the optimum method. The generalized hybrid scheme is given by Equation 8:

$$W_j = C_1 \frac{1.0 \Big/ (1.0 - p_j)}{\sum_{i=1}^{N_{classifiers}} 1.0 \Big/ (1.0 - p_i)} + C_2 \frac{p_j^{\,2}}{\sum_{i=1}^{N_{classifiers}} p_i^{\,2}} \tag{8}$$

Where $C_1 + C_2 = 1.0$. Clearly, varying these coefficients allows the system designer to tune the output for different considerations – accuracy, robustness, lack of false positives for a given class, etc.

The final weighting approach explored is one based on the inverse of the square root of the error, for which the weights are defined by Equation 9:

$$W_j = \frac{1.0 \Big/ \sqrt{1.0 - p_j}}{\sum_{i=1}^{N_{classifiers}} 1.0 \Big/ \sqrt{1.0 - p_i}} \tag{9}$$

The behavior of this weighting approach is similar to the hybrid method and not greatly dissimilar from that of the optimal method.

After the individual weights are determined, classification assignment is given to the class with the highest weight, defined in Equation 10:

$$Classification = \max_i \sum_{j=1}^{N_C} ClassifierWeight_j \left(\sum_{i=1}^{N_C} ClassWeight_{i,j}\right) \tag{10}$$

Where N_C is the number of classifiers, i is the index for the document classes, j is the index for the confidence each particular classifier i has for the classes (expressed as $ClassWeight_{i,j}$), and ClassifierWeight is the weight of the specific classifier per the above.

Conclusions

Existing text summarization engines can be used to summarize non-text (image/video, binary, genetic and other life science) data. These summaries can then be used effectively to classify the data,

which has the benefit of requiring no additional classifiers (assuming summarization engines are readily available, which is the case).

This approach also provides the means to tokenize (and otherwise anonymize) data in a wide variety of non-text data processing applications and so be able to summarize and functionally summarize the data for all the advantages of clustering, tagging, search and classification associated with text summarization.

The only prior solutions we are aware of are associative (a tag is associated with content in a database), derivative (e.g. a digital sign or hash), or ontological (the identifiers are selected from a pre-defined list such as a dictionary or other list/ontology).

References

[1] S.J. Simske, Meta-Algorithmics, New York: Wiley & Sons, 386 pp., 2013.

[2] D. Larlus and F. Jurie, "Latent mixture vocabularies for object categorization and segmentation," Image and Vision Computing 27, pp. 523-534, 2009.

[3] X. Lin, S. Yacoub, J. Burns and S. Simske, "Performance analysis of pattern classifier combination by plurality voting," Pattern Recognition Letters 24, pp. 1959-1969, 2003.

Author Biography

Steve Simske is an HP Fellow and the Director and Chief Technologist of the Content Solutions Lab in Hewlett-Packard Labs. Steve is currently on the IS&T Board. He is also an IS&T Fellow and a member of the World Economic Forum's Global Agenda Councils on Illicit Economy and the Future of Electronics. Steve has advanced degrees in Biomedical, Electrical and Aerospace Engineering, and has more than 100 granted US patents and more than 350 publications.