Applying Bayesian Belief Networks in Approximate String Matching for Robust Keyword-based Retrieval

Björn Schuller, Ronald Müller, Gerhard Rigoll, and Manfred Lang
Institute for Human-Machine Communication
Technische Universität München
D-80290 München, Germany
(schuller | mueller | rigoll | lang)@mmk.ei.tum.de

Abstract

In this work we present a novel approach towards robust keyword-based retrieval. Thereby Bayesian Belief Networks are applied in a word-model based Approximate String Matching algorithm. Apart from proved reliable performance of a working implementation on standard sources like digital text, wholly probabilistic modeling allows for integration of confidence measures and hypotheses obtained from preprocessing stages like handwriting recognition or optical character recognition respecting uncertainties on the lower levels. Furthermore a flexible method to include the modeling of specific error types deriving from humans and various input sources is provided. The remarkable performance of the algorithms presented was tested during extensive evaluation with respect to Levenstein-Distance, which can be seen as basis of state-of-the-art methods in this research field. The tests ran on a 14K database containing common international music titles and four 10K databases consisting of the most frequently used words in English, German, French, and Dutch language.

1. Introduction

Due to the rapidly growing amount of digital information available at giant archives and the Internet, information retrieval became an established field of research especially during the last decade. Within this area, text retrieval is playing a major role, as text is still the most widely spread form of presentation. Thereby query methods try to find matches of the string retrieval request in the textual data available. If the search engine applies solely exact string matching, many originally adequate hits might be missed because of erroneous data either in the retrieval request or at the retrieved data itself. Such errors may derive from several sources: On the one hand retrieval requests can be corrupted and uncertain due to orthographical or typing errors of the user or results of a handwriting recognition unit on handheld or tablet PC's. On the other hand the textual information database may contain typing and spelling errors as well as uncertainties and hypothesis of pre-positioned optical character recognition stages. Furthermore during transmission over lossy channels random deletion of several characters may appear. Therefore Approximate String Matching is a must to achieve adequate robustness in keyword-based retrieval scenarios.

Unlike Boolean string comparison, such soft matching requires a measurement of similarity of two strings S_A and S_B. Early approaches on this topic have been made by V. Levenstein [1]. He proposes that similarity can be determined by the minimum number of operations to turn string S_A into S_B by editing S_A. Thereby three kinds of edit operations are defined: Substitution, deletion, and insertion of one character at a time. This minimum number of necessary alterations is referred to as Levenstein-Distance and is widely used as the basis of state-of-the-art soft matching algorithms [2]. Unlike these, a novel approach towards Soft String Matching is presented in this work: It uses fully probabilistic modelling of character sequences with Bayesian Belief Networks [3]. This allows the processing of uncertainty in input strings as well as the calculation of real confidences for different matching hypotheses. Thus all knowledge lying in the uncertainty of both system input and system output can be conserved until a final decision has to be made.

2. Approximate String Matching

Our method of determining similarity between S_A and S_B quantitatively requires to build up a model of one string, say S_A, at first. In a subsequent step

information of the reference pattern S_B is fed into this model in order to draw a comparison. The corresponding algorithm starts with securing invariance against character cases by switching all to lowercase. In the following the input sequence is split into units of characters. A unit may consist in one or more characters, which represent meta-characters as phonemes or spelling variations. This is the first step considering source adapted error modeling, here exemplarily shown and implemented for orthographic mistakes. The latter appear due to ambiguities in spelling as different combinations of characters may represent same phonemes. Common examples in English are character units like doubled and single consonants or "*ea*", "*ee*", "*ie*", and "*ei*", all inter alia standing for "*i*", their respective phonetic transcription. If such a character unit representing a phoneme is observed in the string to be modeled, the characters of the unit will not be separated and their similar units will be kept as alternatives. As our target application is a music retrieval database with international titles, we used 37 different character units to cover prevalent spelling mistakes in the languages contained. As mentioned, the error modeling also has to take into account the error characteristic of the source. Considering a handwriting recognition engine, the alternative units consist of characters that are probably confused due to the confusion matrix of the engine applied. After the accomplishment of hence described steps, a graph of parent-child relations between the string to be modeled, its character units, and their alternatives can be constructed. This graph constitutes the structure of a discrete Bayesian Belief Network, which is applied as mathematical model in our algorithm. Figure 1 shows the corresponding structure for the word "ease".

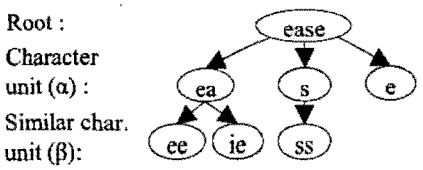

Figure 1. Exemplary Belief Network graph for "ease"

Next to the constitution of the network structure, a number of parameters have to be set. Since discrete Belief Networks with soft evidences are used, the parameters are the probabilistic dependencies between each node and its parent node as well as the initial probability of the root node itself. Concerning individual importance of all character units α on the midlevel, we assume equal properties. Comparable conditions exist between similar character units β and their parent nodes. Likewise, overall three values must be set to define the complete network. These are the probability of the root node $P(Root)$ and the conditional probabilities $P(\alpha|Root)$ as well as $P(\beta|\alpha)$. The strong correlation expected between character units and their similar alternatives lead to setting $P(\beta|\alpha)$ to 0.95, which proved suitable during evaluation. Unlike this, the establishment of the remaining parameters has to be adapted to the number of units contained in the character sequence attended. The mathematical reasons for this will be discussed in the description of the matching algorithm. So far it can be motivated by an interpretation of $P(Root)$ and $P(\alpha|Root)$: $P(Root)$ can be seen as the probability of appearance of the string modeled. A monotone decrease of probability of occurrence was observed for strings with increasingly more than four characters. On the other hand a single character unit of the string gets less significant the greater their number is. Considering these aspects we have two indirect proportionalities to the length L of the string. Since the construction of model is to be automated, a function for each parameter in dependency of L must be used. Two functions have been found heuristically, which meet requirements described later on.

$$P(Root) = p - 8 \cdot 10^{-3} \cdot L \;;\; P(\alpha \mid Root) = q - 8 \cdot 10^{-3} \cdot L$$

The letters p and q stand for initial probabilities. Best results could be achieved with following settings for: $p = 0.25$, and $q = 0.75$ for character confidences equal 1. Otherwise recognition probabilities of the co-domain [0 ; 1] are mapped on the range [0.55 ; 0.75] and will be put in for q for the corresponding character. The reason for the need of mapping is that only values >0.5 for $P(\alpha|Root)$ can have a positive impact on $P(Root)$ in case of evidence.

After the execution of all steps described, a Bayesian Belief Network has been built that is suitable for the final task of measuring similarity of the modeled string to any other character sequence.

In determining similarity of two strings, following two factors are playing a major role:

1. Number of identical or similar character units
2. Degree of adherence to the order of the units

In order to introduce notations used in the following, let S_A be the modelled string and S_B the reference string. Furthermore say N_X is the length of S_X in number of units, whereby $X \in \{A,B\}$, and let $U_{X,n}$ be a unit at index n in string X, with $1 \leq n \leq N_X$.

The important two factors mentioned above, are processed together in three steps.

For initialisation ($n = 0$) the variable $I_B(n)$ is set to -1. At first S_B is scanned for each unit $U_{A,n}$. In case that $U_{A,n}$ is found, the index of the position in S_B will be stored in $I_B(n)$. Additionally the distance D_n to the previous place of finding $I_B(n-1)$ is calculated as follows:

$$D_n = I_B(n) - I_B(n-1) - 1$$

D_n actually keeps the number of character units that have been let out in S_B between the previous found unit S_A and the actual found unit of S_A. In case that $U_{A,n}$ can not be discovered in S_B, the search extends to the similar units of $U_{A,n}$. If this is successful, the distance is not stored in D but in a separate list labelled as DS. Otherwise, $I_B(n)$ takes the value of $I_B(n-1)$, its forerunner, and D_n therefore results in -1. Furthermore found units in S_B must be denoted to avoid double matching. After finishing this first step we get a list of values D_n, each corresponding with the n-th unit of the modelled string S_A.

The second step is applied to reduce computation time. Thereby it is important to know that the similarity calculus will perform with comprehension of the valid entries in the lists D and DS. Negative values indicate that either the accordant unit $U_{A,n}$ could not be found or that a retrace had to take place concerning the previous unit of S_A. Both possibilities do not argue for similarity of S_A and S_B. Therefore those entries are denoted as invalid. Same takes place if any values larger than 3 appear, as this means that more than 3 units in S_B were let out till a new matching succeeded. Now normally a decent number of valid entries are left. If the sum of them in D and DS is larger than M percent of N_A the algorithm proceeds to step three. Otherwise similarity of S_A and S_B is assumed to be too small and not worth computing. Evaluative experiments proved that at a value of $M = 0.3$ no negative impact concerning recognition rate occurred. On the other hand the computing time decreased up to 55% depending on the content of used databases.

As mentioned, step three takes on the final calculation of similarity. The principal is to let nodes of the Bayesian Belief Network model achieve evidence, only if their corresponding character unit was found in valid order. Achieving evidence means that the probability $P(\alpha)$ or $P(\beta)$ of those nodes is set to a definite value, being propagated especially to the root node. At the end $P(Root)$ will indicate the degree of similarity of S_A and S_B. In Belief Networks a direct proportionality is present between $P(Root)$ and $P(\alpha)$ as well as $P(\beta)$, due to the setting of the network parameters described. On the one hand all units, whose corresponding entry in D or DS is zero get full evidence, meaning $P(\alpha) = 1$ or $P(\beta) = 1$. This seems reasonable since $D_n = 0$ indicates that a matching occurred in correct order. For values $D_n \in \{1,2,3\}$ the evidence has to weaken incrementally to allow for the order aberration. An adequate function to implement this proved to be

$$P_n(\alpha) = P_n(\beta) = 1 - D_n \cdot 0.1 \ .$$

Now a short review which cases have been covered hence: For $N_A > N_B$, $P(Root)$ will be reduced by the fact that maximum N_B nodes of the model will achieve evidence. If $N_A < N_B$, the occurrence of excrescent units at the beginning of S_B and between units, matching with S_A, is punished by reducing evidence probabilities. However, the possibility that S_B consists of S_A and a suffix, consider e.g. adverbs like "common" (S_A) and "commonly" (S_B), is still not handled. Therefore the evidence of the node modelling the last matching unit $U_{A,i}$ will be weakened according to the number of following character units in S_B, which is $N_B - i$. The reduced evidence probability P_i' is calculated to

$$P_i' = P_i - 0.05 \cdot (N_B - i)$$

Thereby P_i may be $P_i(\alpha)$ or $P_i(\beta)$. So evidence of the node of "n" in "common" is reduced by 10%, as it contains two additional units at the end.

Now the setting of the network parameters $P(Root)$ and $P(\alpha|Root)$ is getting reasonable. Imagine two models of two different strings S_A and $S_{A'}$ with $N_A < N_{A'}$ and a reference string S_B with $N_A \leq N_B$. Say S_B, S_A, and $S_{A'}$ are containing an equivalent sub-string S_y of length $N_y \leq N_A$. The nodes representing the units of S_y will achieve equally high evidences during the execution of the matching algorithm with S_B in both models. If the parameters $P(Root)$ and $P(\alpha|Root)$ were not adapted to the length of S_A and $S_{A'}$, this would result in almost equal probabilities for $P(Root)$ in the models of S_A and $S_{A'}$. The decay of mentioned net parameters for longer strings assures that the shorter string S_A will achieve a greater probability as it is relatively more similar to S_B.

Due to computation time and storage costs the modelling is not done on database entries but on the query request. As the modelling procedure is wholly deterministic, it does not make a difference in similarity measurement whether S_A is modelled and S_B

constitutes the reference or the other way round. This helps to reduce computation time to a fraction.

3. Evaluation

In order to examine the performance of the described algorithms, a running implementation has been tested on five databases, one containing 14,186 titles of common western music and four comprising the 10,000 most frequent words of English, German, Dutch and French each [4]. Table 1 shows statistical parameters of the distributions of the string length L in characters and the inter Levenstein Distance LD in the applied databases.

Table 1. Distribution of L and LD in the databases.

	L_{Min}	L_{Max}	L_{μ}	L_{σ}	LD_{μ}	LD_{σ}
Titles	4	39	16.9	2.51	17.9	6.32
English	1	18	7.11	2.52	7.33	2.02
German	1	27	8.16	3.09	8.25	2.57
Dutch	1	26	7.60	2.89	7.79	2.36
French	1	19	7.72	2.67	7.81	2.13

In the evaluation sets at first 1,000 entries are selected from each database coincidentally. In a subsequent step each entry is treated randomly with editing operations proposed by Levenstein until a defined relative Levenstein-distance *Rel. LD* in percent is achieved. While all errors are randomly distributed, this can be seen as the worst case as no adequate error modeling can be applied. We build up four times six test corpora for each database. In the first six sets all edit operations were equally distributed (Table 2). In further sets we focused on single edit operations.

Table 2. Perform. at equally distributed edit operations

Rel. LD	5%	10%	20%	30%	40%	50%
Titles	99.7	99.2	97.6	95.7	92.6	89.3
English	99.5	98.1	91.7	77.2	67.4	55.0
German	98.8	95.6	89.5	74.3	68.9	51.7
Dutch	98.9	95.8	91.8	76.7	67.0	55.6
French	97.1	93.4	89.1	70.8	62.6	50.2

To show a comparison in performance, the entries in Table 3 and 4 show the absolute gain $\Delta_{BN\text{-}LD}$ in percent of our approach compared to the very standard Levenstein algorithm, where every edit operation is counted by costs of "1".

Table 3. $\Delta_{BN\text{-}LD}$ for equally distributed edit operations

Rel. LD	5%	10%	20%	30%	40%	50%
Titles	-0.3	-0.8	-2.1	-2.8	-6.4	-8.2
English	-0.5	-1.9	-7.0	-5.8	-4.5	-5.5

Table 4. $\Delta_{BN\text{-}LD}$ for exclusive deletion of characters

Rel. LD	5%	10%	20%	30%	40%	50%
Titles	0.0	-0.1	0.1	2.3	11.9	40.5
English	0.3	0.2	8.5	24.9	40.1	41.9

While the proposed probabilistic algorithm shows slightly worse recognition rates for equally distributed disturbances by substitutions, deletions, and insertions, the achieved gains on lossy channel transmission resulting in random deletion of up to 50% of the originally sent characters are outstanding (Table 4).

4. Conclusion

We proposed a novel approach towards robust Approximate String Matching. As mathematical model Bayesian Belief Networks are applied to allow for probabilistic error modeling and adequate integration of confidences and hypotheses provided by various sources of the input and the retrieval text corpus. In order to be comparable with state-of-the-art algorithms, evaluation considered relative Levenstein-distance. Thereby increased robustness could be proved especially at high degrees of input corruption. The presented methods could be successfully integrated in a multimodal Music Retrieval System combining handwritten, speech and humming input. In future works further refinement of automated error modeling and extended evaluation will be forced.

5. References

[1] V. Levenstein: "Binary codes capable of correcting insertions and reversals," *Soviet Physics Doklady*, 10:707-710, 1966.

[2] G. Navarro: "A Guided Tour to Approximate String Matching," *ACM Computing Surveys*, 33(1), pp. 31–88, 2001.

[3] F. V. Jensen: *An Introduction to Bayesian Networks*, UCL Press, 1996.

[4] U. Quasthoff , et al. : *Projekt Deutscher Wortschatz*, University of Leipzig, Institute of Computer Science, http://wortschatz.informatik.uni-leipzig.de.