eda_twitter_data

March 19, 2022

1 Projeto Análise da Popularidade dos Candidatos a Eleição de 2022 no Brasil

1.1 Pontos de análise:

- Total de tweets por candidato
- Media de tweets por dia por candidato
- Total de tweets com retweets por canditatos
- Total de tweets com likes por canditatos
- Assuntos mais citados dos candidatos com aplicação de modelos de análise de sentimentos:
  - **Modelo Bert**, ref.: https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment

    Description: This a bert-base-multilingual-uncased model finetuned for sentiment analysis on product reviews in six languages: English, Dutch, German, French, Spanish and Italian. It predicts the sentiment of the review as a number of stars (between 1 and 5). This model is intended for direct use as a sentiment analysis model for product reviews in any of the six languages above, or for further finetuning on related sentiment analysis tasks.
  - **Modelo Flair**, ref.: https://towardsdatascience.com/text-classification-with-state-of-the-art-nlp-library-flair-b541d7add21f

    Description: Flair delivers state-of-the-art performance in solving NLP problems such as named entity recognition (NER), part-of-speech tagging (PoS), sense disambiguation and text classification. It's an NLP framework built on top of PyTorch.

1.2 Analises Textuais

- Top assuntos mais citados nos tweets por sentimento por candidatos (Aplicar Trigrama)
- Top assuntos mais citados nos tweets por sentimento por candidatos (Aplicar LDA)
- Representatividade dos tweets positivos:
- % Tweets positivos
- % Retweets
- % Likes

- % Termos mais relevantes por candidato (Trigrama)
- % Tópicos mais relevantes por candidato (LDA)

[1]:
```
import pandas as pd
import numpy as np
import warnings
from helpers import (
    chart_tweets_trigrams,
    chart_tweets,
    chart_tweets_sentiment,
    nuvem_palavras,
    get_analysis_trigram_dataframe,
    get_dataframe_tokens,
    get_topicos,
    get_tweets_trigram,
    get_analysis_trigram,
    get_analysis_topics
)
import nltk
import spacy
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('rslp')
nltk.download('averaged_perceptron_tagger')
nlp = spacy.load("pt_core_news_lg")
```

```
[nltk_data] Downloading package punkt to /home/daholive/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /home/daholive/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/daholive/nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package rslp to /home/daholive/nltk_data…
[nltk_data]   Package rslp is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/daholive/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[2]:
```
warnings.filterwarnings('ignore')
```

[3]:
```
%matplotlib inline
```

[4]:
```
# dataframe tweets
dataframe = pd.read_parquet("/home/daholive/Documents/twitter_ellection_brazil/
 ↪datasource/tweets_preprocessing.parquet")
```

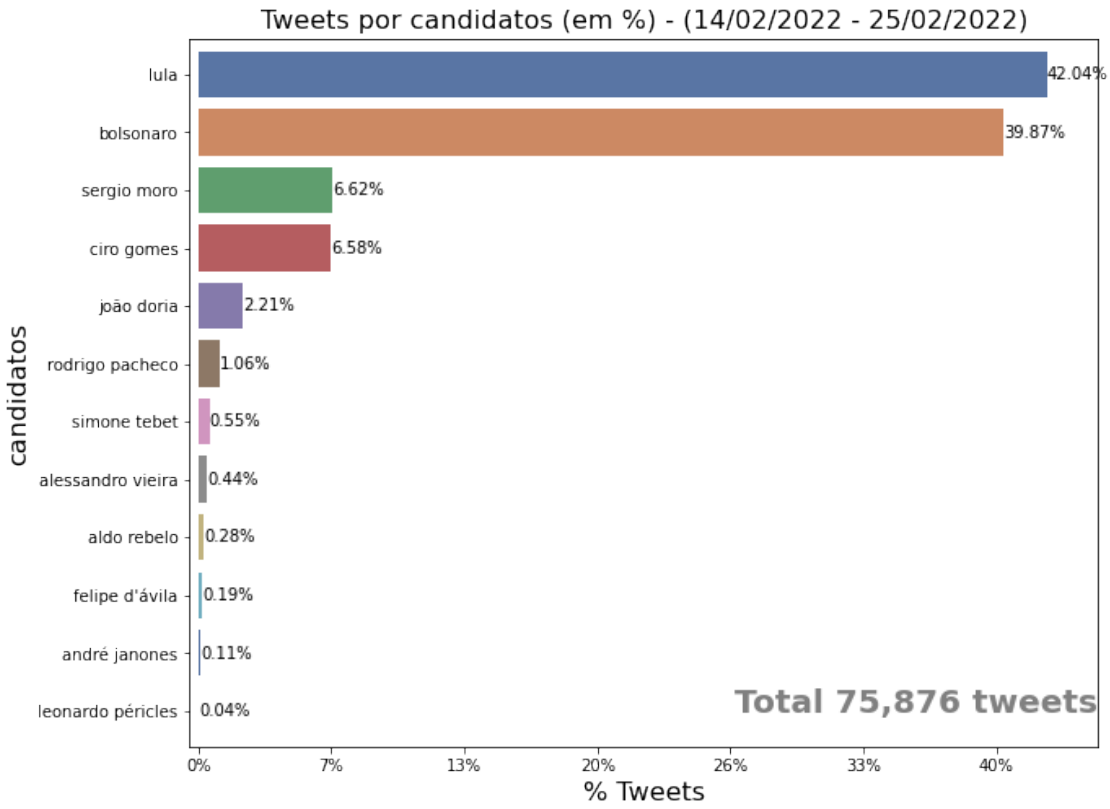
[5]:
```
# adjust to numeric format
dataframe['flair_sentiment_accuracy'] = pd.
 ↪to_numeric(dataframe['flair_sentiment_accuracy'],errors='coerce')
```

1.3 Tweets por candidato (em %)

[6]:
```
min_date = pd.to_datetime(dataframe["dated_at_tz"]).min().strftime("%d/%m/%Y")
max_date = pd.to_datetime(dataframe["dated_at_tz"]).max().strftime("%d/%m/%Y")

df1 = dataframe.groupby(["query"]).agg({
    "twitter_id": "nunique"
}).reset_index().rename(columns={"twitter_id":"qtde"}).sort_values(by=['qtde'],␣
 ↪ascending=False)

chart_tweets(
    df = df1,
    params = {
        "x": "query",
        "y": "qtde",
        "annotation_w":3e4,
        "annotation_h":8,
        "x_title": "% Tweets",
        "y_title": "candidatos",
        "title":f"Tweets por candidatos (em %) - ({min_date} - {max_date})"
    }
)
```

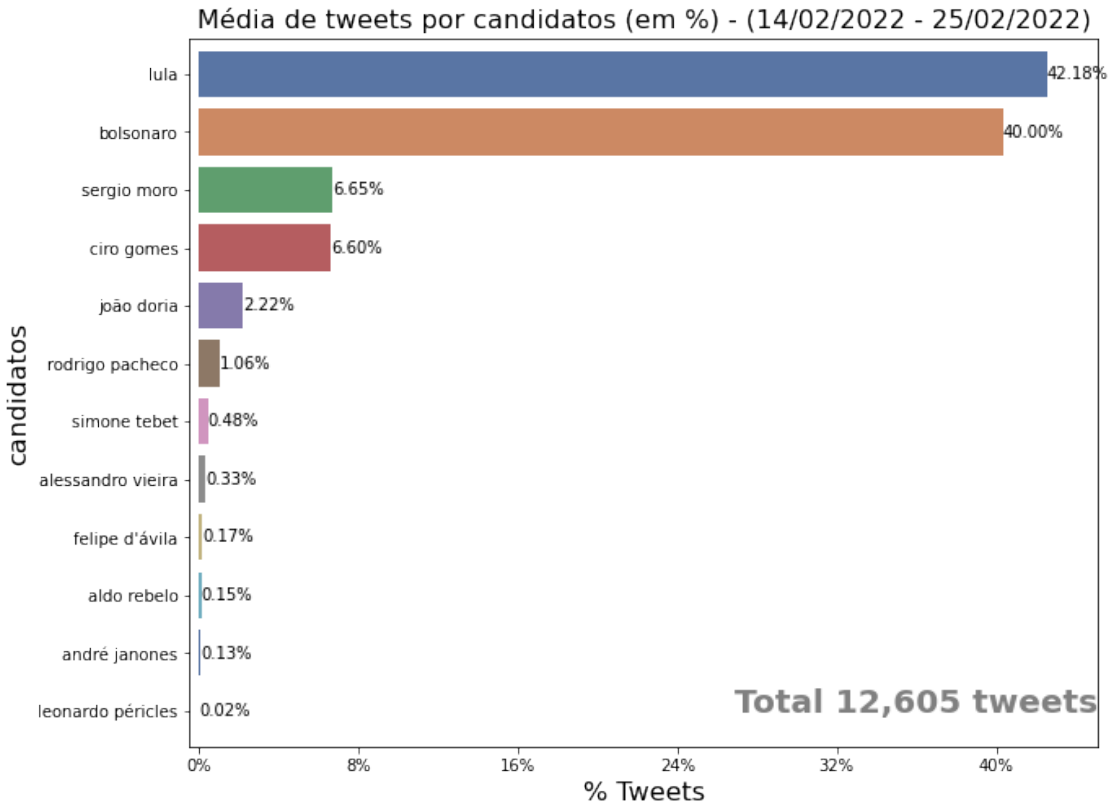


1.4 Média de tweets por candidato (em %)

[7]:
```
df2 = dataframe.groupby(["query","dated_at_tz"]).agg({
    "twitter_id": "nunique"
}).reset_index().rename(columns={"twitter_id":"qtde","dated_at_tz": "data"}).
  ↪groupby([
    "query"
]).agg({
    "qtde": lambda x: round(x.mean())
}).reset_index().sort_values(by=['qtde'], ascending=False)

chart_tweets(
    df = df2,
    params = {
        "x": "query",
        "y": "qtde",
        "annotation_w":5e3,
        "annotation_h":8,
        "x_title": "% Tweets",
        "y_title": "candidatos",
```

```
        "title":f"Média de tweets por candidatos (em %) - ({min_date} -␣
 ↪{max_date})"
    }
)
```

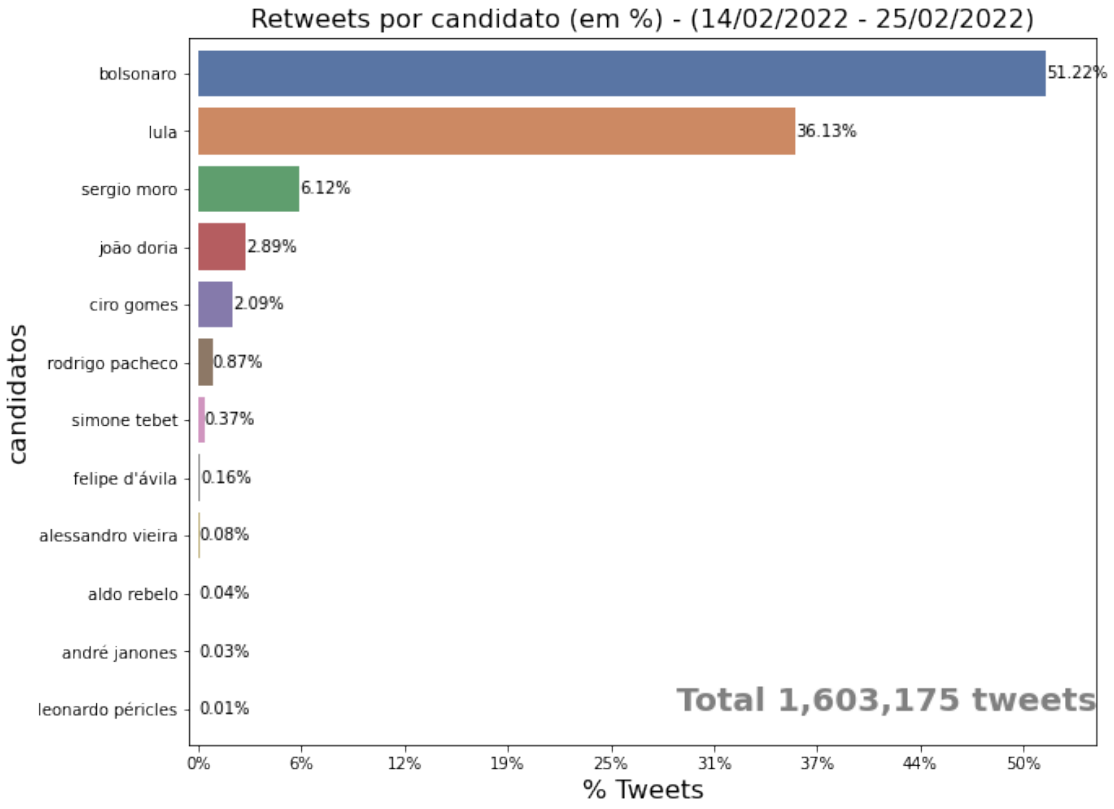


1.5 Média de retweets por candidato (em %)

[8]:
```
df3 = dataframe.groupby(["query"]).agg({
    "retweet_count": "sum"
}).reset_index().rename(columns={"retweet_count":"qtde"}).
 ↪sort_values(by=['qtde'], ascending=False)
```

[9]:
```
chart_tweets(
    df = df3,
    params = {
        "x": "query",
        "y": "qtde",
        "annotation_w":5e3,
        "annotation_h":8,
        "x_title": "% Tweets",
        "y_title": "candidatos",
```

```
        "title":f"Retweets por candidato (em %) - ({min_date} - {max_date})"
    }
)
```

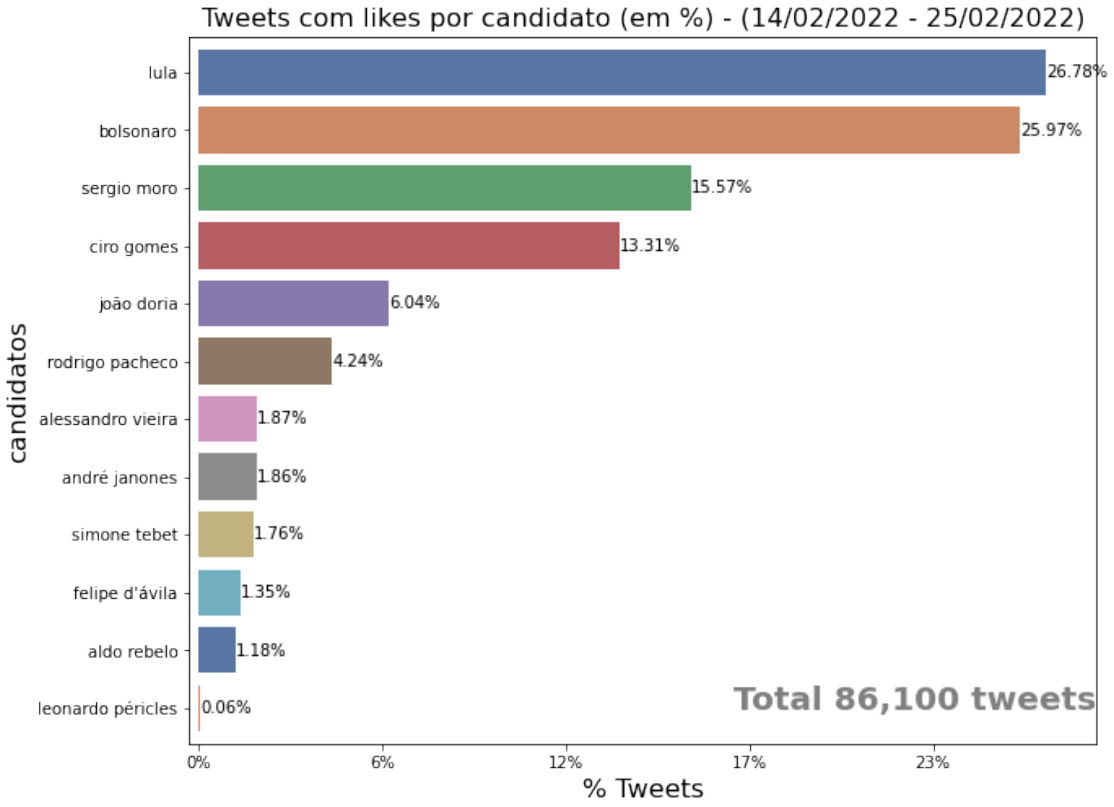


1.6 Média de likes por candidato (em %)

[10]:
```
df4 = dataframe.groupby(["query"]).agg({
    "like_count": "sum"
}).reset_index().rename(columns={"like_count":"qtde"}).sort_values(by=['qtde'],␣
↪ascending=False)
```

[11]:
```
chart_tweets(
    df = df4,
    params = {
        "x": "query",
        "y": "qtde",
        "annotation_w":5e3,
        "annotation_h":8,
        "x_title": "% Tweets",
        "y_title": "candidatos",
```

```
        "title":f"Tweets com likes por candidato (em %) - ({min_date} -␣
 ↪{max_date})"
    }
)
```



1.7 Exploratory data analysis - Bert Model

Classificação dos Tweets utilizando modelo Bert (https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment)

[12]:
```
dataframe["bert_sentiment_level_text"].value_counts()
```

[12]:
```
VERY_NEGATIVE    40516
VERY_POSITIVE    16598
NEUTRAL           7982
POSITIVE          7071
NEGATIVE          3709
Name: bert_sentiment_level_text, dtype: int64
```

Total de Tweets classificados como "VERY_POSITIVE" a serem analisados

[13]:
```
total_tweets_unicos_positivos = dataframe[
    dataframe["bert_sentiment_level_text"]=="VERY_POSITIVE"
]["twitter_id"].nunique()
total_tweets_unicos_positivos
```

[13]: 16598

Distribuição dos Tweets classificados como "VERY_POSITIVE" por candidato

[14]:
```
dataframe[
    dataframe["bert_sentiment_level_text"]=="VERY_POSITIVE"
][["query","bert_sentiment_level_text"]].value_counts()
```

[14]:
```
query              bert_sentiment_level_text
lula               VERY_POSITIVE                6636
bolsonaro          VERY_POSITIVE                6412
ciro gomes         VERY_POSITIVE                1539
sergio moro        VERY_POSITIVE                1186
joão doria         VERY_POSITIVE                 444
rodrigo pacheco    VERY_POSITIVE                 130
aldo rebelo        VERY_POSITIVE                  78
simone tebet       VERY_POSITIVE                  65
alessandro vieira  VERY_POSITIVE                  61
felipe d'ávila     VERY_POSITIVE                  24
andré janones      VERY_POSITIVE                  13
leonardo péricles  VERY_POSITIVE                  10
dtype: int64
```

Total de autores dos Tweets

[15]:
```
total_tweets_autores_unicos_positivos = dataframe[
    dataframe["bert_sentiment_level_text"]=="VERY_POSITIVE"
]["author_id"].nunique()
total_tweets_autores_unicos_positivos
```

[15]: 11920

Tabela analítica dos percentuais de tweets, retweets, likes e usuários ativos (autores) por candidato em percenutal

[16]:
```
# percentual de tweets positivos por candidato
dataframe[
    dataframe["bert_sentiment_level_text"]=="VERY_POSITIVE"
].groupby(["query"]).agg({
    "twitter_id": lambda x: round(len(x)/total_tweets_unicos_positivos,4)*100,
    "have_retweet": lambda x: round(sum(x)/total_tweets_unicos_positivos,4)*100,
    "have_like": lambda x: round(sum(x)/total_tweets_unicos_positivos,4)*100,
```

```
    "author_id": lambda x: round(len(np.unique(x))/
 ↪total_tweets_autores_unicos_positivos,4)*100
}).reset_index().rename(
    columns={
        "twitter_id":"perc_tweets",
        "have_retweet": "perc_retweets",
        "have_like": "perc_like",
        "author_id": "perc_active_users",
    }
).sort_values(by = "perc_tweets", ascending=False)
```

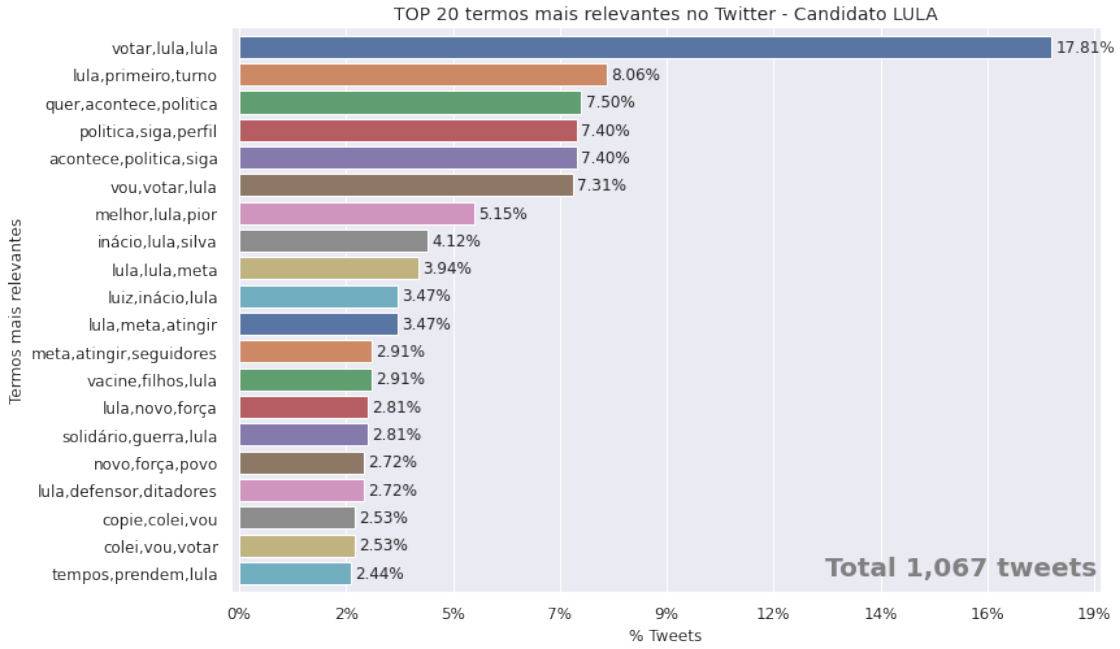
[16]:

	query	perc_tweets	perc_retweets	perc_like	perc_active_users
8	lula	39.98	12.93	6.48	43.17
3	bolsonaro	38.63	14.51	5.04	45.48
4	ciro gomes	9.27	4.63	3.40	6.23
10	sergio moro	7.15	3.40	1.98	6.53
6	joão doria	2.68	1.14	0.88	2.54
9	rodrigo pacheco	0.78	0.33	0.16	0.95
0	aldo rebelo	0.47	0.32	0.25	0.52
11	simone tebet	0.39	0.17	0.17	0.48
1	alessandro vieira	0.37	0.16	0.15	0.50
5	felipe d'ávila	0.14	0.08	0.04	0.18
2	andré janones	0.08	0.03	0.04	0.10
7	leonardo péricles	0.06	0.02	0.02	0.07

Função GET_ANALYSIS_TRIGRAM Função que consolida outras funções em um único pipeline de análise: - Processa os termos mais citados, através da identificação dos trigramas com melhor SCORE, aplicando o método estatístico Teste-t de Student - Gera gráfico dos TOP 20 termos mais relevantes - Gera Wordcloud dos termos

Função aplicada para o candidato LULA

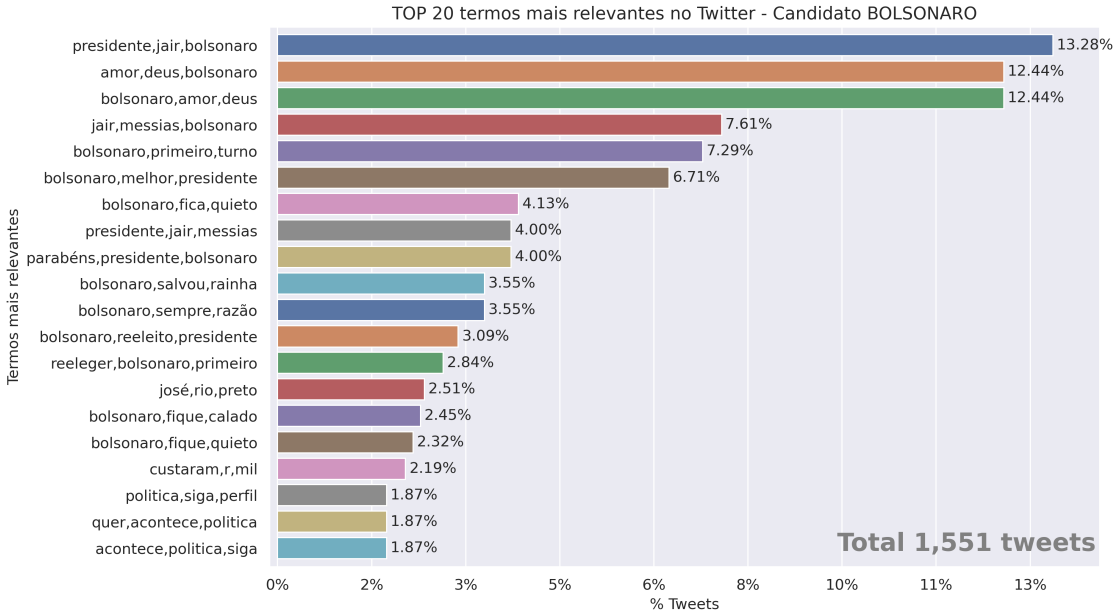
[17]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="lula",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=True
)
```




```
Finished: 59.648357629776
```

Função aplicada para o candidato BOLSONARO

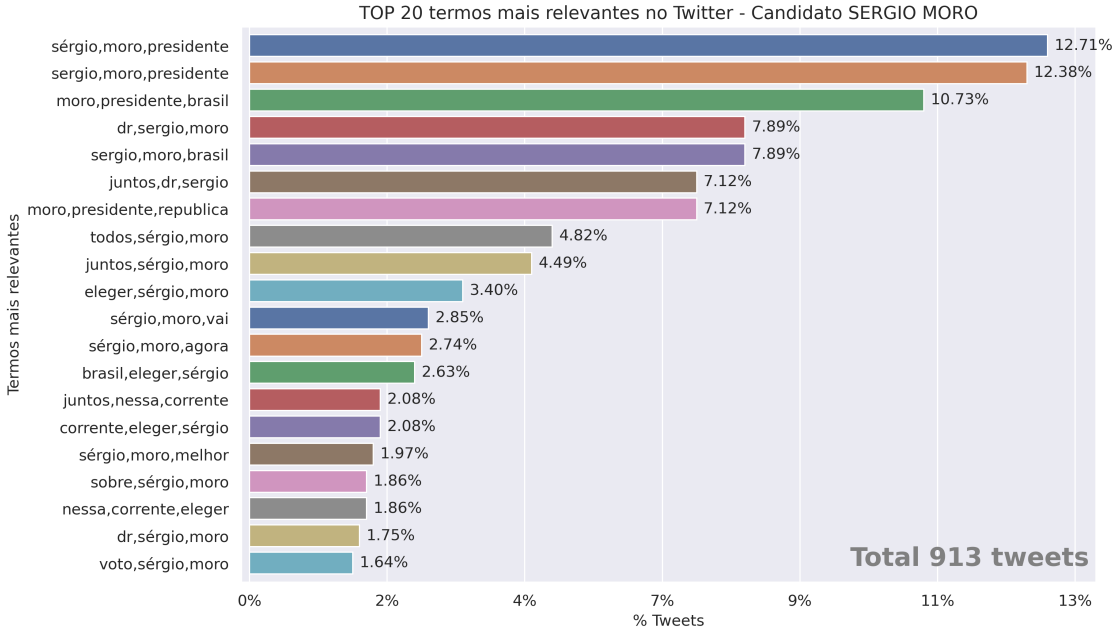
[18]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="bolsonaro",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 54.2503023147583
```

Função aplicada para o candidato SERGIO MORO

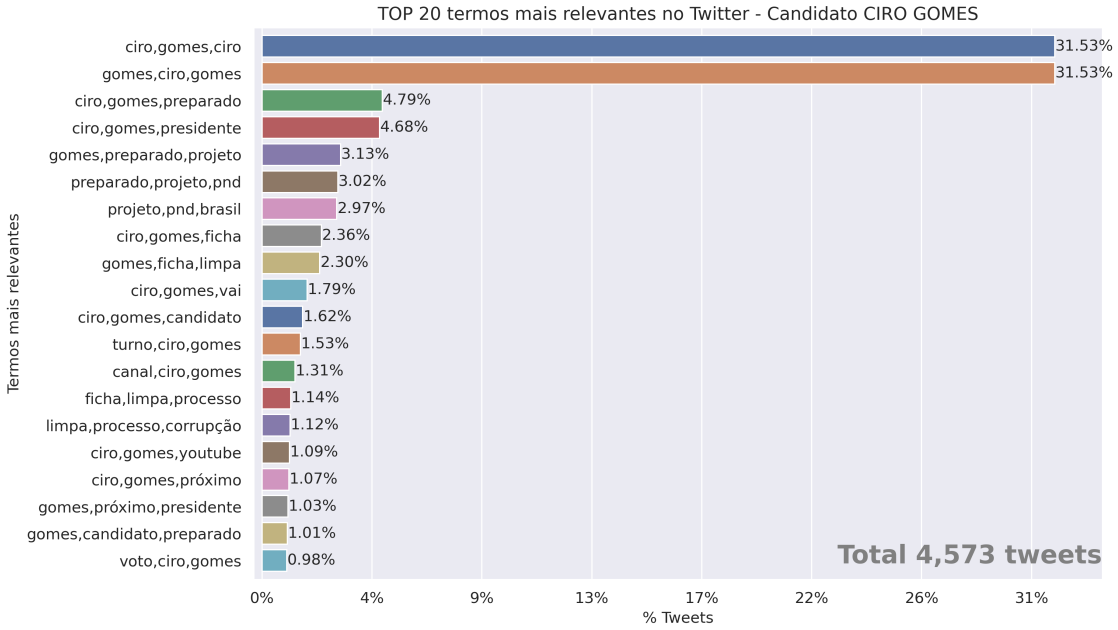
[19]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="sergio moro",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 9.843285083770752
```

Função aplicada para o candidato CIRO GOMES

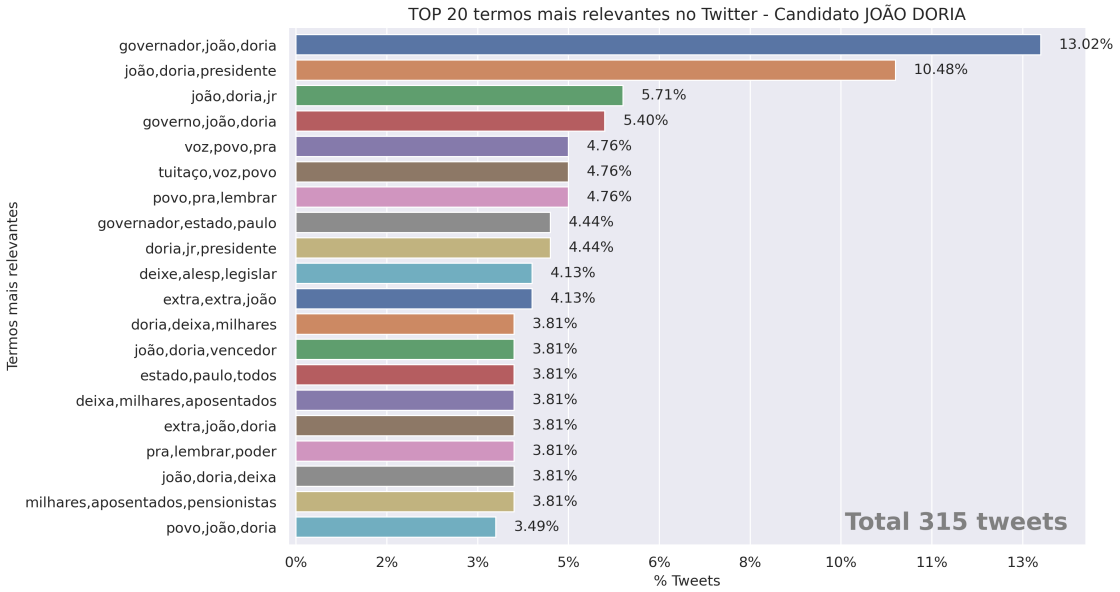
[20]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="ciro gomes",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 11.95229196548462
```

Função aplicada para o candidato JOÃO DORIA

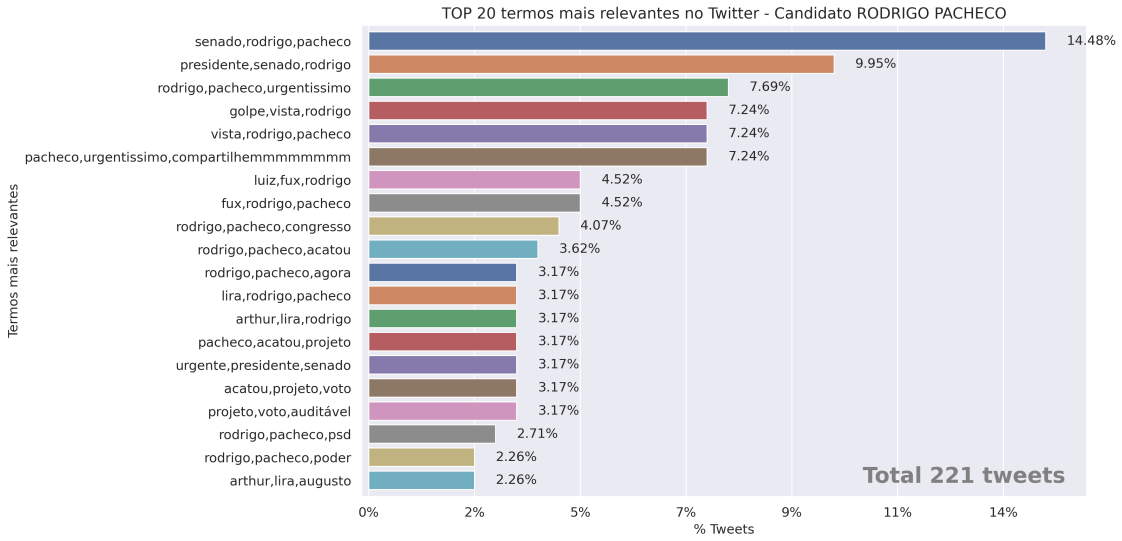
[21]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="joão doria",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 4.405893802642822
```

Função aplicada para o candidato RODRIGO PACHECO

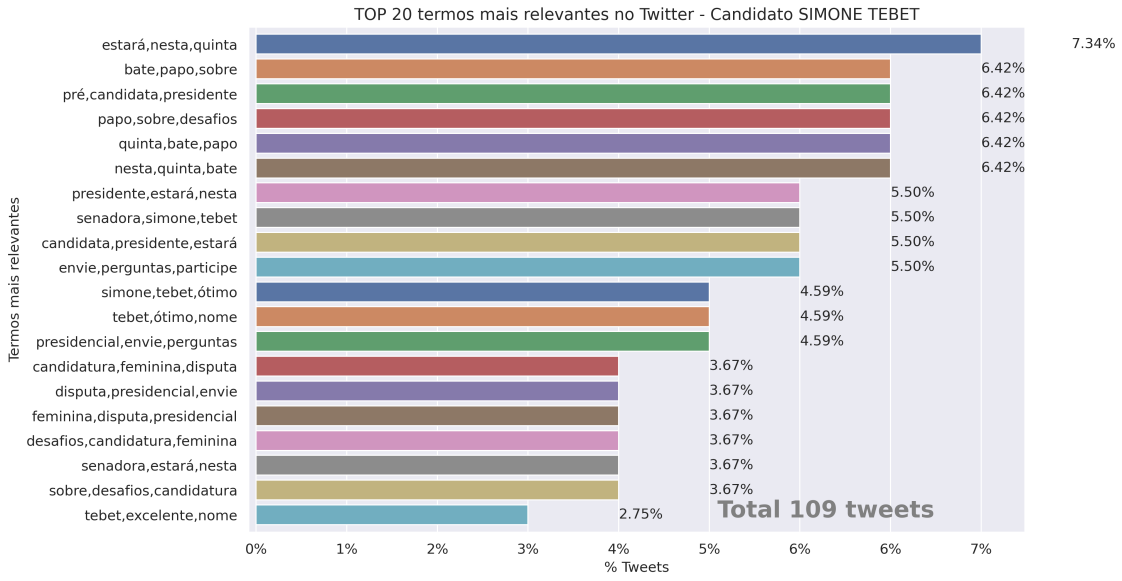
[22]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="rodrigo pacheco",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.98860502243042
```

Função aplicada para o candidato SIMONE TEBET

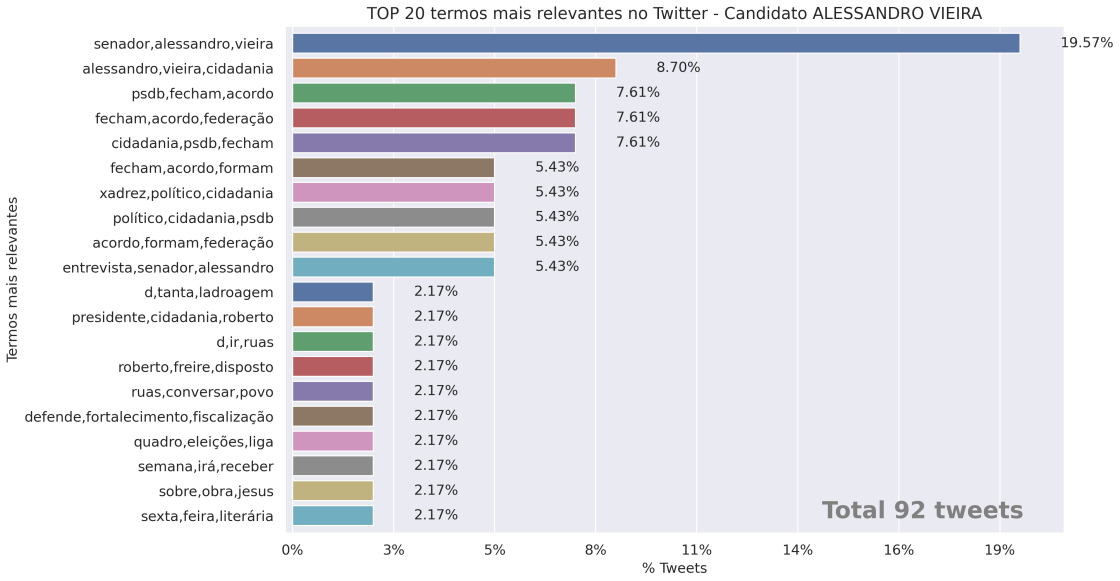
[23]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="simone tebet",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```



```
Finished: 1.393446922302246
```

Função aplicada para o candidato ALESSANDRO VIEIRA

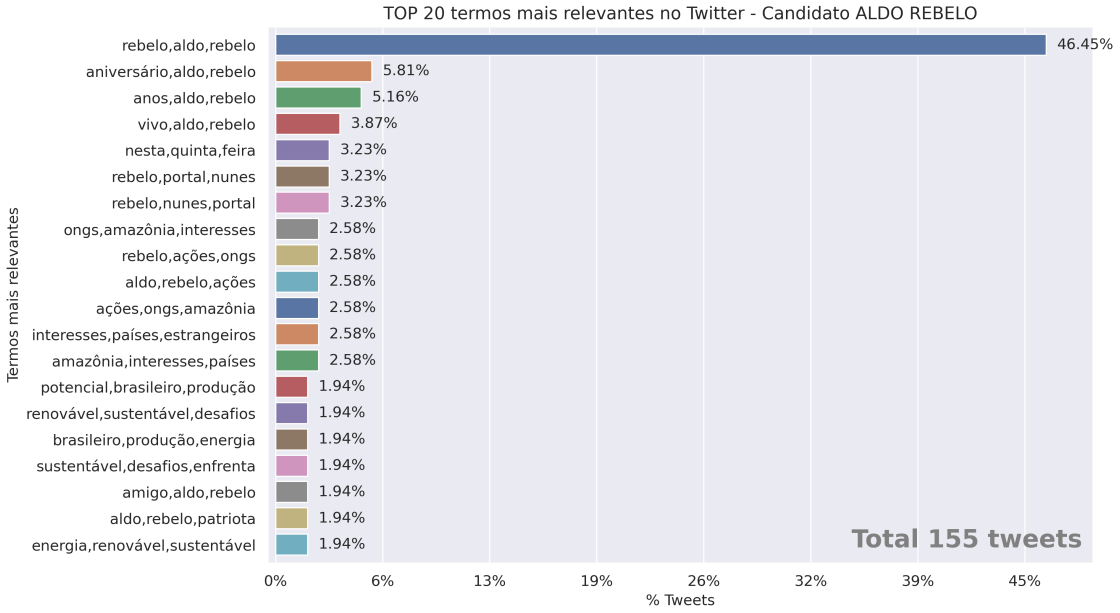
[24]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="alessandro vieira",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.3455677032470703
```

Função aplicada para o candidato ALDO RABELO

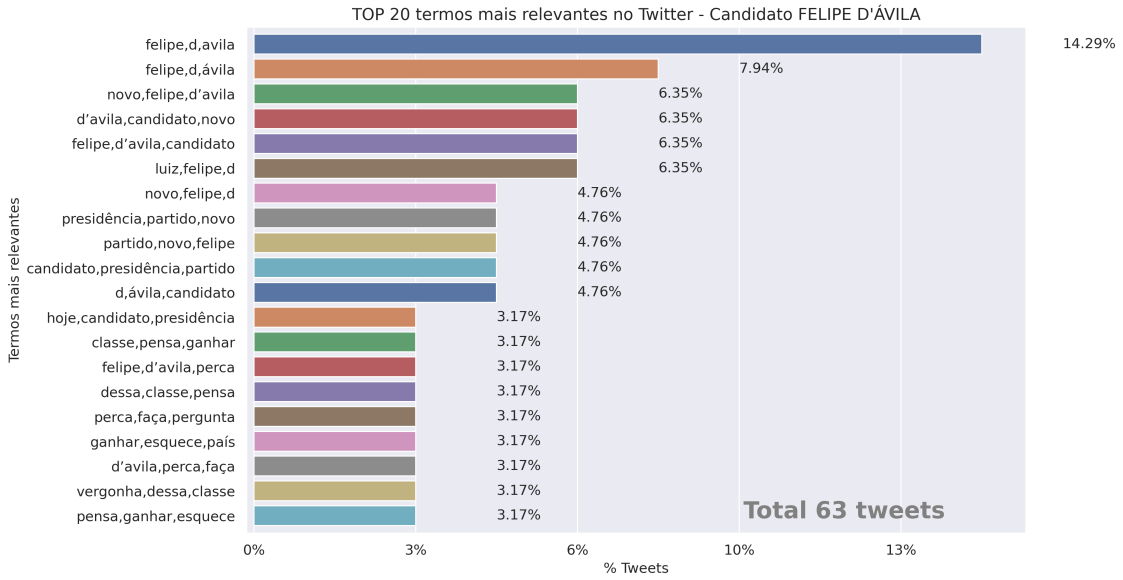
[25]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="aldo rebelo",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.557093858718872
```

Função aplicada para o candidato FELIPE D'AVILA

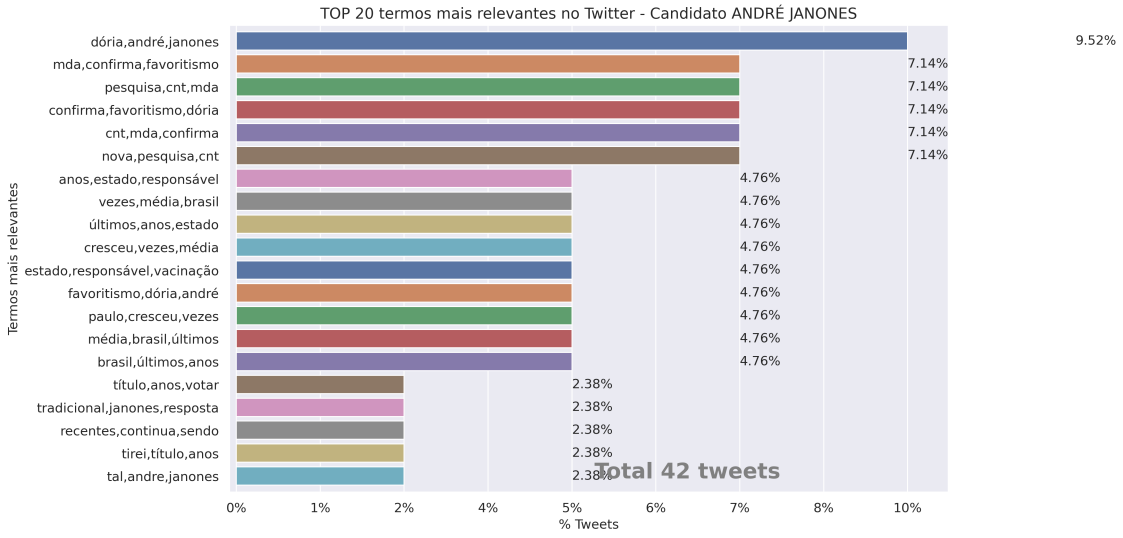
[26]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="felipe d'ávila",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```



Finished: 1.0341143608093262

Função aplicada para o candidato ANDRÉ JANONES

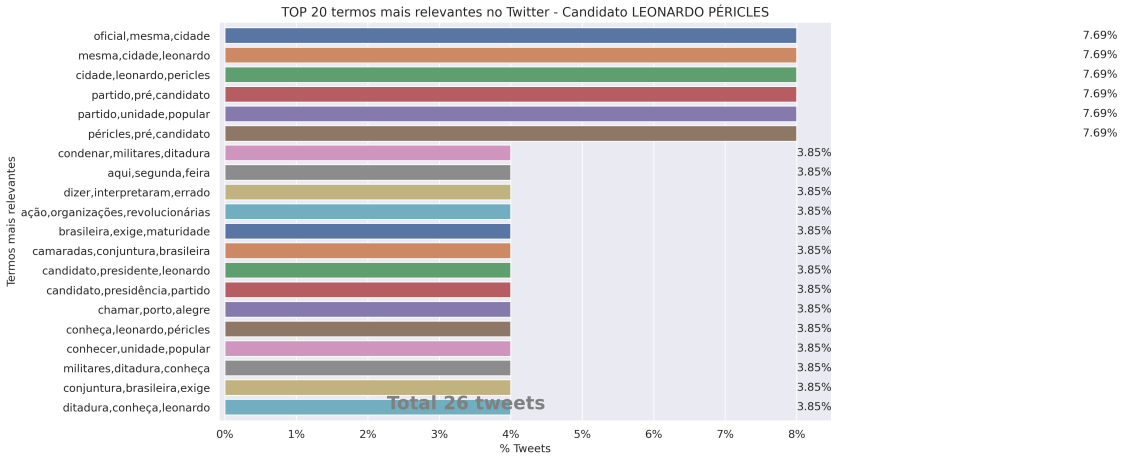
[27]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="andré janones",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```



```
Finished: 1.0565361976623535
```

Função aplicada para o candidato LEONARDO PÉRICLES

[28]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="bert",
    candidate="leonardo péricles",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.1802122592926025
```

1.8 Exploratory data analysis - Bert FLAIR

Classificação dos Tweets utilizando modelo Flair (https://github.com/flairNLP/flair)

[29]:
```
dataframe["flair_sentiment_text"].value_counts()
```

[29]:
```
NEGATIVE    48466
POSITIVE    27410
Name: flair_sentiment_text, dtype: int64
```

Total de Tweets classificados como "POSITIVE" e acurácia do modelo maior ou igual a 90% a serem analisados

[30]:
```
total_tweets_unicos_positivos = dataframe[
    (dataframe["flair_sentiment_text"]=="POSITIVE") &
```

```
    (dataframe["flair_sentiment_accuracy"]>=0.90)
][["twitter_id"].nunique()
total_tweets_unicos_positivos
```

[30]: 18154

Distribuição dos Tweets classificados como "POSITIVE" por candidato

[31]:
```
dataframe[
    (dataframe["flair_sentiment_text"]=="POSITIVE") &
    (dataframe["flair_sentiment_accuracy"]>=0.90)
].groupby(["query"]).agg({
    "twitter_id": lambda x: round(len(x)/total_tweets_unicos_positivos,4)*100,
    "have_retweet": lambda x: round(sum(x)/total_tweets_unicos_positivos,4)*100,
    "have_like": lambda x: round(sum(x)/total_tweets_unicos_positivos,4)*100,
    "author_id": lambda x: round(len(np.unique(x))/
 ↪total_tweets_autores_unicos_positivos,4)*100
}).reset_index().rename(
    columns={
        "twitter_id":"perc_tweets",
        "have_retweet": "perc_retweets",
        "have_like": "perc_like",
        "author_id": "perc_active_users",
    }
).sort_values(by = "perc_tweets", ascending=False)
```

[31]:

	query	perc_tweets	perc_retweets	perc_like	perc_active_users
3	bolsonaro	37.67	13.58	5.22	47.23
8	lula	37.48	10.93	6.87	45.51
4	ciro gomes	12.76	6.04	4.80	8.62
10	sergio moro	6.49	2.89	1.83	6.59
6	joão doria	2.65	1.15	0.93	2.67
9	rodrigo pacheco	0.92	0.40	0.31	1.29
11	simone tebet	0.67	0.27	0.25	
1	alessandro vieira	0.54	0.23	0.25	
0	aldo rebelo	0.47	0.32	0.28	
5	felipe d'ávila	0.15	0.06	0.04	
2	andré janones	0.14	0.04	0.05	
7	leonardo péricles	0.06	0.02	0.02	

```
11              0.90
1               0.74
0               0.54
5               0.21
2               0.19
7               0.08
```

Total de autores dos Tweets

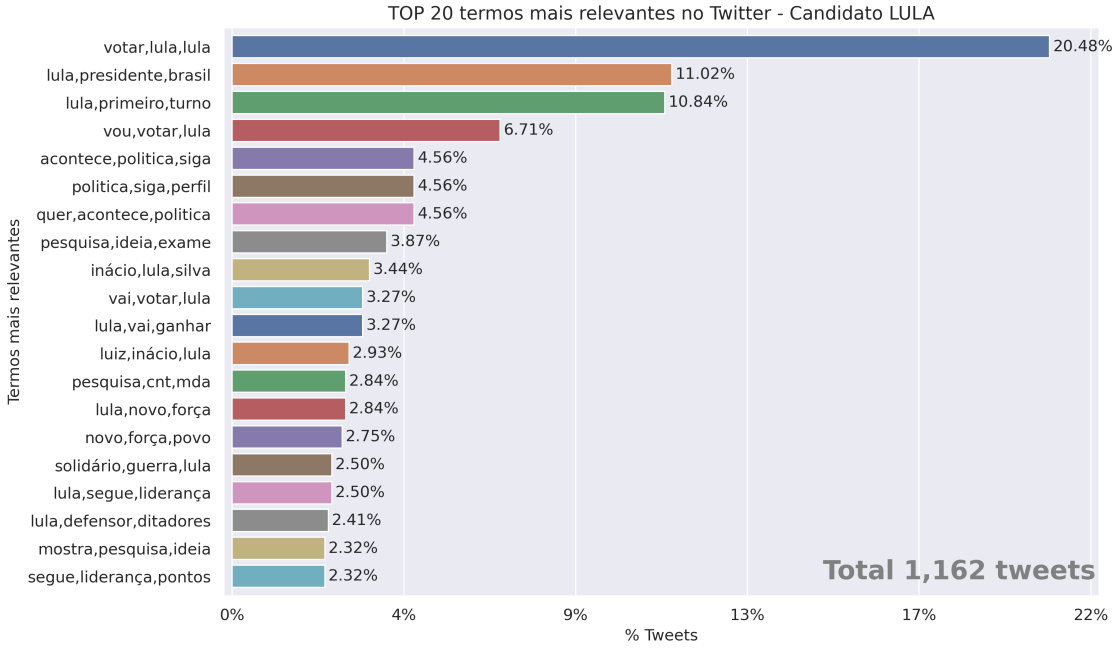
[32]:
```
total_tweets_autores_unicos_positivos = dataframe[
    (dataframe["flair_sentiment_text"]=="POSITIVE") &
    (dataframe["flair_sentiment_accuracy"]>=0.90)
]["author_id"].nunique()
total_tweets_autores_unicos_positivos
```

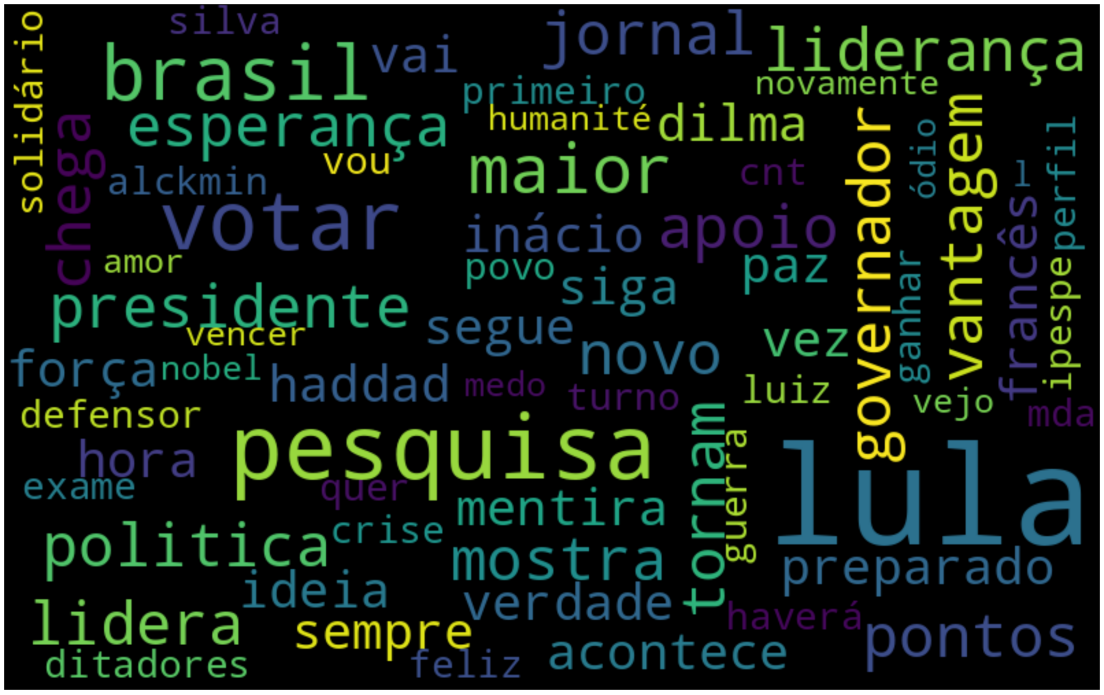
[32]: 12717

Função GET_ANALYSIS_TRIGRAM Função que consolida outras funções em um único pipeline de análise: - Processa os termos mais citados, através da identificação dos trigramas com melhor SCORE, aplicando o método estatístico Teste-t de Student - Gera gráfico dos TOP 20 termos mais relevantes - Gera Wordcloud dos termos

Função aplicada para o candidato LULA

[33]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="lula",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=True
)
```

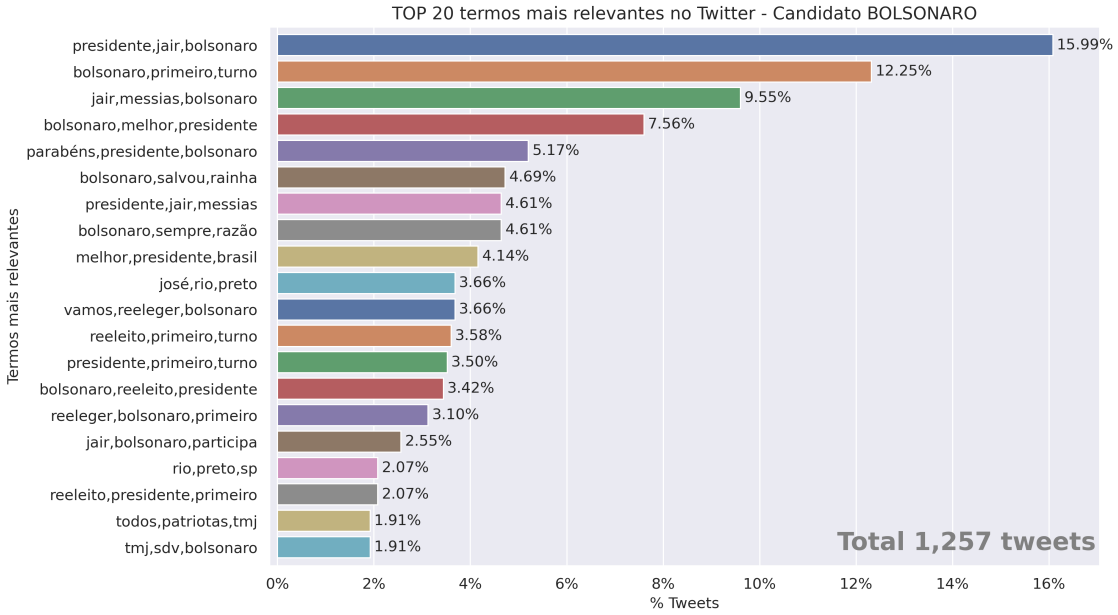





```
Finished: 60.22979664802551
```

Função aplicada para o candidato BOLSONARO

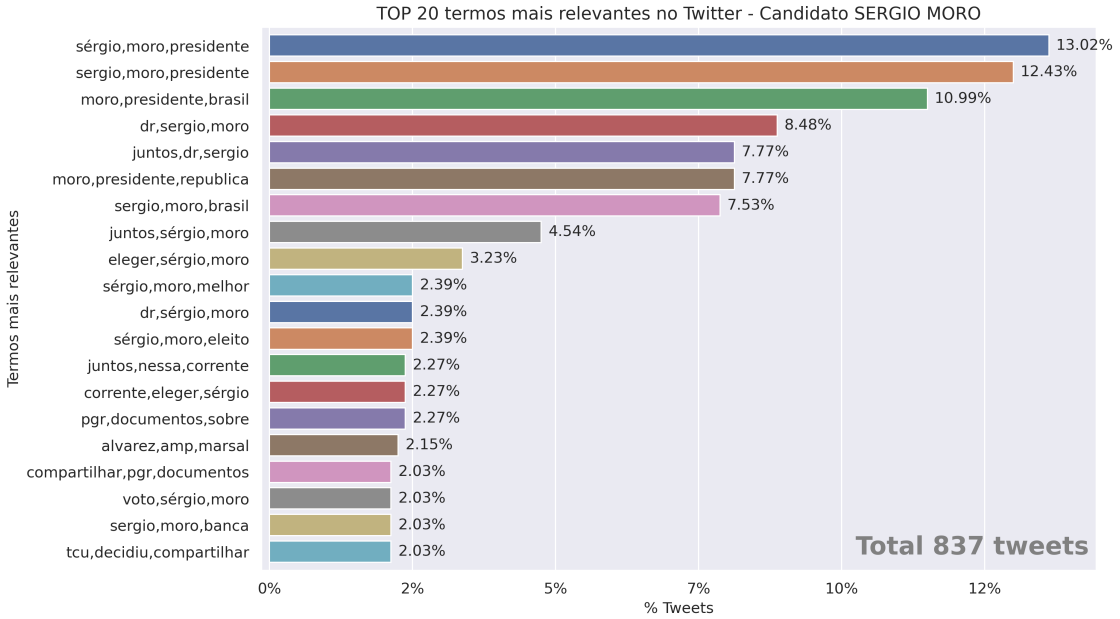
[34]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="bolsonaro",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 58.19952130317688
```

Função aplicada para o candidato SERGIO MORO

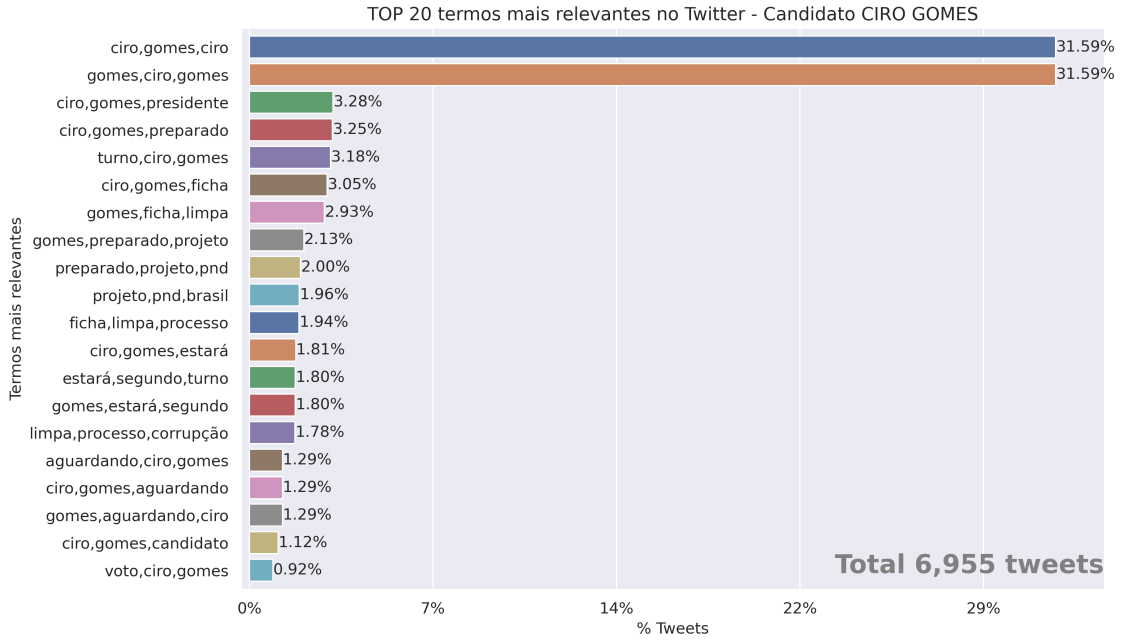
[35]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="sergio moro",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 9.953444957733154
```

Função aplicada para o candidato CIRO GOMES

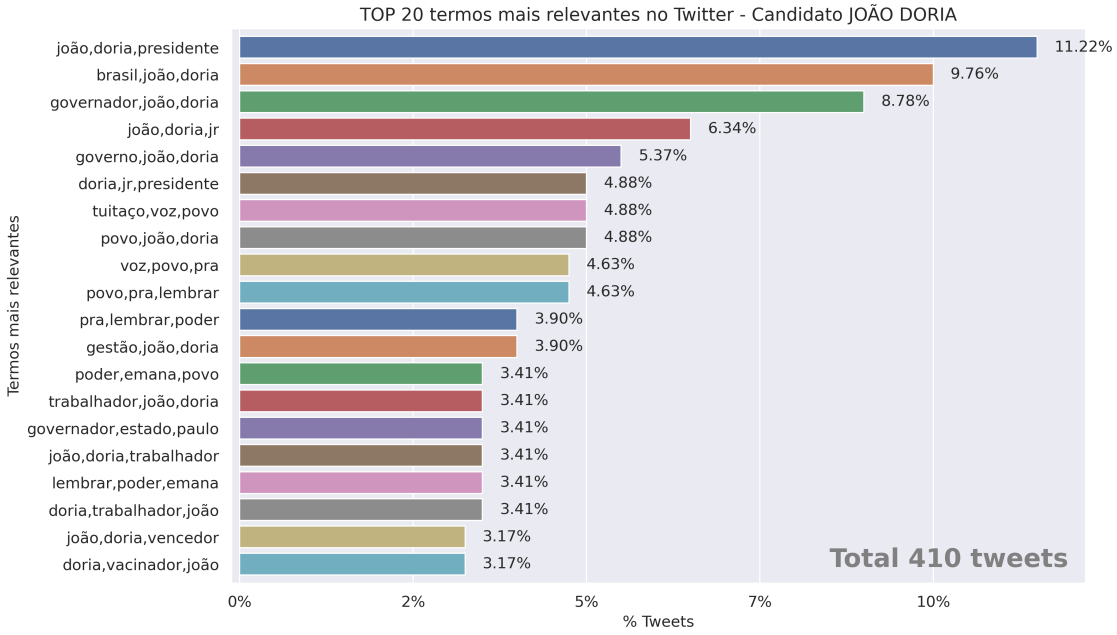
[36]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="ciro gomes",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 17.497204780578613
```

Função aplicada para o candidato JOÃO DORIA

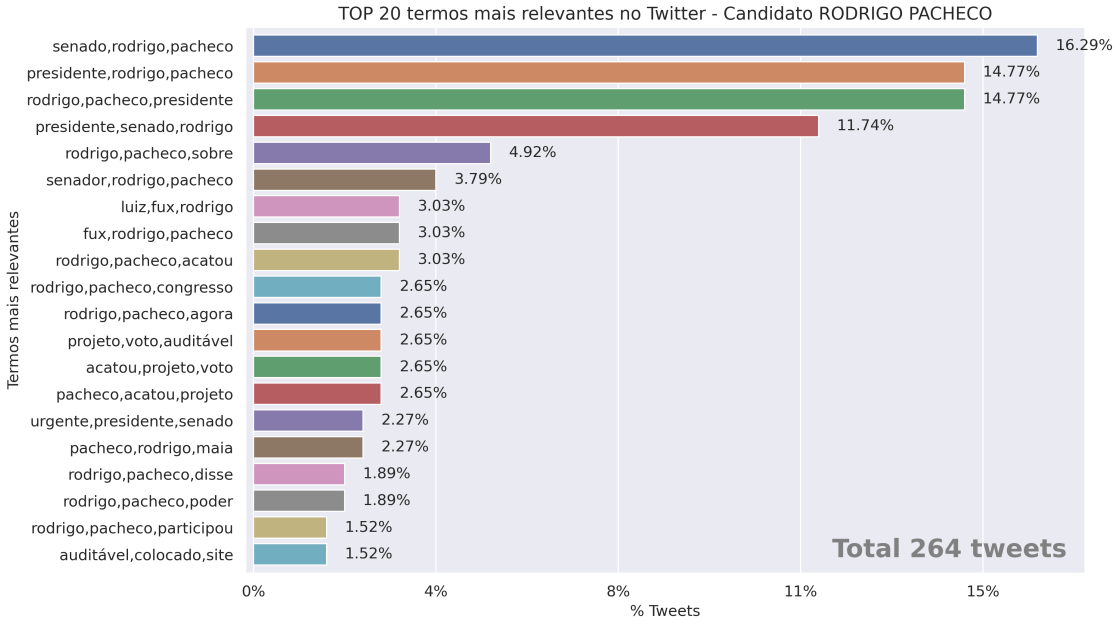
[37]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="joão doria",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 4.4265220165252686
```

Função aplicada para o candidato RODRIGO PACHECO

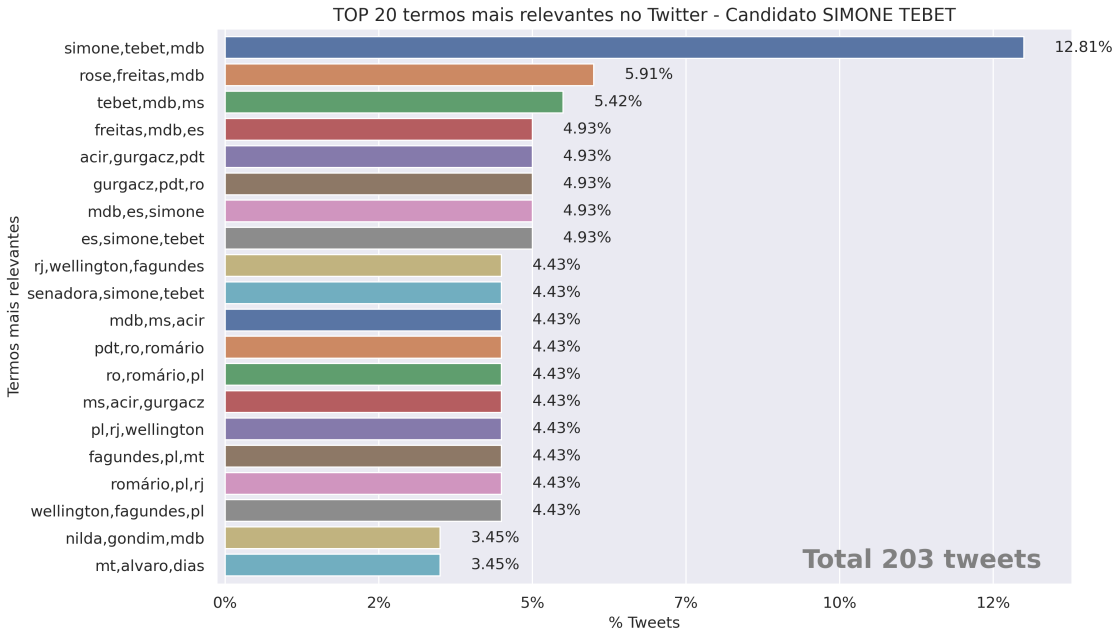
[38]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="rodrigo pacheco",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 2.297269344329834
```

Função aplicada para o candidato SIMONE TEBET

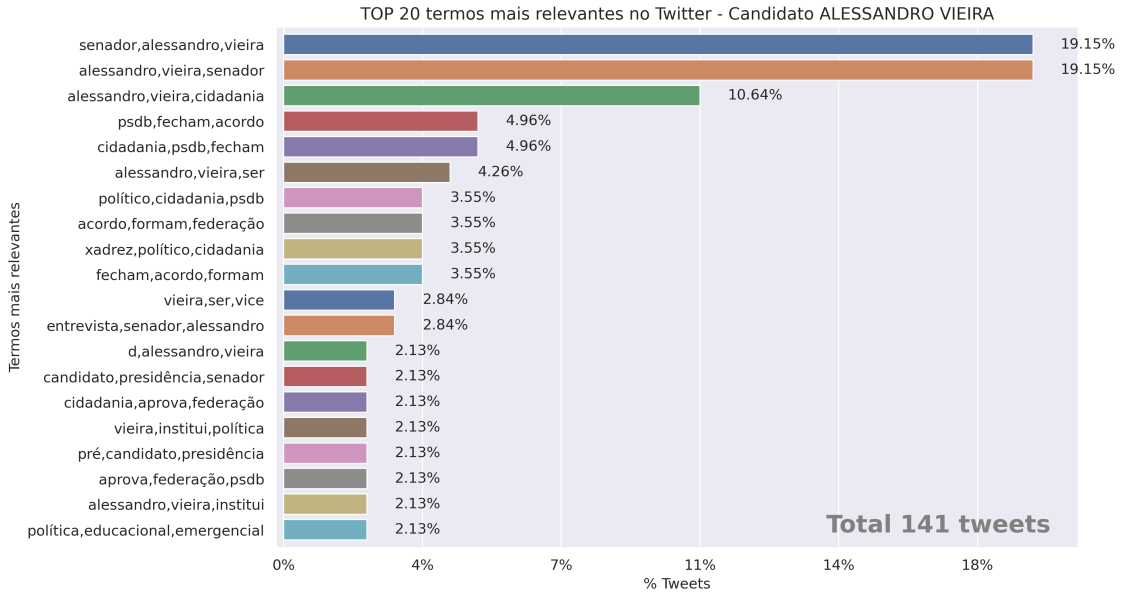
[39]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="simone tebet",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.8591887950897217
```

Função aplicada para o candidato ALESSANDRO VIEIRA

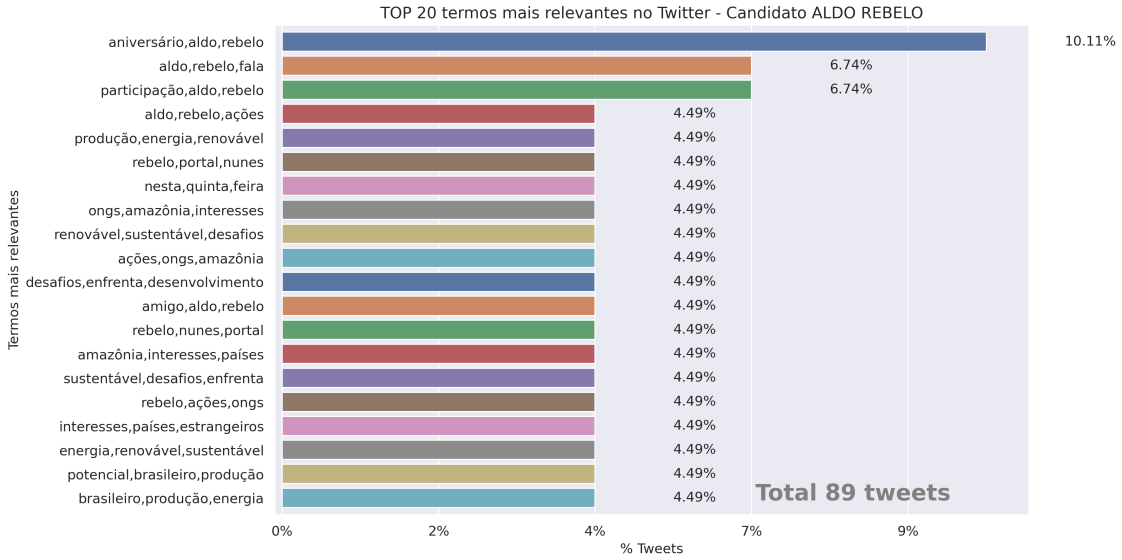
[40]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="alessandro vieira",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.6580369472503662
```

Função aplicada para o candidato ALDO RABELO

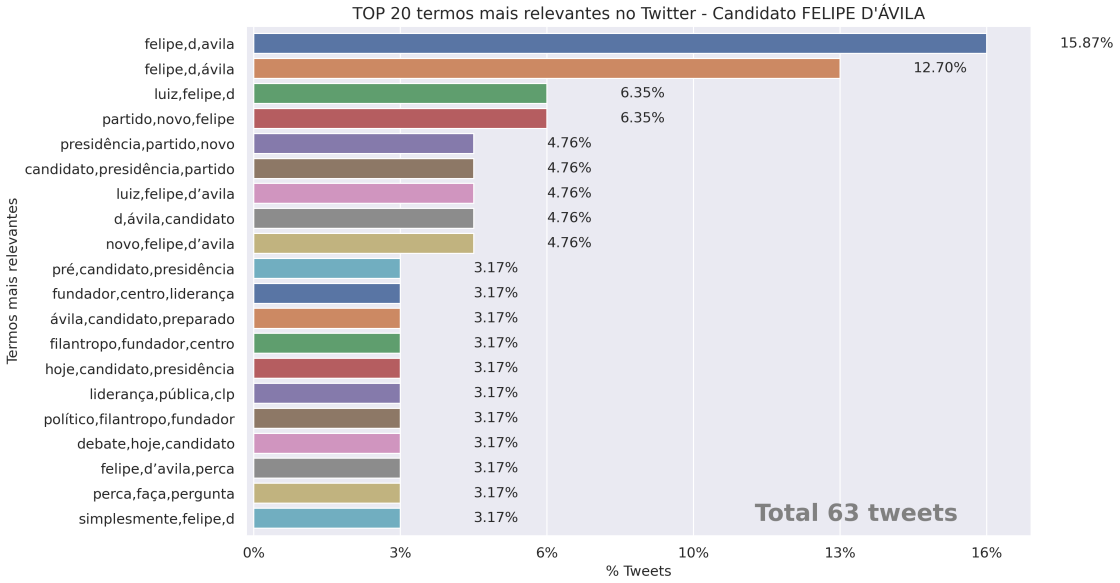
[41]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="aldo rebelo",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.5686683654785156
```

Função aplicada para o candidato FELIPE D'AVILA

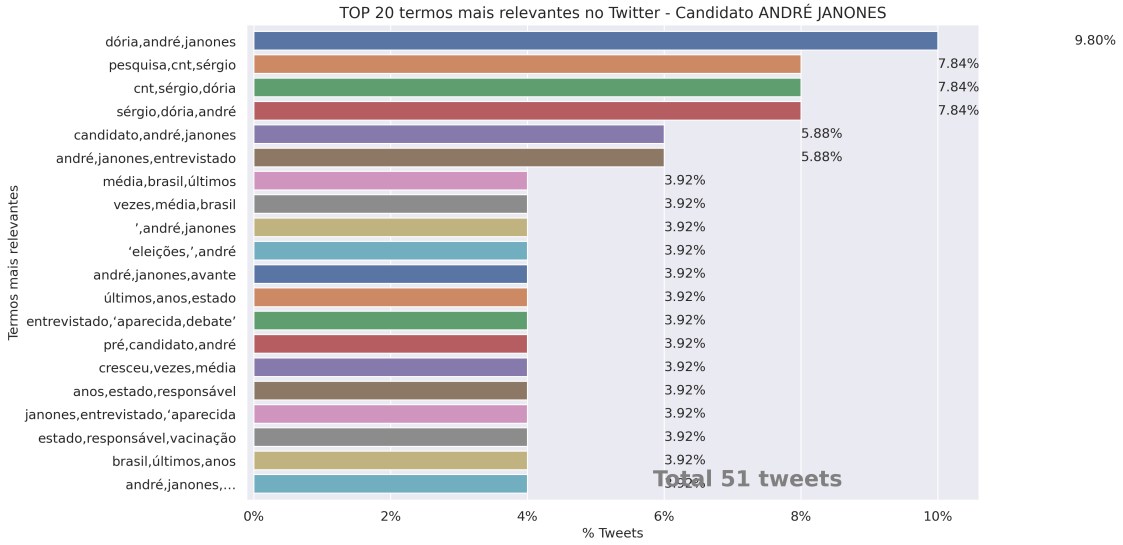
[42]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="felipe d'ávila",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.130528211593628
```

Função aplicada para o candidato ANDRÉ JANONES

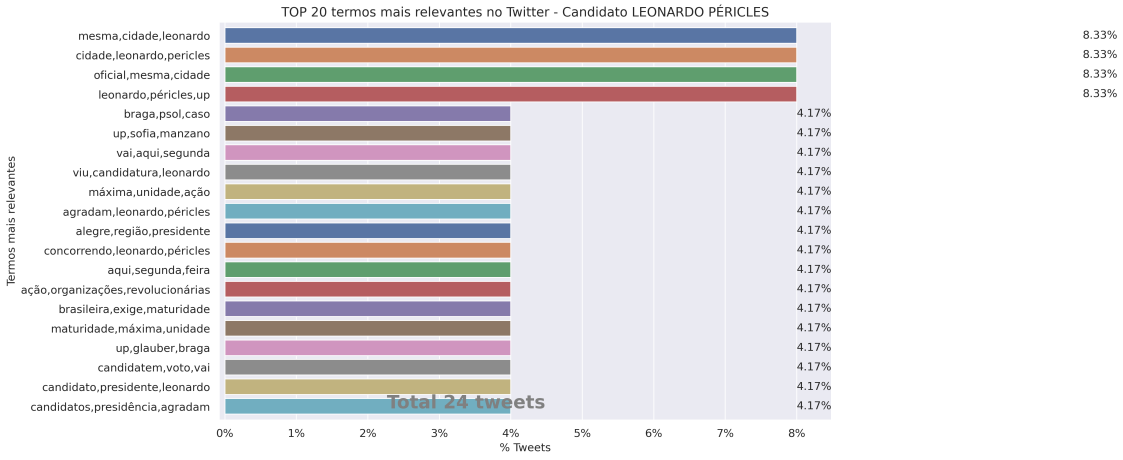
[43]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="andré janones",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```




```
Finished: 1.1406772136688232
```

Função aplicada para o candidato LEONARDO PÉRICLES

[44]:
```
get_analysis_trigram(
    dataframe=dataframe,
    model="flair",
    candidate="leonardo péricles",
    num_trigram=50,
    chart_limit_terms=20,
    model_stats="Teste-t",
    wordcloud=False
)
```



```
Finished: 1.1840524673461914
```

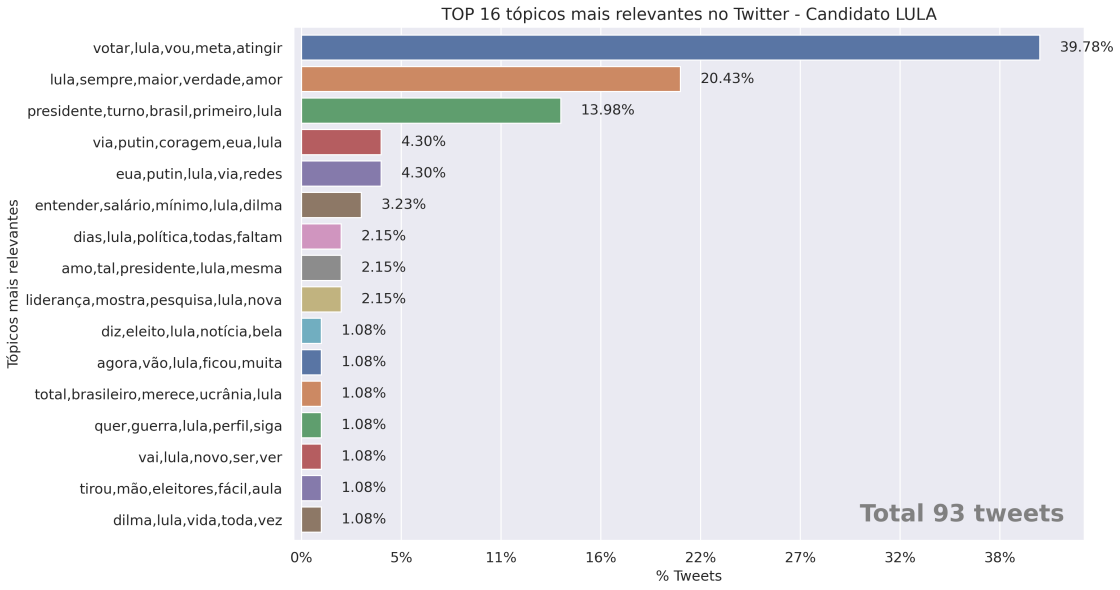
1.9 Exploratory data analysis - Análise de tópicos - Bert Model

Função GET_ANALYSIS_TOPICS Função que consolida outras funções em um único pipeline de análise: - Processa os tópicos mais citados, através da aplicação do modelo LDA (Latent Dirichlet allocation),que é um modelo estatístico generativo. Ele permite que conjuntos de observações sejam explicados por variáveis latentes que explicam por que algumas partes dos dados são semelhantes. - Gera gráfico dos TOP 20 tópicos mais relevantes - Gera Wordcloud dos tópicos

Função aplicada para o candidato Lula

[45]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="lula",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

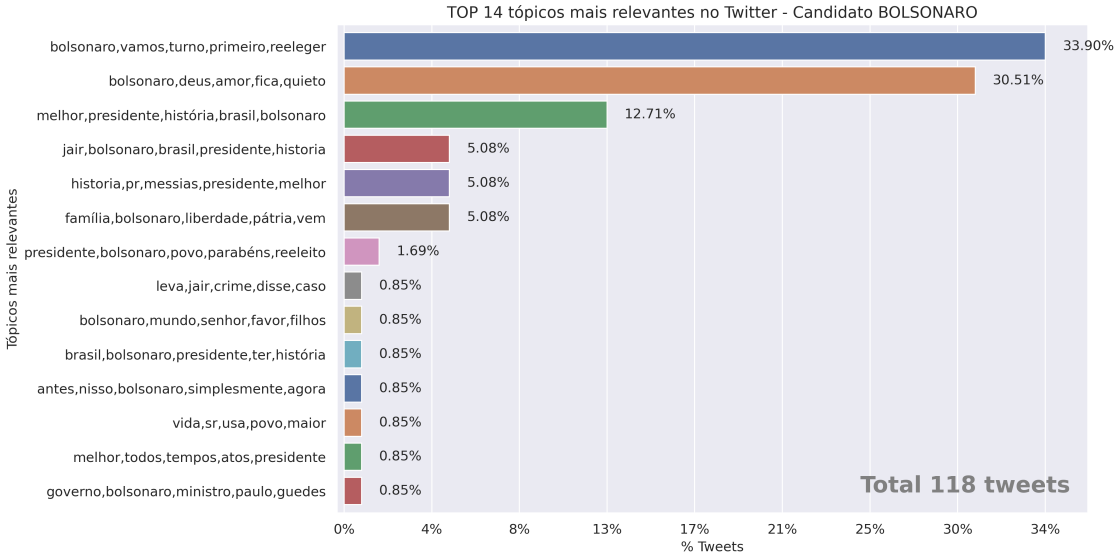
```
done in 3.785s.
```



Função aplicada para o candidato bolsonaro

[46]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="bolsonaro",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

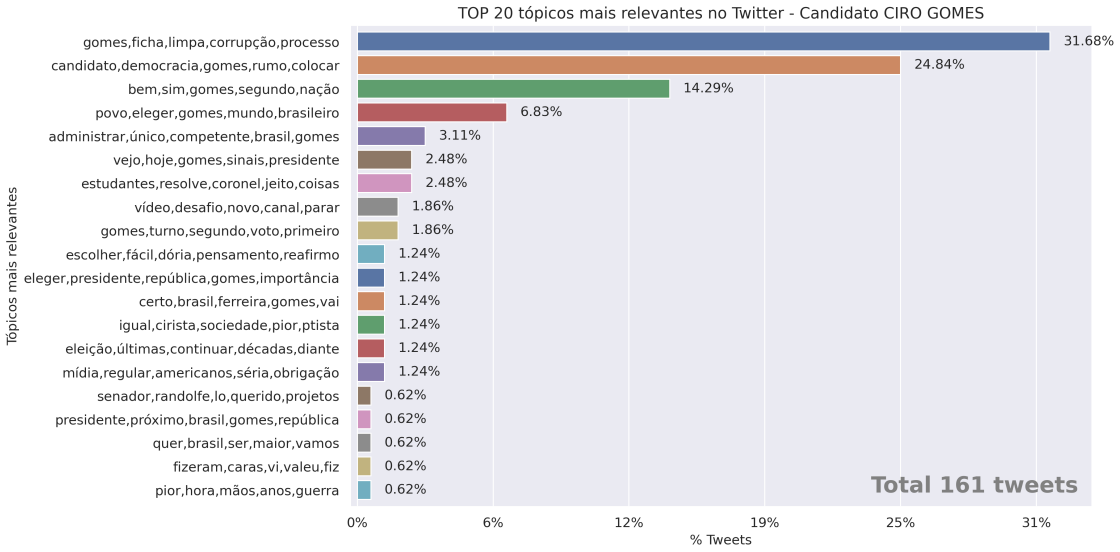
done in 3.681s.



Função aplicada para o candidato ciro gomes

[47]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="ciro gomes",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

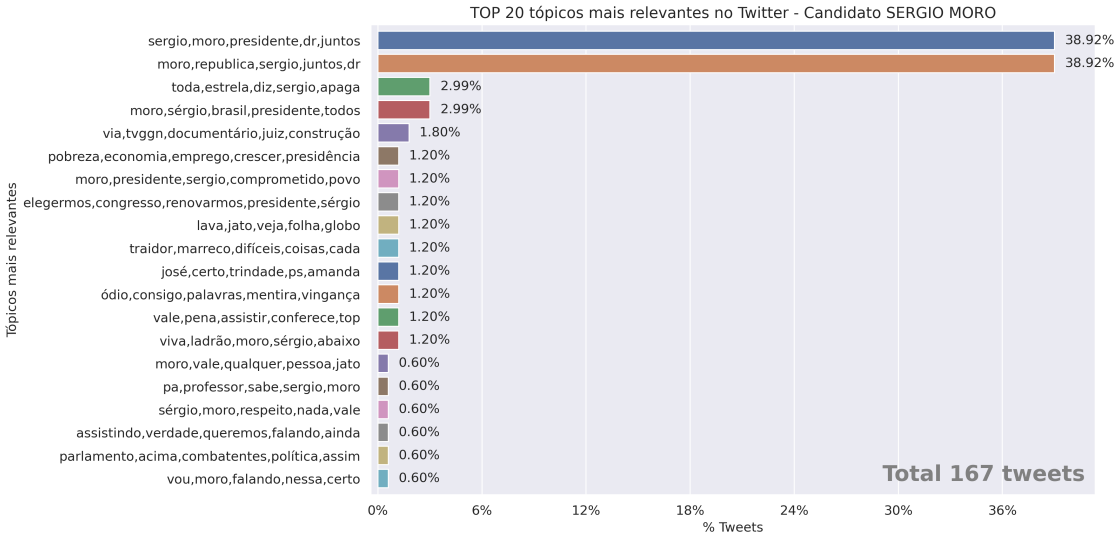
done in 0.956s.



Função aplicada para o candidato sergio moro

[48]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="sergio moro",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

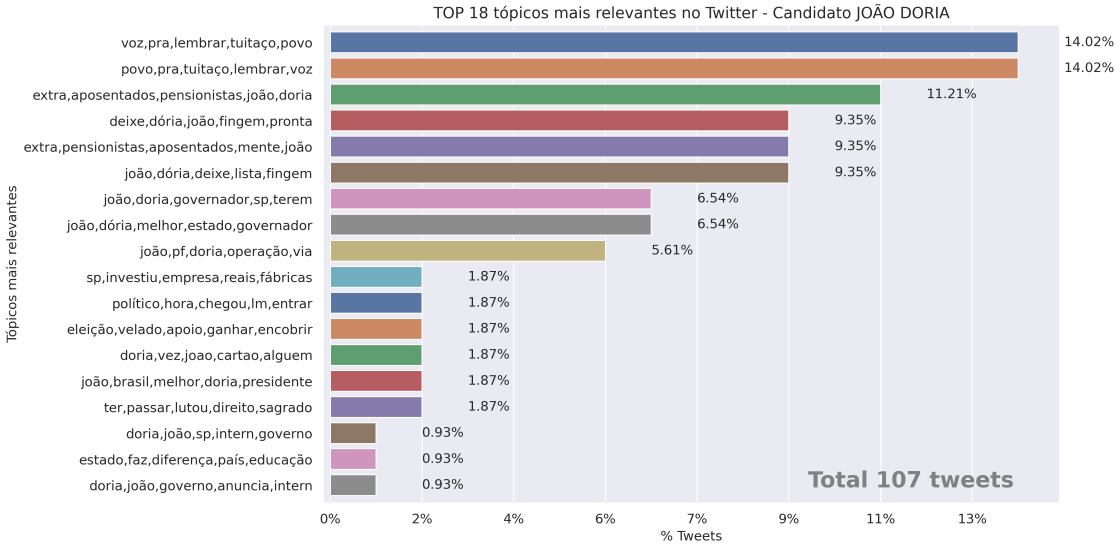
```
done in 0.790s.
```



Função aplicada para o candidato joão doria

[49]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="joão doria",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

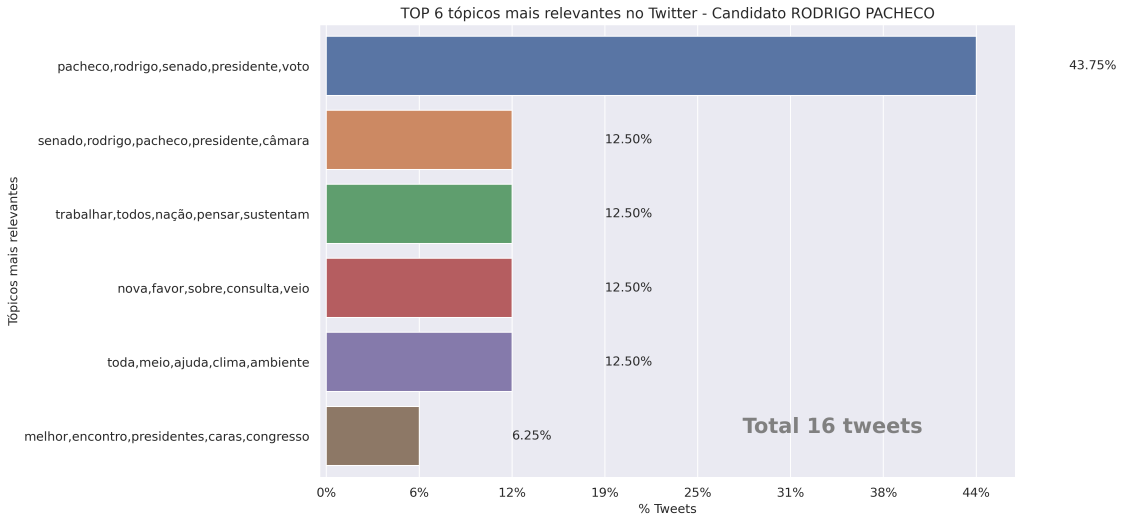
```
done in 0.300s.
```



Função aplicada para o candidato rodrigo pacheco

[50]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="rodrigo pacheco",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

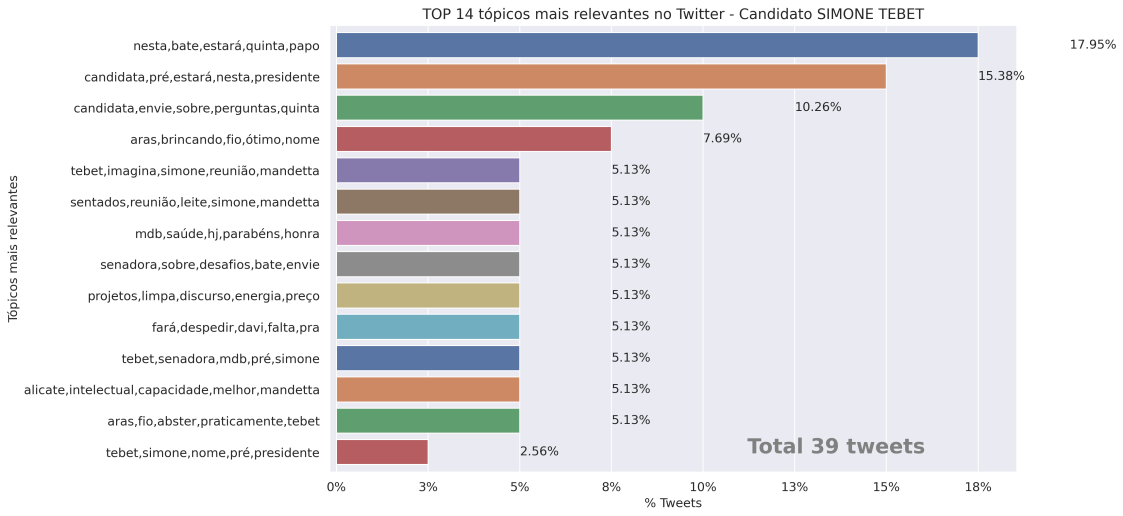
```
done in 0.103s.
```



Função aplicada para o candidato simone tebet

[51]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="simone tebet",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

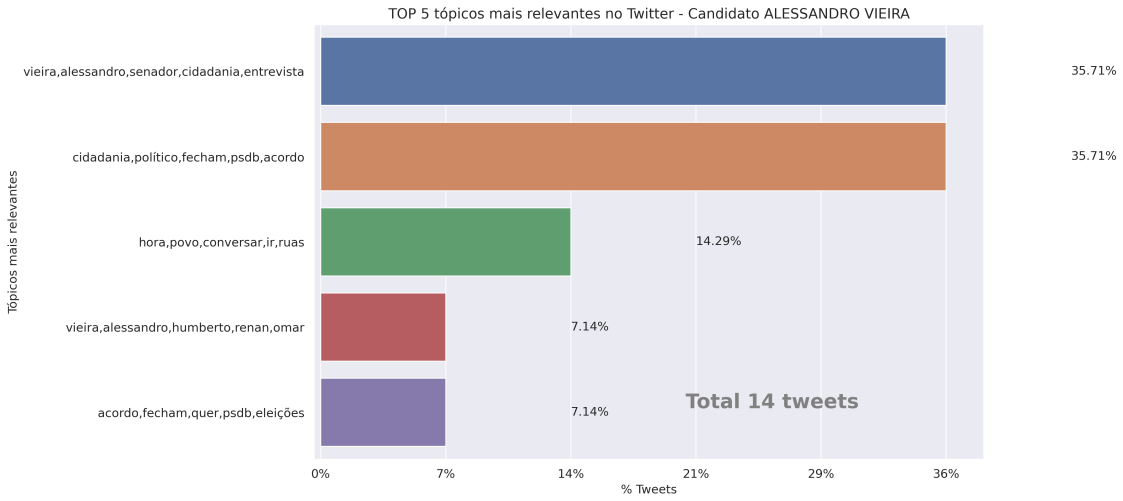
```
done in 0.050s.
```



Função aplicada para o candidato alessandro vieira

[52]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="alessandro vieira",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

```
done in 0.041s.
```

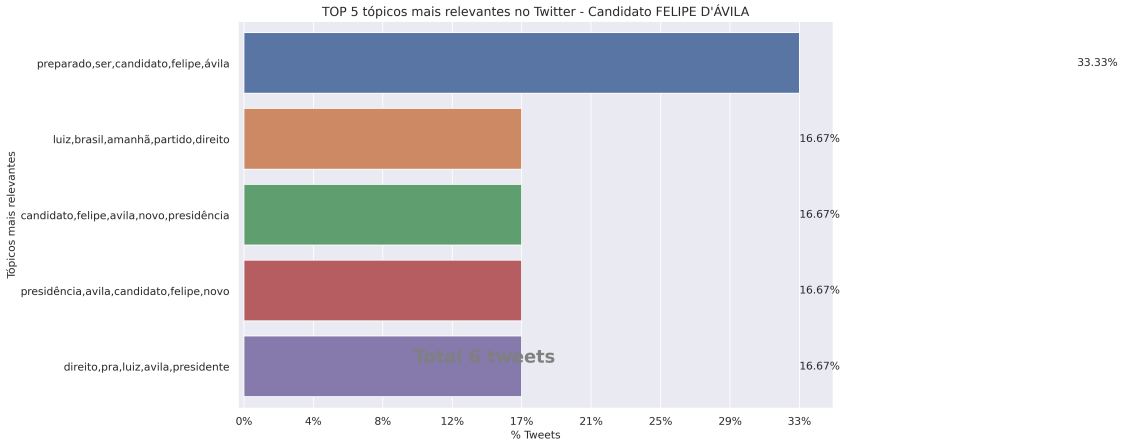


Função aplicada para o candidato felipe d'ávila

[53]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="felipe d'ávila",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
```

```
)
```

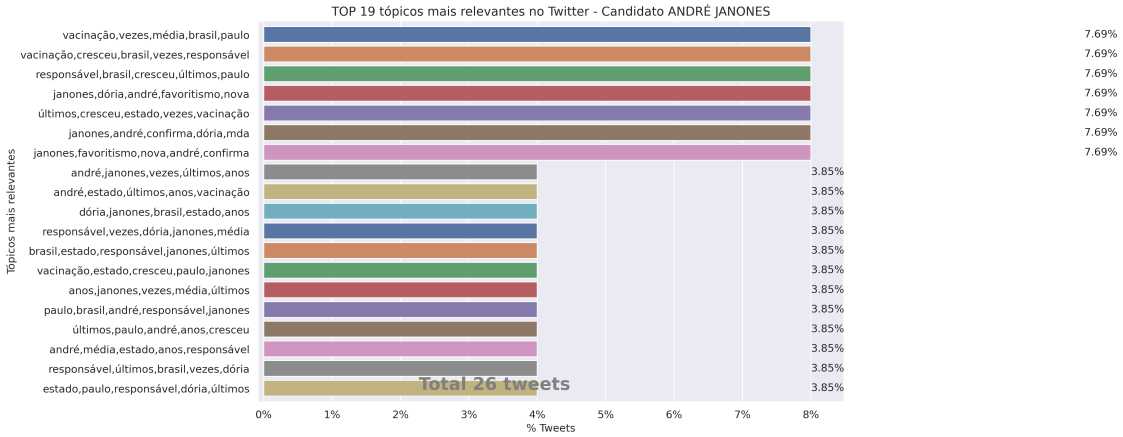
```
done in 0.019s.
```



Função aplicada para o candidato andré janones

[54]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="andré janones",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

```
done in 0.012s.
```



Função aplicada para o candidato leonardo péricles

[55]:
```
get_analysis_topics(
    dataframe=dataframe,
    model="bert",
    candidate="leonardo péricles",
    token_column="text_clean_tokens",
    num_features=1000,
    num_topicos=100,
    num_top_words=5,
    chart_limit_terms=20,
    wordcloud=False
)
```

```
done in 0.008s.
```