Match Harry Potter Spells With Their Definition Documentation

Release 1.0

James Brill

Dec 15, 2016

CONTENTS:

1 About **1**

2 Current Features **3**

3 Usage Notes **5**

3.1 Harry Potter Spell Generator Functions 5

4 Indices and tables **9**

Python Module Index **11**

CHAPTER ONE

ABOUT

This is a research project which aims to explore computational creativity using recent advancements in natural language processing and machine learning. In order to execute this research, this program will be created and used to match harry potter spells to their definitions. The research will focus on how good two different types of vector representation of words (GloVe and Word2Vec) are at modelling semantic simalarity.

CHAPTER

TWO

CURRENT FEATURES

- Can convert a given sentence into a Harry Potter spell with random language and type.
- Uses Google's Word2Vec to get semantically similar words.
- It is non-determinstic, it will not supply the output every time.

CHAPTER

THREE

USAGE NOTES

- This program uses various python packages such as gensim and translate. To install all the relevant python packages go to the base directory of the project and run the command “pip install -r requirements.txt”.
- This program is designed to run through a command line interface and does not have a GUI.

Harry Potter Spell Generator Functions

`hp_spells.`**`calcProb`**(*data*)

Calculates the probabilities for spells of each type.

Parameters **`data`** (*`[[[str, str], int]..]`*) – List of spell types and origin language with frequency.

Returns A list of type of spells and their associated probabilities.

`hp_spells.`**`checkStoredWords`**(*kwords*, *word*)

This function updates a list of known words with a new word. If the spell type and language exists in the list the value is append by 1 otherwise, it is appended to the end of the list with a value of 1.

Parameters

- **`kwords`** (*`[[[str, str], int]..]`*) – List of spell types and language with associated frequencies.
- **`word`** (*`str`*) – One being the spell type and the other being the origin language.

Returns the updated list of known words.

`hp_spells.`**`contains`**(*string*, *char*)

Checks to see if a string contains a character.

Parameters

- **`string`** (*`str`*) – The string to be checked.
- **`char`** (*`str`*) – The character to be looked for.

Returns Boolean value.

`hp_spells.`**`count_instances`**(*fname*)

Reads supplied file, where it splits it up. Then it appends each word to the data set building a list of words and frequencies using checkStoredWords(kwords, word).

Parameters **`fname`** (*`str`*) – This is the name of the CSV file in which the spell data is stored.

Returns returns a list of languages and the probabilities for each one.

`hp_spells.`**`f`**(*str*)

Returns the first two chacters from the string.

Parameters **`str`** (*`str`*) – A word that is passed.

Returns a string that only contains the first two letters.

`hp_spells.`**`generateScale`**(*data*)

This stacks the probabilities of spells so that each spell has a boundary in which it a spell can be selected over another.

Parameters **`data`** (*`[[[str,str],int,float]..]`*) – list of spell names and their associated frequencies and probabilities.

Returns a list of spells and the value between 0-1 in which that name will be selected.

`hp_spells.`**`generateSpell`**(*sentence*)

Generates a Spell from a sentence.

Parameters **`sentence`** (*`str`*) – string which is the definition of the spell you want to create.

Returns list containing the spell and the spell type.

`hp_spells.`**`getSpellType`**(*scale*, *rndNum*)

Selects a spell according to the random number passed.

Parameters

- **`scale`** (*`[(str,str,float).]`*) – A list of tuples which contains the probability associated with each spell and type.
- **`rndNum`** (*`float`*) – The random number used to select a spell type.

Returns A string which is the spell type.

`hp_spells.`**`langCode`**(*language*)

Converts a language name into a language code for the translator.

Parameters **`language`** (*`tr`*) – Full name of the language, for example latin.

Returns The string code for the language.

`hp_spells.`**`pigLatin`**(*source*)

Takes a source string and converts it from english to pig latin.

Parameters **`source`** (*`str`*) – Takes string of english words and changes it into pig latin.

Returns a string containing pig latin words.

`hp_spells.`**`sentenceToWord`**(*sentence*)

Takes a string and converts it into a vector. Then from that it picks a similar word that doesn't contain an underscore.

Parameters **`sentence`** (*`str`*) – A string which contains a sentence to be converted into one word.

Returns A string containing a similar word.

`hp_spells.`**`totalSpells`**(*data*)

Counts the number of spells in the dataset.

Parameters **`data`** (*`[[[str,str], int]..]`*) – List of spell types and origin language with frequency.

Returns an integer value of total number of spells.

`hp_spells.`**`translate2`**(*word*, *lang*)

Translates a word to a target language.

Parameters

- **`word`** (*`str`*) – The word you want to convert.
- **`lang`** (*`str`*) – the lang code of the language you want to convert to.

Returns a string containing the translated word in the latin alphabet.

CHAPTER FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

hp_spells, 5

INDEX

C

calcProb() (in module hp_spells), 5
checkStoredWords() (in module hp_spells), 5
contains() (in module hp_spells), 5
count_instances() (in module hp_spells), 5

F

f() (in module hp_spells), 6

G

generateScale() (in module hp_spells), 6
generateSpell() (in module hp_spells), 6
getSpellType() (in module hp_spells), 6

H

hp_spells (module), 5

L

langCode() (in module hp_spells), 6

P

pigLatin() (in module hp_spells), 6

S

sentenceToWord() (in module hp_spells), 6

T

totalSpells() (in module hp_spells), 7
translate2() (in module hp_spells), 7