Named Entity Recognition - Técnicas de Machine Learning e suas comparações

Luiza Barros Reis Soezima

Parte 1

Descrição das ferramentas

OpenNLP

(https://opennlp.apache.org/docs/1.5.3/manual/opennlp.html)
Apache OpenNLP library
JAVA

- The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text
- It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution
- OpenNLP also included maximum entropy and perceptron based machine learning.
- https://sematext.com/blog/entity-extraction-opennlp-tutorial/

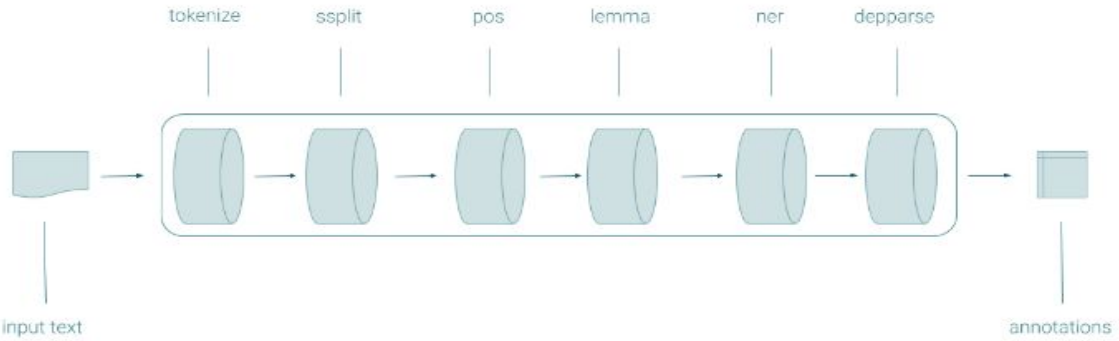
CoreNLP

(https://stanfordnlp.github.io/CoreNLP/)

https://stanfordnlp.github.io/CoreNLP/#:~:text=CoreNLP%20enables%20users%20to%20derive,%2C%20quote%20attributions%2C%20and%20relations.

- CoreNLP is your one stop shop for natural language processing in Java! CoreNLP enables users to derive linguistic annotations for text, including token and sentence boundaries, parts of speech, named entities, numeric and time values, dependency and constituency parses, coreference, sentiment, quote attributions, and relations. CoreNLP currently supports 6 languages: Arabic, Chinese, English, French, German, and Spanish.
- The pipeline itself is composed by 6 annotators. Each of these annotators will process the input text sequentially, the intermediate outputs of the processing sometimes being used as inputs by some other annotator.
- https://towardsdatascience.com/intro-to-stanfords-corenlp-and-java-for-python-programmers-c2586215aab6

Annotator 1: Tokenization → turns raw text into tokens.
Annotator 2: Sentence Splitting → divides raw text into sentences.
Annotator 3: Part of Speech (POS) Tagging → assigns part of speech labels to tokens, such as whether they are verbs or nouns. Each token in the text will be given a tag.
Annotator 4: Lemmatization → converts every word into its lemma, its dictionary form. For example the word “was” is mapped to “be”.
Annotator 5: Named Entity Recognition (NER) → Recognises when an entity (a person, country, organization etc…) is named in a text. It also recognises numerical entities such as dates.
Annotator 6: Dependency Parsing → Will parse the text and highlight dependencies between words.



StanfordNLP

(https://nlp.stanford.edu/software/)

- StanfordNLP is the combination of the software package used by the Stanford team in the CoNLL 2018 Shared Task on Universal Dependency Parsing, and the group's official Python interface to the Stanford CoreNLP sNative Python implementation requiring minimal effort to set up
- Full neural network pipeline for robust text analytics, including: Tokenization, Multi-word token (MWT) expansion, Lemmatization, Parts-of-speech (POS) and morphological feature tagging, Dependency Parsing
- Pretrained neural models supporting 53 (human) languages featured in 73 treebanks
- A stable officially maintained Python interface to CoreNLPoftware.
- https://towardsdatascience.com/natural-language-processing-using-stanfords-corenlp-d9e64c1e1024

SpaCy

(https://spacy.io/)

- SpaCy is a free, open-source library for NLP in Python. It’s written in Cython and is designed to build information extraction or natural language understanding systems. It’s built for production use and provides a concise and user-friendly API.
- The Spacy NER system contains a word embedding strategy using sub word features and "Bloom" embed, and a deep convolution neural network with residual connections. The system is designed to give a good balance of efficiency, accuracy and adaptability.

NAME	DESCRIPTION
Tokenization	Segmenting text into words, punctuations marks etc.
Part-of-speech (POS) **Tagging**	Assigning word types to tokens, like verb or noun.
Dependency Parsing	Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
Lemmatization	Assigning the base forms of words. For example, the lemma of “was” is “be”, and the lemma of “rats” is “rat”.
Sentence Boundary Detection (SBD)	Finding and segmenting individual sentences.
Named Entity Recognition (NER)	Labelling named “real-world” objects, like persons, companies or locations.
Entity Linking (EL)	Disambiguating textual entities to unique identifiers in a knowledge base.
Similarity	Comparing words, text spans and documents and how similar they are to each other.
Text Classification	Assigning categories or labels to a whole document, or parts of a document.
Rule-based Matching	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
Training	Updating and improving a statistical model’s predictions.
Serialization	Saving objects to files or byte strings.

NLTK

(https://www.nltk.org/)

- The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).
- It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.
- NLTK includes more than 50 corpora and lexical sources such as the Penn Treebank Corpus, Open Multilingual Wordnet, Problem Report Corpus, and Lin's Dependency Thesaurus.
- https://pythonprogramming.net/named-entity-recognition-nltk-tutorial/
- https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

GATE

(http://services.gate.ac.uk/annie/)
(https://gate.ac.uk/ie/annie.html)

- General Architecture for Text Engineering or GATE is a Java suite of tools originally developed at the University of Sheffield beginning in 1995 and now used worldwide by a wide community of scientists, companies, teachers and students for many natural language processing tasks, including information extraction in many languages.
- ANNIE is a vanilla information extraction system comprising a set of core PRs: Tokeniser, Sentence Splitter, POS tagger, Gazetteers, Semantic tagger (JAPE transducer), Orthomatcher (orthographic coreference)
- https://en.wikipedia.org/wiki/General_Architecture_for_Text_Engineering

Parte 2

Técnicas aplicadas: Avaliação de artigos

A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate

Xavier Schmitt* , Sylvain Kubler† , Jeremy Robert * , Mike Papadakis* , Yves LeTraon*

Introdução:

- Corpora diferentes trazem dificuldade para comparar performance de software de NER.
- Critérios usados para selecionar um software NER: performance, custo, documentação, licença
- NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate
- Base utilizada: CoNLL2003 e GMB
- Resultados mostram que StanfordNLP performa de 15% a 30% melhor no corpora usado do que os outros softwares
- Não conseguiu recuperar os mesmos resultados da literatura : diferenças de até 66%

NER Software Evaluation

- NER software constituem de 3 tasks: Corpus -> **[Tokenização - > Parte do documento - > NER] -** > Avaliação
- Tabela 1 mostra informações sobre corus, software, métricas consideradas para avaliação dentro do histórico de NER softwares.
- No estudo *[16] A. Pinto, H. Gonçalo Oliveira, and A. Oliveira Alves. 2016. Comparing the performance of different NLP toolkits in formal and social media text. In: 5th Symposium on Languages, Applica* temos que NLTK e OpenNLP tem resultados similares ou melhores que o StanfordNLP usando mesmos corpus. Cada experimento tem um desempenho, sendo impossível de replicar experimentos
- No experimento [16] apenas 24% os NER softwares fornecem informação sobre o tipo de classificador usado e 16% fornecem detalhes da versão de software
- Existem muitas inconsistências experimentais para executar julgamentos entre os softwares em MUITOS artigos e mesmo na documentação

Reference	Corpus	Software	License	Classifier	Version
[16]	CoNLL 2003 Ritter MSM2013	OpenNLP	Apache Software Lic.	N/A	N/A
		StanfordNLP	GNU GPL	CoNLL, ACE, MUC	**3.6.0**
		NLTK	Apache Lic. v2	ACE	N/A
		Pattern	BSD	N/A	N/A
		TweetNLP	GPL v2	N/A	N/A
		TweeterNLP	GNU GPL	N/A	N/A
		TwitIE	N/A	N/A	N/A
[21]	Wikigold	SpaCy	MIT License	N/A	N/A
		StanfordNER	GNU GPL	CoNLL MUC6/7, ACE	v3.6
		NLTK	Apache Lic. v2	N/A	N/A
		Alias-i LingPipe	Royalty Free Lic. v1	MUC6	4.1
[17]	MSM2013 Ritter UMBC	Annie (Gate)	GPL v3	Gazetter	Gate v8
		StanfordNER	GNU GPL	CoNLL, ACE	N/A
		DBpedia Spotlight	Apache Lic. v2	N/A	N/A
		Lupedia	N/A	N/A	N/A
		Ritter et al.	GPL v3	N/A	N/A
		Alchemy API	Non Commercial	N/A	N/A
		NERD-ML	GPL v3	N/A	N/A
		YODIE	N/A	N/A	N/A
		Zemanta	Non Commercial	N/A	N/A
		TextRazor	Non Commercial	N/A	N/A
[20]	CoNLL 2003	StanfordNLP	GNU GPL	N/A	N/A
		Annie (Gate)	GPL v3	N/A	N/A
[22]	OntoNotes 5	SpaCy	MIT License	Small, Medium, Large OntoNotes 5	v2.x
[23]	CoNLL 2003	StanfordNLP	GNU GPL	N/A	N/A

TABLE I

EXPERIMENTAL CONDITIONS OF NER SOFTWARE EVALUATION STUDIES IN THE LITERATURE. THIS TABLE SHOWS THE LACK OF INFORMATION TO BE ABLE TO REPRODUCE THE EXPERIMENTS. AS COMPLEMENTARY INFORMATION, FIG. 2 SHOWS THE DIVERGENCE IN RESULTS FOUND IN THE LITERATURE (FROM 15% TO 50%), WHICH MOTIVATES US IN PROPOSING A REPLICABLE COMPARISON STUDY OF NER SOFTWARE.

NER Softwares:

- StanfordNLP é um JAVA toolkit que providencia uma grande gama de ferramentas como PoS, NER tagger (https://stanfordnlp.github.io/CoreNLP/)
  PoS(https://nlp.stanford.edu/software/tagger.shtml) - diz referência ao tool que permite classificar uma palavra como parte do discurso - verbo, objeto, sujeito, artigo, etc.

- SpaCy é conhecido pela rapidez em parsing

- NLTK oferece uma gama de bibliotecas e módulos para NLP simbólico ou estatístico.

NER softwares Comparisons

- Corpus Selection: Para executar comparações temos que ter os corpus de uso bem definidos: não deve ser de domínio fechado para ser possível tentar replicar. GMB(Groningen Meaning Bank: newswire texts e textos do ANC e fábulas de Esopo) e CoNLL(Reuters Corpus: dados de artigos da Reuters)
- Escolhas de Software:
    Gratuidade,
    Uso ilimitado,
    Documentação,
    Disponível para linux,
    Reconhecimento de Person(PER), Organization (ORG) e Location (LOC) - entidades
- Avaliação de performance: Comparação do conjunto de entidades marcadas identificadas pelo software com os que foram especificados no Gold Data. Outputs foram convertidos para o mesmo IOB (inside-outside-beginning) tag usado no CoNLL
- SpaCy e NLTK usam Geopolitical Entity (GPE) como entidade de LOC(location).

Performance Evaluation:

- Avaliação usando P(precision), R(Recall) e F(F-measure).
- OpenNLP, NLTK, Fate e SpaCy tem performance muito baixa para detecção de ORG(organization)
- OpenNLP tem baixo Recall scoring comparado com os outros mas tem alta precisão
- StanfordNLP tem um classifier default que é treinado parcialmente no CoNLL 2003, assim, os resultados na base GMB traz resultados mais REAIS com um corpus específico.

Name	Prog.	Vers.	Algo	Classifiers
Stanford NLP	Java	3.9.2	CRF	CoNLL, MUC6-7, ACE
OpenNLP	Java	1.9	Max. entropy	
SpaCy	Python	2.0.16	Neural (2.0)	OntoNotes
NLTK	Python	3.4	Max entropy	Stanford NER
GATE	Java	8.5.1	JAPE	

TABLE II
NER SOFTWARE SELECTED FOR EVALUATION

		CoNLL 2003			GMB		
Software	Entity	P	R	F1	P	R	F1
Stanford NLP	LOC	91.30	88.73	90	83.10	63.64	72.08
	ORG	86.32	80.92	83.53	71.40	47.42	56.99
	PER	92.72	82.68	87.41	78.59	84.70	81.53
	Overall	90.06	73.67	**81.05**	79.81	63.74	**70.88**
NLTK	LOC	52.47	65.47	58.26	77.13	77.1	77.12
	ORG	36.20	24.80	29.44	42.06	35.54	38.53
	PER	61.09	66.11	63.50	38.07	55.87	45.28
	Overall	51.78	45.56	**48.47**	60.96	63.91	**62.40**
Gate	LOC	59.63	78.63	67.82	79.03	48.16	59.85
	ORG	50.58	21.29	29.96	45.08	37.68	41.05
	PER	69.53	62.67	65.92	46.53	53.70	49.86
	Overall	61.48	47.44	**53.55**	61.72	46.78	**53.22**
OpenNLP	LOC	76.54	52.22	62.08	84.34	45.84	59.40
	ORG	38.06	14.87	21.39	59.27	30.64	40.39
	PER	83.94	37.17	51.52	62.34	41.98	50.17
	Overall	68.68	30.44	**42.18**	37.35	41.71	**39.41**
SpaCy	LOC	73.38	75.36	74.36	77.04	56.64	65.28
	ORG	40.95	36.24	38.45	41.20	36.50	38.70
	PER	66.89	56.22	61.09	67.41	69.14	68.27
	Overall	60.94	49.01	**54.33**	66.15	54.32	**59.66**

TABLE III

DETAILED RESULTS OF THE EVALUATION STUDY. STANFORDNLP OUTPERFORMS THE OTHER SOFTWARE ON BOTH CORPUS, BUT THE DISPARITY IS NOT AS HIGH ON GMB THAN IT IS ON CONLL 2003 CORPUS.

- NLTK, Gate e OpenNLP traz muitas diferenças entre cada estudo, o que provavelmente deve ser causado pela diferenças de experimento (classifier, versão, etc.)
- Concluímos que é muito difícil de definir unicamente o melhor software

		[16]	[20]	[17]	Current study (Overall)
StanfordNLP	P	70	N/A	88	90.06
	R	70	N/A	87	73.67
	F1	70	89	87	**81.05**
NLTK	P	88	N/A	N/A	51.78
	R	89	N/A	N/A	45.56
	F1	89	N/A	N/A	**48.47**
Gate	P	N/A	78	N/A	61.48
	R	N/A	74	N/A	47.44
	F1	N/A	77	N/A	**53.55**
OpenNLP	P	88	N/A	N/A	68.68
	R	88	N/A	N/A	30.44
	F1	88	N/A	N/A	**42.18**

Our result is [0; 20]% distant from the corresponding study
Our result is [20; 40]% distant from the corresponding study
Our result is [40; 60]% distant from the corresponding study
Our result is [60; 100]% distant from the corresponding study

TABLE IV
RESULT COMPARISON ON CONLL 2003 BETWEEN OUR RESULTS AND EXISTING STUDIES.

A Comparative Analysis of Machine Learning Named Entity Recognition Tools for the Brazilian and European Portuguese Language Variants

Blind Review version

Introdução:

- Motivado pelo uso de text mining para estruturar e extrair informação de textos.
- Objetivo de pesquisar Machine Learning NER tools aplicados no português brasileiro e de portugal.
- HAREM corpus usado para treinar e avaliar.
- Resultados foram avaliados usando k-fold e precisão, recall e f-measure.
- 1.Stanford CoreNLP, 2. Apache OpenNLP, 3.spaCy.
- Variantes de português interfere no resultado.

Fundamentação Teórica

“A tarefa de Reconhecimento de Entidades Nomeadas (REN) ou Reconhecimento de Entidades Mencionadas (REM) é usada para identificar entidades nomeadas de textos escritos em forma livre e classificá-las em um conjunto pré definido de tipos, como: pessoa, organização e localização.
Na tarefa de reconhecimento de entidades nomeadas, vários algoritmos de aprendizado de máquina podem ser utilizados. Dentre eles, se destacam: Conditional Random Fields (CRF), Maximum-entropy Markov Model (MEMM), Support Vector Machine (SVM) e Árvore de Decisão [Jiang et al. 2012].”

Comparação de trabalhos relacionados

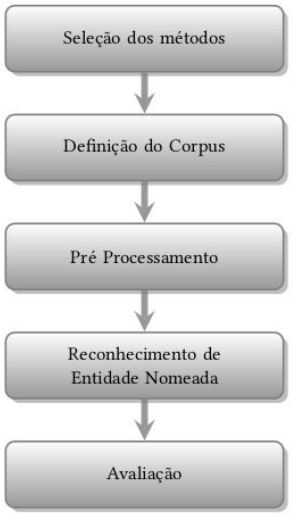
Table 2: Comparação dos trabalhos relacionados

Trabalho	Métodos	Pré-processamento	Métricas	Corpora	Entidades reconhecidas	Ranking Geral (F-measure)
Pires, Devezas e Nunes	CoreNLP OpenNLP SpaCy NLTK	Tokenização Segmentação de Sentenças POS tagging	Precisão Recall F-measure	HAREM	Pessoa, organização, tempo, local, obra, acontecimento, abstração, coisa, valor, variado	1. CoreNLP (56.10%) 2. OpenNLP (53.63%) 3. SpaCy (46.81%) 4. NLTK (30.97%)
Amaral, Fonseca Lopes e Vieira	FreeLing LanguageTasks PALAVRAS NERP-CRF	POS tagging	Precisão Recall F-measure	HAREM	Pessoa, local, organização	1. PALAVRAS (57%) 2. LanguageTasks (55%) 3. FreeLing (54%) 4. NERP-CRF (53%)
Fonseca, Chiele, Vieira e Vanin	OpenNLP	Remoção das POS Adição de marcação das categorias	Precisão Recall F-measure	Amazônia HAREM	Pessoa, organização, tempo, local, obra, acontecimento, abstração, coisa, valor, variado	1. OpenNLP (38,06%)

Métodos

- Corpus a ser usado bem definido (HAREM).
- Pré-processamento: tokenização, segmentação de sentenças.
- Ferramenta própria (haremfmt) que define o nível de especificidade de tratamento (documento, sentença e entidade).
- spaCy aceita o formato JSON como entrada, porém também possui um comando que transforma formatos como IOB, CoNLL para o formato requerido pela ferramenta.

Figure 1: Processo adotado neste trabalho para REN.



Métodos

“A ferramenta OpenNLP requer um formato de arquivo próprio como entrada. Neste arquivos, as entidades são anotadas a partir dos delimitadores start e end, como mostrado na Tabela 5. Então, o comando TokenNameFinderTrainer é executado com os parâmetros: arquivo em que o modelo será salvo, arquivo de entrada anotado, codificação do arquivo de entrada e língua.”

Table 5: Exemplo de anotação no formato do OpenNLP.

```
<START:PESSOA> João <END>
gosta de comer
<START:COISA> maçã <END>
```

“A ferramenta CoreNLP também requer que o arquivo esteja em um formato específico, no qual é somente possível ter um token por linha, tendo as anotações em outra coluna, separada por um carácter de tabulação. Quando não há anotação, o carácter “O” é usado. A Tabela 6 mostra o formato de entrada requerido por essa ferramenta.”

Table 6: Exemplo de anotação no formato do CoreNLP.

```
João PESSOA
gosta O
de O
maçã COISA
```

Avaliação

- Duas avaliações foram feitas: geral usando a coleção dourada do HAREM e outra usando comparação entre as variantes de portugues do corpus.
- Para avaliação geral foi usado cross-validation, k-fold (10 folds)
- Avaliação de frequência de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos.

Resultados

“Na avaliação por sentença e desconsiderando variantes, obteve-se o seguinte resultado: Stanford CoreNLP com o melhor F-measure (69,24%), seguido do Apache OpenNLP (63,50%) e spaCy (33,31%).”

Table 7: Avaliação geral a nível de parágrafo.

Ferramenta	Precisão	Recall	F-measure
Stanford CoreNLP	71,65%	66,99%	69,24%
Apache OpenNLP	65,39%	61,74%	63,50%
spaCy	36,47%	30,68%	33,31%

Conclusão

“Com base nos resultados, pôde-se perceber que a ferramenta Stanford CoreNLP apresenta um desempenho superior, embora a Apache OpenNLP possa ser mais indicada caso tempo e recursos computacionais sejam um fator importante. Já com relação à análise das variantes linguísticas para as ferramentas apresentadas, pôde-se verificar que há um impacto no modelo, já que a utilização de uma variante para treino e outra para teste resultou em uma queda no desempenho. ”

Benchmarking Named Entity Recognition Tools for Portuguese

André Pires, José Devezas, and Sérgio Nunes

Introdução:

- OBjetivo do trabalho é avaliar a performance comparativa entre Stanford CoreNLP, OpenNLP< spaCy e NLTK aplicados ao dataset HAREM.
- Os resultados mostram que StanfordNLP tem melhor f-measure (56,1%), OpenNLP(53,63%), spaCy 46,81%, e NLTK (30,97%)
- Estudos provam que para obter consistência, tem0se que haver uma avaliação sistemática, onde é preciso estabelecer quais pontuações serão feitas, e se haverá match parcial a ser considerado.
- Possibilidade de aplicação de hiperparâmetros como forma de melhorar a performance;

Related works and tools description:

- Inicialmente tinha-se técnicas de matching de padrões.
- Depois introduziu-se o uso de dicionários onde o algoritmo de extração baseia-se no match de texto-palavra. Isso garantiu mais precisão mas baixo recall e f-measure e , negativamente constante atualização dos dicionários devido às alterações ortográficas.
- ML foi introduzida. Inicialmente supervisionada, mas semi-supervisionada tem suas vantagens por precisar de menos exemplos para o treinamento.
- Critérios de escolha de software:
    - Software completamente livre/gratuito
    - Software é independente de linguagem e domínio
    - Software permite treinamento com modelo próprio de NER com entidades customizáveis.
- Nenhum dos modelos tem modelos em Português, sendo necessário utilizar algum corpus em português para treinamento.
- Stanford CoreNLP, OpenNLP e NLTK utilizam algoritmos conhecidos para execução de NER (CRF ou Max-entropy).
- spaCy utiliza uma implementação própria de um perceptron estruturado (Thinc linear model)

Table 1. Overview of the assessed tools.

Tool	License	Version	Language	NER classifiers	Default language models
Stanford CoreNLP	GNU General Public License	3.7.0 (2016)	Java	Conditional Random Fields	Arabic, Chinese, English, French, German, Spanish
OpenNLP	Apache License	1.7.2 (2017)	Java	Maximum Entropy	Dutch, English, Spanish
spaCy	MIT License	1.7.2 (2017)	Python	Thinc linear model	English, German
NLTK	Apache License	3.2.2 (2016)	Python	Naive Bayes, Decision Tree, Maximum Entropy	English

Corpus specifications:

- HAREM gold standard collection - conjunto de textos multi gêneros em português onde entidades nomeadas são identificadas, classificadas semanticamente e marcadas morfologicamente no contexto (Tagging).
- HAREM está no formato XML e no experimento não pode ser usado diretamente como input, sendo transformado para o uso em cada software/tool.

Tools/ Software running specifications

Stanford CoreNLP

- Precisa de um arquivo tokenizado de entrada, onde cada linha contém um token e separa as classes de entidade através de *tabs*.
- Após a execução do NER, adiciona-se IOB tagging aos output para poder fazer análise comparativa dos resultados.

Treinamento do classifier:

```
java - cp stanford - corenlp . jar edu . stanford . nlp . ie . crf .\
        CRFClassifier - prop < file . prop >
```

(file.prop são os hiperparâmetros)

Classificação dos documentos:

```
java - cp stanford - corenlp . jar edu . stanford . nlp . ie . crf .\
        CRFClassifier - loadClassifier <ner - model . ser . gz > - -\
        testFile < file_test . txt >
```

(file.test são os inputs não anotados a serem classificados)

Tools/ Software running specifications

OpenNLP

- Precisa a separação de uma sentença a cada linha na entrada, onde as entidades são anotadas com tag de início (<START:tag-name>) e fim (<END>)
- Converte-se as tags de entidade do HAREM em XML para o formato pedido.
- Usa-se o NLTK para segmentação das sentação para poder separar as sentenças no input.
- Treina-se o modelo com o input e classifica-se:
- Executamos o NER, convertemos o output para o formato CoNLL e adicionamos tags IOB.

Treinamento do modelo:

```
opennlp TokenNameFinderTrainer - model < model . bin > - lang <pt >\
        - data < training_data . txt > - encoding <UTF -8 >
```

Classificação do texto:

```
opennlp TokenNameFinder < model . bin > < < corpus_test . txt > > \ <output file >
```

Tools/ Software running specifications

spaCy:

- Precisa que o dataset de entrada esteja no formato standoff,ou seja, em dois arquivos: um com o texto pleno, e outro com as anotações de entidade, contendo separados por tabs com marcação de posição de beggining e ending da entidade, junto com a classe.
- O texto original foi separado em sentenças, usando a conversão feita pro OpenNLP.
- Arquivo resultante foi usado para treinar o modelo NER no spaCY (script no repositório do spaCy).
- NER classifiers mudam de EntityRecognizer para BeamEntityRecognizer. Verificar as mudanças na documentação.
- Resultado convertido para CoNLL e aplicado IOB.

Transformação do dataset:

1. Search until <START:tag> tag
2. Save starting position
3. Delete matched tag
4. Search for <END> tag
5. Save end position
6. Delete matched tag
7. Save standoff = (beginPos, endPos, tag)
8. Repeat from step 1 until no matches occur
9. Output to a tab separated file

Tools/ Software running specifications

NLTK

- Precisa que o dataset de input esteja no formato CoNLL com tags IOB. PRecisa que tenha os tags POS. OU seja, o input possui tab separation, onde cada linha tem tokens, POS tags e entity tag no formato IOB.
- Precisa de 3 passos: namely tokenizing and POS tagging, tokenizing while keeping entity tags and POS tagging, tokenizing while keeping the entity tags and join files.
- Treinamos os classificadores NLTK com NLTK trainers.
- Usa-se outro leitor de documentos: `nltk.corpus.reader.conll-.ConllChunkCorpusReader` , onde tem-se que especificar as entidades
- Classifica-se o texto, lembrando que o dataset precisa ter anotações de POS tag e classificado com chunker.parse(tagged)
- O parser retorna o resultado num formato de árvore que é então convertido para o CoNLL usando nltk.chunk.util.tree2conlltags(ner result).

Evaluation

- Apenas foram considerados matches exatos, ou seja, entity tags e limites tem que estar corretos para contar como match correto.
- Avaliação dos resultados usando tags IOB
- Foi feita 10-fold cross validation, com 4 repetições e calculada a média de precisão e recall e a macro-average do f-measure. Ou seja, para cada repetição, splitamos os dataset em 10 folds iguais, com sets diferentes de treinamento e teste, então rodamos cada ferramenta de NER para cada fold e também para cada repetição e nível (categoria, tipo e subtipo). Resultando em 480 runs.

Results

- Resultados extraídos da repetição de 10-fold cross validation.
- Tabela 4 mostra a comparação de todas as ferramentas para a maior level de classe entidade(categoria)
- Tabela 5 mostra a comparação entre as ferramentas usando a média do f-measure para todos os levels.
- NLTK sempre mostra um resultado inferior
- Stanford CoreNLP exige muita demanda computacional, sendo muito custoso executar para todos os levels

Table 4. Results for categories only.

Tool	Precision	Recall	F-measure
Stanford CoreNLP	58.84%	53.60%	56.10%
OpenNLP	55.43%	51.94%	53.63%
SpaCy	51.21%	43.10%	46.81%
NLTK	30.58%	31.38%	30.97%

Table 5. F-measure for all levels.

Tool	Categories	Types	Subtypes
Stanford CoreNLP	56.10%	-	-
OpenNLP	53.63%	48.53%	50.74%
SpaCy	46.81%	44.04%	37.86%
NLTK	30.97%	28.82%	21.91%

Results

- Para o NLTK, pode-se executar classifiers diferentes:

Table 6. Results for the category level in NLTK, for all classifiers.

Classifier	Precision	Recall	F-measure
NaiveBayes	30.58%	31.38%	30.97%
Maxent	18.19%	0.58%	1.13%
DecisionTree	21.84%	25.72%	23.62%

- Variação de hiperparâmetros: como cada ferramenta é muito custosa, não é possível testar todos os parâmetros, então proforma-se split de 70% dos dados de treino e 30% dos dados de testing. Isso leva a valores diferentes, mas dá-se ideia de como cada parâmetro pode influenciar no resultado.

Table 7. Summary results for all tools, category level.

Tool	Default F-measure	Best Configurations	Best F-measure
OpenNLP	50.90%	cutoff=4	52.38%
		iterations = 170	51.52%
SpaCy	45.70%	iterations=110	46.60%
Stanford CoreNLP	54.14%	tolerance=1e-3	54.31%
NLTK DT	26.14%	entropy_cutoff=0.08	26.36%
		support_cutoff=16	26.18%
NLTK ME	1.11%	min_lldelta=0, iterations=100	35.24%

Conclusions

- O melhor f-measure configura pro Stanford CoreNLP(56,1%) que possui o melhor resultado em comparação com o esperado (1% de diferença).

Referências:

https://www.g2.com/categories/natural-language-understanding-nlu

https://nanonets.com/blog/named-entity-recognition-with-nltk-and-spacy/#:~:text=Named%20entity%20recognition%20(NER)%20is,person%2C%20location%2C%20organisation%20etc.

https://pythonprogramming.net/named-entity-recognition-nltk-tutorial/

https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

[1] André Pires, José Luís Devezas, and Sérgio Nunes. 2017. Benchmarking Named Entity Recognition Tools for Portuguese.

[2]Schmitt, Xavier & Kubler, Sylvain & Robert, Jérémy & Papadakis, Mike & LeTraon, Yves. (2019). A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate. 338-343. 10.1109/SNAMS.2019.8931850.

[3]Pinto, Alexandre & Gonçalo Oliveira, Hugo & Alves, Ana. (2016). Comparing the Performance of Different NLP Toolkits in Formal and Social Media Text. 51. 3:1-. 10.4230/OASIcs.SLATE.2016.3.

[4]Dozier, Christopher & Light, Marc & Vachher, Arun & Veeramachaneni, Sriharsha & Wudali, Ramdev. (2010). Named Entity Recognition and Resolution in Legal Text. 27-43. 10.1007/978-3-642-12837-0_2.