

Approximate String Matching for Geographic Names and Personal Names

Clodoveu Davis

clodoveu@pucminas.br

Emerson de Salles

emersondesalles@gmail.com

PUC Minas

The Problem

- ▶ Approximate string matching is an important set of techniques, useful in many applications
 - Information retrieval
 - Digital libraries
 - Ontology integration
 - Computational biology
- ▶ Interest in these techniques reflects the current need to deal with uncertain or semi-structured data, and with varying views of the world

An Experiment

- ▶ Ask the person next to you: "Where do you live?"

(assuming you do not already know the answer)

An Experiment

- ▶ Depending on the context of the conversation, the answer may be a city name, a state, a country, or even an address
- ▶ Anyway, the response will refer to one or more places, using their names

An Experiment

- ▶ The reference to places by name can be imprecise and ambiguous
 - “Paris, France”, or “Paris, Texas”?
 - “London” or “Londres”
 - “São Paulo”: the capital city, or the state?
 - Have I spelled “Itaquaquecetuba” correctly?
 - “Campos do Jordão” or “Campos do Jordao”

An Experiment

- ▶ Errors and misinterpretations can derive from spelling difficulties, use of language or local particularities
 - Proper nouns
 - Foreign words
 - Phonetics
 - Translation between different alphabets
 - Accent marks and their phonetic interpretation

Context

- ▶ Our interest in this subject comes from previous and ongoing research directed at address geocoding, gazetteers, and geographic information retrieval
 - Matching place names
 - Dealing with semantic uncertainty in ontologies and metadata
 - Since many places are named after people, our interest includes personal names as well

The Problem

- ▶ Find out if a string S matches a pattern P within a given similarity threshold δ
 - $0 \leq \delta \leq 1$, or δ can be expressed as a maximum number of errors
 - Create a function $f(S, P)$ that measures the similarity between the two strings



The text in this image is heavily distorted and illegible, appearing as a series of horizontal, wavy lines with a blue and white color scheme.

The Problem

- ▶ There are several metrics for similarity, divided into two main groups
 - **Phonetic matching:** assumes a similarity between individual characters and the sound they usually produce
 - **Pattern matching:** compares characters from each string to detect and quantify points in common
- ▶ It has been shown that phonetic matching is systematically outperformed by pattern matching, even though phonetic algorithms run faster

Pattern Matching

- ▶ One of the most widely known methods for approximate pattern matching is the Levenshtein edit distance
 - Determines the number of insertions, exclusions and substitutions required to transform S into P

		P	a	r	a	n	a	g	u	á
	0	1	2	3	4	5	6	7	8	9
P	1	0	1	2	3	4	5	6	7	8
a	2	1	0	1	1	2	2	3	4	5
r	3	2	1	0	1	2	3	3	4	5
a	4	3	1	1	0	1	2	3	4	5
n	5	4	2	2	1	0	1	2	3	4
a	6	5	2	3	1	1	0	1	2	3
p	7	6	3	3	2	2	1	1	2	3
u	8	7	4	4	3	3	2	2	1	2
ã	9	8	4	5	3	4	2	3	2	2

Matching Personal and Geographic Names

- ▶ Personal and geographic names share some characteristics
 - Usually short strings, and a small number of words
 - Use of special characters or accent marks
 - Abbreviations
 - Titles or professional descriptions
 - Inversions
 - Omission of parts
 - Stopwords
- ▶ Personal names are frequently used as place names
 - 67% of the street names in Belo Horizonte have more than a single word; of these, 65% are personal names

Matching Personal and Geographic Names

- ▶ Our approach
 - Match pairs of single words
 - ▶ Treatment of accent marks
 - ▶ First similarity threshold
 - Match multiword strings considering the results of individual word matching
 - ▶ Treatment of abbreviations
 - ▶ Treatment of inversions
 - ▶ Treatment of stopwords
 - Calculate a similarity index for ranking of alternative matches

Single Word Matching

- ▶ We use a variation of the Levenshtein edit distance method, adapted to deal with accent marks optionally
- ▶ The actual edit distance calculation is preceded by faster tests, in order to determine if a match is at all possible

- Difference between lengths

$$\text{abs}(|S| - |P|) / \max(|S|, |P|) > 1.0 - \delta$$

- Bag test

$$\text{bag}(S, P) = \max(|X - Y|, |Y - X|)$$

Example

- ▶ Considering $\delta = 0.80$, or 2 errors in 10-character strings
 - “Pres” and “Presidente” fail the length test, since the difference of lengths is 6 and δ only allows for 2 differences
 - “Kubtscheck” and “Kubitschek” can match
 - ▶ The difference in length is 1
 - ▶ The bag test also returns 1 (the first string has an extra ‘c’, while the second string has an extra ‘i’)
 - ▶ The strings have to go through the Levenshtein test, giving $f(S, P) = 0.8$, so the words are considered to match within the similarity threshold

Single Word Matching

- ▶ In our implementation, characters that are supposed to match are arranged in groups
- ▶ Different sets of groups can be prepared for various matching jobs, including phonetic equivalence sets

<i>Id</i>	<i>Group</i>
L1	a, á, ã, à, ä, â
L2	e, é, è, ë, ê
L3	i, í, ì, ï, î
L4	o, ó, õ, ò, ö, ô
L5	u, ú, ù, ü, û
L6	n, ñ
L7	c, ç
...	one group for each consonant (lowercase)

<i>Id</i>	<i>Group</i>
U1	A, Á, Ã, À, Ä, Â
U2	E, É, È, Ë, Ê
U3	I, Í, Ì, Ï, Î
U4	O, Ó, Õ, Ò, Ö, Ô
U5	U, Ú, Ù, Ü, Û
U6	N, Ñ
U7	C, Ç
...	one group for each consonant (uppercase)

Matching Multiword Strings

- ▶ The strings S and P are divided into words, considering a set of whitespace characters
- ▶ Points are preserved as the last character of the preceding word, since they can indicate abbreviations
- ▶ Stopwords can be preserved or eliminated

Matching Multiword Strings

- ▶ Phase 1: checking for standard abbreviations
 - Each word is tested against a list of standard abbreviations, and translated into its meaning, if found
- ▶ Phase 2: checking for non-standard abbreviations
 - Candidates are 1-character capitalized words and words that end with a point
 - A match is considered when all characters in the abbreviation match the candidate word exactly
 - A similarity value is calculated

$$f_{NSA}(S[i], P[j]) = \frac{|S[i]|}{|P[j]|}$$

Matching Multiword Strings

► Phase 3: word by word matching

- A matrix similar to Levenshtein's is used, performing a row-wise traversal to the right
- At the end of each row, we select the best match for each word in the pattern, as a word match or as a non-standard abbreviation match

	Antônio	Carlos	de	Souza
Antonio	0.857	*	**	*
C.	**	0.167	0.000	0.000
de	**	**	1.000	**
Sousa	*	*	***	0.800

(*) discarded: bag test; (**) discarded: length test; (***) discarded: previous match

Matching Multiword Strings

► Another example

	Antônio	Carlos	de	Souza
Antonio	0.857	*	**	*
Coelho	*	*	**	*
de	**	**	1.000	**
Sousa	*	*	***	0.800

(*) discarded: bag test; (**) discarded: length test; (***) discarded: previous match

Matching Multiword Strings

► Phase 4: verifying inversions

- The sequence of matches is analyzed, and inversions are counted
- In the example below, the order of matches is 4 / 1 / 3, therefore there is one inversion

	Antônio	Carlos	de	Souza
Souza	**	*	**	0.800
A.	0.143	0.000	0.000	**
C.	0.000	0.167	0.000	0.000

(*) discarded: bag test; (**) discarded: length test; (***) discarded: previous match

Similarity Measures

► Multiword similarity

- Sum of the similarity values found for each matching pair of words divided by the number of matching words
 - At each pair, the similarity may indicate a regular match, or a non-standard abbreviation match
- Average similarity of matching words

$$f_{MW}(S, P) = \frac{\sum_{j=1}^{|P|_w} \left[\max_{i=1}^{|S|_w} (f_{LD}(S[i], P[j]), f_{NSA}(S[i], P[j])) \right]}{v}$$

Similarity Measures

► Valid matches

- Fraction of the words of the pattern for which a match has been found

$$f_{VM}(S, P) = \frac{v}{\max(|S|_w, |P|_w)}$$

Similarity Measures

► Inversions

- A penalty for each detected inversion, as a ratio between the number of inversions and the number of matching words

$$f_{INV}(S, P) = 1.0 - \frac{n_I}{v}$$

Similarity Measures

► Overall similarity

- Weighted average of the three previous measures

- Weights can be determined according to the requirements of the matching effort

$$f(S, P) = w_{MW} f_{MW}(S, P) + w_{VM} f_{VM}(S, P) + w_{INV} f_{INV}(S, P)$$

$$w_{MW} + w_{VM} + w_{INV} = 1$$

Similarity Measures

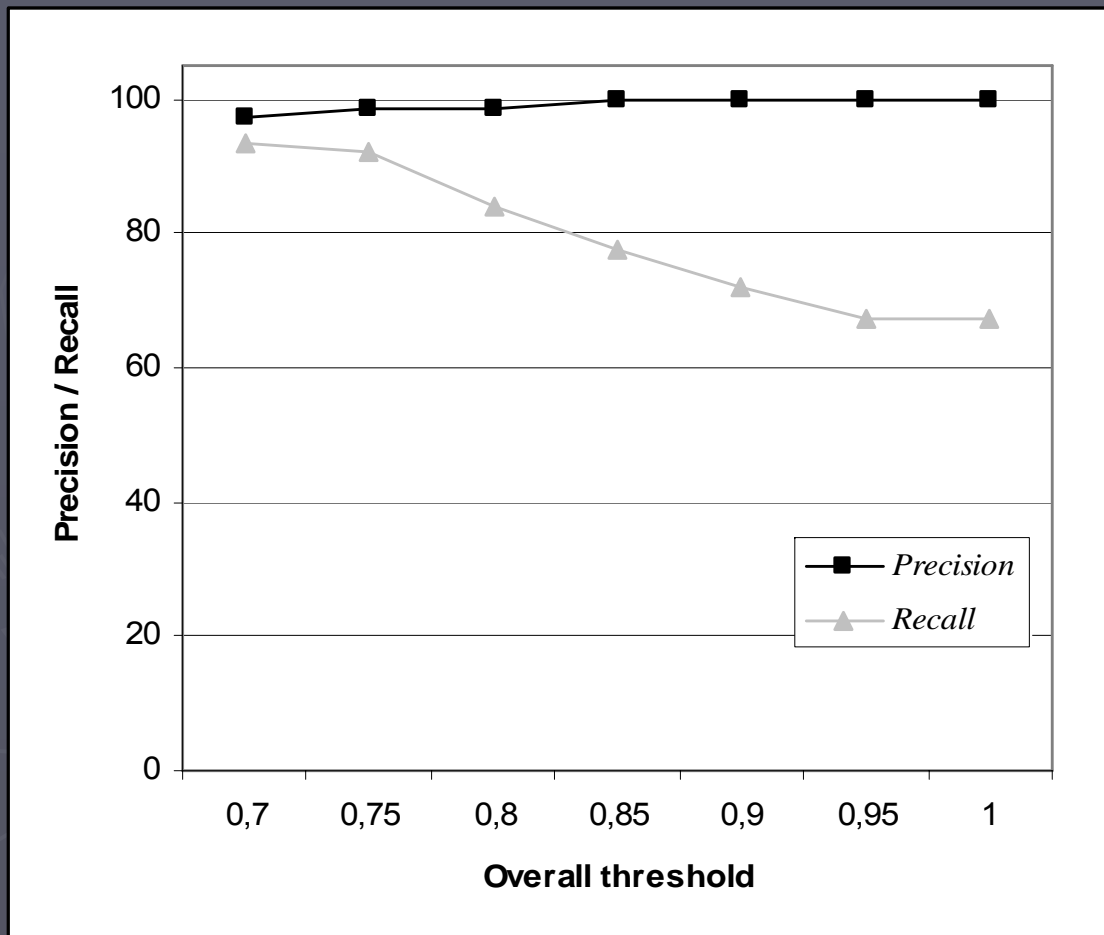
- ▶ Example: “Antônio Carlos de Souza” against three alternatives, considering equal weights

	f_{MW}	f_{VM}	f_{INV}	f
Antonio C. de Sousa	0.706	1.000	1.000	0.902
Antonio Coelho de Sousa	0.664	0.750	1.000	0.805
Sousa, A. C.	0.278	0.750	0.670	0.566

Experimental Results

- ▶ First experiment: personal names
 - Names of researchers found in the results page of a CNPq grant bid have been compared against research paper author names found in the Brazilian Digital Library on Computing (BDBComp)
 - ▶ 85 names to be found in a 9,222 names database
 - ▶ Inversions discarded
 - ▶ Stopwords, accent marks and case-sensitivity enabled
 - ▶ $\delta = 0.85$

First Experiment Results



Precision-recall graphic

$f(S, P)$ threshold	Precision (%)	Recall (%)
0,70	97,3	93,4
0,75	98,6	92,1
0,80	98,5	84,2
0,85	100,0	77,6
0,90	100,0	72,0
0,95	100,0	67,1
1,00	100,0	67,1

Decreasing recall rate shows the drop in efficiency when the process approaches exact matching

Experimental Results

▶ Second experiment

- A hundred street names were compared against Belo Horizonte's official street names catalog
 - ▶ Street names were randomly selected from a list of 4,700 manually typed records as part of a data collection effort
 - ▶ Names on the list included several problems for automatic geocoding, such as abbreviations, omission of parts, and misspellings
 - ▶ All addresses were manually geocoded by Prodabel, in order to provide a frame of reference for the experiment; any other available data were allowed to be used to resolve ambiguities

Second Experiment

- ▶ 87 of the 100 street names were manually matched by Prodabel's technicians
 - The remaining 13 were unrecognizable, definitively ambiguous or from other city
- ▶ Our method used $\delta = 0.85$, case- and accent-mark insensitiveness; stopwords and inversions were considered
 - A 94% precision and a 71% recall rates were obtained
 - If case- and accent-mark sensitivity was used, recall would drop to 35%

Conclusions and Future Work

- ▶ Our experiments have preliminarily showed the validity and the effectiveness of the proposed method
- ▶ The method is flexible, and can be adapted to a number of different approximate string matching needs
- ▶ Further evaluation is required, in order to compare the method with others and to assess its computational efficiency
- ▶ More tests with larger datasets are scheduled

