# Knapp daneben ist nicht immer auch vorbei

## Unscharfe String-Vergleiche mit dem SoundsLike Phonetic Tokenizer

von Mark Boland

Die Anwendung phonetischer Zeichenkettensuche mit der SoundsLike-Phonetic-Tokenizer-DLL anhand einer kleinen und einfachen Städtesuche in einer Datenbank.


Jeder, der schon einmal mit Datenbanken gearbeitet hat, kommt früher oder später an zwei Aufgaben nicht vorbei: Erstens eine schnelle Suche eines bestimmten Datensatzes anhand eines Wertes in einer Datenbank und zweitens ein Bereinigen des Datenbestandes von doppelten Datensätzen. Das wirklich Gemeine an diesen beiden Aufgaben ist, dass der Inhalt der Vergleichsfelder gar nicht übereinstimmt. Es gibt in SQL zwar die Möglichkeit der Wildcard(*)-Suche und VB bietet einen *Like*-Operator. Aber was ist, wenn die Werte wirklich unterschiedlich, aber phonetisch gleich sind (z.B. Meier und Mayer). Oder was, wenn ein Kunde in einem Onlineshop den Film „Forrest Gump" sucht, aber „Forest Gamb" eingibt. Natürlich soll der Shop den Film schnell finden, ansonsten kauft der Kunde woanders.

**kurz & bündig**

**Inhalt**

Ein phonetischer String-Vergleich und die Anwendung der DLL *SoundsLike Phonetic Tokenizer* bei der Suche in einer Datenbank

**Zusammenfassung**

Auch wenn die Bibliothek *Pamphylia SoundsLike Phonetic Tokenizer* nicht kostenlos ist, löst sie ein Problem, das in sehr vielen Anwendungen auftritt und ist daher eine lohnende Anschaffung

**Quellcode**

VB

**Quellcode auf CD**

### Soundex und Levenshtein-Distanz

Dieses Problem ist kein neues Problem und so wurde schon Anfang des letztes Jahrhunderts der *Soundex-Algorithmus* entwickelt und in den 60er Jahren der *Levenshtein-Distanz-Algorithmus*. Beide Algorithmen gehen von Ähnlichkeiten aus. Wobei Soundex auf phonetische Merkmale der Vokale und Konsonanten in der englischen Sprache basiert und der Levenshtein-Algorithmus sich mit der Anzahl der Einzelunterschiede von zwei Zeichenketten beschäftigt. Auf die Algorithmen möchte der Autor nicht weiter eingehen, da diese auch reichlich im Internet vertreten sind – Beschreibungen mit Quellcode finden Sie unter [1] und [2]. Der Soundex in seiner damaligen Form ist für die meisten Anwendungen heutzutage unzureichend. Der Levenshtein-Distanz-Algorithmus verbraucht viel Zeit. Bei jeder Suchanfrage müsste der Algorithmus mit dem Suchwert gegen alle Werte in der Datenbank verglichen werden. Bei großen Datenbeständen wie bei Onlineshops würde das viel zu lange dauern. Es wird also eine Vorauswahl der Daten benötigt, um die Analysemenge gering zu halten.

### Kommerzielle Produkte

Seit dem Internet-Boom und dem Entdecken des Internethandels ist das Thema Zeichenkettensuche wieder sehr aktuell geworden. Mittlerweile gibt es zahlreiche Firmen, die sich hauptsächlich mit der Suche von ähnlichen Zeichenketten in Datenbanken beschäftigen. Große Onlineshops wie *Amazon* haben natürlich eigene Suchmaschinen entwickelt – kleinere Shops kaufen dazu. Da es sich um einen lukrativen Markt handelt, kosten diese Produkte ganz schnell vier- bis fünfstellige Beträge. Für den „Otto-Normal-Entwickler" ist das natürlich viel zu kostspielig. Deshalb möchte der Autor eine recht günstige Entwicklerkomponente vorstellen, die sich vor den Großen nicht zu verstecken braucht, die *Pamphylia SoundsLike Phonetic Tokenizer DLL* [3] – ein zugegeben langer Name für eine kleine DLL.

### Der Pamphylia SoundsLike Phonetic Tokenizer

Die Soundslike-DLL vereinigt die traditionellen String-Analysen wie Soundex oder Levenshtein und bietet dazu noch weitere mächtige Features wie Nickname-Auflösung (z.B. Robert->Bob), eine prozentuale Trefferwahrscheinlichkeit und die Möglichkeit die Daten im Vorfeld phonetisch zu berechnen. Gerade der letzte Punkt ist der Schlüssel zum Erfolg. Mit diesem Feature ist es möglich, große Datenbestände im Vorfeld zu durchlaufen und phonetische Schlüsselwerte in der Datenbank wegzuspeichern. Bei einer Suche kann der in Frage kommende Datenbestand im Vorfeld so weit verkleinert werden, dass die zu analysierende Datenmenge gering bleibt.

Das Beispielprogramm (Abbildung 1) enthält eine rund 16.000 Datensätze umfassende Datenbank mit Städtenamen und benutzt die Kernroutinen von SoundsLike, um die Standardabläufe darzustellen. Das Programm errechnet den Soundex-Wert sowie den normalen und den erweiterten Token. Der Soundex ist hier nur zur Vollständigkeit halber mit aufgeführt, wird aber nicht weiter verwendet. Mit den Token-Zahlen wird eine SQL-Abfrage auf

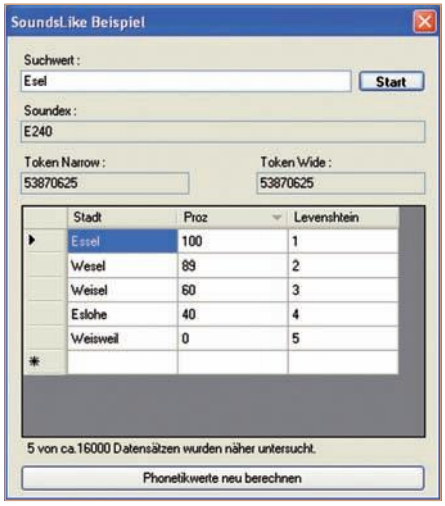


Abb. 1: Das SoundsLike-Beispielprogramm

die Städtedatenbank durchgeführt. Diese wurde vorher einmalig auf die Token-Abfrage vorbereitet. Die Vorberechnungsroutine befindet sich im Programm hinter der Schaltfläche PHONETIKWERTE NEU BERECHNEN. Das Beispielprogramm benutzt die v.1.2 Beta. Um das Beispielprogramm lauffähig zu bekommen, benötigen Sie einen Demo-Lizenz-Schlüssel, den Sie unter [4] erhalten. Dies ist natürlich noch eine kleine Datenbank und nur zur Demonstration gedacht. Wenn größere Produktdatenbestände vorberechnet werden, sollten die Token in einer Extra-Tabelle angelegt werden, sodass man mehrere Token pro Produktdatensatz anlegen kann. Die zu suchenden Werte sollten Wort für Wort berechnet und in die Token-Tabelle geschrieben werden. Das zu Beginn zitierte „Forest-Gamb"-Beispiel hätte demnach zwei Datensätze in der Token-Tabelle.

## Die Funktionen der DLL

Das Vorberechnen der Daten ist recht simpel. Nachdem das *SoundsLike*-Objekt mit dem Lizenzschlüssel instanziiert wurde, gibt es ein paar Methoden um die Daten zu vergleichen bzw. für den Vergleich vorzubereiten. Folgend das Vorberechnen der Daten aus dem Städtebeispiel:

```
For Each row In dt.Rows
  SL.StringToTokenize = row("Stadt")
  row("Narrow") = SL.TokenNarrow
  row("Wide") = SL.TokenWide
Next
```

Erst wird der String übergeben, danach lassen sich die beiden Token-Eigenschaften auslesen, die in der Datenbank gespeichert werden. SoundsLike bietet zwei Token-Werte an, wobei der *TokenNarrow* nähere Ähnlichkeiten darstellt und *TokenWide* ebenfalls bei größeren Differenzen noch übereinstimmen kann. Die berechneten Token-Werte werden auch in der Maske des Beispiels angezeigt. Nachdem nach der Sucheingabe die Token-Werte des Suchbegriffs errechnet wurden, kann mit diesen eine SQL-Abfrage auf die Datenbank gestartet werden. Da es mehrere Zieldatensätze geben kann, müssen die Daten weiter analysiert werden. Dies geschieht hier mit der *ProbabilityOfMatch*-Funktion. Sie liefert eine prozentuale Ähnlichkeit der beiden Zeichenketten zurück. Der Levenshtein-Distanz-Wert wird nur errechnet, um diese Möglichkeit vorzustellen:

```
For Each row In dt.Rows
  row(1) = SL.sup  ProbabilityOfMatch
                   (Me.txtSearchValue.Text, row("Stadt"))
  row(2) = SL.sup  LevenshteinDistance
                   (Me.txtSearchValue.Text, row("Stadt"))
Next
```

Unter der Anzeige der Suchtreffer steht die Anzahl der analysierten Datensätze zum Vergleich zu den 16.000 möglichen Zeichenketten-Vergleichen.

## Kreativität ist gefragt

Die SoundsLike-DLL bietet preisgünstig eine Menge mächtiger Funktionen, um Zeichenketten in großen Datenbanken vorzuberechnen, schnell zu suchen und zu finden. Die Einsatzgebiete sind vielfältig. Ob von der Online-Katalog-Suche oder der Suche nach möglichen Dubletten im Adressbestand. Hier ist wieder die Kreativität des Entwicklers gefragt, wie er die verschiedenen bestehenden Algorithmen kombiniert und auf seine Datenbank anwendet. Auch ein Soundex kann in einer „deutschsprachigen" Datenbank als Zusatzabfrage noch hilfreich sein.

**Mark Boland** arbeitet seit mehreren Jahren als selbstständiger Software-Entwickler auf Basis von .NET, VB6 und VBA.

### Links & Literatur

[1] de.wikipedia.org/wiki/Soundex
[2] de.wikipedia.org/wiki/Levenshtein-Distanz
[3] www.soundsliketokenizer.net
[4] www.soundsliketokenizer.net/SoundsLikeDownload.aspx