Language Independent Text Summarization of Western European Languages using Shape Coding of Text Elements

Ahmed A. Saleh
PPMEC, Department of Mechanical Engineering
University of Brasilia, Brasilia-DF, Brazil
ahmdsalh@yahoo.com

Li Weigang
TransLab, Department of Computer Science
University of Brasilia, Brasilia-DF, Brazil
weigang@unb.br

Abstract—**The majority of text summarization techniques in literature depend, in one way or another, on language dependent pre-structured lexicons, databases, taggers and/or parsers. Such techniques require a prior knowledge of the language of the text being summarized. In this paper we propose an extractive text summarization tool, *UnB Language Independent Text Summarizer* (*UnB-LITS*), which is capable of performing text summarization in a language independent manner. The new model depends on intrinsic characteristics of the text being summarized rather than its language and thus eliminates the need for language dependent lexicons, databases, taggers or parsers. Within this tool, we develop an innovative way of coding the shapes of text elements (words, n-grams, sentences and paragraphs), in addition to proposing language independent algorithms that is capable of normalizing words and performing relative stemming or lemmatization. The proposed algorithms and Shape-Coding routine enable the *UnB-LITS tool* to extract intrinsic features of document elements and score them statistically to extract a representative extractive summary independent of the document language. In this paper we focused on single document summarization of western European languages. The tool was tested on hundreds of documents written in English, Portuguese, French and Spanish and showed better performance as compared with the results obtained in literature as well as from commercial summarizers.**

Keywords—Extractive Summarization; Language Independent Summarization; UnB-LITS; Shape-Coding.

I. Introduction

As the amount of textual Information has increased exponentially during the last two decades, especially with the rise of the Internet and electronic media, then developing automatic text summarization model becomes of significant importance [4,5,7]. Text summarization model is essential for information retrieval; inferring knowledge; text processing; and reduction of dimensionality of texts for subsequent classification and understanding [1,2,11].

Since machine text summarization has started in the 1950s by Luhn [10] the goal was to make the computer-generated summary as good and representative of the original text as the human generated one. Generated Summaries can be either abstractive (text rephrasing) or extractive (extract important elements as sentences, key phrases, etc.). In addition, machine text summarization can be: a) *Supervised*, that requires training datasets of labeled documents to select important document elements; b) *Unsupervised*, by applying heuristic rules to extract relevant and important document element; or c) *Semi-Supervised*, which combines labeled and unlabeled data [9].

Text summarization can be performed with a wide range of techniques. The majority of those techniques used in text summarization depend on pre-structured lexicons, databases, parsers and taggers [8]. Such techniques require a prior knowledge of the language of the text being summarized and in certain situation requires knowledge of the contextual domain.

Models presented in literature are heavily dependent, in one step or another, on a language related tools, like parsers, taggers or lexicons. And in some models, it is even required to add hand-coded rules that are language and/or contextual dependent, to decide the selection of some attributes and features. In addition, extraction of key phrases, Named Entities (NE's) [6] or Concrete Concepts (CC's) is highly dependent on parsers, lexicons and taggers, which by definition are language dependent. Efficient tagger or parsers for particular languages are not always available. In addition, preparing a lexicon for a language is not an easy task.

A major limitation of many extractive summarization methods is their language dependency. Where, in text summarization the pre-processing steps include removal of *stop-words* (unimportant words) and stemming or lemmatization of words. Removal of stop-words requires lexicons of those words, in addition stemming and lemmatization needs a language specific dictionary. Other methods require language specific taggers or parsers.

As such, the problem solution is to develop a model that can perform all steps of text summarization in language independent manner. Where, the removal of unimportant words (or excluding their influence), as well as, combining similar words through stemming or lemmatization should be carried out independent of any language specific tool. In addition, the model should be capable of reflecting the importance of key phrases, CC's and Named Entities without the need of language/context dependent lexicons or databases

To tackle the aforementioned problems, we propose an extractive text summarization tool, *UnB Language Independent Text Summarizer* (*UnB-LITS*), which is capable of performing text summarization in a *language independent*

The research is partially supported by the National Council for Scientific and Technological Development (CNPq) by the contract of 304903/2013-2.

manner. The new model depends on intrinsic characteristics of the text being summarized rather than its language and thus eliminates the need for language dependent lexicons, databases, taggers or parsers. Within this tool, we propose an innovative way of coding the shapes of text elements (words, n-grams, sentences and paragraphs), in addition to proposing language independent algorithms capable of normalizing words and performing relative stemming/lemmatization. The proposed algorithms and *Shape-Coding* routine enable the *UnB-LITS* tool to extract intrinsic features of document elements and score them statistically to extract a representative extractive summary independent of the document language. The proposed algorithm is explained in the following 3 sections: II) Shape-Coding; III) Features Extraction and Scoring; and IV) Summary Extraction. Section V explains the experiment, while sections VI presents the Results and section VII is the conclusion of the paper.

II. Proposed Shape-Coding Definition and Approach

Shape-Coding is the process of encoding the main shape features of a document element (word, n-grams, sentence and paragraph) using a set of predefined set of codes. The aim of *shape-coding* is to create few intuitive classes of coded elements that can be manipulated statistically. *Shape-coding* takes into consideration both the nature of the characters forming a document element (numbers or letters) as well as the case of that element (capital, small or mixed) to extract the most influential words, n-grams, key phrases and sentences.

A. Shape-Coding of Words and N-grams

Words are the building blocks of a document or text. As such, coding the shape of a word will be reflected, directly, in coding its parent *n-grams* and, indirectly, in coding the containing sentences and paragraphs. *Shape-coding* of a word means transferring each character in the word to its corresponding code in a process that results in a compact code, which in turn, reflects the word's important shape features.

In words *shape-coding*, the code set consists of 6 elements (or letters) that are used to reflect the word shape features. The elements of this code set are {X, x, C, c, N, n} as in Table 1.

TABLE 1. WORD CODE SET

Code Element	Indication
X	Indicates a single capital letter
x	Indicates a single small letter
C	Indicates 1 or more capital letters
c	Indicates 1 or more small letters
N	Indicates a single numeric character
n	Indicates 1 or more numeric characters

Shape coding of a word is done in four main steps. Table 2 shows detailed word shape-coding examples using the following steps:

i. Remove all non-alphanumeric characters as: {. , " / & ; : @ etc.}.
ii. Change all numeric characters to "N".
iii. Change all letters to "X" and "x" for capital and small letters respectively.
iv. Group sequential repeated codes using "C", "c" and "n" for repeated "X", "x" and "N" respectively. "C", "c" or "n" are used to replace a sequence of similar characters of a size of 2 or more. The first character of the identical sequence is kept unchanged while "C", "c" or "n" replace all the following similar characters.

TABLE 2. SHAPE CODING OF WORDS AND NUMBERS

Word	Coding
game	game → xxxx → xc
USA	USA → XXX → XC
U.S.A.	U.S.A. → USA → XXX → XC
UnB-LITS	UnB-LITS → UnBLITS → XxXXXXX → XcXC
2-way	2-way → 2way → Nxxx → Nxc
25.44	25.44 → 2544 → NNNN → Nn
9	9 → N → Nn

The *shape-coding* of words results in coding those words into a small set of equivalent classes that represent their shapes. Where, for example, all numbers are normalized to "Nn", while the majority of words in the texts are converted to "xc". On the other hand, names of persons, cities are converted to "Xxc" that is less common than "xc" reflecting their importance. In addition, the step of removing non-alphanumeric characters adds more power to the proposed model as it helps in normalizing some similar words (especially abbreviations), where, for example, words like "USA" or "U.S.A." referring to the United States of America will be treated the same way in our model as both words will have the same coded word shape "XC".

In addition, word formatting can be coded easily using the proposed shape-coding technique; simply by applying the word formatting on the coded word shape. For example, a word like "***Brazil***" with bold and italic formatting will have a coded shape "***Xxc***" where the word shape become bold and italic as well to reflect the formatting of the original word.

The rareness of a word shape in a document reflects the importance of that word. For example a capitalized word like USA will have a word shape of "XC", or a word like the name of our proposed tool "UnB-LITS" will have a shape "XcXC". Both shapes "XC" and "XxXC" are more rare in a document than all other common words (verbs, nouns, adjectives, etc.) that have the shape "xc". In addition, words with rare word shapes are most likely to be a Named-Entity (name of a person, city, country, tool, abbreviations, etc.).

As a conclusion, *shape-coding* of words results in: a) encoding words into a small set of equivalent classes; b) normalize numbers and similar words; c) identifies important words and Named Entities.

On the other hand, *shape-coding* of n-grams is performed simply by concatenating the shapes of the n-gram's individual words. Where, a *bigram* can be shape-coded by coding its two individual words and then join them together with a space delimiter. Some examples are listed in TABLE 3.

TABLE 3. SHAPE-CODING OF N-GRAMAS

Class	n-grams	Coded n-grams
Bigram	school bus	xc xc
Trigrams	189 square feet	Nn xc xc
4-grams	Arab Republic of Egypt	Xxc Xxc xc Xxc
5-grams	BFC bought 88% of BankAtlantic	XC xc Nn xc XxcXxc

B. Shape-Coding of Sentences

Shape-coding of a sentence means transferring some of its main features to a representative code, resulting in a single compact code reflecting the sentence's important features. Sentence shape-coding is much simpler than word's shape-coding, where the code set consists of only 2 elements {Z, z} that are used to reflect the sentence's features. These elements and their indications are explained as Z: Indicates a sentence with a capital initial letter, and z: Indicates a sentence with an initial lower case letter.

Before start coding a sentence, it is worth mentioning that, alike the word's shape-code, the sentence shape-code has a fixed length of three codes whatever the shapes of the individual words of the sentence are. Shape coding of a sentence is done as follows:

i. If the sentence starts in a word with capital first letter then add "***Z***" to the beginning of the code, while if it starts with a lower case letter (which is very rare) then add "***z***" to the beginning of the shape-code.

ii. Count the number of words starting in lower case letter (*L*) and the number of words starting an upper case letter (*U*).

iii. Calculate the ratio of words with initial capital letter in the sentence to its total number of words and then decide the second part of the *shape-code* as per equation (1).

$$ShapeCode = \begin{Bmatrix} ZZ \ \ if \ \ \dfrac{U}{L+U} \geq \omega_1 \\ zZ \ \ if \ \ \dfrac{U}{L+U} < \omega_1 \ and \ \omega_2 \\ zz \ \ if \ \ \dfrac{U}{L+U} < \omega_2 \end{Bmatrix} \qquad (1)$$

where, ω_1 and ω_2 are the code-shaping thresholds that are decided by the model designer. In this study the default values of these thresholds are: ω_1=0.8 and ω_2=0.4.

iv. A sentence that starts with a word with initial capital letter, its first code is "Z", then coding the rest of the sentence will be performed as follows:

a) If the ratio of words that starts in capital letter is more than 0.8 then the code part "***ZZ***" will be added to the shape-code to be: "***ZZZ***".

b) If the ratio is more than 0.4 and less than 0.8 then the code part "***Zz***" will be added to the code to be: "***ZZz***".

c) On the other hand, if the ratio of words that starts in capital letter is less than 0.4, then the code part "***zz***" will be added to the shape-code to be: "***Zzz***".

v. If the sentence starts with a lower case word, then the sentence can be coded as follows: "***zzz***", "***zZz***" or "***zZZ***".

vi. In cases where the sentence starts with a number, then consider the number as an upper case letter, as such, the *code-shape* of the sentence will start with "***Z***".

The rareness of a sentence shape in a document reflects the importance of that sentence. For example the title of a document is a sentence with shape-code "***ZZZ***" which is more rare than normal sentences with "**Zzz**" code and thus has heavier weight.

C. Shape- Coding of Paragraphs

Shape-coding of a paragraph means transferring some of its main features to a representative code in a process that results in a single compact code reflecting the paragraph's important features. The paragraph's *code set* consists of 7 elements {B, N, O, S, M, P, p} whose indications are explained in TABEL 4.

TABEL 4. PARAGRAPH CODE SET

Code Element	Indication
N	Indicates Numbered Paragraphs
B	Indicates Bulleted Paragraph
O	Indicates Ordinary Paragraph (not numbered or bulleted)
S	Indicates Single Sentence Paragraph
M	Indicates Multiple Sentences Paragraph
P	Indicates a paragraph with a capital initial letter
p	Indicates a paragraph with an initial lower case letter

Similar to sentence shape-code, the paragraph shape-code has a fixed length of five codes whatever the shape of the size of the paragraph is. The five length coded shape has pre-defined fixed position for every main feature in the paragraph: a) The first letter in the code is for bullets and numbering; b) the second letter is for the number of sentences forming the paragraph; while c) the last three letters are for the spread of words that starts with upper and lower case letters within a paragraph. Shape coding of a paragraph is done in the following main steps:

i. The first letter in the paragraph's coded-shape is to indicate whether the paragraph is *numbered*, *bulleted* or *ordinary* paragraph. Thus the first letter in the code will be "***N***", "***B***" or "***O***" respectively.

ii. The second letter in the paragraph's coded-shape is to indicate how many sentences composes the paragraph. Thus the second letter in the code will be "***S***" or "***M***".

iii. If the paragraph starts in a word with capital first letter then add "***P***" to the third position of the code, while if it starts with a lower case letter (which is rare) then add "***p***".

iv. Count the number of words that starts in lower case letter (*L*) and the number of those words that starts with an upper case letter (*U*) in the entire paragraph.

v. Calculate the ratio of words with initial capital letter in the paragraph to its total number of words and then decide the last two letters of the *shape-code* as seen in equation (2).

$$ShapeCode = \begin{Bmatrix} PP \ \ if \ \ \dfrac{U}{L+U} \geq \omega_1 \\ pP \ \ if \ \ \dfrac{U}{L+U} < \omega_1 \ and \ \omega_2 \\ pp \ \ if \ \ \dfrac{U}{L+U} < \omega_2 \end{Bmatrix} \qquad (2)$$

where, ω_1 and ω_2 are the code-shaping thresholds, similar to those of the sentence *shape-coding*, and decided by the model designer. In this study the default values are: ω_1=0.8 and ω_2=0.4 as mentioned in the previous sub-section.

vi. Defining the last two letters in the paragraph coded-shape can be done as follows:

a) If the ratio of words that starts in capital letter is more than 0.8 then the code part "***PP***" will be added to the shape-code. Therefore the last three letters in the code will be: "***PPP***" which indicates that the paragraph is almost fully capitalized.
b) While, if the ratio is more than 0.4 and less than 0.8 then the code part "***Pp***" will be added to the shape-code. Therefore the last three letters in the code will be: "***PPp***" which indicates that the paragraph has mixed case words in considerable amount.
c) On the other hand, if the ratio of words that starts in capital letter is less than 0.4, then the code part "***pp***" will be added to the shape-code. Therefore the last three letters in the code will be: "***Ppp***", that implies that the paragraph is mostly lower case, similar to the majority of paragraphs in a Latin text.

vii. In cases where the paragraph starts with a number, not a numbered list, then consider the number as an upper case letter. As such, the third letter of the *code-shape* of the paragraph will start with "***P***".

S*hape-coding* of paragraphs results in encoding those paragraphs into a small set of equivalent classes that represent their shapes. As in the case of words and sentences, the rareness of a paragraph shape in a document reflects the importance of that paragraph. For example a single sentence paragraph with mostly capitalized words has a coded-shape "***OSPPP***", which is most probably represents a title, where normal paragraphs in the text will have a shape like "***OMPpp***". Thus the rareness of "***OSPPP***" is a reflection to its importance and therefore has heavier weight.

III. Feature Extraction Procedure by Language Independent Text Summarization Model (UNB-LITS)

A. *Words Scoring and Features Extraction*

Words are the main building blocks of all other document elements starting from *n-grams* up to paragraphs. Extraction of word features is a process of extracting statistical values that depend on the word form (spelling) as well as its coded shape. A *word score* is a combination of the extracted features weights. Scoring higher document elements (n-grams, sentences and paragraphs) will depend heavily on the scores of words forming that element. As such, word feature extraction and its subsequent scoring is the most crucial step in the whole model. The extracted features are: word shape, word shape frequency, word form frequency, and overall word Score.

On the contrary of TF-IDF scoring, word feature extraction and scoring in the proposed model depends on two assumptions: a) rare words in a text have more influence on the meaning of the text and should get heavier weights; and b) rare or less frequent word shapes may indicate a Named Entity that implies more influence in the text, hence, should receive heavier weights. In conclusion, the more rare the word's form and shape are, the higher the word score and thus the more important it is for summarizing the text.

The process of word features extraction involves many steps that begins with language independent pre-processing; *shape-coding;* calculating frequencies of shapes and forms; and ending with computing the final word score. Fig. 1 shows the process of word features extraction starting from the basic preprocessing till computing the word score. The steps of word features extraction are:

i. *Extract Words (Tokenization)*: A document is segmented into its individual tokens (words).
ii. *Removal of special characters*: All none alphanumeric characters, as commas, hyphens, points, semicolons, etc., are removed from each individual token.
iii. *Normalize Numbers*: If a token is a number, then convert it to "####". This normalization step of numbers is crucial to avoid assigning false high weights for each number. Since numbers tend to be different in a document, without normalization, the model will consider every number as a rare influential word leading to an over estimated importance of the number and its containing sentence.
iv. *Normalize Word Forms:* Coded shapes depend on the word form and its case (upper case or lower case letters or mixed). And since the rareness of word shape reflects its importance (usually Named Entities are capitalized or start with upper case letter), thus it's crucial to normalize word case such that upper case words are those of NE's and not for words that accidently appear at the beginning of a sentence. To achieve this normalization, each word that starts with a capital letter is searched in the whole document. If the word was found in any other position in the document with lower case letters, then, the word case is changed to lower case.
v. *Combining Similar Words*: Some words are similar and should be treated as one when counting the frequency of occurrence of words. For example, in English, words like "*year*" and "*years*"; "*automatic*" and "*automatically*"; or "*allow*" and "*allowed*"; and in Portuguese, "*embaixada*" and "*embaixadas*"; or "*quero*" and "*quer*". In supervised text summarization techniques *stemming* or *lemmatization* are used to perform the process of combining similar words, which requires database of words or hand coded rules that are language dependent. And since our proposed model is language independent, so it is important that we construct a technique that combines similar words depending on word forms and their degree of similarity whatever the language is. The proposed technique computes the length of sequence of common letters between two words starting from the letter at position 1. If the *Degree of Similarity* (*DoS*), equation (3), exceeds a predefined threshold, then the longer word is changed to the smaller one (in this study the threshold is set to 0.7). Examples are listed in TABLE 4.

$$DoS = \frac{\textit{length of common letters sequence}}{\textit{length of the longer word}} \quad (3)$$

vi. *Word Shape-Coding:* After normalizing tokens by converting numbers to "####", change the word case to the appropriate one and combine similar words, each character in the token is mapped to its appropriate shape code as mentioned in the previous section.

vii. *Compute Word Frequency and its Word Form Weight (WfW):* Get the count of each word in the text to be summarized. This word frequency is used to compute the weight of the word form (*WfW)*. Since, the proposed model assumes that a rare word has more influence on the text than a more common word, thus as seen in equation (4) , the *WfW* is computed by taking the inverse of the natural logarithm (*ln* or log of base 2) of the word count.

$$WfW = \frac{1}{\ln\left(count\left(word\right)+1\right)} \tag{4}$$

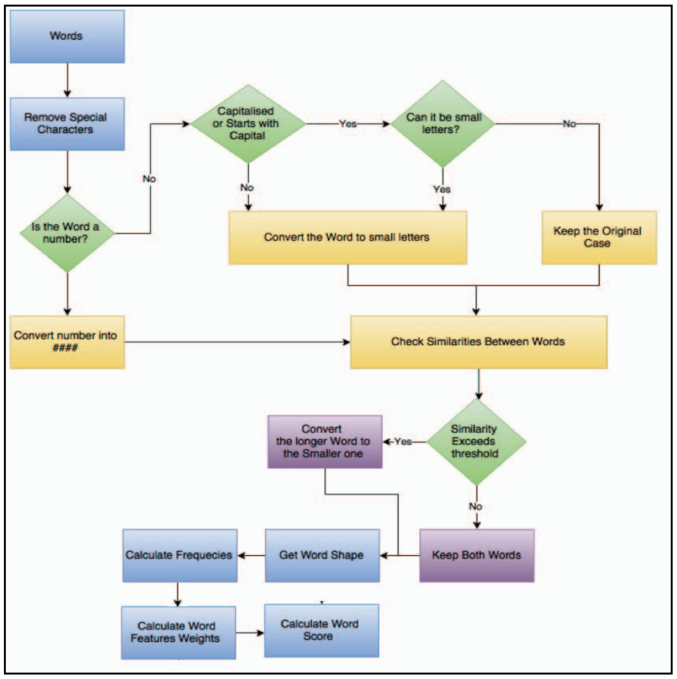
Words
Remove Special Characters
Is the Word a number?
No
Capitalised or Starts with Capital
Yes
Can it be small letters?
No
No
Yes
Convert the Word to small letters
Keep the Original Case
Convert number into ####
Check Similarities Between Words
Similarity Exceeds threshold
Yes
Convert the longer Word to the Smaller one
No
Keep Both Words
Get Word Shape
Calculate Frequecies
Calculate Word Features Weights
Calculate Word Score

Figure 1. Process of word features extraction and Word Score computation.

TABLE 4. COMBINING SIMILAR WORDS USING THRESHOLD 0.7

Word 1	**Word 2**	**Number of Similar letters**	***DoS***	**Action Performed**
An	And	2	2/3 = 0.66	Both words are maintained
University	Universities	9	9/12= 0.75	"*Universities*" is changed to "*University*"
Quer	Quero	4	4/5 = 0.8	"*Quero*" is changed to "*Quer*"

The natural logarithm is inverted in order to give non-linear higher weights to rare words in comparison to abundant words. While, the add 1 normalization is done to avoid dividing by zero in case of a word that was mentioned only once, as *ln(1) = 0*.

The impact of the natural logarithmic weight and the assumption of the higher influence of rare words are better understood using an example. Where, a word like "the" may be mentioned in a piece of text around 40 times, while a word like "*USA*" is mentioned twice. Thus the *WfW* of "the" in this case is equal to 1/(ln(40+1)) = 0.2693, while a rare word "*USA*" will get a *WfW* = 0.9102. This result reflects clearly the difference in importance between these two words and the subsequent effect on evaluating the importance of particular sentence.

viii. *Compute Coded Shape Frequency and its Word Shape Weight (WsW):* Get the count of each word shape in the text to be summarized. This shape frequency is used to compute the weight of the word shape (*WsW)*. Since, the proposed model assumes that a rare coded shape has more influence on the text than a more common shape, thus as seen in equation (5) the *WsW* is computed by taking the inverse of the logarithm (log of base 10) of the coded shape count.

$$WsW = \frac{1}{\log\left(count\left(shape\right)+1\right)} \tag{5}$$

It is worth mentioning that logarithm of base 10 is used to compute *WsW* rather than the natural logarithm used for computing *WfW* since this technique will give much higher weights for word shapes in comparison to their forms. If two words have equal frequency of occurrence, then the word with more rare coded shape will have get *Word Score* (*WS*).

ix. *Computing Word Score (WS):* Computing the *Word Score* (*WS*) of a token is simply done by combining its *Word form Weight* (*WfW*) and its *Word shape Weight* (*WsW*) as seen in equation (6) below.

$$WS = \left(2-\alpha\right)WfW + \alpha WsW \tag{6}$$

where α is a constant greater than or equal to 0 and less than or equal to 2. It is used to adjust the relative weights of each term of the *WS* computing equation. Usually α is set to 1 in order to maintain the relative weights of both terms of the equation imposed by the difference between *log* and *ln*. However, in certain situations where short texts are to be summarized, it is recommended to give larger weight for the shape term *WsW,* as in case of short text it may happen that unimportant words may occur in an unusual low frequency. In this situation higher α is used to give more weights for named entities.

As discussed previously rare words with rare shapes will get higher *WS* than the most abundant ones. Where a common words like "*the*" with frequency of occurrence = 40 and its coded shape "*xc*" frequency = 300, will have WS=1 × 0.2693+1 × 0.4035 = 0.6728. While, the word "*USA*" that exists 2 times and its coded shape "*XC*" exits 3 times, thus its WS=1 × 0.9102 +1 × 1.6610 = 2.5712. The *WS* reflects the importance of the token "*USA*" as compared to the token "*the*".

As discussed above, the word shape is given higher weight than the word form by using the natural logarithm for computing *WfW* and logarithm of base 10 for computing *WsW*. This technique of computing weights is useful in giving a relatively lower score for relatively unimportant words with very common shape "*xc*" that may accidently appear fewer times in the text to be summarized.

B. N-grams Scoring and Features Extraction

As defined earlier an *n-gram* is a sequence of *n* words from a given sequence of text, where *n* is the size of the window that forms the sequence of words. Extraction of *n-gram* features is a process of extracting statistical values that depend on the n-gram's form (spelling) and its coded shape. The *n-gram score* (*nGS*) is a combination of the extracted feature weights. In this model we are going to extract features for *bigrams*, *trigrams*, *4-grams* and *5-grams*. The extracted features are: N-gram Shape; N-gram Shape frequency; N-gram Form frequency; Overall N-gram Score (*nGS*; Also it can be denoted as *2GS*, *3GS*, *4GS* or *5GS* for bigrams, trigrams, 4-grams and 5-grams respectively).

N-gram feature extraction and scoring depends on one assumptions: high probability of an *n-gram* form and shape in a text implies that it has more influence on the meaning of the text than less probable ones and should get heavier weights (in the contrary to the word scoring assumption). In addition, the *WS's* of words forming the n-gram will influence the overall score of that n-gram (*nGS)*.

The process of *n-gram* features extraction involves many steps that depend mainly on the previous step of *Word Features Extraction*. It starts with concatenating normalized tokens and their shapes; and ends up with computing the final n-Gram Score (*nGS*). No preprocessing or normalization techniques are applied in this step as the tokens are already normalized and pre-processed before. The steps of *n-Gram* features extraction are:

i. *Extract N-Grams*
ii. *Get N-Grams Shape*: This is simply done by concatenating the coded shapes of the tokens composing the n-gram separated by a single space character.
iii. *Compute N-gram Frequency and its N-gram Form Weight (nGfW):* Get the count of each *n-gram* in the text to be summarized. The *n-gram* frequency is used to compute the Maximum Likelihood Estimate (MLE) of that n-gram form. Since, the proposed model assumes that the more likely (probable) the *n-gram* is the more influential it is, thus the *nGS* is the product of the MLE of the *n-gram* form (*nGfW*) and shape (*nGsW*). Equation (7) shows the way to compute *nGfW* .

$$nGfW = \left(\prod_{i=1}^{n} P\left(w_i \mid w_1 ,\ldots,\ w_{i-1}\right)\right) \times \sum_{i=1}^{n} WS\left(word_i\right) \tag{7}$$

$$where,\ \mathrm{P}\left(w_i \mid w_1,\ldots,w_{i-1}\right) = \frac{count\left(w_1,\ldots,w_i\right)}{count\left(w_1,\ldots,w_{i-1}\right)} \tag{8}$$

where, w_i is the word at position *i* of that *n-gram*. While, $\sum WS(word_i)$ is the sum of word score of all the words that form the n-gram. This term is included to reduce the *nGfW* value for probable n-grams that are made from weak words and give higher weights for n-grams made from stronger words.

As seen in equation (8), $P(w_i|w_1,w_2,\ldots,w_{i-1})$ is the maximum likelihood estimate of an n-gram formed from *i* words, which is calculated as the probability of the word w_i given the sequence of words $(w_1,w_2,\ldots,w_{i-1})$, which is equal to ratio of the frequency of occurrence of the whole sequence of words $(w_1,w_2,\ldots,w_{i-1})$ to the frequency of occurrence of the sequence formed from the words $(w_1,w_2,\ldots,w_{i-1})$ [3].

iv. Compute *N-gram Shape Frequency and its N-gram Shape Weight (nGsW*) as per equation (9).

$$nGsW = \left(\prod_{i=1}^{n} P\left(s_i \mid s_1 ,\ldots,\ s_{i-1}\right)\right) \tag{9}$$

$$where,\ \mathrm{P}\left(s_i \mid s_1,\ldots,s_{i-1}\right) = \frac{count\left(s_1,\ldots,s_i\right)}{count\left(s_1,\ldots,s_{i-1}\right)} \tag{10}$$

where, s_i is the shape at position *i* of that *n-gram*. And as seen in equation (10), $P(s_i|s_1,s_2,\ldots,s_{i-1})$ is the maximum likelihood estimate of an n-gram formed from *i* coded shapes.

v. *Computing N-gram Score (nGS): is simply done* by combining *n-gram form Weight* (*nGfW*) and its *n-gram shape Weight* (*nGsW*) as seen in equation (11).

$$nGS = nGfW + nGsW \tag{11}$$

As discussed previously high probable n-grams form and shapes composed of rare words will get higher *nGS* than the less probable and/or weak ones. For example in a text about astronomy, a relatively abundant bigram like "*Solar System*" will get higher *nGS* than less abundant one like "*the flare*". However in certain circumstances, a highly probable n-gram can receive lower *nGS* than less probable ones. This occurs when the n-gram is composed of weak words with weak *WS*.

C. Paragraph Scoring and Features Extraction

The paragraph score (*PS*) is obtained by computing the weight of the paragraph coded-shape as seen in equation (12).

$$PS = \frac{1}{\log\left(count\left(shape\right)+1\right)} \tag{12}$$

The logarithm is inverted in order to give non-linear higher weights to rare paragraph shapes in comparison to abundant ones.

D. Sentence Scoring and Features Extraction

Sentence scoring is the ultimate goal of the proposed automatic summarization algorithm. After all the sentences in a text are scored, then, they are arranged in descending order according to their overall score *Sentence Score* (*SC*). Sentence scoring depends on both intrinsic properties within the sentence itself, like the sentence shape, as well as the scores of all other documents elements. Equation (13) is used to calculate the Sentence coded-shape Weight (*SsW*).

$$SsW = \frac{1}{\log\left(count\left(shape\right)+1\right)} \tag{13}$$

Computing the final sentence score (*SC*) is explained in the following section.

IV. EXTRACTING THE SUMMARY

After preprocessing the text (in language independent manner), code the text elements, extract their features and score them, then the final sentence score (*SC*) is calculated as seen in equation (14).

$$SC^{j}=\eta\left(\lambda_1\sum_{i=1}^{N_w^j}\frac{WS_i}{\max(WS)}+\lambda_2\sum_{i=1}^{N_{2g}^j}\frac{2GS_i}{\max(2GS)}+\lambda_3\sum_{i=1}^{N_{3g}^j}\frac{3GS_i}{\max(3GS)}+\lambda_4\sum_{i=1}^{N_{4g}^j}\frac{4GS_i}{\max(4GS)}+\lambda_5\sum_{i=1}^{N_{5g}^j}\frac{5GS_i}{\max(5GS)}\right)+\beta\frac{SsW^j}{\max(SsW)}+\delta\frac{PS^j}{\max(PS)} \quad (14)$$

where, N_w^j, N_{2g}^j, N_{3g}^j, N_{4g}^j and N_{5g}^j are the total number of words, bigrams, trigrams, 4grams and 5grams in the sentence j respectively. The weights λ_{1-5} are weights given to word and n-grams scores respectively, to help tweaking the sentence-scoring performance, where $\lambda_1+\lambda_2+\lambda_3+\lambda_4+\lambda_5=1$. In addition, η, β *and* δ are weights ranging between 0 and 1 to adjust the relative importance of the sum of words and n-grams scores (*WS* and *nGS*); the sentence shape (*SsW);* and the Paragraph Score (*PS*) respectively.

While, max(..) means the maximum value of that score in the entire text (not only the sentence). All scores are normalized by dividing them with the maximum value of the score noticed in the entire text. This is done in order to keep the scores of all elements between 0 and 1 and prevent any unwanted effect of off-scale scores for some terms.

After the document elements are scored properly, all sentences in the text are then arranged in a descending order of their scores (*SC*). Then the top *N* sentences are selected, where the number of sentences to be selected (*N*) depends on the Degree of Compression of the summary. Finally, the top sentences (with the highest scores) are extracted and ordered according to their order in the original text to produce the required summary.

V. EXPERIMENT

A. Datasets and Evaluation Methodology

UnB-LITS is applied for single document summarization of collections of news articles written in four different western European languages: English, Portuguese, French and Spanish. For English language, the experiment is run using a dataset provided in the Document Understanding Conference (DUC 2002)[1]. The dataset consists of 567 news articles and two manually generated summarizes (100 words) per article. For Portuguese, TeMario [2], a Brazilian Portuguese text collection, is used. The collection contains 100 news articles and their manually produced summaries (25-30% of the original document). While, for French and Spanish, two datasets, 40 documents each, were obtained from French and Spanish news websites respectively. Each article has two manually generated summaries of 100 words length.

The results of the summary are evaluated using the F-Score of ROUGE-1 and ROUGE-2 [12]. Results of English and Portuguese experiments are compared with results mentioned in literature using state-of-the-art algorithms. However for French and Spanish experiments the F-Score is compared with the results obtained by commercial automatic summarizers; Apple OSX summarizer (OS X El Capitan version 10.11.1) and the online Automatic Text Summarizer (www.autosummarizer.com).

B. Parameter Selection

In order to achieve the best summarization performance, each language dataset is divided into two sub-sets: a) *Training Set* used to determine the optimum model parameters that achieve the highest ROUGE score; and b) *Test Set* used to evaluate the model.

A deterministic optimization technique is carried out to determine the optimum parameters values. Where, the weights of the λ_{1-5} have varied in the range [0,1] with 0.2 intervals, while the η, β *and* δ has been set to 1 to give equal weights for the three terms of the formula (n-grams, paragraphs and sentences). The ROUGE metrics was calculated for the entire *Training Set*, and then the weights combination that maximizes the ROUGE score was selected (Table 5).

TABLE 5. PARAMETERS WEIGHTS OPTIMIZATION

Language	**English**	**Portuguese**	**French**	**Spanish**
Training Set Size	236	50	20	20
Parameters Value	$\lambda_1=0.6$ $\lambda_{2-5}=0.1$	$\lambda_{1-5}=0.2$	$\lambda_2=0.8$ $\lambda_{1,3-5}=0.05$	$\lambda_1=0.4$ $\lambda_{2-5}=0.15$

VI. RESULTS AND DISCUSSIONS

UnB-LITS is applied on the test sets of the four languages, which is composed of the second half of the language datasets. UnB-LITS generated summary is formed of 100 words per document. Table 6 shows the F-score of *UnB-LITS* applied on DUC 2002 English dataset and compared with the best single document summarization systems reported in literature as presented in [13].

TABLE 6. SYSTEMS COMPARISON (DUC 2002 ENGLISH DATASET).

Algorithm	**R-1 F-Score**	**R-2 F-Score**
UnB-LITS	***0.4927***	***0.2377***
DUC2002 Baseline	0.4751	0.2240
Text Summarization using key concepts	0.4855	0.2304
Ccsnsa.v2: Supervised sentence extraction with Hidden Markov & Logistic Regression Models	0.4830	0.2297
Wpdv-xtr.v1 Supervised sentence classification with WPDV (Weighted Prob. Distribution Voting)	0.4757	0.2218
ULeth 131m: Unsupervised sentence extraction + text segmentation	0.4599	0.2018
Kul.2002: unsupervised sentence extraction and topic segmentation	0.4685	0.2147
Ntt.duc02: Supervised classification with SVM	0.4651	0.2155

1 http://www-nlpir.nist.gov/projects/duc/past_duc/duc2002/test.html

2 http://www.linguateca.pt/Repositorio/TeMario/

For Portuguese Language, the size of *UnB-LITS* generated summaries is relative to the size of the original document to achieve 25-30% compression rate. Table 7 shows the F-score of *UnB-LITS* applied on TeMario Brazilian Portuguese dataset and compared with other algorithms reported in [14].

While, for Spanish and French Languages, Tables 8 and 9 compares the performance of *UnB-LITS* applied on those languages dataset and compared with commercial tools. Generated summaries size is 100 words per document.

TABLE 7. SYSTEMS RESULTS COMPARISON USING ROUGE MEASURE ON TEMARIO PORTUGUESE DATASET.

Algorithm	**R-1 F-Score**	**R-2 F-Score**
UnB-LITS	***0.57***	***0.22***
MMR (λ =0.5)	0.43	0.15
Support Sets (Manhattan Distance and Support set cardinality = 2)	0.52	0.19
KP-Centrality (10 Key Phrases)	0.54	0.20
LSA	0.56	0.20
GRASSHOPPER	0.54	0.19
LexRank	0.55	0.20

TABLE 8. SYSTEMS RESULTS COMPARISON USING ROUGE MEASURE ON SPANISH DATASET.

System	**R-1 F-Score**	**R-2 F-Score**
UnB-LITS	***0.6778***	***0.4652***
Apple OSX Summarizer	0.5310	0.2451
Autosummarizer.com	0.5377	0.2640

TABLE 9. SYSTEMS RESULTS COMPARISON USING ROUGE MEASURE ON FRENCH DATASET.

System	**R-1 F-Score**	**R-2 F-Score**
UnB-LITS	***0.5762***	***0.3389***
Apple OSX Summarizer	0.5075	0.2760
Autosummarizer.com	0.4577	0.2221

UnB-LITS applied on the four datasets demonstrates better performance over all other listed systems as evaluated by F-score of ROUGE-1 and ROUGE-2 metrics as shown in the tables above.

VII. CONCLUSIONS

In this paper a new automatic text summarization model is introduced, which is capable of performing text summarization in a language independent manner. The proposed tool, ***UnB-LITS*** or "***UnB-Language Independent Text Summarizer***", generates efficient language independent extractive summary when evaluated against human generated summaries. The tool extracts intrinsic features of document elements (words, n-grams, sentences and paragraphs) using an innovative way of *Shape-Coding* of those elements. *Shape-Coding* is performed by transferring the document element to a normalized shape that fits into relatively small number of coded classes, which in turn, reflect the importance of elements with rare shapes.

The proposed tool is totally free of any particular language tools dependency due to the fact that it is capable of removing or neutralizing the effect of unimportant words (stop words) without the need of stop words lexicons that are by definition language dependent. The proposed model is capable, as well, of grouping similar words together in a way similar to stemming without the need of language dictionaries. As such, the proposed model preserves the weight of potential words and sentences, with the ability to extract strong and important key phrases with no dependency on external language databases or corpora.

The proposed tool, *UnB-LITS*, was tested on news datasets written in English, Portuguese, French and Spanish. The obtained results were evaluated against human-generated summaries using ROUGE-1 and ROUGE-2 metrics. In case of English and Portuguese the results were compared with algorithms and systems listed in literature. While, for French and Spanish results were compared with those obtained by Apple OSX integrated summarizer as well as the online Automatic summarizer. *UnB-LITS* achieved better performance over other tools in all of the four languages. These results were achieved with no dependency on any specific language related lexicons, parsers or corpora, which proves the quality of the proposed contributions.

In future work, the proposed algorithm can be extended and implemented as a system to summarize multi-documents and to cover other languages.

REFERENCES

[1] A. P. M. Pal and D. Saha. "An Approach To Automatic Text Summarization Using Simplified Lesk Algorithm And Wordnet." *International Journal of Control Theory and Computer Modeling (IJCTCM)* 3, no. 4/5 (2013): 15-23.

[2] C. H. Keong, Z. X. Hu and L. H. Beng. Fusion of simplified entity networks from unstructured text. In Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on (pp. 1-7). IEEE.

[3] D. Jurafsky and J. Martin. Speech and Language Processing. 2nd. 2008.

[4] D. Radev, H. Jing, M. Stys, and D. Tam. "Centroid-based summarization of multiple documents." Information Processing and Management 40, no. 6 (2004): 919-938.

[5] F. Flores and V. Moreira. "Assessing the impact of Stemming Accuracy on Information Retrieval–A multilingual perspective." Information Processing & Management , 2016.

[6] H. L. Chieu and L. N. Teow. "Combining local and non-local information with dual decomposition for named entity recognition from text." In Information Fusion (FUSION), 2012, 15th International Conference on (pp. 231-238). IEEE.

[7] H. Silber and K. McCoy. "Efficiently Computed Lexical Chains As an Intermediate Representation for Automatic Text Summarization. Co." Computational Linguistics 28, no. 4 (2002): 487-496.

[8] I. Mani and M. Maybury. In Advances in Automatic Text Summarization. 1999.

[9] K. Wong, M. Wu, and W. Li. "Extractive Summarization Using Supervised and Semi-supervised Learning." 22nd International Conference on Computational Linguistics. Manchester, 2008. 985-992.9

[10] Luhn, H. "The automatic creation of literature abstracts ." IBM Journal of Research Development 2, no. 2 (1958): 159–165.

[11] M. Gambhir and V. Gupta. "Recent automatic text summarization techniques: a survey." *Artificial Intelligence Review*, 2016: 1-66.

[12] C.Y. Lin. "Rouge: A package for automatic evaluation of summaries." In Text summarization branches out: Proceedings of the ACL-04 workshop, vol. 8. 2004.

[13] K. Sarkar. "Automatic single document text summarization using key concepts in documents." Journal of information processing systems 9, no. 4 (2013): 602-620.

[14] M. Aparício, P. Figueiredo, F. Raposo, D. Martins de Matos, R. Ribeiro, and L. Marujo. "Summarization of films and documentaries based on subtitles and scripts." Pattern Recognition Letters 73 (2016): 7-12.