UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Disciplina: Processamento de Linguagem Natural

Unidade 4 – Análise Sintática, Modelagem e Extração de Informações

Data de entrega: 20/05

Equipe: Felipe Bona, João Martinho

A partir dos dados textuais já coletados, cada equipe deverá apresentar e documentar as seguintes tarefas:

1) **Extração de entidades** (PER, ORG, LOC) e/ou **atributos linguísticos** (NOUN, VERB, ADJ etc.), apresentando uma análise estatística básica (entidades mais frequentes, frequência ao longo do tempo etc.), com comentários sobre a importância (ou não) disso para entender a base de dados **(3 pontos).**

   Implementação no código:

```
def analyze_entities(self, text, verses_text):
    # Dicionário de entidades pré-definidas
    entity_dict = {
         'PESSOA': ['Deus', 'Adão', 'Eva', 'Noé', 'Moisés', 'Jesus',
'Paulo',...],
                  'LOCAL':  ['terra',  'céus',  'Egito',  'Israel',
'Jerusalém',...],
                'ORGANIZAÇÃO':  ['Israel',  'Igreja',  'Sacerdotes',
'Levitas',...]
    }

    # Dicionário de categorias gramaticais
    pos_patterns = {
        'SUBSTANTIVO': r'\b(terra|céus|luz|águas|homem|...)\b',
        'VERBO': r'\b(criou|disse|fez|viu|chamou|...)\b',
        'ADJETIVO': r'\b(bom|santo|justo|sábio|eterno|...)\b'
    }
```

   Análise estatística:

- O código calcula frequência de cada entidade e atributo linguístico;
- Mostra distribuição por versículo (ex: "Versículo 1: PESSOA=2, LOCAL=1, ORGANIZAÇÃO=0"), apresenta totais gerais de cada categoria;
- Relevância para textos bíblicos;
- Identificação de personagens centrais (Deus, Jesus) ajuda na análise narrativa;
- Locais como "Egito" e "Jerusalém" contextualizam eventos históricos;
- Atributos como "criou" (VERBO) e "santo" (ADJETIVO) revelam aspectos teológicos.

2) Aplicação e avaliação qualitativa de abordagens extrativas **– sumarização abstrativa, sumarização extrativa e extração de palavras-chave**. Incluir comentários sobre aplicabilidade e relevância para o conjunto de dados. **(3 pontos)**

```
def generate_summaries(self, text, num_verses):
                    parser     =     PlaintextParser.from_string(text,
Tokenizer("portuguese"))
    num_sentences = max(3, num_verses // 5)

    # LexRank
    lex_rank = LexRankSummarizer()
    lex_summary = lex_rank(parser.document, num_sentences)

    # LSA
    lsa_summarizer = LsaSummarizer()
    lsa_summary = lsa_summarizer(parser.document, num_sentences)
```

Extração de Palavras-chave:

```
def extract_keywords(self, verses_text):
          vectorizer   =   TfidfVectorizer(stop_words=list(stopwords),
max_features=100)
    X = vectorizer.fit_transform(verses_text)
    feature_names = vectorizer.get_feature_names_out()
    tfidf_scores = X.max(axis=0).toarray()[0]
      return sorted(zip(feature_names, tfidf_scores), key=lambda x:
x[1], reverse=True)
```

Avaliação qualitativa:

- LexRank é melhor para destacar eventos narrativos centrais;
- LSA identifica melhor temas latentes e relações conceituais;
- TF-IDF é eficaz para identificar termos centrais, mas pode perder contexto.

3) Proposta e/ou aplicação de **classificação de textos por aprendizado supervisionado**, incluindo avaliação quantitativa e qualitativa das métricas (precisão, acurácia, recall etc.), estratégias para rotulação dos dados e resultados obtidos/esperados. **(4 pontos)**

```
def classify_text(self, verses_text):
    # Dados de treino rotulados manualmente
    sample_texts = [
        ("Deus criou os céus e a terra", "positive"),
        ("A terra era sem forma e vazia", "negative"),
        ...
    ]


  # Vetorização e treinamento
    vectorizer = TfidfVectorizer(max_features=1000, stop_words=...)
    X = vectorizer.fit_transform(texts)
    y = labels
         X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3)


    # Modelo Naive Bayes
    classifier = MultinomialNB(alpha=1.0)
    classifier.fit(X_train, y_train)


    # Avaliação
    cv_scores = cross_val_score(classifier, X, y, cv=5)
    y_pred = classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
```

Métricas e avaliação:

- Acurácia: medida de acertos gerais;
- Precisão: capacidade de não classificar incorretamente como positivo;
- Recall: capacidade de encontrar todos os positivos;
- F1-Score: média harmônica entre precisão e recall;
- Validação cruzada: mostra robustez do modelo.

Estratégia de rotulação:

- Baseada em contexto teológico;
- Positivo: criação, bênçãos, esperança;
- Negativo: julgamento, pecado, destruição;
- Neutro: descrições factuais.