Assignment 2 - Word Embeddings on Harry Potter Books

Submitted by: Shivani Naik

In [1]:
```
# install libraries and upgrade gensim for compatibility
!pip install glove-python-binary
!pip install --upgrade gensim
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages (4.2.0)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-packages (from gensim) (1.4.1)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.7/dist-packages (from gensim) (6.0.0)
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3.7/dist-packages (from gensim) (1.21.6)
```

In [2]:
```
# Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import logging
import nltk
from wordcloud import WordCloud
from nltk.stem import  WordNetLemmatizer
from sklearn.manifold import TSNE
from gensim.scripts.glove2word2vec import glove2word2vec
from gensim.models import KeyedVectors
from gensim.models import *
from glove import Corpus, Glove
from nltk.corpus import stopwords
%matplotlib inline
```

In [3]:
```
# Download nltk packages
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[3]: True

Word Embedding Class

The following class encapsulates all functions.

In [44]:
```
class WordEmbedding():
  def __init__(self):
    '''Reads the data files for all 7 Harry Potter books and appends them'''
    text_file = open("HPBook1.txt", "r")
    self.book1 = text_file.read()
    text_file.close()

    text_file = open("HPBook2.txt", "r")
    self.book2 = text_file.read()
    text_file.close()

    text_file = open("HPBook3.txt", "r")
    self.book3 = text_file.read()
    text_file.close()

    text_file = open("HPBook4.txt", "r")
    self.book4 = text_file.read()
    text_file.close()

    text_file = open("HPBook5.txt", "r")
    self.book5 = text_file.read()
    text_file.close()

    text_file = open("HPBook6.txt", "r")
    self.book6 = text_file.read()
    text_file.close()

    text_file = open("HPBook7.txt", "r")
    self.book7 = text_file.read()
    text_file.close()

    self.all_books = self.book1 + self.book2 + self.book3 + self.book4 + self.book5 + self.book6 + self.book7

  def generate_wordcloud(self, text):
    word_cloud = WordCloud(collocations = False, background_color = 'white').generate(text)
    plt.figure(figsize=(15,8))
    plt.imshow(word_cloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()
  def toLower(self, x):
    '''Converts string to lowercase'''
    return x.lower()

  def sentenceTokenize(self, x):
    '''Tokenizes document into sentences'''
    sent_tokenizer = nltk.data.load("tokenizers/punkt/english.pickle")
    sentences = sent_tokenizer.tokenize(x)
    return sentences

  def preprocess_sentences(self, all_sentences):
    '''Tokenizes sentences into words, removes punctuations, stopwords and
    performs tokenization'''
    word_tokenizer = nltk.RegexpTokenizer(r"\w+")
    sentences = []
    special_characters = re.compile("[^A-Za-z0-9 ]")
    for s in all_sentences:
      # remove punctuation
      s = re.sub(special_characters, " ", s)
      # Word tokenize
      words = word_tokenizer.tokenize(s)
      # Remove Stopwords
      words = self.removeStopwords(words)
      # Perform lemmatization
      words = self.wordnet_lemmatize(words)
      sentences.append(words)
    return sentences

  def removeStopwords(self, sentence):
    '''Removes stopwords from a sentence'''
    stop_words = stopwords.words('english')
    tokens = [token for token in sentence if token not in stop_words]
    return tokens

  def wordnet_lemmatize(self, sentence):
    '''Lemmatizes tokens in a sentence'''
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token, pos='v') for token in sentence]
    return tokens

  def complete_preprocess(self, text):
    '''Performs complete preprocessing on document'''
    #Convert text to lowercase
    text_lower = self.toLower(text)
    #Sentence tokenize the document
    sentences = self.sentenceTokenize(text_lower)
    #Preprocess all sentences
    preprocessed_sentences = self.preprocess_sentences(sentences)
    return preprocessed_sentences

  def word2vec_model(self, sentences,num_feature, min_word_count,
                     window_size, down_sampling,  sg):
    '''Creates and trains Word2Vec model'''
    num_thread = 5
    logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
    model = word2vec.Word2Vec(sentences,
                              #iter = iteration,
                              vector_size=num_feature,
                              min_count = min_word_count,
                              window = window_size,
                              sample = down_sampling,
                              workers=num_thread,
                              sg = sg,
                              epochs = 20)
    return model

  def fasttext_model(self, sentences, num_feature, window_size, min_word_count):
    '''Creates and trains Fasttext model'''
    num_thread = 5
    logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
    model = FastText(sentences,
                              vector_size=num_feature,
                              window=window_size,
                              min_count=min_word_count,
                              workers=num_thread,
                              epochs = 20)
    return model

  def glove_model(self, sentences, window_size, num_features, lr, iterations):
    '''Creates and trains GloVe model'''
    num_thread = 5
    corpus = Corpus()
    # Create word co occurence matrix
    corpus.fit(sentences, window=window_size)
    glove = Glove(no_components=num_features, learning_rate=lr)
    # Fit model
    glove.fit(corpus.matrix, epochs=iterations, no_threads=num_thread)
    glove.add_dictionary(corpus.dictionary)
    return glove

  def top_10_frequent_words(self, model):
    '''Returns top 10 frequent words'''
    # sort model vocab according to top frequent words
    model.sorted_vocab
    top_words = model.wv.index_to_key[:10]
    return top_words

  def top_10_frequent_words_2(self, w2vec_model):
    '''Returns top 10 frequent words'''
    # sort model vocab according to top frequent words
    top_words = w2vec_model.wv.index_to_key[:10]
    return top_words

  def getDictionary(self, model):
    '''Creates dictionary'''
    return model.wv.index_to_key

  def most_similar_words(self, model, words):
    '''Returns most similar words to a list of words'''
    for word in words:
      print("Most similar to ", word,": ", model.wv.most_similar(word))

  def most_similar_words_glove(self, model, words):
    '''Returns most similar words to a list of words for GloVe model'''
    for word in words:
      print("Most similar to ", word,": ", model.most_similar(word))

  def odd_word(self, model, word_list):
    '''Returns odd word from list of words with a provided model'''
    word_list = [x.lower() for x in word_list]
    odd = model.wv.doesnt_match(word_list)
    return odd

  def save_glove_as_gensim(self, model_glove, glove_input_file, word2vec_output_file):
    '''Saves the glove model as a gensim model to use gensim functions'''
    # Create a dataframe as GloVe format
    df = pd.DataFrame(zip(model_glove.dictionary.keys(), model_glove.word_vectors))
    split_df = pd.DataFrame(df[1].to_list())
    df = df.drop(1, axis=1)
    df = pd.concat([df, split_df], axis=1)
    # save embeddings
    df.to_csv('glove_embeddings.txt', header=None, index=None, sep = " ")
    # convert to word2vec file
    glove2word2vec(glove_input_file, word2vec_output_file)
    # load model as gensim model
    glove_model_w2v = KeyedVectors.load_word2vec_format(word2vec_output_file, binary=False)
    return glove_model_w2v

  def odd_word_glove(self, model, word_list):
    '''Returns odd word from list of words with a provided model'''
    word_list = [x.lower() for x in word_list]
    odd = model.doesnt_match(word_list)
    return odd

  def plot_tsne(self, model, num_words = 500):
    '''Plots TSNE for num_words frequent words from vocab'''
    labels = []
    words = []
    # Get tokens and labels
    for word in model.wv.index_to_key:
        words.append(model.wv[word])
        labels.append(word)

    # fit TSNE
    tsne = TSNE(perplexity=5, n_components=2, init='pca', n_iter=5000, random_state=0)
    T = tsne.fit_transform(words)

    x = []
    y = []
    for value in T[:num_words]:
        x.append(value[0])
        y.append(value[1])

    # create scatter plot
    plt.figure(figsize=(16, 16))
    for i in range(len(x)):
        plt.scatter(x[i],y[i])
        plt.annotate(labels[i],
                     xy=(x[i], y[i]),
                     xytext=(5, 2),
                     textcoords='offset points',
                     ha='right',
                     va='bottom')
    plt.show()

  def plot_tsne_glove(self, model, num_words = 500):
    '''Plots TSNE for num_words frequent words from vocab'''
    labels = []
    words = []
    # Get tokens and labels
    for word in model.index_to_key:
        words.append(model[word])
        labels.append(word)

    # fit TSNE
    tsne = TSNE(perplexity=5, n_components=2, init='pca', n_iter=5000, random_state=0)
    T = tsne.fit_transform(words)

    x = []
    y = []
    for value in T[:num_words]:
        x.append(value[0])
        y.append(value[1])

    # create scatter plot
    plt.figure(figsize=(16, 16))
    for i in range(len(x)):
        plt.scatter(x[i],y[i])
        plt.annotate(labels[i],
                     xy=(x[i], y[i]),
                     xytext=(5, 2),
                     textcoords='offset points',
                     ha='right',
                     va='bottom')
    plt.show()
```

In [45]:
```
# Instantiate WordEmbedding class
we = WordEmbedding()
```

Preprocessing and EDA

I have concatenated all 7 Harry Potter books. Let's perform preprocessing on these books.

In [6]:
```
book_sentences = we.complete_preprocess(we.all_books)
```

In [7]:
```
print("Number of sentences: ",len(book_sentences))
```

```
Number of sentences:  70367
```

In [8]:
```
# Sample tokenized and preprocessed sentences
for i in book_sentences[:10]:
    print("{}\n".format(i))
```

```
['text', 'chapter', 'book', 'boy', 'live', 'mr', 'mrs', 'dursley', 'number', 'four', 'privet', 'drive', 'proud', 'say', 'perfectly', 'normal', 'thank', 'much']

['last', 'people', 'expect', 'involve', 'anything', 'strange', 'mysterious', 'hold', 'nonsense']

['mr', 'dursley', 'director', 'firm', 'call', 'grunnings', 'make', 'drill']

['big', 'beefy', 'man', 'hardly', 'neck', 'although', 'large', 'mustache']

['mrs', 'dursley', 'thin', 'blonde', 'nearly', 'twice', 'usual', 'amount', 'neck', 'come', 'useful', 'spend', 'much', 'time', 'crane', 'garden', 'fence', 'spy', 'neighbor']

['dursleys', 'small', 'son', 'call', 'dudley', 'opinion', 'finer', 'boy', 'anywhere']

['dursleys', 'everything', 'want', 'also', 'secret', 'greatest', 'fear', 'somebody', 'would', 'discover']

['think', 'could', 'bear', 'anyone', 'find', 'potter']

['mrs', 'potter', 'mrs', 'dursley', 'sister', 'meet', 'several', 'years', 'fact', 'mrs', 'dursley', 'pretend', 'sister', 'sister', 'good', 'nothing', 'husband', 'undursleyish', 'possible']

['dursleys', 'shudder', 'think', 'neighbor', 'would', 'say', 'potter', 'arrive', 'street']
```

Wordcloud

Word cloud shows most frequent words are Harry, Hermione, Rone, Said, Dumbledore.

In [9]:
```
we.generate_wordcloud(we.all_books)
```

Word2vec

Let us create and train a Word2Vec embedding model on the preprocessed data. I have experimented with various parameters of the model:

- vector_size
- window_size
- min_word_count
- epochs
- down_sampling

Observations:

- vector_size 100 gives good results, they don't improve significantly after increasing it to 256

In [10]:
```
# Word2Vec paramters
num_feature = 100
min_word_count = 5
window_size = 10
down_sampling = 0.001
iteration = 20
sg=1

# Create Word2Vec model
w2vec_model = we.word2vec_model(book_sentences,  num_feature, min_word_count,
                  window_size, down_sampling, sg)
```

```
2022-06-05 17:00:28,366 : INFO : collecting all words and their counts
2022-06-05 17:00:28,378 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2022-06-05 17:00:28,448 : INFO : PROGRESS: at sentence #10000, processed 84155 words, keeping 6437 word types
2022-06-05 17:00:28,478 : INFO : PROGRESS: at sentence #20000, processed 170625 words, keeping 9109 word types
2022-06-05 17:00:28,518 : INFO : PROGRESS: at sentence #30000, processed 240148 words, keeping 10603 word types
2022-06-05 17:00:28,550 : INFO : PROGRESS: at sentence #40000, processed 302503 words, keeping 12003 word types
2022-06-05 17:00:28,580 : INFO : PROGRESS: at sentence #50000, processed 361226 words, keeping 13077 word types
2022-06-05 17:00:28,615 : INFO : PROGRESS: at sentence #60000, processed 446600 words, keeping 14766 word types
2022-06-05 17:00:28,657 : INFO : PROGRESS: at sentence #70000, processed 569522 words, keeping 16492 word types
2022-06-05 17:00:28,662 : INFO : collected 16548 word types from a corpus of 574796 raw words and 70367 sentences
2022-06-05 17:00:28,666 : INFO : Creating a fresh vocabulary
2022-06-05 17:00:28,717 : INFO : Word2Vec lifecycle event {'msg': 'effective_min_count=5 retains 6691 unique words (40.43% of original 16548, drops 9857)', 'datetime': '2022-06-05T17:00:28.717845', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'prepare_vocab'}
2022-06-05 17:00:28,720 : INFO : Word2Vec lifecycle event {'msg': 'effective_min_count=5 leaves 557991 word corpus (97.08% of original 574796, drops 16805)', 'datetime': '2022-06-05T17:00:28.720545', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'prepare_vocab'}
2022-06-05 17:00:28,779 : INFO : deleting the raw counts dictionary of 16548 items
2022-06-05 17:00:28,782 : INFO : sample=0.001 downsamples 43 most-common words
2022-06-05 17:00:28,788 : INFO : Word2Vec lifecycle event {'msg': 'downsampling leaves estimated 498727.55312674557 word corpus (89.4%% of prior 557991)', 'datetime': '2022-06-05T17:00:28.788900', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'prepare_vocab'}
2022-06-05 17:00:28,885 : INFO : estimated required memory for 6691 words and 100 dimensions: 8698300 bytes
2022-06-05 17:00:28,889 : INFO : resetting layer weights
2022-06-05 17:00:28,901 : INFO : Word2Vec lifecycle event {'update': False, 'trim_rule': 'None', 'datetime': '2022-06-05T17:00:28.901898', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'build_vocab'}
2022-06-05 17:00:28,903 : INFO : Word2Vec lifecycle event {'msg': 'training model with 5 workers on 6691 vocabulary and 100 features, using sg=1 hs=0 sample=0.001 negative=5 window=10 shrink_windows=True', 'datetime': '2022-06-05T17:00:28.903866', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'train'}
2022-06-05 17:00:30,107 : INFO : EPOCH 0 - PROGRESS: at 26.36% examples, 120989 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:31,131 : INFO : EPOCH 0 - PROGRESS: at 65.96% examples, 135616 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:32,256 : INFO : EPOCH 0 - PROGRESS: at 91.37% examples, 128671 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:32,654 : INFO : EPOCH 0: training on 574796 raw words (498900 effective words) took 3.7s, 134875 effective words/s
2022-06-05 17:00:33,669 : INFO : EPOCH 1 - PROGRESS: at 24.87% examples, 130366 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:34,740 : INFO : EPOCH 1 - PROGRESS: at 63.67% examples, 138736 words/s, in_qsize 9, out_qsize 2
2022-06-05 17:00:35,837 : INFO : EPOCH 1 - PROGRESS: at 92.36% examples, 136650 words/s, in_qsize 7, out_qsize 1
2022-06-05 17:00:36,184 : INFO : EPOCH 1: training on 574796 raw words (498662 effective words) took 3.5s, 141824 effective words/s
2022-06-05 17:00:37,208 : INFO : EPOCH 2 - PROGRESS: at 24.87% examples, 129668 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:00:38,299 : INFO : EPOCH 2 - PROGRESS: at 63.70% examples, 136594 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:39,339 : INFO : EPOCH 2 - PROGRESS: at 91.43% examples, 135286 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:00:39,747 : INFO : EPOCH 2: training on 574796 raw words (498839 effective words) took 3.5s, 140663 effective words/s
2022-06-05 17:00:40,849 : INFO : EPOCH 3 - PROGRESS: at 26.36% examples, 128351 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:41,912 : INFO : EPOCH 3 - PROGRESS: at 65.96% examples, 137363 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:42,939 : INFO : EPOCH 3 - PROGRESS: at 92.32% examples, 136447 words/s, in_qsize 7, out_qsize 1
2022-06-05 17:00:43,242 : INFO : EPOCH 3: training on 574796 raw words (498802 effective words) took 3.5s, 143398 effective words/s
2022-06-05 17:00:44,298 : INFO : EPOCH 4 - PROGRESS: at 26.36% examples, 133370 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:45,338 : INFO : EPOCH 4 - PROGRESS: at 65.96% examples, 141525 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:46,369 : INFO : EPOCH 4 - PROGRESS: at 92.36% examples, 138985 words/s, in_qsize 8, out_qsize 0
2022-06-05 17:00:46,706 : INFO : EPOCH 4: training on 574796 raw words (498813 effective words) took 3.5s, 144450 effective words/s
2022-06-05 17:00:47,908 : INFO : EPOCH 5 - PROGRESS: at 26.36% examples, 118486 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:00:48,980 : INFO : EPOCH 5 - PROGRESS: at 70.62% examples, 138905 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:00:50,069 : INFO : EPOCH 5 - PROGRESS: at 96.40% examples, 140129 words/s, in_qsize 4, out_qsize 1
2022-06-05 17:00:50,196 : INFO : EPOCH 5: training on 574796 raw words (498478 effective words) took 3.5s, 143921 effective words/s
2022-06-05 17:00:51,265 : INFO : EPOCH 6 - PROGRESS: at 26.36% examples, 132997 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:52,305 : INFO : EPOCH 6 - PROGRESS: at 68.31% examples, 145188 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:53,456 : INFO : EPOCH 6 - PROGRESS: at 95.30% examples, 141528 words/s, in_qsize 5, out_qsize 0
2022-06-05 17:00:53,626 : INFO : EPOCH 6: training on 574796 raw words (498821 effective words) took 3.4s, 146100 effective words/s
2022-06-05 17:00:54,673 : INFO : EPOCH 7 - PROGRESS: at 26.36% examples, 135374 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:55,713 : INFO : EPOCH 7 - PROGRESS: at 63.64% examples, 138356 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:56,826 : INFO : EPOCH 7 - PROGRESS: at 93.25% examples, 138844 words/s, in_qsize 7, out_qsize 0
2022-06-05 17:00:57,132 : INFO : EPOCH 7: training on 574796 raw words (498710 effective words) took 3.5s, 142969 effective words/s
2022-06-05 17:00:58,218 : INFO : EPOCH 8 - PROGRESS: at 26.80% examples, 130206 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:00:59,274 : INFO : EPOCH 8 - PROGRESS: at 65.96% examples, 138876 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:00,277 : INFO : EPOCH 8 - PROGRESS: at 92.39% examples, 138531 words/s, in_qsize 8, out_qsize 0
2022-06-05 17:01:00,592 : INFO : EPOCH 8: training on 574796 raw words (498972 effective words) took 3.4s, 144867 effective words/s
2022-06-05 17:01:01,630 : INFO : EPOCH 9 - PROGRESS: at 24.87% examples, 127544 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:01:02,715 : INFO : EPOCH 9 - PROGRESS: at 63.67% examples, 135812 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:01:03,742 : INFO : EPOCH 9 - PROGRESS: at 92.36% examples, 138099 words/s, in_qsize 8, out_qsize 0
2022-06-05 17:01:04,109 : INFO : EPOCH 9: training on 574796 raw words (498562 effective words) took 3.5s, 142353 effective words/s
2022-06-05 17:01:05,260 : INFO : EPOCH 10 - PROGRESS: at 26.36% examples, 124549 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:06,327 : INFO : EPOCH 10 - PROGRESS: at 70.62% examples, 142485 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:01:07,354 : INFO : EPOCH 10 - PROGRESS: at 94.21% examples, 139921 words/s, in_qsize 6, out_qsize 0
2022-06-05 17:01:07,632 : INFO : EPOCH 10: training on 574796 raw words (498562 effective words) took 3.5s, 142541 effective words/s
2022-06-05 17:01:08,816 : INFO : EPOCH 11 - PROGRESS: at 26.74% examples, 119234 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:09,863 : INFO : EPOCH 11 - PROGRESS: at 70.73% examples, 141141 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:10,867 : INFO : EPOCH 11 - PROGRESS: at 94.21% examples, 139974 words/s, in_qsize 6, out_qsize 0
2022-06-05 17:01:11,111 : INFO : EPOCH 11: training on 574796 raw words (498780 effective words) took 3.5s, 144043 effective words/s
2022-06-05 17:01:12,239 : INFO : EPOCH 12 - PROGRESS: at 26.36% examples, 127410 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:01:13,241 : INFO : EPOCH 12 - PROGRESS: at 68.31% examples, 144986 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:14,296 : INFO : EPOCH 12 - PROGRESS: at 94.21% examples, 142995 words/s, in_qsize 6, out_qsize 0
2022-06-05 17:01:14,544 : INFO : EPOCH 12: training on 574796 raw words (498745 effective words) took 3.4s, 146735 effective words/s
2022-06-05 17:01:15,606 : INFO : EPOCH 13 - PROGRESS: at 25.17% examples, 124788 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:01:16,697 : INFO : EPOCH 13 - PROGRESS: at 68.31% examples, 142218 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:17,784 : INFO : EPOCH 13 - PROGRESS: at 93.32% examples, 137032 words/s, in_qsize 6, out_qsize 1
2022-06-05 17:01:18,061 : INFO : EPOCH 13: training on 574796 raw words (498759 effective words) took 3.5s, 142460 effective words/s
2022-06-05 17:01:19,103 : INFO : EPOCH 14 - PROGRESS: at 24.87% examples, 128455 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:01:20,228 : INFO : EPOCH 14 - PROGRESS: at 63.64% examples, 133742 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:01:21,295 : INFO : EPOCH 14 - PROGRESS: at 93.32% examples, 137716 words/s, in_qsize 7, out_qsize 0
2022-06-05 17:01:21,577 : INFO : EPOCH 14: training on 574796 raw words (498724 effective words) took 3.5s, 142862 effective words/s
2022-06-05 17:01:22,706 : INFO : EPOCH 15 - PROGRESS: at 26.36% examples, 125113 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:23,714 : INFO : EPOCH 15 - PROGRESS: at 70.62% examples, 147322 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:01:24,717 : INFO : EPOCH 15 - PROGRESS: at 93.25% examples, 141469 words/s, in_qsize 7, out_qsize 0
2022-06-05 17:01:25,049 : INFO : EPOCH 15: training on 574796 raw words (498898 effective words) took 3.5s, 144316 effective words/s
2022-06-05 17:01:26,148 : INFO : EPOCH 16 - PROGRESS: at 26.36% examples, 128718 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:01:27,165 : INFO : EPOCH 16 - PROGRESS: at 65.96% examples, 140463 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:28,176 : INFO : EPOCH 16 - PROGRESS: at 94.21% examples, 144763 words/s, in_qsize 6, out_qsize 0
2022-06-05 17:01:28,493 : INFO : EPOCH 16: training on 574796 raw words (498501 effective words) took 3.4s, 145464 effective words/s
2022-06-05 17:01:29,532 : INFO : EPOCH 17 - PROGRESS: at 26.36% examples, 135885 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:01:30,546 : INFO : EPOCH 17 - PROGRESS: at 63.64% examples, 140502 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:31,618 : INFO : EPOCH 17 - PROGRESS: at 92.36% examples, 139240 words/s, in_qsize 8, out_qsize 0
2022-06-05 17:01:31,969 : INFO : EPOCH 17: training on 574796 raw words (498713 effective words) took 3.5s, 144041 effective words/s
2022-06-05 17:01:33,156 : INFO : EPOCH 18 - PROGRESS: at 26.80% examples, 118888 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:34,174 : INFO : EPOCH 18 - PROGRESS: at 68.26% examples, 139438 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:35,184 : INFO : EPOCH 18 - PROGRESS: at 92.36% examples, 135403 words/s, in_qsize 8, out_qsize 0
2022-06-05 17:01:35,569 : INFO : EPOCH 18: training on 574796 raw words (498884 effective words) took 3.6s, 139134 effective words/s
2022-06-05 17:01:36,607 : INFO : EPOCH 19 - PROGRESS: at 24.87% examples, 127781 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:37,695 : INFO : EPOCH 19 - PROGRESS: at 51.55% examples, 115349 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:38,769 : INFO : EPOCH 19 - PROGRESS: at 75.29% examples, 103603 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:39,838 : INFO : EPOCH 19 - PROGRESS: at 95.30% examples, 107964 words/s, in_qsize 5, out_qsize 0
2022-06-05 17:01:39,935 : INFO : EPOCH 19: training on 574796 raw words (498952 effective words) took 4.4s, 114681 effective words/s
2022-06-05 17:01:39,938 : INFO : Word2Vec lifecycle event {'msg': 'training on 11495920 raw words (9975077 effective words) took 71.0s, 140442 effective words/s', 'datetime': '2022-06-05T17:01:39.938439', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'train'}
2022-06-05 17:01:39,940 : INFO : Word2Vec lifecycle event {'params': 'Word2Vec<vocab=6691, vector_size=100, alpha=0.025>', 'datetime': '2022-06-05T17:01:39.940262', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'created'}
```

In [11]:
```
print("Total number of words: {}".format(len(w2vec_model.wv.key_to_index)))
```

```
Total number of words: 6691
```

Get Dictionary

In [43]:
```
# Words are ordered according to decreasing frequency.
#Top frequency words appear at the top
we.getDictionary(w2vec_model)[:20]
```

Out[43]:
```
['harry',
 'say',
 'ron',
 'look',
 'hermione',
 'get',
 'go',
 'know',
 'think',
 'back',
 'dumbledore',
 'could',
 'see',
 'one',
 'like',
 'come',
 'tell',
 'would',
 'around',
 'eye']
```

Top 10 words with Word2Vec

We order the vocab by most frequent words using sorted_vocab attribute of Word2Vec and retrieve the top 10. We get words like Harry, Ron, Hermione in the top words.

In [13]:
```
top_words = we.top_10_frequent_words(w2vec_model)
top_words
```

Out[13]:
```
['harry',
 'say',
 'ron',
 'look',
 'hermione',
 'get',
 'go',
 'know',
 'think',
 'back']
```

Most similar words to top 10 words with Word2Vec

We see words like Hermione and Ron are most similar to each other.

In [14]:
```
we.most_similar_words(w2vec_model, top_words)
```

```
Most similar to  harry :  [('ron', 0.7265907526016235), ('silencio', 0.6914387345314026), ('hermione', 0.6812456250190735), ('sceptically', 0.6685629487037659), ('tremulously', 0.6664249897003174), ('options', 0.658836305141449), ('goodnight', 0.6429656147956848), ('tensely', 0.638627290725708), ('exasperatedly', 0.6361172795295715), ('back', 0.6331034898757935)]
Most similar to  say :  [('ask', 0.7165927886962891), ('well', 0.7138505578041077), ('scoff', 0.6681712865829468), ('oh', 0.6663616299629211), ('know', 0.6596862077713013), ('hmmm', 0.6513105630874634), ('yes', 0.6503114104270935), ('harry', 0.6323049664497375), ('think', 0.6315926313400269), ('yeah', 0.6279149055480957)]
Most similar to  ron :  [('hermione', 0.8546380996704102), ('harry', 0.7265907526016235), ('goodnight', 0.6727980375289917), ('scandalize', 0.633036732673645), ('ginny', 0.6240652203559875), ('aaah', 0.6045141220092773), ('george', 0.5981253385543823), ('thunderstruck', 0.5979795455932617), ('infuriate', 0.5922998785972595), ('fred', 0.588123083114624)]
Most similar to  look :  [('nonplus', 0.6068686246871948), ('remotely', 0.5933437943458557), ('star', 0.5910016894340515), ('glance', 0.5866807699203491), ('expression', 0.581293523311615), ('puzzle', 0.5663690567016602), ('glum', 0.5637786984443665), ('malevolent', 0.5631609559059143), ('rather', 0.558380663394928), ('exasperation', 0.558159351348877)]
Most similar to  hermione :  [('ron', 0.8546380400657654), ('harry', 0.6812455058097839), ('say', 0.5993678569793701), ('aggressively', 0.5951573848724365), ('doubtfully', 0.5911186933517456), ('uh', 0.5743845105171204), ('alas', 0.5723267197608948), ('vector', 0.5716705918312073), ('plea', 0.5686533451080322), ('scoff', 0.5628979802131653)]
Most similar to  get :  [('go', 0.7277942895889282), ('talkin', 0.6312040090560913), ('know', 0.6294001340866089), ('think', 0.6260281205177307), ('scoff', 0.6253399848937988), ('probably', 0.6235625743865967), ('migh', 0.623383641242981), ('well', 0.6160168647766113), ('want', 0.6120638251304626), ('myst', 0.6111989617347717)]
Most similar to  go :  [('come', 0.7431937456130981), ('get', 0.727794349193573), ('think', 0.6481966376304626), ('evasively', 0.6255102157592773), ('kip', 0.6227594614028931), ('silencio', 0.6163314580917358), ('quicker', 0.6113894581794739), ('tell', 0.610897421836853), ('know', 0.6103991270065308), ('cop', 0.6097822785377502)]
Most similar to  know :  [('tell', 0.7449623942375183), ('scrim', 0.731433629989624), ('well', 0.7224025726318359), ('mean', 0.7189705967903137), ('think', 0.714867889881134), ('scoff', 0.7121427059173584), ('cop', 0.7114754915237427), ('nutters', 0.6841086149215698), ('genius', 0.6833103895187378), ('justify', 0.6719285845756531)]
Most similar to  think :  [('know', 0.7148679494857788), ('want', 0.6714436411857605), ('scrim', 0.6690962314605713), ('scoff', 0.6671172976493835), ('tell', 0.657023549079895), ('well', 0.6550671458244324), ('go', 0.6481966376304626), ('hesitantly', 0.646798312664032), ('risky', 0.6342747807502747), ('really', 0.6338464021682739)]
Most similar to  back :  [('harry', 0.6331034898757935), ('toward', 0.5963829755783081), ('scruff', 0.5942357182502747), ('kip', 0.5903924107551575), ('straight', 0.5812329649925232), ('onwards', 0.5755997896194458), ('ron', 0.5593794584274292), ('goodnight', 0.5557909607887268), ('aargh', 0.5522965788841248), ('immobilize', 0.548739492893219)]
```

To test the model further, I have checked the similar words to some magic spells from Harry Potter. We see, for *Avada (Avada Kedavra* spell), most similar is *Kedavra*. Also, for *Hogwarts*, most similar is *School*.

In [15]:
```
print(w2vec_model.wv.most_similar("avada"))
print(w2vec_model.wv.most_similar("expecto"))
print(w2vec_model.wv.most_similar("lumos"))
print(w2vec_model.wv.most_similar("dumbledore"))
print(w2vec_model.wv.most_similar("hogwarts"))
```

```
[('kedavra', 0.9569318890571594), ('expelliarmus', 0.6656011343002319), ('unforgivable', 0.6196795701980591), ('crucio', 0.6014413237571716), ('impedimenta', 0.5960601568222046), ('rebound', 0.578210175037384), ('waitress', 0.5771713256835938), ('whine', 0.5730738639831543), ('stupefy', 0.5638341903686523), ('jet', 0.5617681741714478)]
[('patronum', 0.9717077612876892), ('otter', 0.6616460084915161), ('wisp', 0.605968177318573), ('fog', 0.6028965711593628), ('petrificus', 0.5945016145706177), ('protego', 0.5840536952018738), ('totalus', 0.5772987604141235), ('stag', 0.5599668622016907), ('clammy', 0.5473964810371399), ('aargh', 0.5438020825386047)]
[('ignite', 0.7599468231201172), ('wand', 0.6119696497917175), ('reparo', 0.5890833193419556), ('light', 0.5837756395339966), ('vapour', 0.5604777336120605), ('imperio', 0.5588639378547668), ('arch', 0.5546481013298035), ('tip', 0.5527899861335754), ('alohomora', 0.5438332557678223), ('skyward', 0.5427126288414001)]
[('headmaster', 0.6794047951698303), ('courteously', 0.6230568885803223), ('dippet', 0.6062681078910828), ('eccentric', 0.5933166146278381), ('sibyll', 0.58150315284729), ('severus', 0.5798885822296143), ('apprehend', 0.5795800089836121), ('dryly', 0.5770117044448853), ('behaviour', 0.5727391839027405), ('impertinent', 0.5712749361991882)]
[('school', 0.7524427175521851), ('witchcraft', 0.6838997602462769), ('wizardry', 0.6371129155158997), ('educate', 0.6184692978858948), ('express', 0.6029987931251526), ('participate', 0.5891844034194946), ('host', 0.583256721496582), ('expulsion', 0.5609077215194702), ('cop', 0.5506772398948669), ('educational', 0.548794686794281)]
```

Odd word using Word2Vec

In [16]:
```
word_list = ["Harry","Hermione", "train"] #walle
odd_word = we.odd_word(w2vec_model, word_list)
print("The odd word is ", odd_word)
```

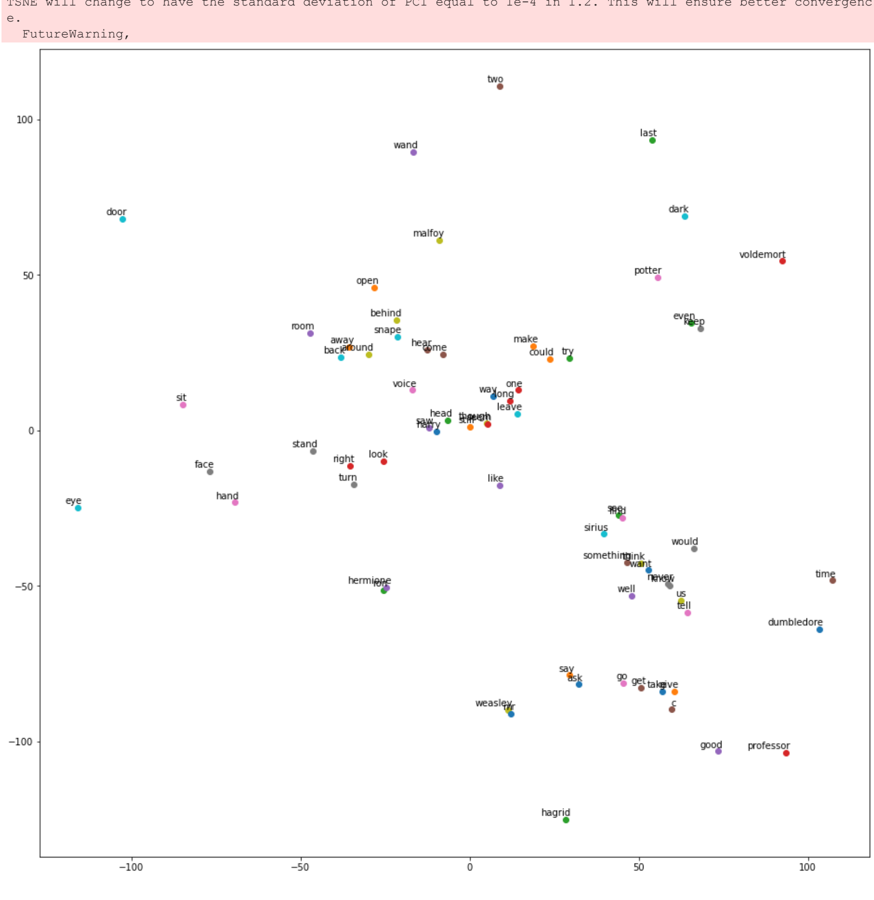
```
The odd word is  train
```

Visualising Word2Vec Embeddings with TSNE

To avoid overly crowded plot, let us plot top word embedding using TSNE. We see words like hermione ron are near , mr and weasley are together, dark, voldemort, potter are close.

In [31]:
```
we.plot_tsne(w2vec_model, num_words = 70)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: FutureWarning: The default learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:986: FutureWarning: The PCA initialization in TSNE will change to have the standard deviation of PC1 equal to 1e-4 in 1.2. This will ensure better convergence.
  FutureWarning,
```



Fasttext

Let us create and train a FastText embedding model on the preprocessed data. I have experimented with various parameters of the model:

- vector_size
- window_size
- min_word_count
- epochs
- down_sampling

Observations:

In [17]:
```
# Fasttext parameters
num_feature = 100
min_word_count = 5
window_size = 10
down_sampling = 0.001
iteration = 20

# Create a fastext model
model_fastText = we.fasttext_model(book_sentences, num_feature, window_size,
                                   min_word_count)
```

```
2022-06-05 17:01:40,104 : INFO : collecting all words and their counts
2022-06-05 17:01:40,106 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2022-06-05 17:01:40,187 : INFO : PROGRESS: at sentence #10000, processed 84155 words, keeping 6437 word types
2022-06-05 17:01:40,213 : INFO : PROGRESS: at sentence #20000, processed 170625 words, keeping 9109 word types
2022-06-05 17:01:40,238 : INFO : PROGRESS: at sentence #30000, processed 240148 words, keeping 10603 word types
2022-06-05 17:01:40,261 : INFO : PROGRESS: at sentence #40000, processed 302503 words, keeping 12003 word types
2022-06-05 17:01:40,285 : INFO : PROGRESS: at sentence #50000, processed 361226 words, keeping 13077 word types
2022-06-05 17:01:40,321 : INFO : PROGRESS: at sentence #60000, processed 446600 words, keeping 14766 word types
2022-06-05 17:01:40,357 : INFO : PROGRESS: at sentence #70000, processed 569522 words, keeping 16492 word types
2022-06-05 17:01:40,362 : INFO : collected 16548 word types from a corpus of 574796 raw words and 70367 sentences
2022-06-05 17:01:40,364 : INFO : Creating a fresh vocabulary
2022-06-05 17:01:40,413 : INFO : FastText lifecycle event {'msg': 'effective_min_count=5 retains 6691 unique words (40.43% of original 16548, drops 9857)', 'datetime': '2022-06-05T17:01:40.412913', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'prepare_vocab'}
2022-06-05 17:01:40,417 : INFO : FastText lifecycle event {'msg': 'effective_min_count=5 leaves 557991 word corpus (97.08% of original 574796, drops 16805)', 'datetime': '2022-06-05T17:01:40.417860', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'prepare_vocab'}
2022-06-05 17:01:40,466 : INFO : deleting the raw counts dictionary of 16548 items
2022-06-05 17:01:40,469 : INFO : sample=0.001 downsamples 43 most-common words
2022-06-05 17:01:40,473 : INFO : FastText lifecycle event {'msg': 'downsampling leaves estimated 498727.55312674557 word corpus (89.4%% of prior 557991)', 'datetime': '2022-06-05T17:01:40.473817', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'prepare_vocab'}
2022-06-05 17:01:40,628 : INFO : estimated required memory for 6691 words, 2000000 buckets and 100 dimensions: 809905252 bytes
2022-06-05 17:01:40,630 : INFO : resetting layer weights
2022-06-05 17:01:42,323 : INFO : FastText lifecycle event {'update': False, 'trim_rule': 'None', 'datetime': '2022-06-05T17:01:42.323307', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'build_vocab'}
2022-06-05 17:01:42,326 : INFO : FastText lifecycle event {'msg': 'training model with 5 workers on 6691 vocabulary and 100 features, using sg=0 hs=0 sample=0.001 negative=5 window=10 shrink_windows=True', 'datetime': '2022-06-05T17:01:42.326127', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'train'}
2022-06-05 17:01:43,409 : INFO : EPOCH 0 - PROGRESS: at 18.31% examples, 89700 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:01:44,496 : INFO : EPOCH 0 - PROGRESS: at 49.44% examples, 109048 words/s, in_qsize 9, out_qsize 1
2022-06-05 17:01:45,549 : INFO : EPOCH 0 - PROGRESS: at 82.64% examples, 116229 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:46,558 : INFO : EPOCH 0 - PROGRESS: at 100.00% examples, 118229 words/s, in_qsize 0, out_qsize 1
2022-06-05 17:01:46,560 : INFO : EPOCH 0: training on 574796 raw words (498525 effective words) took 4.2s, 118174 effective words/s
2022-06-05 17:01:47,675 : INFO : EPOCH 1 - PROGRESS: at 20.05% examples, 95095 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:01:48,751 : INFO : EPOCH 1 - PROGRESS: at 49.44% examples, 107882 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:49,884 : INFO : EPOCH 1 - PROGRESS: at 82.64% examples, 112714 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:50,844 : INFO : EPOCH 1: training on 574796 raw words (498936 effective words) took 4.3s, 116861 effective words/s
2022-06-05 17:01:51,936 : INFO : EPOCH 2 - PROGRESS: at 19.90% examples, 97864 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:52,967 : INFO : EPOCH 2 - PROGRESS: at 49.24% examples, 111332 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:54,113 : INFO : EPOCH 2 - PROGRESS: at 82.64% examples, 114579 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:55,039 : INFO : EPOCH 2: training on 574796 raw words (499040 effective words) took 4.2s, 119307 effective words/s
2022-06-05 17:01:56,098 : INFO : EPOCH 3 - PROGRESS: at 19.90% examples, 99854 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:57,172 : INFO : EPOCH 3 - PROGRESS: at 51.74% examples, 114839 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:58,192 : INFO : EPOCH 3 - PROGRESS: at 82.64% examples, 118782 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:01:59,218 : INFO : EPOCH 3 - PROGRESS: at 100.00% examples, 119698 words/s, in_qsize 0, out_qsize 1
2022-06-05 17:01:59,221 : INFO : EPOCH 3: training on 574796 raw words (498519 effective words) took 4.2s, 119616 effective words/s
2022-06-05 17:02:00,257 : INFO : EPOCH 4 - PROGRESS: at 20.05% examples, 102459 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:01,292 : INFO : EPOCH 4 - PROGRESS: at 51.74% examples, 118485 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:02,333 : INFO : EPOCH 4 - PROGRESS: at 81.34% examples, 117658 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:02:03,341 : INFO : EPOCH 4 - PROGRESS: at 96.90% examples, 115093 words/s, in_qsize 3, out_qsize 1
2022-06-05 17:02:03,409 : INFO : EPOCH 4: training on 574796 raw words (498747 effective words) took 4.2s, 119543 effective words/s
2022-06-05 17:02:04,455 : INFO : EPOCH 5 - PROGRESS: at 19.92% examples, 101415 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:05,595 : INFO : EPOCH 5 - PROGRESS: at 51.55% examples, 112115 words/s, in_qsize 10, out_qsize 1
2022-06-05 17:02:06,638 : INFO : EPOCH 5 - PROGRESS: at 82.64% examples, 115958 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:07,578 : INFO : EPOCH 5: training on 574796 raw words (498638 effective words) took 4.2s, 120048 effective words/s
2022-06-05 17:02:08,624 : INFO : EPOCH 6 - PROGRESS: at 19.90% examples, 100791 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:09,695 : INFO : EPOCH 6 - PROGRESS: at 49.24% examples, 111495 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:10,698 : INFO : EPOCH 6 - PROGRESS: at 79.45% examples, 114439 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:11,746 : INFO : EPOCH 6 - PROGRESS: at 96.40% examples, 112465 words/s, in_qsize 4, out_qsize 1
2022-06-05 17:02:11,845 : INFO : EPOCH 6: training on 574796 raw words (498480 effective words) took 4.3s, 117113 effective words/s
2022-06-05 17:02:12,917 : INFO : EPOCH 7 - PROGRESS: at 19.92% examples, 98908 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:13,933 : INFO : EPOCH 7 - PROGRESS: at 49.24% examples, 113313 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:02:14,959 : INFO : EPOCH 7 - PROGRESS: at 79.77% examples, 114868 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:02:15,985 : INFO : EPOCH 7 - PROGRESS: at 98.81% examples, 118808 words/s, in_qsize 1, out_qsize 1
2022-06-05 17:02:15,999 : INFO : EPOCH 7: training on 574796 raw words (498871 effective words) took 4.1s, 120567 effective words/s
2022-06-05 17:02:17,048 : INFO : EPOCH 8 - PROGRESS: at 18.31% examples, 92383 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:02:18,139 : INFO : EPOCH 8 - PROGRESS: at 49.64% examples, 110256 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:19,202 : INFO : EPOCH 8 - PROGRESS: at 81.07% examples, 114217 words/s, in_qsize 10, out_qsize 1
2022-06-05 17:02:20,202 : INFO : EPOCH 8 - PROGRESS: at 98.81% examples, 116897 words/s, in_qsize 1, out_qsize 1
2022-06-05 17:02:20,227 : INFO : EPOCH 8: training on 574796 raw words (498705 effective words) took 4.2s, 118320 effective words/s
2022-06-05 17:02:21,269 : INFO : EPOCH 9 - PROGRESS: at 19.90% examples, 102325 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:02:22,333 : INFO : EPOCH 9 - PROGRESS: at 49.24% examples, 112688 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:23,482 : INFO : EPOCH 9 - PROGRESS: at 82.64% examples, 115363 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:24,408 : INFO : EPOCH 9: training on 574796 raw words (498922 effective words) took 4.2s, 119961 effective words/s
2022-06-05 17:02:25,499 : INFO : EPOCH 10 - PROGRESS: at 19.90% examples, 97119 words/s, in_qsize 10, out_qsize 1
2022-06-05 17:02:26,575 : INFO : EPOCH 10 - PROGRESS: at 51.74% examples, 113925 words/s, in_qsize 9, out_qsize 1
2022-06-05 17:02:27,573 : INFO : EPOCH 10 - PROGRESS: at 82.64% examples, 118425 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:28,556 : INFO : EPOCH 10: training on 574796 raw words (498758 effective words) took 4.1s, 120702 effective words/s
2022-06-05 17:02:29,642 : INFO : EPOCH 11 - PROGRESS: at 21.51% examples, 105800 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:02:30,653 : INFO : EPOCH 11 - PROGRESS: at 49.24% examples, 112951 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:31,696 : INFO : EPOCH 11 - PROGRESS: at 81.01% examples, 116760 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:32,708 : INFO : EPOCH 11 - PROGRESS: at 97.35% examples, 115252 words/s, in_qsize 3, out_qsize 0
2022-06-05 17:02:32,784 : INFO : EPOCH 11: training on 574796 raw words (498743 effective words) took 4.2s, 118445 effective words/s
2022-06-05 17:02:33,828 : INFO : EPOCH 12 - PROGRESS: at 18.31% examples, 92862 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:34,847 : INFO : EPOCH 12 - PROGRESS: at 47.13% examples, 110137 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:35,914 : INFO : EPOCH 12 - PROGRESS: at 79.45% examples, 114114 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:36,927 : INFO : EPOCH 12 - PROGRESS: at 96.40% examples, 113206 words/s, in_qsize 4, out_qsize 1
2022-06-05 17:02:37,030 : INFO : EPOCH 12: training on 574796 raw words (498523 effective words) took 4.2s, 117748 effective words/s
2022-06-05 17:02:38,065 : INFO : EPOCH 13 - PROGRESS: at 18.31% examples, 95233 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:39,083 : INFO : EPOCH 13 - PROGRESS: at 47.13% examples, 111766 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:40,243 : INFO : EPOCH 13 - PROGRESS: at 81.01% examples, 114534 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:41,256 : INFO : EPOCH 13 - PROGRESS: at 97.35% examples, 113580 words/s, in_qsize 3, out_qsize 1
2022-06-05 17:02:41,307 : INFO : EPOCH 13: training on 574796 raw words (498984 effective words) took 4.2s, 117465 effective words/s
2022-06-05 17:02:42,328 : INFO : EPOCH 14 - PROGRESS: at 18.31% examples, 95118 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:43,483 : INFO : EPOCH 14 - PROGRESS: at 49.24% examples, 108672 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:02:44,574 : INFO : EPOCH 14 - PROGRESS: at 82.64% examples, 114604 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:45,569 : INFO : EPOCH 14: training on 574796 raw words (498666 effective words) took 4.2s, 117384 effective words/s
2022-06-05 17:02:46,692 : INFO : EPOCH 15 - PROGRESS: at 19.90% examples, 95136 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:02:47,737 : INFO : EPOCH 15 - PROGRESS: at 51.74% examples, 113678 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:02:48,799 : INFO : EPOCH 15 - PROGRESS: at 82.64% examples, 116386 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:49,775 : INFO : EPOCH 15: training on 574796 raw words (498606 effective words) took 4.2s, 119313 effective words/s
2022-06-05 17:02:50,903 : INFO : EPOCH 16 - PROGRESS: at 19.90% examples, 93973 words/s, in_qsize 8, out_qsize 1
2022-06-05 17:02:51,928 : INFO : EPOCH 16 - PROGRESS: at 51.74% examples, 114075 words/s, in_qsize 10, out_qsize 0
2022-06-05 17:02:52,948 : INFO : EPOCH 16 - PROGRESS: at 82.64% examples, 118235 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:53,948 : INFO : EPOCH 16 - PROGRESS: at 98.31% examples, 116845 words/s, in_qsize 2, out_qsize 1
2022-06-05 17:02:53,991 : INFO : EPOCH 16: training on 574796 raw words (499035 effective words) took 4.2s, 118831 effective words/s
2022-06-05 17:02:55,088 : INFO : EPOCH 17 - PROGRESS: at 20.05% examples, 97415 words/s, in_qsize 9, out_qsize 1
2022-06-05 17:02:56,102 : INFO : EPOCH 17 - PROGRESS: at 49.44% examples, 111967 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:57,121 : INFO : EPOCH 17 - PROGRESS: at 79.45% examples, 114248 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:02:58,135 : INFO : EPOCH 17 - PROGRESS: at 97.35% examples, 115336 words/s, in_qsize 3, out_qsize 1
2022-06-05 17:02:58,218 : INFO : EPOCH 17: training on 574796 raw words (498433 effective words) took 4.2s, 118348 effective words/s
2022-06-05 17:02:59,236 : INFO : EPOCH 18 - PROGRESS: at 18.32% examples, 95283 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:03:00,353 : INFO : EPOCH 18 - PROGRESS: at 49.64% examples, 110480 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:03:01,384 : INFO : EPOCH 18 - PROGRESS: at 81.01% examples, 115531 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:03:02,385 : INFO : EPOCH 18 - PROGRESS: at 98.31% examples, 116754 words/s, in_qsize 2, out_qsize 1
2022-06-05 17:03:02,401 : INFO : EPOCH 18: training on 574796 raw words (498549 effective words) took 4.2s, 119537 effective words/s
2022-06-05 17:03:03,421 : INFO : EPOCH 19 - PROGRESS: at 18.31% examples, 95246 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:03:04,479 : INFO : EPOCH 19 - PROGRESS: at 49.24% examples, 113802 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:03:05,493 : INFO : EPOCH 19 - PROGRESS: at 81.01% examples, 118442 words/s, in_qsize 9, out_qsize 0
2022-06-05 17:03:06,501 : INFO : EPOCH 19 - PROGRESS: at 99.04% examples, 119916 words/s, in_qsize 1, out_qsize 1
2022-06-05 17:03:06,523 : INFO : EPOCH 19: training on 574796 raw words (498716 effective words) took 4.1s, 121407 effective words/s
2022-06-05 17:03:06,524 : INFO : FastText lifecycle event {'msg': 'training on 11495920 raw words (9974396 effective words) took 84.2s, 118468 effective words/s', 'datetime': '2022-06-05T17:03:06.524485', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'train'}
2022-06-05 17:03:06,857 : INFO : FastText lifecycle event {'params': 'FastText<vocab=6691, vector_size=100, alpha=0.025>', 'datetime': '2022-06-05T17:03:06.857341', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'created'}
```

Top 10 Words with Fasttext

In [18]:
```
top_words = we.top_10_frequent_words(model_fastText)
top_words
```

Out[18]:
```
['harry',
 'say',
 'ron',
 'look',
 'hermione',
 'get',
 'go',
 'know',
 'think',
 'back']
```

Most similar words to top 10 words with FastText

We see words like Hermione and Ron are most similar to each other. Similarly *back* and *aback*, *say* and *sayin* are similar to each other.

In [19]:
```
we.most_similar_words(model_fastText, top_words)
```

```
Most similar to  harry :  [('harp', 0.5479258298873901), ('arry', 0.5343486666679382), ('harsh', 0.49138596653938293), ('barry', 0.48100706934928894), ('harshly', 0.47930610179901123), ('hopefully', 0.4710060954093933), ('harper', 0.4606828987598419), ('hardly', 0.4522239863872528), ('harness', 0.4437425136566162), ('hard', 0.4433293044567108)]
Most similar to  say :  [('sayin', 0.7130235433578491), ('ask', 0.6018242239952087), ('oh', 0.5920661091804504), ('ah', 0.5869526267051697), ('aaah', 0.577444314956665), ('sad', 0.5720301866531372), ('sarcasm', 0.5597436428070068), ('aaaah', 0.5479901432991028), ('okay', 0.5233922600746155), ('ooh', 0.521248996257782)]
Most similar to  ron :  [('rons', 0.7233307361602783), ('iron', 0.6739728450775146), ('ronan', 0.6580653786659241), ('baron', 0.6393399238586426), ('moron', 0.61771160364151), ('apron', 0.6166530847549438), ('ginny', 0.5256892442703247), ('crookshanks', 0.4998105764389038), ('ok', 0.4722720682621002), ('seamus', 0.41372522711753845)]
Most similar to  look :  [('overlook', 0.6970016956329346), ('lookout', 0.650980532169342), ('lookin', 0.6288907527923584), ('glance', 0.5989559888839722), ('lookouts', 0.5968016386032104), ('ok', 0.5692983865737915), ('lony', 0.5543243288993835), ('cook', 0.5477511286735535), ('loopy', 0.5312240123748779), ('gaze', 0.5196946859359741)]
Most similar to  hermione :  [('hermiones', 0.9478340744972229), ('hermi', 0.9133964776992798), ('hermy', 0.8135080337524414), ('hermes', 0.6915450096130371), ('herbs', 0.5712769031524658), ('herd', 0.5264253616333008), ('hero', 0.46489816904067993), ('throne', 0.41368716955184937), ('hone', 0.40824320912361145), ('harry', 0.40729933977127075)]
Most similar to  get :  [('gettin', 0.696559727191925), ('go', 0.6351714134216309), ('fidget', 0.6239123940467834), ('target', 0.5742014050483704), ('reckon', 0.5635852217674255), ('bet', 0.5623549222946167), ('put', 0.5077040791511536), ('okay', 0.49128615856170654), ('load', 0.48666349053382874), ('anyway', 0.47439008951187134)]
Most similar to  go :  [('okay', 0.6792644262313843), ('get', 0.6351714134216309), ('stay', 0.6334949135780334), ('anyway', 0.5594459176063538), ('10', 0.5568896532058716), ('tomorrow', 0.5242606401443481), ('gonna', 0.5151916742324829), ('goad', 0.4820300042629242), ('maybe', 0.4663326144218445), ('bet', 0.4626072943210602)]
Most similar to  know :  [('knowin', 0.6815596222877502), ('do', 0.6134558320045471), ('believe', 0.5182393193244934), ('important', 0.5178598761558533), ('underestimate', 0.5090644359588623), ('wonderful', 0.5071111917495728), ('knowledge', 0.5066526532173157), ('doubt', 0.5043128132820129), ('impose', 0.503737211227417), ('maybe', 0.5031765103340149)]
Most similar to  think :  [('worry', 0.5189785957336426), ('thinner', 0.51204514503479), ('know', 0.5021456480026245), ('things', 0.49130523204803467), ('thing', 0.4820689260959625), ('idea', 0.45928430557250977), ('hope', 0.44800296425819397), ('sometime', 0.43995410203933716), ('imagine', 0.4386306405067444), ('anything', 0.43211162090301514)]
Most similar to  back :  [('aback', 0.7613603472709656), ('backside', 0.722427248954773), ('ridgeback', 0.7075477838516235), ('knack', 0.6796066761016846), ('jack', 0.6740654706954956), ('hack', 0.6697696447372437), ('whack', 0.655268669128418), ('rucksack', 0.6502652764320374), ('sack', 0.6435987949371338), ('rack', 0.6332429051399231)]
```

To test the model further, I have checked the similar words to some magic spells from Harry Potter. We see, for *Avada (Avada Kedavra* spell), most similar is *Kedavra*. Also, for *Hogwarts*, most similar is *ogwarts*:.

In [20]:
```
print(model_fastText.wv.most_similar("avada"))
print(model_fastText.wv.most_similar("lumos"))
print(model_fastText.wv.most_similar("expecto"))
print(model_fastText.wv.most_similar("hogwarts"))
print(model_fastText.wv.most_similar("dumbledore"))
```

```
[('kedavra', 0.8791778087615967), ('expelliarmus', 0.6212646961212158), ('cruciatus', 0.6167282462120056), ('crucio', 0.6042208671569824), ('curse', 0.5595863461494446), ('bella', 0.5191957354545593), ('bellatrix', 0.5095393657684326), ('killin', 0.5089583396911621), ('whoosh', 0.5022376179695129), ('volkov', 0.4997921288013458)]
[('twirl', 0.54984050989151), ('luminous', 0.5468977093696594), ('glum', 0.5309141278266907), ('wave', 0.5050281286239624), ('patronum', 0.49939408898353577), ('directly', 0.489840567111969), ('flick', 0.4864705502986908), ('tap', 0.48542240262031555), ('raise', 0.481509268283844), ('lumpy', 0.4684904217720032)]
[('expect', 0.7805984616279602), ('unexpected', 0.7061405181884766), ('expel', 0.6880911588668823), ('expelliarmus', 0.6699967980384827), ('expert', 0.6298949718475342), ('expensive', 0.5893750190734863), ('patronum', 0.5768509507179026), ('expectantly', 0.5745901465415955), ('explode', 0.5129415392875671), ('explore', 0.49326473474502563)]
[('ogwarts', 0.967029869556427), ('warts', 0.7906294465065002), ('school', 0.7063440680503845), ('ramparts', 0.6305966973304749), ('boggarts', 0.6025255918502808), ('arts', 0.601914644241333), ('outskirts', 0.5976708531313797), ('hog', 0.5831590890884399), ('hogsmeade', 0.5824134349822998), ('home', 0.5498879551887512)]
[('dumbledores', 0.9661026000976562), ('dumb', 0.8715276122093201), ('dumble', 0.6987357139587402), ('dummy', 0.6022387742996216), ('dum', 0.5677412152290344), ('dore', 0.5602182149887085), ('headmaster', 0.5428498387336731), ('dumbstruck', 0.5373256802558899), ('headmasters', 0.4469207525253296), ('voldemorts', 0.4423289597034454 3)]
```

Odd word using Fasttext

In [21]:
```
word_list = ["Harry","Hermione", "Walle"] #walle
odd_word = we.odd_word(model_fastText, word_list)
print("The odd word is ", odd_word)
```

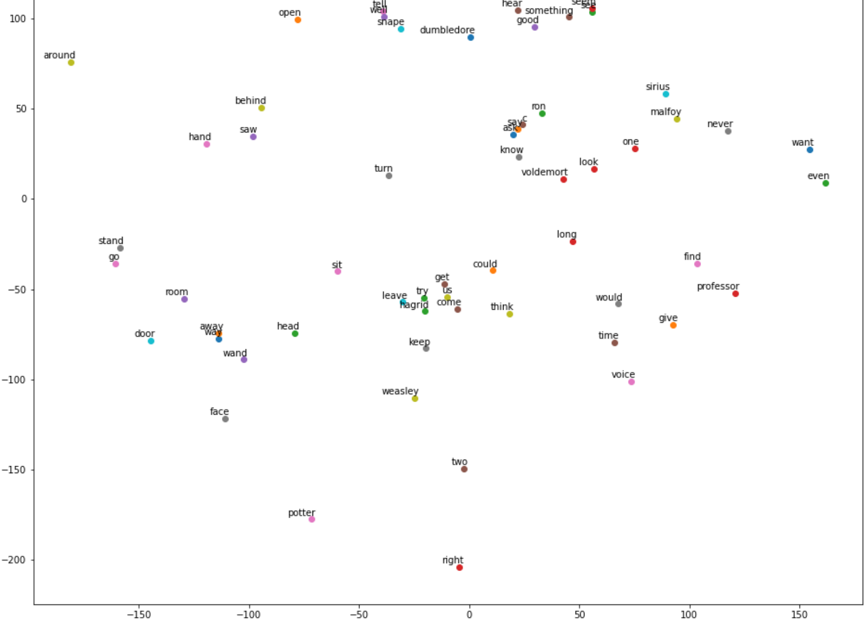
```
The odd word is  walle
```

Visualising Fasttext Embeddings with TSNE

To avoid overly crowded plot, let us plot top word embedding using TSNE.

In [32]:
```
we.plot_tsne(model_fastText, num_words = 70)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: FutureWarning: The default learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:986: FutureWarning: The PCA initialization in TSNE will change to have the standard deviation of PC1 equal to 1e-4 in 1.2. This will ensure better convergence.
  FutureWarning,
```



GloVe

Let us create GloVe word embeddings. Since gensim does not provide support for training a GloVe model, I have used glove-python library. I have experimented with various parameters of the model:

- vector_size
- window_size
- epochs

Observations:

In [22]:
```
window_size = 10
num_features = 100
lr = 0.05
iterations = 20

model_glove = we.glove_model(book_sentences, window_size, num_features, lr, iterations)
```

In [23]:
```
model_glove.save('glove.model')
```

Convert Glove-python model to gensim model

We will convert the glove-python model to a gensim model so we can use gensim model's functionalities. For this, we save the vectors from our trained glove model and treat this saved file as a pretrained model, and load it as a gensim model.

In [24]:
```
glove_input_file = 'glove_embeddings.txt'
word2vec_output_file = 'word2vec.txt'
gensim_glove_model = we.save_glove_as_gensim(model_glove, glove_input_file, word2vec_output_file)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:167: DeprecationWarning: Call to deprecated `glove2word2vec` (KeyedVectors.load_word2vec_format(.., binary=False, no_header=True) loads GLoVE text vectors.).
2022-06-05 17:03:57,523 : INFO : loading projection weights from glove_embeddings.txt
2022-06-05 17:03:59,215 : INFO : KeyedVectors lifecycle event {'msg': 'loaded (16548, 100) matrix of type float32 from glove_embeddings.txt', 'binary': False, 'encoding': 'utf8', 'datetime': '2022-06-05T17:03:59.215097', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'load_word2vec_format'}
2022-06-05 17:03:59,217 : INFO : converting 16548 vectors from glove_embeddings.txt to word2vec.txt
2022-06-05 17:03:59,243 : INFO : storing 16548x100 projection weights into word2vec.txt
2022-06-05 17:04:00,549 : INFO : loading projection weights from word2vec.txt
2022-06-05 17:04:01,681 : INFO : KeyedVectors lifecycle event {'msg': 'loaded (16548, 100) matrix of type float32 from word2vec.txt', 'binary': False, 'encoding': 'utf8', 'datetime': '2022-06-05T17:04:01.681100', 'gensim': '4.2.0', 'python': '3.7.13 (default, Apr 24 2022, 01:04:09) \n[GCC 7.5.0]', 'platform': 'Linux-5.4.188+-x86_64-with-Ubuntu-18.04-bionic', 'event': 'load_word2vec_format'}
```

Top 10 Words

In [46]:
```
top_words = we.top_10_frequent_words_2(gensim_glove_model)
top_words
```

Out[46]:
```
['harry',
 'say',
 'ron',
 'look',
 'hermione',
 'get',
 'go',
 'know',
 'think',
 'back']
```

Most similar words to top 10 words with Glove

We see words like Hermione and Ron are most similar to each other.

In [26]:
```
we.most_similar_words_glove(model_glove, top_words)
```

```
Most similar to  harry :  [('recognise', 0.8920242764912691), ('recognize', 0.8851638641138097), ('release', 0.8363621321737874), ('remind', 0.8245594144648682)]
Most similar to  say :  [('scrimgeour', 0.8941209512199478), ('aberforth', 0.8876266859954253), ('excellent', 0.8781842687072056), ('problem', 0.8756643523985984)]
Most similar to  ron :  [('hermione', 0.9437827451933333), ('hopefully', 0.8850852175642495), ('add', 0.8750077890662885), ('together', 0.8740372002441282)]
Most similar to  look :  [('terrify', 0.9172965599649341), ('round', 0.8967215004735645), ('puzzle', 0.8835046630831572), ('startle', 0.8749498444633205)]
Most similar to  hermione :  [('ron', 0.943782745193333), ('whisper', 0.9044214651971448), ('hopefully', 0.8736637894409676), ('gasp', 0.8435217318957172)]
Most similar to  get :  [('manage', 0.9006999948575642), ('better', 0.8866682892507073), ('stuff', 0.8706698880868858), ('rid', 0.8664509672191718)]
Most similar to  go :  [('let', 0.9505328814641893), ('stay', 0.8883072860096489), ('dinner', 0.8788605161423553), ('home', 0.8672388681531662)]
Most similar to  know :  [('mean', 0.9764131681918871), ('exactly', 0.9516621645683027), ('must', 0.9417828982103208), ('always', 0.9369622967871799)]
Most similar to  think :  [('might', 0.9671288085439728), ('exactly', 0.9359227134593969), ('always', 0.9250160092015055), ('know', 0.9248579246700154)]
Most similar to  back :  [('straight', 0.9107908142602327), ('away', 0.8985640354267357), ('upstairs', 0.8797880394679141), ('walk', 0.875771168890324)]
```

To test the model further, I have checked the similar words to some magic spells from Harry Potter. We see, for *Avada (Avada Kedavra* spell), most similar is *Kedavra*. Also, for *Hogwarts*, most similar is *school*.

In [27]:
```
print(model_glove.most_similar("avada"))
print(model_glove.most_similar("lumos"))
print(model_glove.most_similar("expecto"))
print(model_glove.most_similar("hogwarts"))
print(model_glove.most_similar("dumbledore"))
```

```
[('repel', 0.9817239219921347), ('kedavra', 0.97928894731559), ('seventeenth', 0.9766145368346428), ('mesmerize', 0.9732906002593424)]
[('elder', 0.8595733695982688), ('eyebrows', 0.81342822149693), ('raise', 0.780916413690157), ('flick', 0.7717844294367589)]
[('patronum', 0.9829121078998504), ('plummet', 0.9350777276565848), ('engulf', 0.9342034501856955), ('3', 0.925485034964151)]
[('school', 0.9701746813561194), ('champion', 0.8860520652584163), ('students', 0.8423077861482372), ('express', 0.8377899146113961)]
[('albus', 0.9159896877642514), ('headmaster', 0.8310207445492175), ('snape', 0.8212557100238804), ('trust', 0.8096411244208244)]
```

Odd word using Glove

In [28]:
```
word_list = ["Harry","Hermione", "Wand"]
odd_word = we.odd_word_glove(gensim_glove_model, word_list)
print("The odd word is ", odd_word)
```

```
The odd word is  harry
```

Visualising GloVe Embeddings with TSNE

To avoid overly crowded plot, let us plot top word embedding using TSNE.

In [33]:
```
we.plot_tsne_glove(gensim_glove_model, num_words = 70)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: FutureWarning: The default learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:986: FutureWarning: The PCA initialization in TSNE will change to have the standard deviation of PC1 equal to 1e-4 in 1.2. This will ensure better convergence.
  FutureWarning,
```

Performance measure comparison and conclusion

We can compare model performance using the similarity metrics and overall results returned by all models.

- Word2Vec:
  - We get very similar words like *Harry:Ron* (similarity 0.73), *look:glance* (similarity 0.61), *ron:hermione* (similarity 0.84) which indicates a good performance
  - For the other words like magic spells also it has very significant similarity results, where it returns words that are semantically similar to the spells
- Fasttext:
  - We can see it mostly shows similar words as words that have similar spelling. eg. *harry:arry* (similarity 0.52), _say:sayin (similarity 0.7), *look:overlook* (similarity 0.74), *back:aback* (similarity 0.53)
  - The similarity scores are quite lower than Word2Vec
  - The similarities also do not seem very semantic like word2vec, which showed the character names close to each other instead of similar spelling words.
- GloVe:
  - Glove has high similarity for few pairs like *ron:hermione* (similarity 0.95), but even after changing model parameters, the results do not seem very satisfactory as it returns pairs like *harry:recognise*, *say:scrimgeour* with high similarity scores (0.91, 0.89)

Based on these metrics and observations, Word2Vec model with 100 vector size seems to perform well with respect to metrics as well as semantics on this harry potter dataset

In [29]:
```
print("Word2Vec: \n")
we.most_similar_words(w2vec_model, top_words)
print("\nFasttext: \n")
we.most_similar_words(model_fastText, top_words)
print("\nGlove: \n")
we.most_similar_words_glove(model_glove, top_words)
```

```
Word2Vec:

Most similar to  harry :  [('ron', 0.7265907526016235), ('silencio', 0.6914387345314026), ('hermione', 0.6812456250190735), ('sceptically', 0.6685629487037659), ('tremulously', 0.6664249897003174), ('options', 0.658836305141449), ('goodnight', 0.6429656147956848), ('tensely', 0.638627290725708), ('exasperatedly', 0.6361172795295715), ('back', 0.6331034898757935)]
Most similar to  say :  [('ask', 0.7165927886962891), ('well', 0.7138505578041077), ('scoff', 0.6681712865829468), ('oh', 0.6663616299629211), ('know', 0.6596862077713013), ('hmmm', 0.6513105630874634), ('yes', 0.6503114104270935), ('harry', 0.6323049664497375), ('think', 0.6315926313400269), ('yeah', 0.6279149055480957)]
Most similar to  ron :  [('hermione', 0.8546380996704102), ('harry', 0.7265907526016235), ('goodnight', 0.6727980375289917), ('scandalize', 0.633036732673645), ('ginny', 0.6240652203559875), ('aaah', 0.6045141220092773), ('george', 0.5981253385543823), ('thunderstruck', 0.5979795455932617), ('infuriate', 0.5922998785972595), ('fred', 0.588123083114624)]
Most similar to  look :  [('nonplus', 0.6068686246871948), ('remotely', 0.5933437943458557), ('star', 0.5910016894340515), ('glance', 0.5866807699203491), ('expression', 0.581293523311615), ('puzzle', 0.5663690567016602), ('glum', 0.5637786984443665), ('malevolent', 0.5631609559059143), ('rather', 0.558380663394928), ('exasperation', 0.558159351348877)]
Most similar to  hermione :  [('ron', 0.8546380400657654), ('harry', 0.6812455058097839), ('say', 0.5993678569793701), ('aggressively', 0.5951573848724365), ('doubtfully', 0.5911186933517456), ('uh', 0.5743845105171204), ('alas', 0.5723267197608948), ('vector', 0.5716705918312073), ('plea', 0.5686533451080322), ('scoff', 0.5628979802131653)]
Most similar to  get :  [('go', 0.7277942895889282), ('talkin', 0.6312040090560913), ('know', 0.6294001340866089), ('think', 0.6260281205177307), ('scoff', 0.6253399848937988), ('probably', 0.6235625743865967), ('migh', 0.623383641242981), ('well', 0.6160168647766113), ('want', 0.6120638251304626), ('myst', 0.6111989617347717)]
Most similar to  go :  [('come', 0.7431937456130981), ('get', 0.7277943491935730), ('think', 0.6481966376304626), ('evasively', 0.6255102157592773), ('kip', 0.6227594614028931), ('silencio', 0.6163314580917358), ('quicker', 0.6113894581794739), ('tell', 0.610897421836853), ('know', 0.6103991270065308), ('cop', 0.609782278537750 2)]
Most similar to  know :  [('tell', 0.7449623942375183), ('scrim', 0.731433629989624), ('well', 0.7224025726318359), ('mean', 0.7189705967903137), ('think', 0.714867889881134), ('cop', 0.7121427059173584), ('cop', 0.7114754915237427), ('nutters', 0.6841086149215698), ('genius', 0.6833103895187378), ('justify', 0.6719285845756531)]
Most similar to  think :  [('know', 0.7148679494857788), ('want', 0.6714436411857605), ('scrim', 0.6690962314605713), ('scoff', 0.6671172976493835), ('tell', 0.657023549079895), ('well', 0.6550671458244324), ('go', 0.6481966376304626), ('hesitantly', 0.646798312664032), ('risky', 0.6342747807502747), ('really', 0.6338464021682739)]
Most similar to  back :  [('harry', 0.6331034898757935), ('toward', 0.5963829755783081), ('scruff', 0.5942357182502747), ('kip', 0.5903924107551575), ('straight', 0.5812329649925232), ('onwards', 0.5755997896194458), ('ron', 0.5593794584274292), ('goodnight', 0.5557909607887268), ('aargh', 0.5522965788841248), ('immobilize', 0.548739492893219)]

Fasttext:

Most similar to  harry :  [('harp', 0.5479258298873901), ('arry', 0.5343486666679382), ('harsh', 0.49138596653938293), ('barry', 0.48100706934928894), ('harshly', 0.47930610179901123), ('hopefully', 0.4710060954093933), ('harper', 0.4606828987598419), ('hardly', 0.4522239863872528), ('harness', 0.4437425136566162), ('hard', 0.4433293044567108)]
Most similar to  say :  [('sayin', 0.7130235433578491), ('ask', 0.6018242239952087), ('oh', 0.5920661091804504), ('ah', 0.5869526267051697), ('aaah', 0.577444314956665), ('sad', 0.5720301866531372), ('sarcasm', 0.5597436428070068), ('aaaah', 0.5479901432991028), ('okay', 0.5233922600746155), ('ooh', 0.521248996257782)]
Most similar to  ron :  [('rons', 0.7233307361602783), ('iron', 0.6739728450775146), ('ronan', 0.6580653786659241), ('baron', 0.6393399238586426), ('moron', 0.61771160364151), ('apron', 0.6166530847549438), ('ginny', 0.5256892442703247), ('crookshanks', 0.4998105764389038), ('ok', 0.4722720682621002), ('seamus', 0.41372522711753845)]
Most similar to  look :  [('overlook', 0.6970016956329346), ('lookout', 0.650980532169342), ('lookin', 0.6288907527923584), ('glance', 0.5989559888839722), ('lookouts', 0.5968016386032104), ('ok', 0.5692983865737915), ('lony', 0.5543243288993835), ('cook', 0.5477511286735535), ('loopy', 0.5312240123748779), ('gaze', 0.5196946859359741)]
Most similar to  hermione :  [('hermiones', 0.9478340744972229), ('hermi', 0.9133964776992798), ('hermy', 0.8135080337524414), ('hermes', 0.6915450096130371), ('herbs', 0.5712769031524658), ('herd', 0.5264253616333008), ('hero', 0.46489816904067993), ('throne', 0.41368716955184937), ('hone', 0.40824320912361145), ('harry', 0.40729933977127075)]
Most similar to  get :  [('gettin', 0.696559727191925), ('go', 0.6351714134216309), ('fidget', 0.6239123940467834), ('target', 0.5742014050483704), ('reckon', 0.5635852217674255), ('bet', 0.5623549222946167), ('put', 0.5077040791511536), ('okay', 0.49128615856170654), ('load', 0.48666349053382874), ('anyway', 0.47439008951187134)]
Most similar to  go :  [('okay', 0.6792644262313843), ('get', 0.6351714134216309), ('stay', 0.6334949135780334), ('anyway', 0.5594459176063538), ('10', 0.5568896532058716), ('tomorrow', 0.5242606401443481), ('gonna', 0.5151916742324829), ('goad', 0.4820300042629242), ('maybe', 0.4663326144218445), ('bet', 0.4626072943210602)]
Most similar to  know :  [('knowin', 0.6815596222877502), ('do', 0.6134558320045471), ('believe', 0.5182393193244934), ('important', 0.5178598761558533), ('underestimate', 0.5090644359588623), ('wonderful', 0.5071111917495728), ('knowledge', 0.5066526532173157), ('doubt', 0.5043128132820129), ('impose', 0.503737211227417), ('maybe', 0.5031765103340149)]
Most similar to  think :  [('worry', 0.5189785957336426), ('thinner', 0.51204514503479), ('know', 0.5021456480026245), ('things', 0.49130523204803467), ('thing', 0.4820689260959625), ('idea', 0.45928430557250977), ('hope', 0.44800296425819397), ('sometime', 0.43995410203933716), ('imagine', 0.4386306405067444), ('anything', 0.43211162090301514)]
Most similar to  back :  [('aback', 0.7613603472709656), ('backside', 0.722427248954773), ('ridgeback', 0.7075477838516235), ('knack', 0.6796066761016846), ('jack', 0.6740654706954956), ('hack', 0.6697696447372437), ('whack', 0.655268669128418), ('rucksack', 0.6502652764320374), ('sack', 0.6435987949371338), ('rack', 0.6332429051399231)]

Glove:

Most similar to  harry :  [('recognise', 0.8920242764912691), ('recognize', 0.8851638641138097), ('release', 0.8363621321737874), ('remind', 0.8245594144648682)]
Most similar to  say :  [('scrimgeour', 0.8941209512199478), ('aberforth', 0.8876266859954253), ('excellent', 0.8781842687072056), ('problem', 0.8756643523985984)]
Most similar to  ron :  [('hermione', 0.9437827451933333), ('hopefully', 0.8850852175642495), ('add', 0.8750077890662885), ('together', 0.8740372002441282)]
Most similar to  look :  [('terrify', 0.9172965599649341), ('round', 0.8967215004735645), ('puzzle', 0.8835046630831572), ('startle', 0.8749498444633205)]
Most similar to  hermione :  [('ron', 0.943782745193333), ('whisper', 0.9044214651971448), ('hopefully', 0.8736637894409676), ('gasp', 0.8435217318957172)]
Most similar to  get :  [('manage', 0.9006999948575642), ('better', 0.8866682892507073), ('stuff', 0.8706698880868858), ('rid', 0.8664509672191718)]
Most similar to  go :  [('let', 0.9505328814641893), ('stay', 0.8883072860096489), ('dinner', 0.8788605161423553), ('home', 0.8672388681531662)]
Most similar to  know :  [('mean', 0.9764131681918871), ('exactly', 0.9516621645683027), ('must', 0.9417828982103208), ('always', 0.9369622967871799)]
Most similar to  think :  [('might', 0.9671288085439728), ('exactly', 0.9359227134593969), ('always', 0.9250160092015055), ('know', 0.9248579246700154)]
Most similar to  back :  [('straight', 0.9107908142602327), ('away', 0.8985640354267357), ('upstairs', 0.8797880394679141), ('walk', 0.875771168890324)]
```

In [30]:
```
print("Word2Vec Magic Spells: \n")
print("Most similar to Avada: ", w2vec_model.wv.most_similar("avada"))
print("Most similar to expecto: ",w2vec_model.wv.most_similar("expecto"))
print("Most similar to lumos: ",w2vec_model.wv.most_similar("lumos"))
print("Most similar to dumbledore: ",w2vec_model.wv.most_similar("dumbledore"))
print("Most similar to hogwarts: ",w2vec_model.wv.most_similar("hogwarts"))

print("\nFasttext Magic Spells: \n")
print("Most similar to Avada: ",model_fastText.wv.most_similar("avada"))
print("Most similar to lumos: ",model_fastText.wv.most_similar("lumos"))
print("Most similar to expecto: ",model_fastText.wv.most_similar("expecto"))
print("Most similar to hogwarts: ",model_fastText.wv.most_similar("hogwarts"))
print("Most similar to dumbledore: ",model_fastText.wv.most_similar("dumbledore"))

print("\nGlove Magic Spells: \n")
print("Most similar to Avada: ",model_glove.most_similar("avada"))
print("Most similar to lumos: ",model_glove.most_similar("lumos"))
print("Most similar to expecto: ",model_glove.most_similar("expecto"))
print("Most similar to hogwarts: ",model_glove.most_similar("hogwarts"))
print("Most similar to dumbledore: ",model_glove.most_similar("dumbledore"))
```

```
Word2Vec Magic Spells:

Most similar to Avada:  [('kedavra', 0.9569318890571594), ('expelliarmus', 0.6656011343002319), ('unforgivable', 0.6196795701980591), ('crucio', 0.6014413237571716), ('impedimenta', 0.5960601568222046), ('rebound', 0.578210175037384), ('waitress', 0.5771713256835938), ('whine', 0.5730738639831543), ('stupefy', 0.5638341903686523), ('jet', 0.5617681741714478)]
Most similar to expecto:  [('patronum', 0.9717077612876892), ('otter', 0.6616460084915161), ('wisp', 0.605968177318573), ('fog', 0.6028965711593628), ('petrificus', 0.5945016145706177), ('protego', 0.5840536952018738), ('totalus', 0.5772987604141235), ('stag', 0.5599668622016907), ('clammy', 0.5473964810371399), ('aargh', 0.5438020825386047)]
Most similar to lumos:  [('ignite', 0.7599468231201172), ('wand', 0.6119696497917175), ('reparo', 0.5890833139419556), ('light', 0.5837756395339966), ('vapour', 0.5604777336120605), ('imperio', 0.5588639378547668), ('arch', 0.5546481013298035), ('tip', 0.5527899861335754), ('alohomora', 0.5438332557678223), ('skyward', 0.5427126288414001)]
Most similar to dumbledore:  [('headmaster', 0.6794047951698303), ('courteously', 0.6230568885803223), ('dippet', 0.6062681078910828), ('eccentric', 0.5933166146278381), ('sibyll', 0.58150315284729), ('severus', 0.5798885822296143), ('apprehend', 0.5795800089836121), ('dryly', 0.5770117044448853), ('behaviour', 0.5727391839027405), ('impertinent', 0.5712749361991882)]
Most similar to hogwarts:  [('school', 0.7524427175521851), ('witchcraft', 0.6838997602462769), ('wizardry', 0.6371129155158997), ('educate', 0.6184692978858948), ('express', 0.6029987931251526), ('participate', 0.5891844034194946), ('host', 0.583256721496582), ('expulsion', 0.5609077215194702), ('cop', 0.5506772398948669), ('educational', 0.548794686794281)]

Fasttext Magic Spells:

Most similar to Avada:  [('kedavra', 0.8791778087615967), ('expelliarmus', 0.6212646961212158), ('cruciatus', 0.6167282462120056), ('crucio', 0.6042208671569824), ('curse', 0.5595863461494446), ('bella', 0.5191957354545593), ('bellatrix', 0.5095393657684326), ('killin', 0.5089583396911621), ('whoosh', 0.5022376179695129), ('volkov', 0.4997921288013458)]
Most similar to lumos:  [('twirl', 0.54984050989151), ('luminous', 0.5468977093696594), ('glum', 0.5309141278266907), ('wave', 0.5050281286239624), ('patronum', 0.49939408898353577), ('directly', 0.489840567111969), ('flick', 0.4864705502986908), ('tap', 0.48542240262031555), ('raise', 0.481509268283844), ('lumpy', 0.4684904217720032)]
Most similar to expecto:  [('expect', 0.7805984616279602), ('unexpected', 0.7061405181884766), ('expel', 0.6880911588668823), ('expelliarmus', 0.6699967980384827), ('expert', 0.6298949718475342), ('expensive', 0.5893750190734863), ('patronum', 0.576850950717926), ('expectantly', 0.5745901465415955), ('explode', 0.5129415392875671), ('explore', 0.49326473474502563)]
Most similar to hogwarts:  [('ogwarts', 0.967029869556427), ('warts', 0.7906294465065002), ('school', 0.7063440680503845), ('ramparts', 0.6305966973304749), ('boggarts', 0.6025255918502808), ('arts', 0.601914644241333), ('outskirts', 0.59767085313797), ('hog', 0.5831590890884399), ('hogsmeade', 0.5824134349822998), ('home', 0.5498879551887512)]
Most similar to dumbledore:  [('dumbledores', 0.9661026000976562), ('dumb', 0.8715276122093201), ('dumble', 0.6987357139587402), ('dummy', 0.6022387742996216), ('dum', 0.5677412152290344), ('dore', 0.5602182149887085), ('headmaster', 0.5428498387336731), ('dumbstruck', 0.5373256802558899), ('headmasters', 0.4469207525253296), ('voldemorts', 0.442328959703445 43)]

Glove Magic Spells:

Most similar to Avada:  [('repel', 0.9817239219921347), ('kedavra', 0.97928894731559), ('seventeenth', 0.9766145368346428), ('mesmerize', 0.9732906002593424)]
Most similar to lumos:  [('elder', 0.8595733695982688), ('eyebrows', 0.81342822149693), ('raise', 0.780916413690157), ('flick', 0.771784429367589)]
Most similar to expecto:  [('patronum', 0.9829121078998504), ('plummet', 0.9350777276565848), ('engulf', 0.9342034501856955), ('3', 0.925485034964151)]
Most similar to hogwarts:  [('school', 0.9701746813561194), ('champion', 0.8860520652584163), ('students', 0.8423077861482372), ('express', 0.8377899146113961)]
Most similar to dumbledore:  [('albus', 0.9159896877642514), ('headmaster', 0.8310207445492175), ('snape', 0.8212557100238804), ('trust', 0.8096411244208244)]
```

Convert Notebook to PDF

In [ ]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended texlive-plain-generic
```

In [ ]:
```
!python -m pip install -U notebook-as-pdf
!pyppeteer-install
```