An Analysis of Harry Potter

By Lindsey Feingold

Research Question:

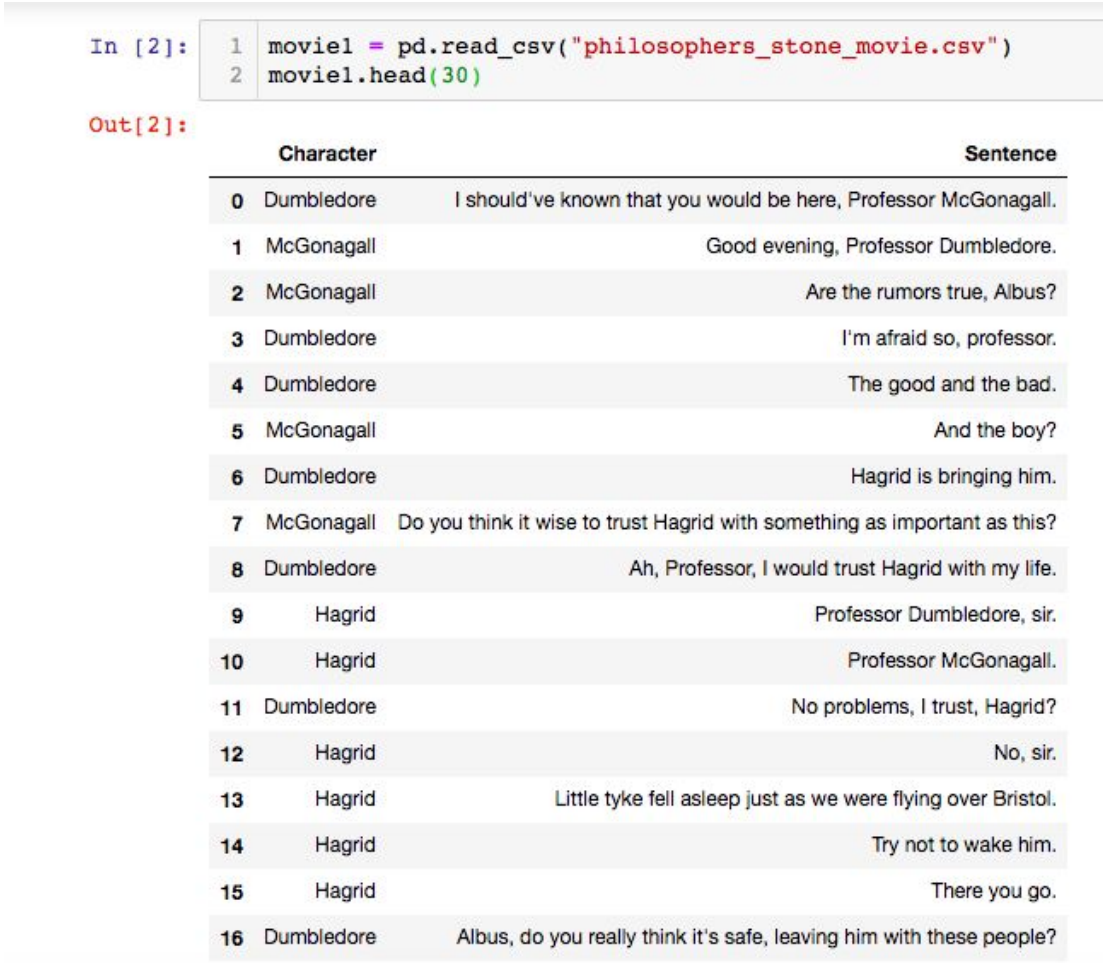
Are the main characters speaking differently in the first book versus the first movie?

NLP methods:

- tf-idf
- Sentence length
- Complexity

Movie csv

- From Kaggle
- Script, split up by each line and character
- Very clean :)

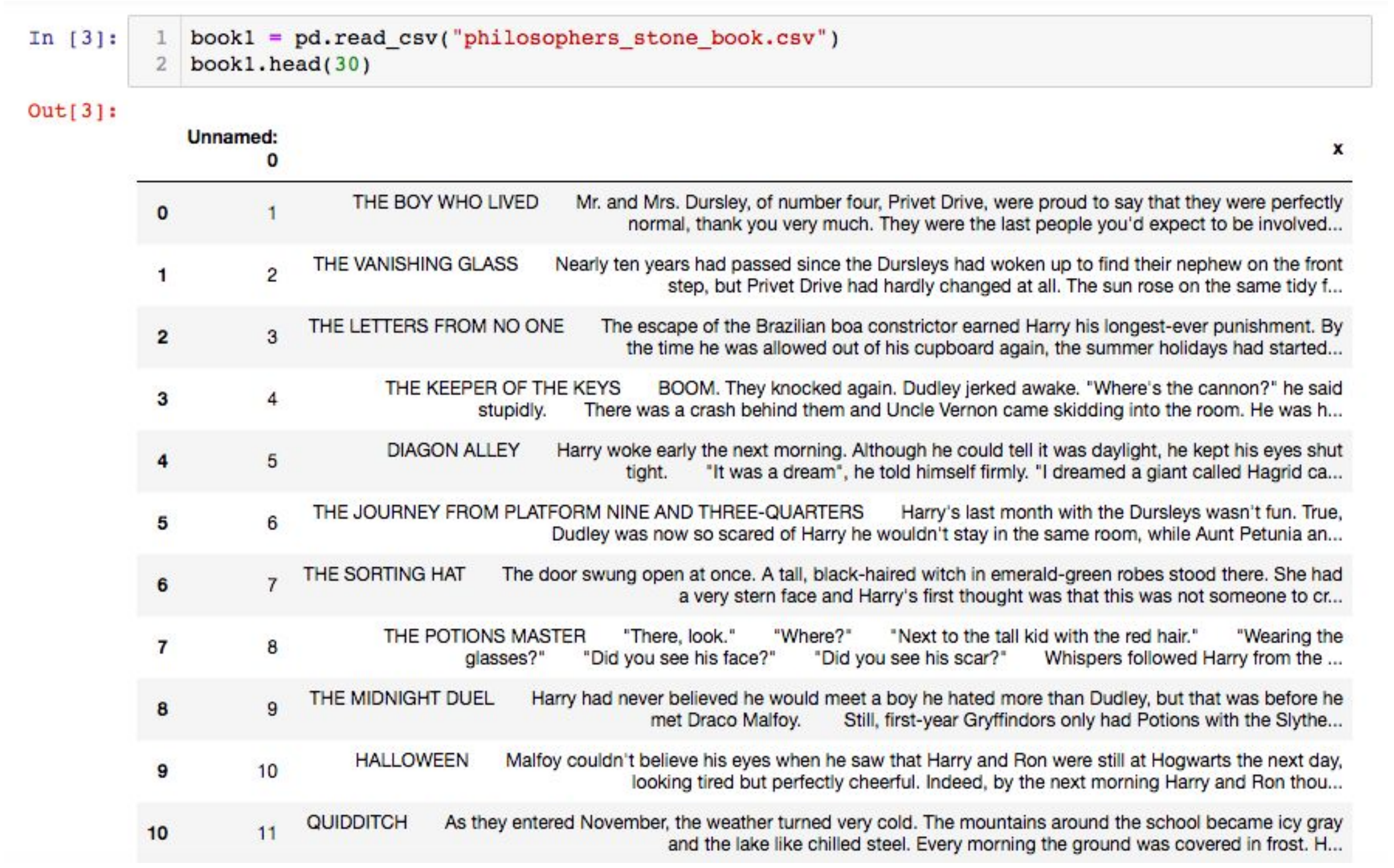


```
In [2]: movie1 = pd.read_csv("philosophers_stone_movie.csv")
        movie1.head(30)
Out[2]:
```

	Character	Sentence
0	Dumbledore	I should've known that you would be here, Professor McGonagall.
1	McGonagall	Good evening, Professor Dumbledore.
2	McGonagall	Are the rumors true, Albus?
3	Dumbledore	I'm afraid so, professor.
4	Dumbledore	The good and the bad.
5	McGonagall	And the boy?
6	Dumbledore	Hagrid is bringing him.
7	McGonagall	Do you think it wise to trust Hagrid with something as important as this?
8	Dumbledore	Ah, Professor, I would trust Hagrid with my life.
9	Hagrid	Professor Dumbledore, sir.
10	Hagrid	Professor McGonagall.
11	Dumbledore	No problems, I trust, Hagrid?
12	Hagrid	No, sir.
13	Hagrid	Little tyke fell asleep just as we were flying over Bristol.
14	Hagrid	Try not to wake him.
15	Hagrid	There you go.
16	Dumbledore	Albus, do you really think it's safe, leaving him with these people?

Book csv

- From Github
- Entire novel
- A mess



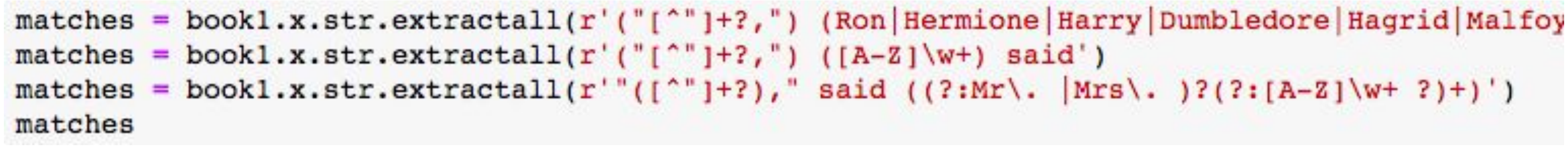
```
In [3]:  book1 = pd.read_csv("philosophers_stone_book.csv")
         book1.head(30)
```

Out[3]:

	Unnamed: 0		x
0	1	THE BOY WHO LIVED	Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved...
1	2	THE VANISHING GLASS	Nearly ten years had passed since the Dursleys had woken up to find their nephew on the front step, but Privet Drive had hardly changed at all. The sun rose on the same tidy f...
2	3	THE LETTERS FROM NO ONE	The escape of the Brazilian boa constrictor earned Harry his longest-ever punishment. By the time he was allowed out of his cupboard again, the summer holidays had started...
3	4	THE KEEPER OF THE KEYS	BOOM. They knocked again. Dudley jerked awake. "Where's the cannon?" he said stupidly. There was a crash behind them and Uncle Vernon came skidding into the room. He was h...
4	5	DIAGON ALLEY	Harry woke early the next morning. Although he could tell it was daylight, he kept his eyes shut tight. "It was a dream", he told himself firmly. "I dreamed a giant called Hagrid ca...
5	6	THE JOURNEY FROM PLATFORM NINE AND THREE-QUARTERS	Harry's last month with the Dursleys wasn't fun. True, Dudley was now so scared of Harry he wouldn't stay in the same room, while Aunt Petunia an...
6	7	THE SORTING HAT	The door swung open at once. A tall, black-haired witch in emerald-green robes stood there. She had a very stern face and Harry's first thought was that this was not someone to cr...
7	8	THE POTIONS MASTER	"There, look." "Where?" "Next to the tall kid with the red hair." "Wearing the glasses?" "Did you see his face?" "Did you see his scar?" Whispers followed Harry from the ...
8	9	THE MIDNIGHT DUEL	Harry had never believed he would meet a boy he hated more than Dudley, but that was before he met Draco Malfoy. Still, first-year Gryffindors only had Potions with the Slythe...
9	10	HALLOWEEN	Malfoy couldn't believe his eyes when he saw that Harry and Ron were still at Hogwarts the next day, looking tired but perfectly cheerful. Indeed, by the next morning Harry and Ron thou...
10	11	QUIDDITCH	As they entered November, the weather turned very cold. The mountains around the school became icy gray and the lake like chilled steel. Every morning the ground was covered in frost. H...

Regex

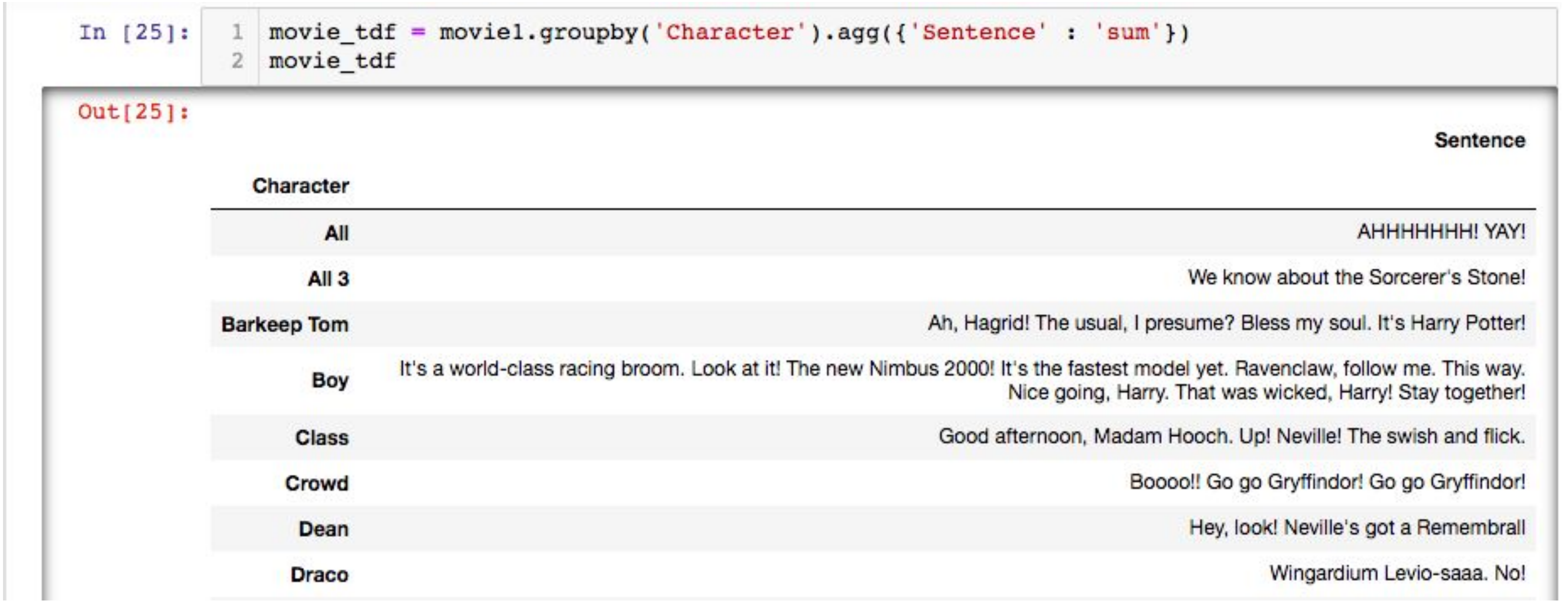
I used regex to find every quote said by a character for the first book. I had to manually going through the csv because some quotes were mismatched, meaning it didn't have a beginning or end quotation mark. It was not a fun process.



```
matches = book1.x.str.extractall(r'("[^"]+?,") (Ron|Hermione|Harry|Dumbledore|Hagrid|Malfoy
matches = book1.x.str.extractall(r'("[^"]+?,") ([A-Z]\w+) said')
matches = book1.x.str.extractall(r'"([^"]+?)," said ((?:Mr\. |Mrs\. )?(?:[A-Z]\w+ ?)+)')
matches
```

tf-idf

- Step 1: turned all columns into a string, then combined all the sentences per character together through aggregation



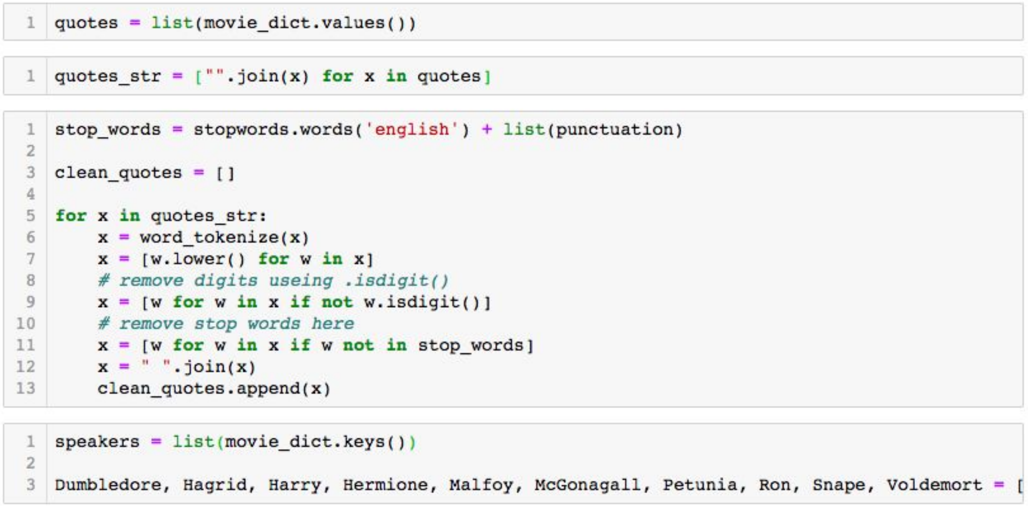
In [25]:
```
movie_tdf = movie1.groupby('Character').agg({'Sentence' : 'sum'})
movie_tdf
```

Out[25]:

Character	Sentence
All	AHHHHHHH! YAY!
All 3	We know about the Sorcerer's Stone!
Barkeep Tom	Ah, Hagrid! The usual, I presume? Bless my soul. It's Harry Potter!
Boy	It's a world-class racing broom. Look at it! The new Nimbus 2000! It's the fastest model yet. Ravenclaw, follow me. This way. Nice going, Harry. That was wicked, Harry! Stay together!
Class	Good afternoon, Madam Hooch. Up! Neville! The swish and flick.
Crowd	Boooo!! Go go Gryffindor! Go go Gryffindor!
Dean	Hey, look! Neville's got a Remembrall
Draco	Wingardium Levio-saaa. No!

tf-idf

- Step 2: chose which characters I wanted to use, and turned that into a dictionary
  - Harry, Hermione, Ron, Dumbledore, Hagrid, Malfoy, McGonagall, Snape, Petunia
- Step 3: joined all the quotes per character together after creating a list of them, added stop words, tokenized the words, and added the list of speakers
- Step 4: ran a vectorizer



```
quotes = list(movie_dict.values())
```

```
quotes_str = ["".join(x) for x in quotes]
```

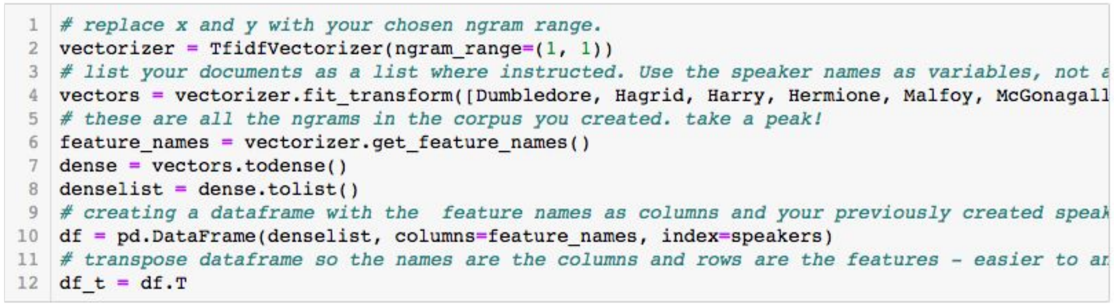
```
stop_words = stopwords.words('english') + list(punctuation)

clean_quotes = []

for x in quotes_str:
    x = word_tokenize(x)
    x = [w.lower() for w in x]
    # remove digits useing .isdigit()
    x = [w for w in x if not w.isdigit()]
    # remove stop words here
    x = [w for w in x if w not in stop_words]
    x = " ".join(x)
    clean_quotes.append(x)
```

```
speakers = list(movie_dict.keys())

Dumbledore, Hagrid, Harry, Hermione, Malfoy, McGonagall, Petunia, Ron, Snape, Voldemort = [
```



```
# replace x and y with your chosen ngram range.
vectorizer = TfidfVectorizer(ngram_range=(1, 1))
# list your documents as a list where instructed. Use the speaker names as variables, not a
vectors = vectorizer.fit_transform([Dumbledore, Hagrid, Harry, Hermione, Malfoy, McGonagall
# these are all the ngrams in the corpus you created. take a peak!
feature_names = vectorizer.get_feature_names()
dense = vectors.todense()
denselist = dense.tolist()
# creating a dataframe with the  feature names as columns and your previously created speak
df = pd.DataFrame(denselist, columns=feature_names, index=speakers)
# transpose dataframe so the names are the columns and rows are the features - easier to ar
df_t = df.T
```

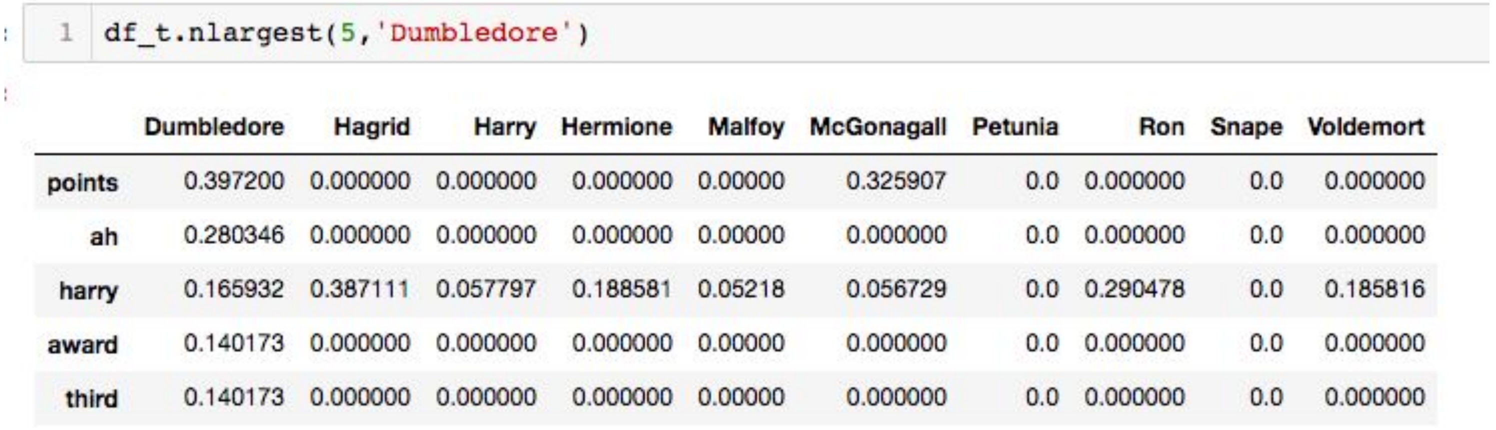
tf-idf results for the movie

- Hagrid, Harry, and Hermione all say Voldemort more than they should
- Harry says other people's names more than anything else, and Hermione, Ron, Draco, Hagrid, and Dumbledore both say Harry's name A LOT

tf-idf results for the book

- Dumbledore, Hagrid, and Harry all say Voldemort more than they should
- Most characters say other people's names a lot less than in the movie
- Top words are more boring

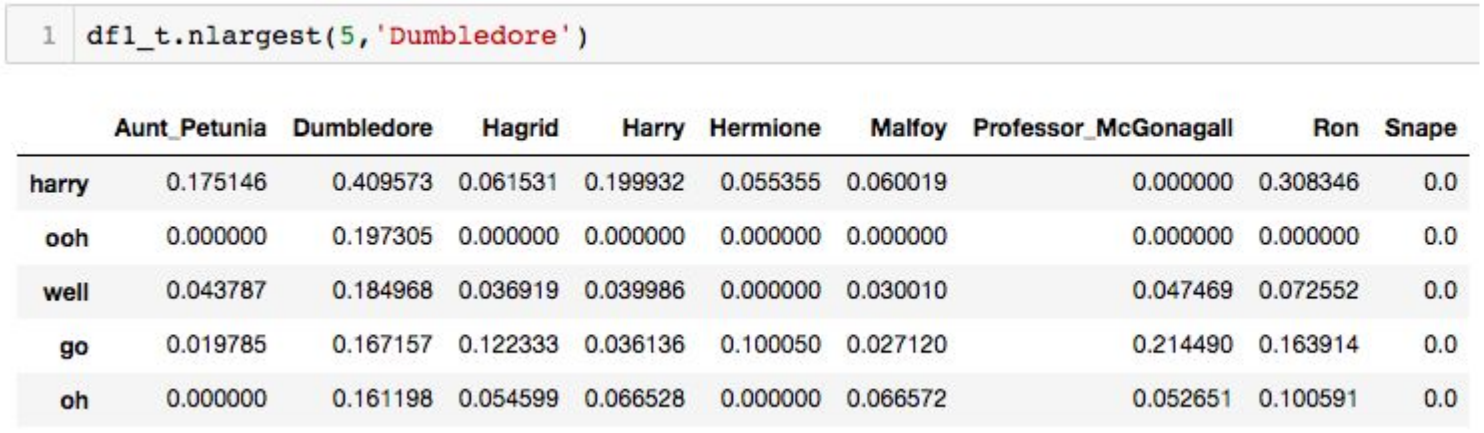
Movie:



```
df_t.nlargest(5,'Dumbledore')
```

	Dumbledore	Hagrid	Harry	Hermione	Malfoy	McGonagall	Petunia	Ron	Snape	Voldemort
points	0.397200	0.000000	0.000000	0.000000	0.00000	0.325907	0.0	0.000000	0.0	0.000000
ah	0.280346	0.000000	0.000000	0.000000	0.00000	0.000000	0.0	0.000000	0.0	0.000000
harry	0.165932	0.387111	0.057797	0.188581	0.05218	0.056729	0.0	0.290478	0.0	0.185816
award	0.140173	0.000000	0.000000	0.000000	0.00000	0.000000	0.0	0.000000	0.0	0.000000
third	0.140173	0.000000	0.000000	0.000000	0.00000	0.000000	0.0	0.000000	0.0	0.000000

Book:

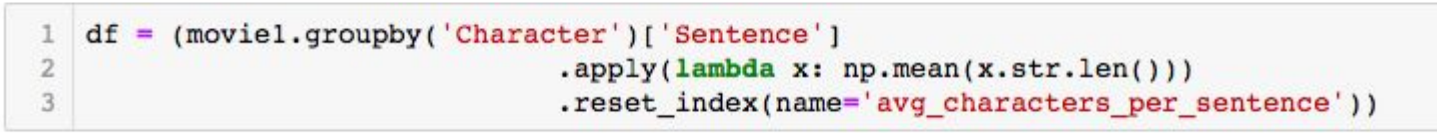


```
df1_t.nlargest(5,'Dumbledore')
```

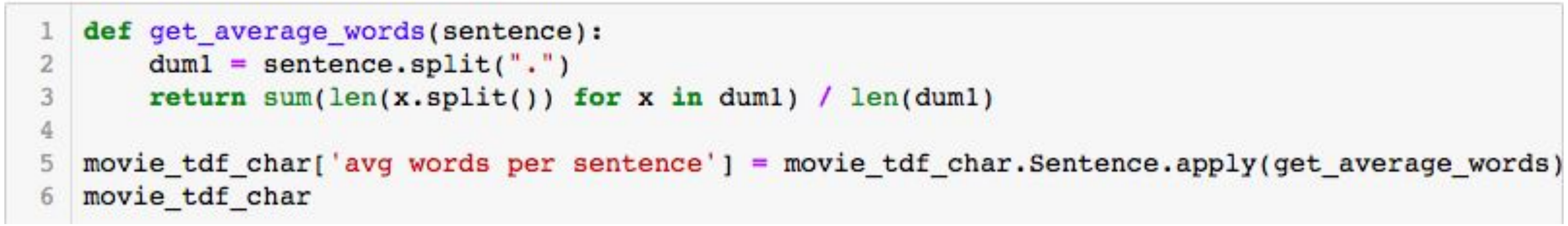
	Aunt_Petunia	Dumbledore	Hagrid	Harry	Hermione	Malfoy	Professor_McGonagall	Ron	Snape
harry	0.175146	0.409573	0.061531	0.199932	0.055355	0.060019	0.000000	0.308346	0.0
ooh	0.000000	0.197305	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
well	0.043787	0.184968	0.036919	0.039986	0.000000	0.030010	0.047469	0.072552	0.0
go	0.019785	0.167157	0.122333	0.036136	0.100050	0.027120	0.214490	0.163914	0.0
oh	0.000000	0.161198	0.054599	0.066528	0.000000	0.066572	0.052651	0.100591	0.0

Sentence length

- Two ways: character-count and word-count

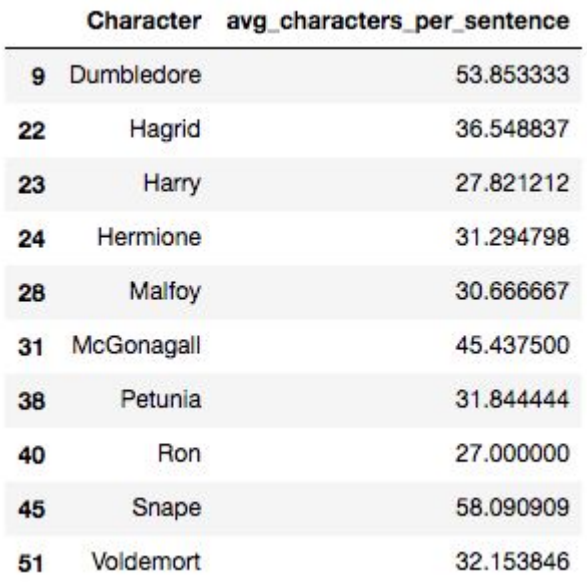


```python
df = (movie1.groupby('Character')['Sentence']
                            .apply(lambda x: np.mean(x.str.len()))
                            .reset_index(name='avg_characters_per_sentence'))
```

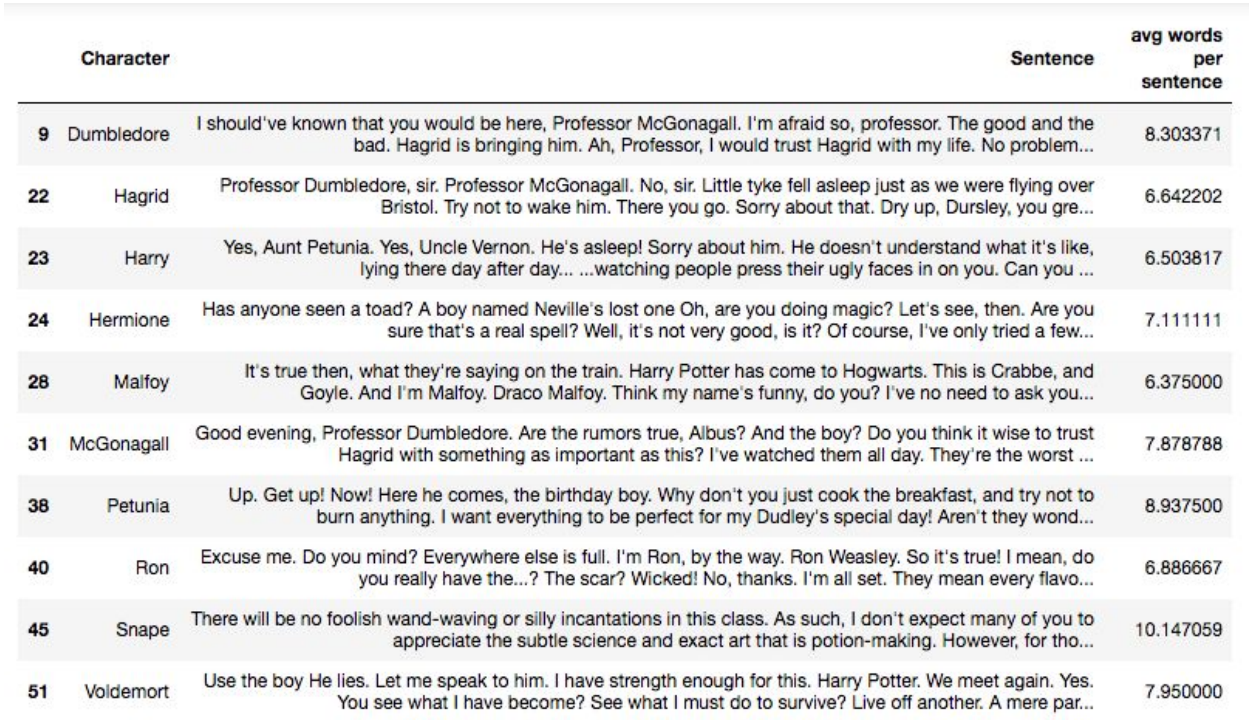


```python
def get_average_words(sentence):
    duml = sentence.split(".")
    return sum(len(x.split()) for x in duml) / len(duml)

movie_tdf_char['avg words per sentence'] = movie_tdf_char.Sentence.apply(get_average_words)
movie_tdf_char
```



	Character	avg_characters_per_sentence
9	Dumbledore	53.853333
22	Hagrid	36.548837
23	Harry	27.821212
24	Hermione	31.294798
28	Malfoy	30.666667
31	McGonagall	45.437500
38	Petunia	31.844444
40	Ron	27.000000
45	Snape	58.090909
51	Voldemort	32.153846



	Character	Sentence	avg words per sentence
9	Dumbledore	I should've known that you would be here, Professor McGonagall. I'm afraid so, professor. The good and the bad. Hagrid is bringing him. Ah, Professor, I would trust Hagrid with my life. No problem...	8.303371
22	Hagrid	Professor Dumbledore, sir. Professor McGonagall. No, sir. Little tyke fell asleep just as we were flying over Bristol. Try not to wake him. There you go. Sorry about that. Dry up, Dursley, you gre...	6.642202
23	Harry	Yes, Aunt Petunia. Yes, Uncle Vernon. He's asleep! Sorry about him. He doesn't understand what it's like, lying there day after day... ...watching people press their ugly faces in on you. Can you ...	6.503817
24	Hermione	Has anyone seen a toad? A boy named Neville's lost one Oh, are you doing magic? Let's see, then. Are you sure that's a real spell? Well, it's not very good, is it? Of course, I've only tried a few...	7.111111
28	Malfoy	It's true then, what they're saying on the train. Harry Potter has come to Hogwarts. This is Crabbe, and Goyle. And I'm Malfoy. Draco Malfoy. Think my name's funny, do you? I've no need to ask you...	6.375000
31	McGonagall	Good evening, Professor Dumbledore. Are the rumors true, Albus? And the boy? Do you think it wise to trust Hagrid with something as important as this? I've watched them all day. They're the worst ...	7.878788
38	Petunia	Up. Get up! Now! Here he comes, the birthday boy. Why don't you just cook the breakfast, and try not to burn anything. I want everything to be perfect for my Dudley's special day! Aren't they wond...	8.937500
40	Ron	Excuse me. Do you mind? Everywhere else is full. I'm Ron, by the way. Ron Weasley. So it's true! I mean, do you really have the...? The scar? Wicked! No, thanks. I'm all set. They mean every flavo...	6.886667
45	Snape	There will be no foolish wand-waving or silly incantations in this class. As such, I don't expect many of you to appreciate the subtle science and exact art that is potion-making. However, for tho...	10.147059
51	Voldemort	Use the boy He lies. Let me speak to him. I have strength enough for this. Harry Potter. We meet again. Yes. You see what I have become? See what I must do to survive? Live off another. A mere par...	7.950000

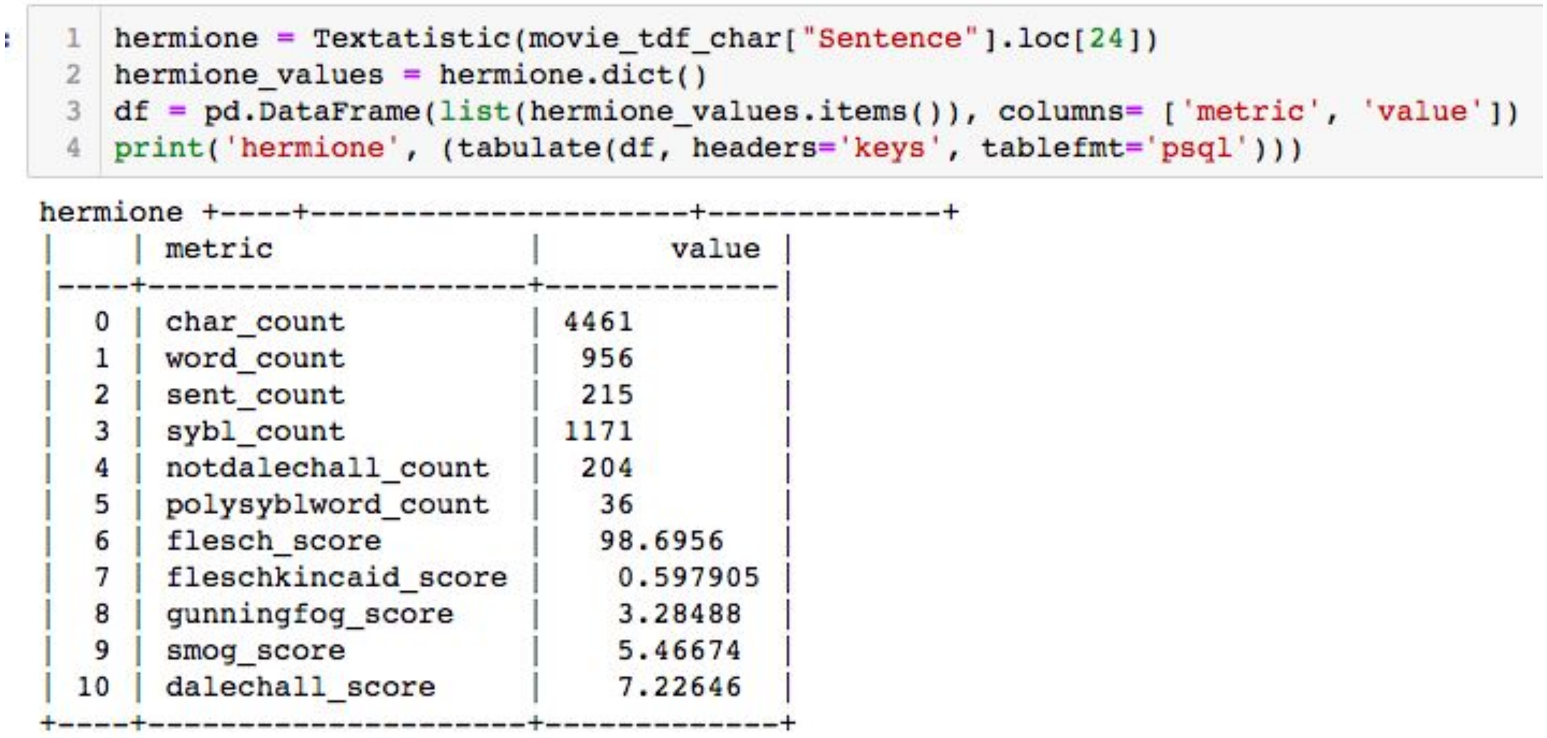
Sentence length results

- Character-count
  - Movie: Snape uses the most characters and Ron uses the least
  - Book: Snape uses the most characters and Harry uses the least

- Word-count
  - Movie: Snape says the most words per sentence on average and Malfoy says the least
  - Book: Snape says the most words per sentence on average and Petunia says the least

Complexity

- I used a textatistic library on each character to calculate the total character count, word count, sentence count, syllable count, and other scores including:
- the Flesch score (the lower the score, the more difficult the text is to read)
- the Dalechall score (it measures a text against a number of words considered familiar to fourth-graders, and the more unfamiliar words used, the higher the reading level will be)



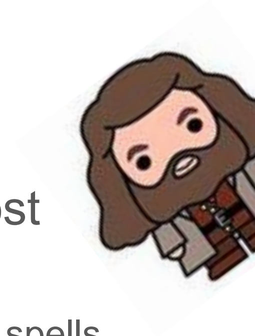
```python
hermione = Textatistic(movie_tdf_char["Sentence"].loc[24])
hermione_values = hermione.dict()
df = pd.DataFrame(list(hermione_values.items()), columns= ['metric', 'value'])
print('hermione', (tabulate(df, headers='keys', tablefmt='psql')))
```

```
hermione +----+--------------------+------------+
|    | metric             |      value |
|----+--------------------+------------|
|  0 | char_count         | 4461       |
|  1 | word_count         |  956       |
|  2 | sent_count         |  215       |
|  3 | sybl_count         | 1171       |
|  4 | notdalechall_count |  204       |
|  5 | polysyblword_count |   36       |
|  6 | flesch_score       |   98.6956  |
|  7 | fleschkincaid_score|    0.597905|
|  8 | gunningfog_score   |    3.28488 |
|  9 | smog_score         |    5.46674 |
| 10 | dalechall_score    |    7.22646 |
+----+--------------------+------------+
```

Complexity results

- Movie: I found that Dumbledore's sentences are the most complex, according to the Flesch score, and that Hermione's sentences are the most complex according to the Dalechall score

- Book: I found that Dumbledore's sentences are still the most complex, according to the Flesch score, and that Hagrid's sentences are the most complex according to the Dalechall score
  - (Hagrid says a lot of words in an accent, which J.K. Rowling emphasizes in how she spells words he says. A list of common words a fourth-grader would recognize might not include "yeh" or "gettin'", it would include "your" and "getting" since that is the correct spelling.)



Overall findings

- Snape is the best represented from the first book to the first movie in terms of complexity

- Dumbledore and Hermione are also represented accurately as having a high number of complex sentences from the book to the movie

- But overall, there are only subtle differences
    - Ex: Harry's Dalechall score for the movie was 6.4 and his score for the book was 6.3
    - Ex: Ron's average words per sentence for the movie was 27 and for the book it was 30

- The top used words were different, but the book had more boring results

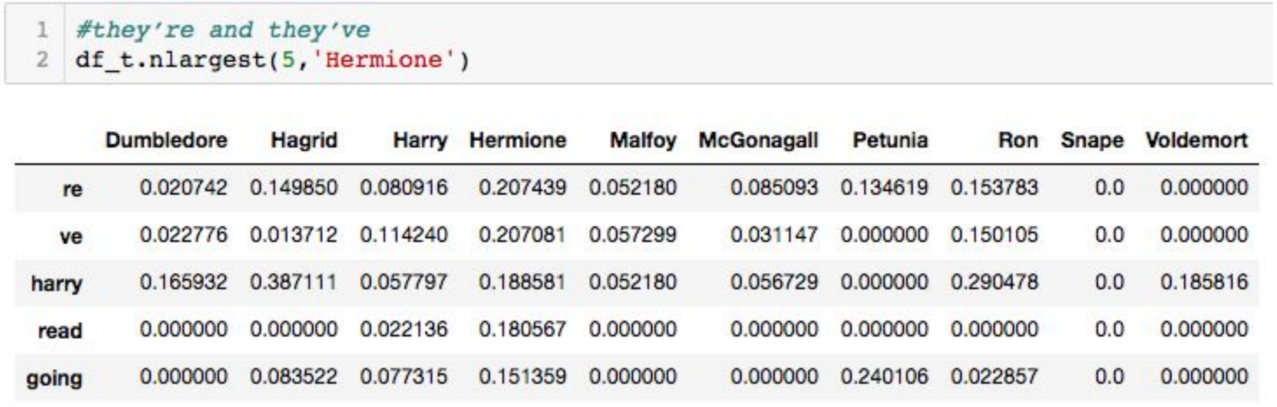
Limitations

- quotes in the book that were attributed as "he said" and "she said" were not taken, which definitely changes things

- Snape was an issue during the tf-idf vectorizing process (wasn't calculated correctly)

- Vectorizing was also an issue with they've and they're -- had ve and re as separate words

- Future: look at other books and movies since books get way more complex



```python
#they're and they've
df_t.nlargest(5,'Hermione')
```

	Dumbledore	Hagrid	Harry	Hermione	Malfoy	McGonagall	Petunia	Ron	Snape	Voldemort
re	0.020742	0.149850	0.080916	0.207439	0.052180	0.085093	0.134619	0.153783	0.0	0.000000
ve	0.022776	0.013712	0.114240	0.207081	0.057299	0.031147	0.000000	0.150105	0.0	0.000000
harry	0.165932	0.387111	0.057797	0.188581	0.052180	0.056729	0.000000	0.290478	0.0	0.185816
read	0.000000	0.000000	0.022136	0.180567	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
going	0.000000	0.083522	0.077315	0.151359	0.000000	0.000000	0.240106	0.022857	0.0	0.000000

The end :)