Sumarização de Notas Técnicas de Documentos Fiscais Utilizando PySpark e Técnicas de Processamento de Linguagem Natural

Luís Fernando Leite, Francis Henrique Pereira Marques, Henrique Dezani

luisleite1997@gmail.com; francishpm@gmail.com; henrique.dezani@fatec.sp.gov.br

Faculdade de Tecnologia de São José do Rio Preto

Resumo: Neste artigo, aprofundamos a exploração das capacidades do Python e PySpark, com foco particular na biblioteca SparkNLP, para sumarizar eficientemente notas técnicas associadas a documentos fiscais (DF-e) divulgadas pelo governo. A escolha do PySpark foi devido a sua capacidade de integração nativa com o ambiente Spark o tornando ideal para o tratamento de vastos volumes de dados textuais, pensando em futura escalabilidade devido ao aumento desses documentos publicados. Utilizando a já segmentada técnica de Análise de Alocação Latente de Dirichlet (LDA), conseguimos desenvolver um modelo baseado em tópicos, que extraí e captura as principais temáticas e tendências presentes nos documentos. Esta abordagem escolhida nos permitiu condensar informações essenciais, simplificando significativamente a compreensão e análise desses documentos, focando no benefício para partes interessadas e profissionais da área fiscal. Em um cenário onde a clareza e a agilidade são essenciais, esta metodologia surge para criar um modelo facilitador, potencializando a extração de insights relevantes de extensos conjuntos de documentos fiscais.

Palavras-chave: pln; linguagem; PySpark; extrativo.

Abstract: *In this article, we delve deeper into the capabilities of Python and PySpark, with a specific focus on the SparkNLP library, to efficiently summarize technical notes associated with fiscal documents (DF-e) released by the government. The choice of PySpark was due to its innate ability to integrate with the Spark environment, making it ideal for processing vast volumes of textual data, especially considering future scalability due to the increasing number of such published documents. By employing the distinct technique of Latent Dirichlet Allocation (LDA), we were able to develop a topic-based model that extract and captures the main themes and trends present in the documents. The chosen approach allowed us to significantly condense essential information, simplifying the understanding and analysis of these documents, with a focus on benefiting stakeholders and professionals in the fiscal field. In a scenario where clarity and agility are paramount, this methodology emerges to create a facilitating model, enhancing the extraction of relevant insights from extensive sets of fiscal documents.*

Keywords: *nlp; language; PySpark; exctrative.*

1. INTRODUÇÃO

Os Documentos Fiscais Eletrônicos (DF-e) são arquivos eletrônicos emitidos pelas empresas ao governo em operações de venda de bens ou prestação de serviços. Para parametrizar esses arquivos, a Secretaria da Receita Federal do Brasil emite Notas Técnicas que definem o *layout* dos documentos, estabelecem novas regras, campos que devem ser preenchidos e outras instruções (ENCAT, 2015). É de extrema importância que desenvolvedores de sistemas, contribuintes de impostos, responsáveis pela empresa e contadores acompanhem regularmente essas Notas Técnicas, uma vez que as mudanças podem ter impactos significativos nas operações fiscais da empresa. No entanto, a leitura das Notas Técnicas pode ser complexa e demorada, e muitas vezes o leitor pode achar que as mudanças não afetam suas emissões fiscais (SERRA, 2021).

2. JUSTIFICATIVA

Acompanhar as mudanças nas Notas Técnicas (NT) dos Documentos Fiscais Eletrônicos (DF-e) é de extrema importância para garantir o cumprimento correto das obrigações fiscais e evitar possíveis sanções. Porém, estas NT possuem algumas complexidades, tais como: (*i*) Terminologia técnica: As notas técnicas do governo muitas vezes usam terminologia técnica específica para a área de assunto, o que pode ser difícil para as pessoas que não estão familiarizadas com o campo; (*ii*) Complexidade do assunto: As notas técnicas do governo frequentemente tratam de assuntos complexos e detalhados que podem ser difíceis de entender sem um conhecimento prévio do assunto; e (*iii*) Formalidade: As notas técnicas do governo geralmente são escritas em um estilo formal e objetivo, o que pode torná-las menos acessíveis para leitores que preferem uma linguagem mais casual e conversacional. Pelo exposto, justifica-se a criação de ferramentas que permitam sumarizar as principais alterações para facilitar o entendimento e agilizar o processo de leitura.

3. OBJETIVOS

Tem-se por objetivo neste trabalho explorar a viabilidade, criação e utilização de um modelo de sumarização de texto, utilizando o modelo de *Topic Modelling* com técnicas de processamento de linguagem natural, que será aplicado usando como base de dados as notas técnicas publicadas pela SEFAZ. O resultado criará uma listagem com os principais tópicos abordados nas documentações a fim de facilitar a compreensão dos principais pontos documentados.

4. FUNDAMENTAÇÃO TEÓRICA

4.1. Nota técnica

É uma publicação oficial, neste caso, do governo, que traz consigo informações sobre os documentos fiscais eletrônicos (DF-e), como exemplo a Nota Fiscal de bem e consumo. Este documento visa orientar, tendo descrito em seu conteúdo o *layout* do arquivo enviado, as regras de validação acerca do documento, possíveis erros, preenchimentos exigidos e toda regra e protocolo esperado pelo servidor do governo (VARGAS, 2022).

4.2. Processamento de Linguagem Natural

O processamento de linguagem natural é um ramo do estudo de inteligências artificiais, onde é trabalhado para que os computadores consigam compreender, gerar e manipular a linguagem humana. É uma técnica muito difundida em diversas aplicações que lidem com linguagem, seja escrita ou em áudio (ORACLE, s.d.).

Se tornou uma área específica de estudo, já que a linguagem humana se trata de dados não estruturados. Em artigo para o site *Towards Data Science*, Diego Lopez Yse (2019) comenta: “Dados gerados a partir de conversas, declarações ou mesmo *tweets* são exemplos de dados não estruturados. Os dados não estruturados não se enquadram perfeitamente na estrutura tradicional de linhas e colunas dos bancos de dados relacionais e representam a grande maioria dos dados disponíveis no mundo real. (...) No entanto, graças aos avanços em disciplinas como a aprendizagem automática, uma grande revolução está em curso neste tema. Hoje em dia não

se trata mais de tentar interpretar um texto ou discurso com base nas suas palavras-chave (a velha forma mecânica), mas de compreender o significado por detrás dessas palavras (a forma cognitiva). Dessa forma é possível detectar figuras de linguagem como a ironia, ou mesmo realizar análises de sentimento".

4.3. Spark, PySpark e Spark NLP

Spark é um *framework* de código aberto, com capacidade para escalabilidade por computação distribuída. Permite o usuário ler e manipular dados, implementar e treinar modelos estatísticos. Por padrão conta com vários algoritmos e modelos implementados, para aprendizado de máquina, como o MLlib, com algoritmos comuns, como classificação, regressão, entre outros, e possui modelo para processamento gráfico, como o GraphX. O Spark pode ser utilizado em diversas linguagens, tais como: Java, Scala, Python, R e inclusive SQL. Sua lista de bibliotecas possui semelhança com as bibliotecas de outras ferramentas para análise de dados (DRABAS; LEE, 2017).

PySpark é uma API para utilização do Spark em ambiente Python, que é utilizada para tratamento de dados em computação distribuída, oferecendo interação com outras bibliotecas para análise exploratória de dados, como exemplo a biblioteca Pandas (TANDON, *et al.,* 2022).

O módulo SparkNLP foi desenvolvido pois as outras bibliotecas que se integravam com o PySpark não tinham como propósito a mesma base de processamento distribuído. Ela utiliza o mesmo conceito das outras bibliotecas de processamento de linguagem natural, com adendo de que a forma de armazenamento das anotações é diferente das outras bibliotecas com o mesmo fim. A maioria das bibliotecas armazenam as anotações no documento objeto, já o Spark realiza o armazenando em colunas diferentes no *dataframe* (TANDON, *et al.*, 2022).

Diferente das outras bibliotecas para processamento de linguagem natural, um dos objetivos do desenvolvimento da biblioteca SparkNLP foi a de unificada, para abranger todas as tarefas de *NLP,* como *tokenização,* análise de sentimento e reconhecimento de entidades nomeadas. Um dos pontos fortes a ser citado e que foi o motivo da escolha para a realização do trabalho, é a capacidade industrial do SparkNLP, porque é utilizado em áreas de negócios como saúde, finanças e jurídica. Pois é possível realizar extração eficiente em um grande volume de dados, assim permitindo a captura de *insights* para áreas específicas de negócios (GURSEV, s.d.).

4.4. Vetorização por *TF-IDF*

TF-IDF, do inglês *term frequency - inverse document frequency*, é um método estatístico que é utilizado para criar a vetorização de palavras, usado nas técnicas de processamento de linguagem natural. Este calcula a frequência de um termo em um documento, levando em consideração também a sua raridade, com objetivo de encontrar a importância de cada termo. Sendo assim, ele considera termos frequentes, e raros dentro de um documento (KARABIBER, *et al.*, s.d.).

4.5. Extração por *Topic Modeling* (Modelagem de Tópicos)

A extração via modelagem de tópicos é uma técnica dentro da área de processamento de linguagem natural, com objetivo de identificar e apresentar um conjunto de tópicos de um documento. Uma das técnicas conhecidas é a *Latent Dirichlet Allocation* (LDA), ou Alocação Latente de Dirichlet, esta é a técnica escolhida para a realização do trabalho (TANDON, *et al.,* 2022).

Para entender melhor o funcionamento da técnica de modelagem de tópicos por LDA, deve-se assumir que cada documento ou conjunto de dados são distribuídos em tópicos ou parágrafos, e cada conjunto destes possuem termos. A técnica LDA visa quebrar o corpus do conjunto de dados ou documento em uma série de tópicos baseado no seu peso dentro do documento (TANDON, *et al.*, 2022).

4.6.Perplexidade

A perplexidade é uma métrica utilizada na avaliação da habilidade preditiva de modelos de linguagem. Ela funciona como uma medida quantitativa para avaliar como um modelo pode antecipar uma determinada amostra, sendo que valores mais baixos de perplexidade indicam uma capacidade mais proficiente de previsão em um conjunto de dados (TANDON, *et al.*, 2022).

5. METODOLOGIA

As informações coletadas sobre o assunto foram obtidas de fóruns e blogs voltados ao tema de ciência de dados, especificamente sobre o tema de processamento de linguagem natural. Foram pesquisados e estudados artigos voltados a esta área.

A coleta de dados foi realizada no site oficial do governo e a base de dados foi montada em arquivo CSV, contendo o conteúdo original dos arquivos das NT. Em seguida, realizou-se uma limpeza nas informações, retirando informação irrelevante. O texto foi então *tokenizado* para que as palavras se tornassem *tokens* individuais. As *stopwords* também foram tratadas, pois não agregam valor ao modelo montado. Os dados textuais foram convertidos para representação numérica em TF-IDF, para permitir a captura da importância relativa de diferentes termos no documento, baseado em sua frequência.

O projeto foi desenvolvido utilizando um ambiente chamado Jupyter Notebook, onde se possibilita a análise exploratória de dados e sua visualização. Apesar de existirem outras linguagens de desenvolvimento, a linguagem inicial utilizada foi o Python, devido à simplicidade e usabilidade em trabalhos para desenvolvimento de modelos de *machine learning.*

6. DESENVOLVIMENTO

6.1. Criação do *dataset* ou base de dados

A criação da base de dados utilizou de documentos extraídos a partir de 2020 até o ano presente de 2023, no domínio da secretaria da fazenda. As informações servidas de modelo foram transformadas em um arquivo CSV (*comma-separated values*), em que duas colunas foram criadas, do nome dos arquivos extraídos, e a segundo coluna contendo seu conteúdo.

6.2. Definição das *Stopwords*

Para a definição de *stopwords* foi utilizada a biblioteca NLTK (*Natural Language Tool Kit*), para utilizar a lista de *stopwords* em língua portuguesa (Figura 1). Foram adicionadas mais

algumas *stopwords* para incorporarem a lista de palavras a serem retiradas do conteúdo principal, pois são palavras comuns dos documentos e que não adicionariam peso valido no resultado (Figura 2).

Também foi utilizada expressão regular para limpar palavras restantes que permaneceram, tendo dois ou menos caracteres (Figura 3).

```
nltk.download("stopwords")

sw = nltk.corpus.stopwords.words("portuguese")
```

Figura 1: Definição das *stopwords* em português, com *Natural Language Tool Kit.*

```
new_stopwords = ['xml', 'dao', 'nt', 'das', 'tag', 'regras', 'resumo',
 'validação', 'alteração', 'validações', 'alterações', 'campos']
sw.extend(new_stopwords)
```

Figura 2: Definição das novas palavras vazias.

```
df = df.withColumn("content",
      regexp_replace("content", r"\b\w{1,2}\b", ""))
df.show(4, vertical=True, truncate=False)
```

Figura 3: Limpeza de texto com expressão regular.

6.3. Criação do *Pipeline* do SparkNLP

Neste passo é criado o *pipeline* onde cada passo para o tratamento do texto bruto vai ocorrer (Figura 4):

```
document_assembler = DocumentAssembler()\
  .setInputCol("content")\
  .setOutputCol("document")\

tokenizer = Tokenizer()\
  .setInputCols(["document"])\
  .setOutputCol("token")

normalizer = Normalizer()\
  .setInputCols(["token"])\
  .setOutputCol("normalized")\
  .setLowercase(True)

stopwords_cleaner = StopWordsCleaner()\
  .setInputCols("normalized")\
  .setOutputCol("cleanTokens")\
  .setCaseSensitive(False)\
  .setStopWords(sw)

finisher = Finisher()\
  .setInputCols(["cleanTokens"])\
  .setOutputCols(["tokens"])\
  .setOutputAsArray(True)\
  .setCleanAnnotations(False)
```

Figura 5: Criação do *pipeline.*

6.3.1.1. *Document Assembler*

Processo inicial do *pipeline*, em que o texto bruto é incorporado para um formato que o SparkNLP compreenda, também são definidas as colunas de entrada e saída, para inicialização do processamento.

6.3.1.2. Criação dos *Tokens*

A *tokenização* é procedimento importante em grande parte de projetos de processamento de linguagem natural, onde a estrutura do texto é quebrada palavra por palavra. O resultado desse procedimento cria um *array* com os *tokens* capturados.

Como exemplo, a sentença exemplo no *dataframe* “D. Evento de “Ciência da Operação ou Ciência da Emissão”, passa a ser representado da seguinte forma: "D., Evento, de, “, Ciência, da, Operação, ou, Ciência, da, Emissão,” ".

6.3.1.3. Padronização com *Normalizer*

Este passo tem por objetivo normalizar o conteúdo do texto e é utilizado para lidar com variações de capitalização, pontuação e outras formas de formatação de texto que não são essenciais para a análise de linguagem natural. Os *tokens* são convertidos para caixa baixa. Caracteres especiais e duplo espaço também são retirados, algo muito comum nas documentações técnicas utilizadas como fonte de dados.

Como exemplo, a sentença citada fica desta forma: "d, evento, de, ciência, da, operação, ou, ciência, da, emissão".

6.3.1.4. *Remoção das Stopwords*

Definidas anteriormente, nesse passo do pipeline, a lista de *stopwords* a serem removidas são chamadas. Utilizando a lista que definimos usando o NLTK.

O resultado da remoção das *stopwords*, são *tokens* limpos de palavras-vazias: “"d, evento, ciência, operação, ciência, emissão".

6.3.2. Finalização e execução do *pipeline*

Passo final da criação do *pipeline*, esse processo tem função de retornar para uma matriz os *tokens* tratados em todos os passos anteriores. Finalizando o *pipeline* de pré-processamento de dados textuais utilizados no trabalho.

Chamando o método '*fit*', o Spark executou todas as transformações em sequência, passando os resultados para o próximo estágio, e finalmente retornar o modelo que será usado (Figura 5).

```
nlp_pipeline = Pipeline(stages=[document_assembler,
                                tokenizer,
                                normalizer,
                                stopwords_cleaner,
                                finisher])
```

Figura 6: Execução do *Pipeline.*

6.4. Criação do *dataframe* tratado e vetorização

Após o *dataset* ter sido passado no *pipeline*, onde todas as tratativas foram executadas. O resultado é um *dataframe* com colunas para cada ação do tratamento dos dados. Como o importante é apenas o último passo, onde os *tokens* estão limpos, foi montado um outro *dataframe* com apenas o nome do arquivo e os *tokens* tratados (Figura 6).

```
tokens_df = processed_df.select('Nome do arquivo', 'tokens')
tokens_df.printSchema()
```

```
root
 |-- Nome do arquivo: string (nullable = true)
 |-- tokens: array (nullable = true)
 |    |-- element: string (containsNull = true)
```

Figura 7: Seleção apenas das colunas uteis no *dataframe.*

6.5. Vetorização por frequência da palavra

Como a máquina não consegue interpretar diretamente um texto, foi realizada a vetorização por TF-IDF (Frequência do termo–inverso da frequência nos documentos), para que uma representação número dos tokens seja criada, baseada na sua frequência e raridade.

Cada termo no documento foi mapeado, criando um índice em um vocabulário.
O resultado é um vetor onde cada índice representa um termo no vocabulário e cada valor representa a frequência desse termo no documento (Figura 7).

```
cv = CountVectorizer(inputCol="tokens", outputCol="raw_features")
cv_model = cv.fit(tokens_df)

vectorized_tokens = cv_model.transform(tokens_df)

idf = IDF(inputCol="raw_features", outputCol="features")
idf_model = idf.fit(vectorized_tokens)
vectorized_df = idf_model.transform(vectorized_tokens)
vectorized_df = vectorized_df.drop(fun.col('raw_features'))
vectorized_df.show(5, vertical=True)
```

Figura 8: Contagem dos vetores e criação do TF-IDF.

6.6. Modelagem por LDA (Alocação Latente de Dirichlet)

Foi criado um laço para o cálculo da perplexidade, para comparação entre a quantidade de tópicos e o resultado do grau de perplexidade do modelo. Foi definido um laço começando em 2 tópicos mínimos, até 10 tópicos, em que 4 tópicos foram o menor número de resultado da perplexidade (Figura 8). Foi então criado o modelo de LDA com o parâmetro k=4, que especifica o número de tópicos que o modelo deve identificar. No LDA, o k é um parâmetro crucial que determina quantos tópicos o modelo tentará descobrir nos dados.

```
O cálculo de perplexidade para 2 tópicos é: 7.242737189156632
O cálculo de perplexidade para 3 tópicos é: 7.138880588618495
O cálculo de perplexidade para 4 tópicos é: 7.083193908543193
O cálculo de perplexidade para 5 tópicos é: 7.089446450642966
O cálculo de perplexidade para 6 tópicos é: 7.084314133027346
O cálculo de perplexidade para 7 tópicos é: 7.14834000901822
O cálculo de perplexidade para 8 tópicos é: 7.201378643005502
O cálculo de perplexidade para 9 tópicos é: 7.278408834232258
```

Figura 9: Resultado do laço de perplexidade.

Posteriormente foi realizada a distribuição dos tópicos para cada documento e as distribuições das palavras em cada tópico.

7. RESULTADOS E DISCUSSÕES

Durante os testes, 3 *datasets* foram testados para medir a eficácia e evolução do modelo criado. Um dos testes foi realizado utilizando a base de dados completa das notas técnicas, a segunda foi utilizando apenas uma nota técnica e o terceiro teste foi se utilizando de três documentos do mesmo segmento ou tema, no caso, alterações de tributação de combustível. O resultado de comparação pelo cálculo de perplexidade entre os três testes, tamanho de *dataset* e velocidade de processamento, é apresentado na Tabela 1. O resultado das discussões está exposto em cada subtópico referente a cada teste.

Modelo	Velocidade (segundos)	Tamanho *dataset* (mb)	Melhor Perplexidade	Conjunto de Documentos
Completo	140	1.1	7.0831	*Dataset completo de 2020 até 2023*
Individual	10	0.019	-	Manual de orientação ao contribuinte
Segmentado	80	0.194	7,4698	3 versões de N.T. de combustível

Tabela 1: Comparação de perplexidade dos modelos.

7.1. Modelo com coleção de documentos

Foram extraídos 4 tópicos principais das documentações técnicas da Secretaria da Fazenda do Estado (SEFAZ) utilizando o modelo de Alocação Latente de Dirichlet (LDA). O modelo foi refinado várias vezes, com o objetivo de diminuir o cálculo da perplexidade e melhorar a qualidade dos tópicos extraídos. Para isso, foram removidas palavras desnecessárias e a lista de palavras vazias (*stopwords*) foi ampliada, resultando em uma análise mais precisa. Os resultados foram representados visualmente por meio de três *wordclouds* (Figuras 9, 10, 11 e 12), cada uma correspondendo a um dos tópicos extraídos, e refletindo os principais temas presentes nas documentações técnicas da SEFAZ, assim, facilitando a visualização humana.

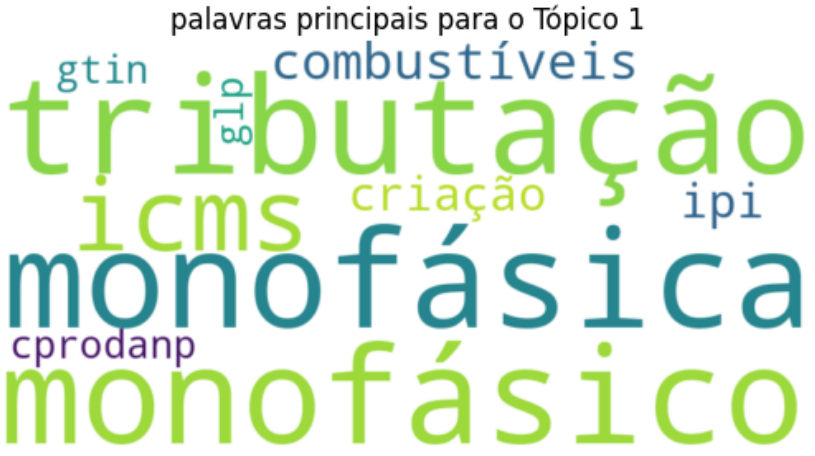

Figura 10: Nuvem de palavras do primeiro tópico.

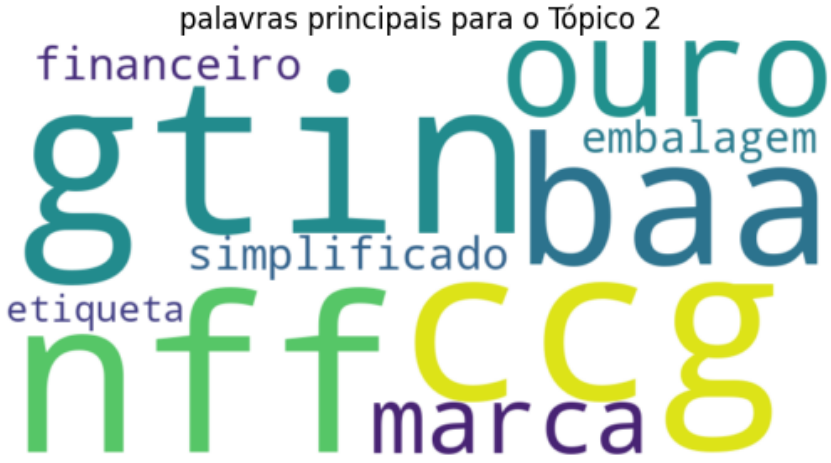

Figura 11: Nuvem de palavras do segundo tópico.

Figura 12: Nuvem de palavras do terceiro tópico.

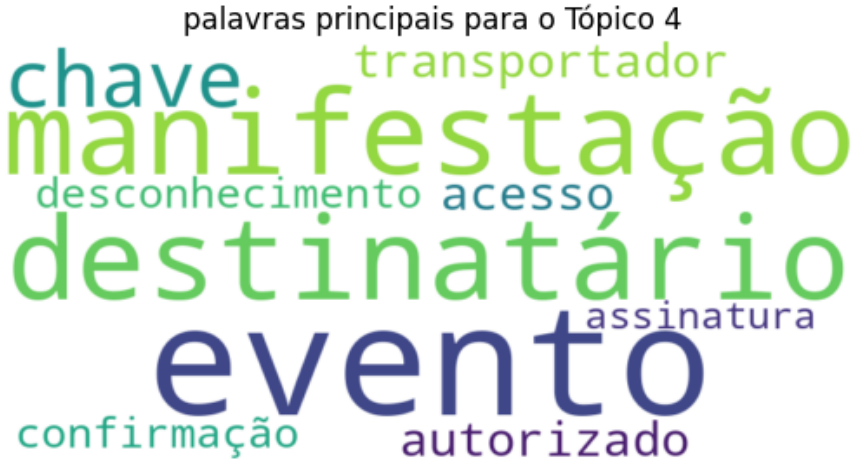

Figura 13: Nuvem de palavras do quarto tópico.

7.2. Análise de documento individual

A aplicação do modelo para o tópico de apenas um documento, utilizou o documento "Manual_de_Orientacao_Contribuinte_v_6.00", presente no *dataset*. o resultado do cálculo de perplexidade retornou 'inf' ou infinito, indicando a incapacidade de mensurar a comparação de modelos pela perplexidade. Sendo executado a modelagem por tópicos, mesmo mediante a impossibilidade de mensurar, o resultado por avaliação humana também ficou comprometido, já que o resultado retornado não se tornou tão claro (Tabela 2).

Conjunto	Tópicos extraídos
Tópico 1	'lote', 'obrigatórios', 'autorização', 'devolverá', 'registro', 'enat', 'acrônimo', 'vinculado', 'assíncronos', 'particular'
Tópico 2	'exceção', 'existirá', 'respectiva', 'pois', 'sistemas', 'consultar', 'dezembro', 'finanças', 'protocolo', 'conceitual'

Tabela 2: Conjunto de tópicos documento individual.

7.3. Análise de documentos agrupados por tema

Para comparação com o modelo principal que utiliza a base de documentos, foi realizada uma análise com apenas 3 documentos com o mesmo tema sobre alterações de combustível. Seu menor índice de perplexidade foi para utilização do mínimo de dois conjuntos de tópicos, tendo 7,4698 o seu resultado. O resultado segue na Tabela 3.

Conjunto	Tópicos extraídos
Tópico 1	'existentes', 'sistema', 'controle', 'lab', 'final', 'verificada', 'vigor', 'contribuinte', 'pai', 'entrada,
Tópico 2	'bases', 'qbcmono', 'qbcmonoret', 'documentação', 'base', 'informadas,', 'revogação', 'exclusão', 'qbcmonoreten', 'dessa'

Tabela 3: Conjunto de tópicos em documentos agrupados.

8. Conclusão

Levando em consideração a escolha de usar SparkNLP pela sua gama de ferramentas e seu uso comercial, a implementação do modelo usando *topic modelling* com LDA com as documentações da SEFAZ, se mostrou uma estratégia válida devido ao resultado obtido pela avaliação humana e pelo cálculo de perplexidade, em uso para coleção de documentos ou para documentos longos, a depender da quantidade de *tokens* no conjunto de dados. A aplicação para documentos individuais trouxe resultado, mas não podendo ser mensurado a perplexidade, devido ao cálculo não conseguir trazer um resultado, apenas retornando os principais tópicos por tópico. A aplicação para conjunto de documentos curtos do mesmo tema, trouxe resultado do cálculo comparativo de perplexidade, mas sua separação por tópicos não foi satisfatória. A visualização com as *wordclouds* proporcionou uma forma intuitiva de identificar os termos e conceitos mais recorrentes, contribuindo para uma interpretação humana das informações.

Os resultados obtidos indicam um potencial de continuidade e melhoria do modelo desenvolvido, principalmente para lidar com longas documentações ou coleção de documentos, não sendo mensurável sua qualidade para o uso em documentos curtos, sendo necessária uma abordagem diferente nesses casos.

REFERÊNCIAS

DRABAS, T; LEE, D. **Learning PySpark.** 35 Livery Street, Sebastopol, Birmingham B3 2PB, UK: Packt Publishing Ltd., February 2017. P. 2 et seg.

ENCAT. **Manual de Orientação do Contribuinte.** v. 6.0, p. 13-14, set. 2015. Disponível em: < https://www.nfe.fazenda.gov.br/portal/exibirArquivo.aspx?conteudo=nebWFce4X9o=>

GURSEV. Introduction to Natural Language Processing with Spark NLP. **Medium.** Disponível em: <https://medium.com/john-snow-labs/introduction-to-natural-language-processing-ce3003bd9242> Acesso em: 15 junho. 2023.

KARABIBER, F; LEWIS, R; MARTIN, B. TF-IDF — Term Frequency-Inverse Document Frequency. **LearnDataSci**. Disponível em: <https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/>. Acesso em: 07 novembro. 2023.

ORACLE. O que é Processamento de Linguagem Natural? **Oracle**. Disponível em: < https://www.oracle.com/br/artificial-intelligence/what-is-natural-language-processing/ >. Acesso em: 05 agosto. 2023.

SERRA, G. Notas Técnicas: o que são e como acompanhá-las. **Tecnospeed,** 2021. Disponível em: < https://blog.tecnospeed.com.br/notas-tecnicas/ >. Acesso em: 01 julho. 2023.

TANDOS, A; RYZA, S; LASERSON, U.; OWEN, S.; WILLS, J. **Advanced Analytics with PySpark, 1.** 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media Inc., 2022. P., 109-124.

VARGAS, L.C. O que é uma Nota Técnica e qual sua importância? **NDD Tech,** 2022.

Disponível em: < https://ndd.tech/blog/compliance-fiscal/o-que-e-uma-nota-tecnica-e-qual-a-sua-importancia/ >. Acesso em: 01 julho. 2023.

YSE, D.L. Your Guide to Natural Language Processing (NLP). **Towards Data Science,** 2019. Disponível em: < https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1 >. Acesso em: 15 junho. 2023.