Punchy’s Stemming Algorithm

**Introduction:**  
In order to understand how it works, please read this brief article explaining the Porter Stemming algorithm: [Porter Stemming Algorithm – Basic Intro | Vijini Mallawaarachchi](https://vijinimallawaarachchi.com/2017/05/09/porter-stemming-algorithm/)   
The algorithm is broken down into steps 1 – 5 which is reflected in the comments in PorterStemmer.cs under the PunchyAPI.Common, Stemmer folder. If you have trouble understanding what each step is accomplishing, refer back to the article.

**Goal:**  
To obtain the stem or lemma of the word so we can send both the lemma and original keyword to PAPI to query the database. This will allow us to get a wider breath of puns related to the input.  
 Example: “jumping” -> “jump”, “assistant” -> “assist”

**Potential Use in Future:**  
Maybe we could run this through all the keywords stored in the database, so that when the user’s keywords are stemmed, we can match them to the stemmed keywords in SQL.

**How it Works:**The stemmer is broken down into 5 steps to reduce the given keyPhrase (assumed this is one word, no spaces or hyphens) to its stem. Every word can be broken down into the following regular expression where C = consonant and V = vowel: **[C](VC)m[V].**

Examples: ‘tree’ = CCVV (m = 0), ‘trouble’ = CCVVCCV (m = 1), ‘TROUBLES’ = CCVVCCVC (m = 2)

The algorithm uses multiple methods to calculate when to remove certain suffixes. Notably the following:

// make this into table format

* GetMCount(string keyPhrase)
  + In the regular expression, m represents the number of times a vowel is followed by a consonant (VC) within [C] and [V]
  + Returns an integer representing the m count
* IsVowel(char c)
  + Returns true if the provided char is a vowel, ‘y’ is NOT considered as a vowel in this case
* HasVowel(string keyPhrase)
  + Returns true if the provided keyPhrase has a vowel, ‘y’ IS considered a vowel when there is a consonant prior to the y (i.e. ‘syzygy’ has ‘y’ vowels)
* CheckDoubleConsonant(string keyPhrase)
  + Returns true if the last two letters of the keyPhrase are the same
* CheckCVC(string keyPhrase)
  + Returns true if the last three letters are CVC, and the 2nd C is not ‘w’, ‘x’, or ‘y’
* String Extensions:
  + Chop(this string keyPhrase, int numChars)
    - Chops off the given number of characters off the end of keyPhrase
    - 0 < numChars < keyPhrase.Length
  + Chop(this string keyPhrase, string removeString)
    - Chops off the removeString from the end of keyPhrase, does nothing if keyPhrase does not end with the removeString.
  + ReplaceEnd(this string keyPhrase, string removeString, string replaceString)
    - Replaces the removeString located at the end of keyPhrase with replaceString, does nothing if keyPhrase does not end with removeString.

**Simplified Idea of Algorithm:**

Steps:

1. Removes any plural suffixes (i.e. ‘ies’). Removes tense suffixes (i.e. ‘ing’). Replaces words that end with 'y' to 'i'.
   1. If Step 1B is successful (see code comments for Step 1B), additional ‘e’ is concatenated to end.
2. Removes suffixes of keyPhrase.
3. Removes more suffixes of keyPhrase.
4. Removes even more suffixes of keyPhrase.
5. Removes remaining ‘e’ or ‘l’ after suffixes have been removed.

**Where it is Implemented:**As of 8/15/2022, the stemmer is used to get the stemmed words from the keywords received from LUIS of the user’s input. Located in PunchyAPI.sln file, MessageController.cs, both the unstemmed keywords and stemmed words are added to a list to be used to query the database for related puns.