Am I your father?
Applying Computational Methods in Detecting Grammatical Similarities in the Dialogues between Star Wars Characters

By Simon Zuberek
November 24th, 2020
CUNY Graduate Center

Introduction

- Final Project for Methods in Computational Linguistics I (Fall 2019)
- Inspired by the last episode of the Star Wars saga
- Python
- Little prior programming background

Introduction

- **“Every son quotes his father, in words and deeds.”** - *Anonymous*
- We know how the actions of Luke resonate with those of his father, but what about his language?

Hypothesis

- What can Luke and Vader *tell* us about their family dynamic.
- If the two characters are indeed patrilineally related then we can hypothesize that this connection *may* be reflected in their dialogue, i.e. that Luke's and Vader's language *may* be similar.
- The project applies basic computational methods to examine whether Luke Skywalker actually *speaks* like his father, Darth Vader


I am your father?

Method: Obtaining the Data

- Screenplays for the "Original Trilogy" (OT: Episodes IV - VI)
- Pre-processed `.txt` files:
  - Only enumerated dialogue lines for every character in each film
  - Saved saved under `ep#.txt` (where # stands for the episode number)

Method: Preparing the "Raw" Data

```
"character" "dialogue"
"1" "THREEPIO" "Did you hear that?  They've shut down the main reactor.  We'll be
destroyed for sure.  This is madness!"
"2" "THREEPIO" "We're doomed!"
"3" "THREEPIO" "There'll be no escape for the Princess this time."
"4" "THREEPIO" "What's that?"
"5" "THREEPIO" "I should have known better than to trust the logic of a half-sized
thermocapsulary dehousing assister..."
"6" "LUKE" "Hurry up!  Come with me!  What are you waiting for?!  Get in gear!"
"7" "THREEPIO" "Artoo! Artoo-Detoo, where are you?"
"8" "THREEPIO" "At last!  Where have you been?"
"9" "THREEPIO" "They're heading in this direction. What are we going to do?  We'll be sent
to the spice mines of Kessel or smashed into who knows what!"
"10" "THREEPIO" "Wait a minute, where are you going?"
```

Method: Preparing the Data (Code)

Removing the line numbers

```
with open('ep4.txt', 'r', encoding = 'utf-8') as input, open('ep4_out.txt', 'w', encoding
= 'utf-8') as output:
        for line in input:
            line = re.sub(r'^"\d+"', '', line).casefold()
            output.write(line)

    path = 'ep4_out.txt'
```

Method: Preparing the Data (Output)

```
"character" "dialogue"
 "threepio" "did you hear that?  they've shut down the main reactor.  we'll be destroyed
for sure.  this is madness!"
 "threepio" "we're doomed!"
 "threepio" "there'll be no escape for the princess this time."
 "threepio" "what's that?"
 "threepio" "i should have known better than to trust the logic of a half-sized
thermocapsulary dehousing assister..."
 "luke" "hurry up!  come with me!  what are you waiting for?!  get in gear!"
 "threepio" "artoo! artoo-detoo, where are you?"
 "threepio" "at last!  where have you been?"
 "threepio" "they're heading in this direction. what are we going to do?  we'll be sent to
the spice mines of kessel or smashed into who knows what!"
 "threepio" "wait a minute, where are you going?"
```


YOU'RE ALL CLEAR KID!
memegenerator.net

Method: Isolating the Characters

- Only interested in Luke's and Vader's lines
- The script will work for other characters appearing in the data.
- Having isolated the lines uttered by either character we want to clean them:
  - Remove the `"Vader"` / `"Luke"` strings
  - The interpunction:
    - Remove the quotation marks
    - Remove the commas
    - **Leave** the periods, question marks, and exclamation points.

Method: Isolating the Characters (Code)

```
with open(path, 'r') as source:
        lines = source.readlines()

        vader = re.compile(r'"vader".+')
        vader_lines = list()

        for line in lines:
            match = vader.search(line)
            if match:
                vader_lines.append(match.group())

        global vader lines str
        vader_lines_str = str()

        for line in vader lines:
            vader lines str += line
            vader lines str = vader lines str.replace('"vader"', '')
            vader lines str = vader lines str.replace('"', '')
            vader_lines_str = vader_lines_str.replace(',', '')

print(f'Lord Vader says: \n {vader_lines_str}')
```

Method: Isolating the Characters (Output)

Lord Vader says:
  where are those transmissions you intercepted? if this is a consular ship... where is the ambassador? commander tear this ship apart until you've found those plans and bring me the ambassador. i want her alive! don't play games with me your highness. you weren't on any mercy mission this time. you passed directly through a restricted system. several transmissions were beamed to this ship by rebel spies. i want to know what happened to the plans they sent you. you're a part of the rebel alliance... and a traitor. take her away! i have traced the rebel spies to her. now she is my only link to find their secret base! leave that to me. send a distress signal and then inform the senate that all aboard were killed! she must have hidden the plans in the escape pod. send a detachment down to retrieve them. see to it personally commander. there'll be no one to stop us this time. the plans you refer to will soon be back in our hands. don't be too proud of this technological terror you've constructed. the ability to destroy a planet is insignificant next to the power of the force. i find your lack of faith disturbing. as you wish. and now your highness we will discuss the location of your hidden rebel base. her resistance to the mind probe is considerable. (etc.)


A boring conversation anyway.

Method: Processing the Dialogue Lines

Process things in order:

1. Split the continuous string into sentences
2. Split the sentences into word tokens
3. POS tag the words

Method: Processing the Dialogues (Code)

Splitting the string into sentences and word tokens

```
from nltk.tokenize import PunktSentenceTokenizer, word_tokenize

def word_sentence_tokenize(text):
  sentence_tokenizer = PunktSentenceTokenizer(text)
  sentence_tokenized = sentence_tokenizer.tokenize(text)
  word_tokenized = list()

  for tokenized_sentence in sentence_tokenized:
    word_tokenized.append(word_tokenize(tokenized_sentence))

  return word_tokenized
```

Method: Splitting into Sentences and Word Tokens (Output)

```
def process_vader():
    vader_tokenized = word_sentence_tokenize(vader)

    single_sentence_tokenized = vader_tokenized[27]
    print(f"Vader's single tokenized sentence: {single_sentence_tokenized}")
```

```
>>> Vader's single tokenized sentence: ['what', 'do', 'you', 'mean', '?']
```

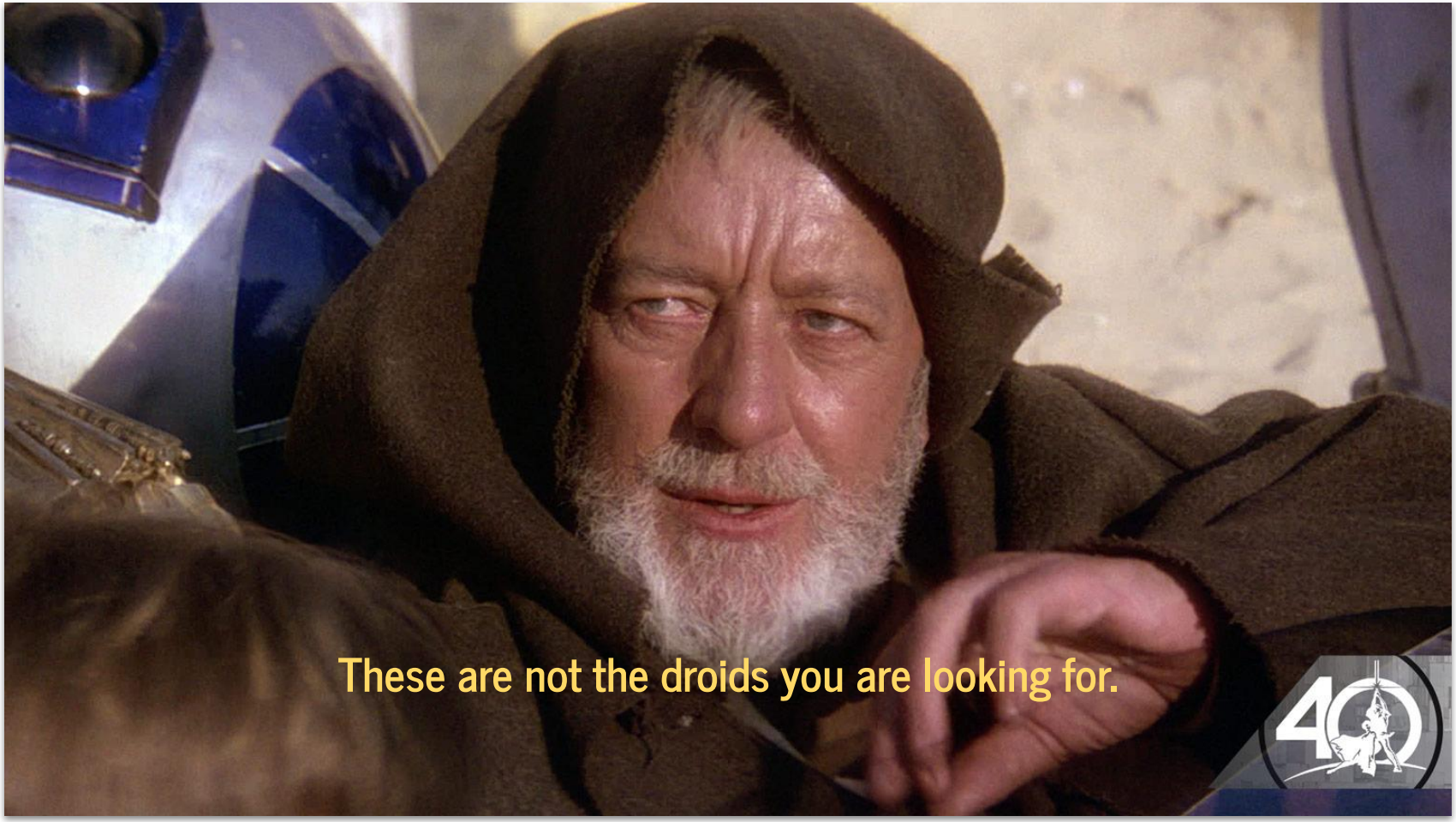
Method: POS Tagging (Output)

```
pos_tagged_vader = list()

    for sentence in vader_tokenized:
        pos_tagged_vader.append(pos_tag(sentence))

    pos_tagged_sentence = pos_tagged_vader[27]
    print()
    print(f"Vader's single part-of-speech tagged sentence: {pos_tagged_sentence}")
```

```
>>> Vader's single part-of-speech tagged sentence: [('what', 'WP'), ('do', 'VBP'), ('you',
'PRP'), ('mean', 'VB'), ('?', '.')]
```

Method: The next steps?

- Everything is broken into sentences, tokenized, and POS-tagged.
- Count the parts of speech? Compare and contrast with Luke?
- None of this is very useful...


These are not the droids you are looking for.

Method: Syntactic Analysis

Grammar: Noun phrases (NPs), Verb Phrases (VPs), Prepositional Phrases (PP)

- NP = (Det.) + n(ADJ) + N
- VP = NP + V + n(ADV) + NP and/or PP
- PP = Prep. + NP

Grammar in RegEx:

- `'NP: {<DT>?<JJ.?>*<NN>}'`
- `'VP: {<DT>?<JJ.?>*<NN><VB.?>((<RB.?>)|(<DT>?<JJ.?>*<NN>)|(<IN><DT>?<JJ.?>*<NN>))*}'`

Method: Syntactic Analysis (Code)

```
np_chunk_grammar = 'NP: {<DT>?<JJ.?>*<NN>}'
    np_chunk_parser = RegexpParser(np_chunk_grammar)

    vp_chunk_grammar = 'VP:
{<DT>?<JJ.?>*<NN><VB.?>((<RB.?>)|(<DT>?<JJ.?>*<NN>)|(<IN><DT>?<JJ.?>*<NN>))*}'
    vp_chunk_parser = RegexpParser(vp_chunk_grammar)

    np_chunked_vader = list()
    vp_chunked_vader = list()

    for sentence in pos_tagged_vader:
        np_chunked_vader.append(np_chunk_parser.parse(sentence))
        vp_chunked_vader.append(vp_chunk_parser.parse(sentence))
```

Method: Vader's NP List (Output)

```
[Tree('S', [('where', 'WRB'), ('are', 'VBP'), ('those', 'DT'), ('transmissions', 'NNS'),
('you', 'PRP'), ('intercepted', 'VBN'), ('?', '.')]), Tree('S', [('if', 'IN'), ('this',
'DT'), ('is', 'VBZ'), Tree('NP', [('a', 'DT'), ('consular', 'JJ'), ('ship', 'NN')]),
('...', ':'), ('where', 'WRB'), ('is', 'VBZ'), Tree('NP', [('the', 'DT'), ('ambassador',
'NN')]), ('?', '.')]), Tree('S', [Tree('NP', [('commander', 'NN')]), Tree('NP', [('tear',
'NN')]), ('this', 'DT'), ('ship', 'JJ'), ('apart', 'RB'), ('until', 'IN'), ('you', 'PRP'),
("'ve", 'VBP'), ('found', 'VBN'), ('those', 'DT'), ('plans', 'NNS'), ('and', 'CC'),
('bring', 'VB'), ('me', 'PRP'), Tree('NP', [('the', 'DT'), ('ambassador', 'NN')]), ('.',
'.')]), Tree('S', [Tree('NP', [('i', 'NN')]), ('want', 'VBP'), ('her', 'PRP$'), ('alive',
'JJ'), ('!', '.')]), Tree('S', [('do', 'VBP'), ("n't", 'RB'), ('play', 'VB'), ('games',
'NNS'), ('with', 'IN'), ('me', 'PRP'), ('your', 'PRP$'), Tree('NP', [('highness', 'NN')]),
('.', '.')]), Tree('S', [('you', 'PRP'), ('were', 'VBD'), ("n't", 'RB'), ('on', 'IN'),
Tree('NP', [('any', 'DT'), ('mercy', 'JJ'), ('mission', 'NN')]), Tree('NP', [('this',
'DT'), ('time', 'NN')]), ('.', '.')]), Tree('S', [('you', 'PRP'), ('passed', 'VBN'),
('directly', 'RB'), ('through', 'IN'), Tree('NP', [('a', 'DT'), ('restricted', 'JJ'),
('system', 'NN')]), ('.', '.')])] etc.
```


Your eyes can deceive you, don't trust them.

Method: Making sense of the VPs and NPs

- Count most common VP and NP chunks for Vader and Luke respectively
- Counter functions that output top 10 of most common grammar chunks

Method: Top NPs and VPs Counters (Code)

Top NPs Counter

```
def np_chunk_counter(chunked_sentences):

    chunks = list()

    for chunked_sentence in chunked_sentences:
        for subtree in
chunked_sentence.subtrees(filter=lambda t:
t.label() == 'NP'):
            chunks.append(tuple(subtree))

    chunk_counter = Counter()

    for chunk in chunks:
        chunk_counter[chunk] += 1

    return chunk_counter.most_common(10)
```

Top VPs Counter

```
def vp_chunk_counter(chunked_sentences):

    chunks = list()

    for chunked_sentence in chunked_sentences:
        for subtree in
chunked_sentence.subtrees(filter=lambda t:
t.label() == 'VP'):
            chunks.append(tuple(subtree))

    chunk_counter = Counter()

    for chunk in chunks:
        chunk_counter[chunk] += 1

    return chunk_counter.most_common(10)
```

Method: Vader's Top NPs and VPs (Output)

Vader's Top NPs

```
top_np_chunks =
np_chunk_counter(np_chunked_vader)

>>> Vader's most-commonly used noun-phrases:
((('i', 'NN'),), 11)
((('the', 'DT'), ('force', 'NN')), 5)
((('the', 'DT'), ('ambassador', 'NN')), 2)
((('highness', 'NN'),), 2)
((('this', 'DT'), ('time', 'NN')), 2)
((('this', 'DT'), ('ship', 'NN')), 2)
((('the', 'DT'), ('rebellion', 'NN')), 2)
((('the', 'DT'), ('end', 'NN')), 2)
((('stay', 'NN'),), 2)
((('formation', 'NN'),), 2)
```

Vader's Top VPs

```
top_vp_chunks = vp_chunk_counter(vp_chunked_vader)

>>> Vader's most-commonly used verb-phrases:
((('i', 'NN'), ('want', 'VBP')), 2)
((('the', 'DT'), ('force', 'NN'), ('is', 'VBZ')), 2)
((('all', 'DT'), ('aboard', 'NN'), ('were', 'VBD')), 1)
((('a', 'DT'), ('planet', 'NN'), ('is', 'VBZ')), 1)
((('probe', 'NN'), ('is', 'VBZ')), 1)
((('i', 'NN'), ('told', 'VBD')), 1)
((('i', 'NN'), ('want', 'VBP'), ('every', 'DT'), ('part',
'NN'), ('of', 'IN'), ('this', 'DT'), ('ship', 'NN')), 1)
((('i', 'NN'), ('have', 'VBP'), ("n't", 'RB')), 1)
((('i', 'NN'), ('felt', 'VBD')), 1)
((('obi-wan', 'NN'), ('is', 'VBZ'), ('here', 'RB')), 1)
```


Never tell me the odds.

Results and Discussion: Episode IV: "A New Hope"

Vader				Luke			
NPs	**#**	**VPs**	**#**	**NPs**	**#**	**VPs**	**#**
I	11	I want	2	I	76	I'm	8
the force	5	the force is	2	threepio	12	I've	5
the ambassador	2	all aboard were	1	all right	11	I was	4
highness	2	a planet is	1	uncle	10	I want	3
this time	2	probe is	1	look	9	I didn't	2
this ship	2	I told	1	something	8	I'm sorry	2
the rebellion	2	I want every part of this ship	1	a lot	6	I'm never	2
the end	2	I haven't	1	owen	6	I thought	2
stay	2	I felt	1	unit	5	I think	2
formation	2	Obi-Wan is here	1	hey	4	I'm not	2

Results and Discussion: Predictions about the Data

Ep. IV: *A New Hope*

- Difference doesn't surprise
- Viewers aren't suspecting that the two characters are related

Ep. V: *The Empire Strikes Back*

- Similarities would be helpful in heightening the tension, complicating the plot, and adding to the film.

Ep. VI: *Return of the Jedi*

- The relationship had been revealed.
- Luke is starting to resemble his father.
- The final confrontation between the father and the son.


No!

Episode IV: A New Hope (new_hope.py)				Episode V: The Empire Strikes Back (empire_strikes_back.py)				Episode VI: Return of the Jedi (return_of_the_jedi.py)			
Vader		Luke		Vader		Luke		Vader		Luke	
NPs (#)	VPs (#)	NPs (#)	VPs (#)	NPs (#)	VPs (#)	NPs (#)	VPs (#)	NPs (#)	VPs (#)	NPs (#)	VPs (#)
I (11)	I want (2)	I (76)	I'm (8)	I (7)	I am (2)	I (45)	I'm (6)	master (7)	the emperor has (2)	I (45)	I've (5)
the force (5)	the force is (2)	threepio (12)	I've (5)	skywalker (5)	sure Skywalker is (1)	Ben (11)	I've (4)	the emperor (6)	I'm here (1)	father (13)	I have (3)
the ambassador (2)	all aboard were (1)	all right (11)	I was (4)	captain (4)	admiral Ozzel came (1)	Artoo (10)	I want (2)	I (5)	The emperor does not share (1)	Solo (5)	I know (2)
highness (2)	a planet is (1)	uncle (10)	I want (3)	the emperor (4)	nothing gets (1)	something (5)	I'm not (2)	the dark side (4)	The emperor is not as (1)	Leia (4)	I'm (2)
this time (2)	probe is (1)	look (9)	I didn't (2)	master (3)	I want that ship (1)	Leia (5)	I don't (2)	son (3)	I am (1)	Jabba (3)	I am Luke Skywalker Jedi knight (1)
this ship (2)	I told (1)	something (8)	I'm sorry (2)	Obi-Wan (3)	I want every ship (1)	Han (4)	old buddy do (1)	schedule (2)	the rebel fleet massing near sullust (1)	master (3)	I seek an audience (1)
the rebellion (2)	I want every part of this ship (1)	a lot (6)	I'm never (2)	destiny (3)	apology accepted captain Needa (1)	hey (4)	I know (1)	the power (2)	shuttle going (1)	Han (3)	wisdom I (1)
the end (2)	I haven't (1)	owen (6)	I thought (2)	father (3)	I have Skywalker (1)	Dagobah (4)	group use (1)	destiny (2)	force has (1)	Yoda (3)	goodwill I (1)
stay (2)	I felt (1)	unit (5)	I think (2)	Luke (3)	I had (1)	Dack (4)	I see (1)	Obi-Wan (2)	son is (1)	Vader (3)	I warn (1)
formation (2)	Obi-Wan is here (1)	hey (4)	I'm not (2)	the system (2)	this facility is (1)	hang (4)	no I'm not (1)	share (1)	i see (1)	captain (2)	I used (1)

Conclusions and Limitations

Conclusions

- No linguistic similarities between Luke and Vader
- Luke is becoming increasingly like his father in everything except the way in which he speaks.

Limitations

- Limited data / small corpus doesn't invite firm conclusions
- Luke tends to speak more than Vader, further skewing the data


Questions and Comments

Simon Zuberek

M.A. in Comp-Ling

simon@zuberek.net

GitHub Repo


May the force be with you.