



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

FRAMEWORK DE EXTRAÇÃO E ETIQUETAMENTO DE INFORMAÇÕES DE
TRÂNSITO PARA LÍNGUA PORTUGUESA

LEONARDO DOS ANJOS TETÉO

Orientador

CARLOS ALBERTO VIEIRA CAMPOS
PEDRO NUNO DE SOUZA MOURA

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2017

Catálogo informatizada pelo autor

T347 Tetéo, Leonardo dos Anjos
Framework de extração e etiquetamento de
informações de trânsito para língua portuguesa /
Leonardo dos Anjos Tetéo. -- Rio de Janeiro, 2017.
76 f

Orientador: Carlos Alberto Vieira Campos.
Coorientador: Pedro Nuno de Souza Moura.
Trabalho de Conclusão de Curso (Graduação) -
Universidade Federal do Estado do Rio de Janeiro,
Graduação em Sistemas de Informação, 2017.

1. cidades inteligentes. 2. aprendizado de
máquina. 3. Conditional Random Fields. 4. Twitter.
5. framework. I. Campos, Carlos Alberto Vieira,
orient. II. Moura, Pedro Nuno de Souza, coorient.
III. Título.

FRAMEWORK DE EXTRAÇÃO E ETIQUETAMENTO DE INFORMAÇÕES DE
TRÂNSITO PARA LÍNGUA PORTUGUESA

LEONARDO DOS ANJOS TETÉO

Projeto de Graduação apresentado à Escola de
Informática Aplicada da Universidade Federal do
Estado do Rio de Janeiro (UNIRIO) para obtenção do
título de Bacharel em Sistemas de Informação.

Aprovado por:

CARLOS ALBERTO VIEIRA CAMPOS (UNIRIO)

PEDRO NUNO DE SOUZA MOURA (UNIRIO)

CARLOS EDUARDO RIBEIRO DE MELLO (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

DEZEMBRO DE 2017

Agradecimentos

Primeiramente, agradeço a Deus porque sem ele nada seria possível. Em segundo lugar agradeço a minha família, principalmente a minha mãe, quem me apoiou desde o início a buscar o sonho de possuir uma educação em nível superior apesar das dificuldades. Também dedico este trabalho ao meu irmão, que foi grande fonte de inspiração na caminhada na área de computação e Sistemas de Informação. Por fim, agradeço a todos os professores que passaram pela vida e transmitiram o conhecimento que hoje demonstro neste trabalho, não só os professores da Universidade Federal do Estado do Rio de Janeiro, o qual agradeço imensamente por todo o conhecimento adquirido sobre Sistemas de Informação, mas também a professores do ensino médio e fundamental. Foi uma longa caminhada e agradeço a todas as pessoas que de algum modo me auxiliariam neste caminho.

RESUMO

Com o grande avanço na tecnologia e nos meios de comunicação, a humanidade passou a depender da internet, bem como a informação que esta provê, sendo possível obter informação de qualquer parte do mundo em tempo real na palma de sua mão. O excesso de informação, porém, é um problema, até mesmo no contexto de uma única cidade, onde muitos eventos ocorrem em paralelo. Neste contexto, este trabalho descreve o desenvolvimento de um *framework* que tem como função facilitar a extração, tratamento e identificação de eventos e sua localidade em *tweets* escritos na Língua Portuguesa utilizando aprendizado de máquina, mais especificamente, a técnica *Conditional Random Fields* (CRF) para realizar Reconhecimento de Entidades Nomeadas. Os resultados obtidos para um modelo CRF treinado a partir de dados coletados, assim como um caso de uso do framework, demonstram o potencial da ferramenta em identificar eventos de trânsito importantes na cidade do Rio de Janeiro, de modo a facilitar ao usuário a busca de informações sobre eventos na localidade de seu interesse.

Palavras-chave: cidades inteligentes, aprendizado de máquina, *Conditional Random Fields*, Twitter, *framework*

ABSTRACT

With the great development of technology and media, humanity became dependent of the internet as well as of the information that it provides, being possible to obtain information from around the world in real-time on the palm of one's hand. The excess of information, however, is a problem, even in the context of an only city, where many events occur in parallel. In this context, this paper describes the development of a framework that seeks to facilitate the extraction, treatment and identification of events and its location in tweets written in Portuguese language using machine learning, more specifically, Conditional Random Fields (CRF) to do Named Entity Recognition (NER). The results obtained for a trained CRF model from data collected as well as the use of the framework demonstrated the potential of the tool in identifying important traffic events in the city of Rio de Janeiro, easing for the user the search for information about events in his location of interest.

Keywords: smart city, machine learning, Conditional Random Fields, Twitter, framework

Índice

1	Introdução.....	13
1.1	Motivação	13
1.2	Objetivos.....	14
1.3	Organização do texto	15
2	Revisão Bibliográfica.....	16
2.1	Cidades Inteligentes & Internet of Things (IoT)	16
2.2	Sistemas de Recomendação.....	19
2.3	Extração de conhecimento de redes sociais em cidades inteligentes	22
2.4	Aprendizado de máquina	25
2.1.1	N-Gram	25
2.1.2	Named Entity Recognition.....	26
2.1.3	Conditional Random Fields (CRF)	27
2.1.4	Convolutional Neural Network (CNN).....	31
2.1.5	Outras técnicas de Named Entity Recognition	33
3	O Framework.....	34
3.1	Visão geral.....	34
3.2	Componentes	37
3.2.1	Ferramenta de extração e tratamento	38
3.2.1.1	Twitter API	39
3.2.1.2	Tratamento e armazenamento	41
3.2.2	Banco de dados	44
3.2.3	Message Broker (RabbitMQ).....	45
3.2.4	Ferramenta de Etiquetamento e Comunicação	46
3.2.4.1	Etiquetamento	47
3.2.4.2	Extração e Geolocalização	49
3.2.4.3	Treinamento	51

4	Resultados	55
4.1	Resultados preliminares.....	55
4.2	Resultados finais.....	64
5	Caso de Uso.....	68
6	Conclusão e trabalhos futuros	73

Lista de Tabelas

Tabela 1 - Resultados retirados de [Farajidavar et al. 2017]	32
Tabela 2 - Resultado do teste de desempenho do banco de dados	44
Tabela 3 - Números das amostras de treinamento	53
Tabela 4 - Precisão, Recall e F-Measure dos modelos 1, 2 e 3	57
Tabela 5 - Precisão, Recall e F-Measure do modelo 4	58
Tabela 6 - Precisão, Recall e F-Measure do modelo 5	58
Tabela 7 - Precisão, Recall e F-Measure do Modelo 6.....	59
Tabela 8 - Precisão, Recall e F-Measure do modelo 7	60
Tabela 9 - Precision, Recall e F-Mesure do modelo 8.....	61
Tabela 10 - Precisão, Recall e F-Measure do modelo 9	62
Tabela 11 - Resultados dos testes finais	65
Tabela 12 - Matriz de confusão referente à Amostra Final 1	66
Tabela 13 - Matriz de confusão referente à Amostra Final 2	66
Tabela 14 - Matriz de Confusão referente à Amostra Final 3	66

Lista de Figuras

Figura 1 - Cidades inteligentes e suas diferentes áreas (adaptada de [Hancke and De Silva 2013])	17
Figura 2 - Sistema de recomendação em três dimensões (adaptada de [Adomavicius et al. 2005]).....	21
Figura 3 - Exemplo de tweet etiquetado	27
Figura 4 - Grafo básico gerado por um modelo CRF	28
Figura 5 - Exemplo de grafo com tokens e tags	28
Figura 6 - Exemplo mais complexo de um grafo CRF.....	29
Figura 7 - Funcionamento de um CNN [Britz 2015]	31
Figura 8 - Visão geral do framework	35
Figura 9 - Diagrama de classe de dados da aplicação	38
Figura 10 - Exemplo de utilização da biblioteca Tweetinvi em C#	41
Figura 11 - Trecho de código que mostra a utilização de paralelismo	43
Figura 12 - Tabela do banco de dados	45
Figura 13 - Envio de mensagem com conceito de produtor-consumidor (adaptado de [Pivotal 2017]).....	46
Figura 14 - Exemplo de utilização da biblioteca StanfordNER	48
Figura 15 - Exemplo de tweet etiquetado	48
Figura 16 - Resultado do Google Places API Web Service	50
Figura 17 - Resultado da Nominatim API.....	50
Figura 18 - Processo de treinamento	52
Figura 19 - Gráfico comparativo entre modelos (Amostra 2.1)	55
Figura 20 - Gráfico comparativo entre modelos.....	57
Figura 21 - Gráfico comparitvo dos modelos 1 a 5	58
Figura 22 - Resultados do Modelo 6	59
Figura 23 - Resultados do modelo 7	60
Figura 24 - Resultados do modelo 8.....	61
Figura 25 - Resultados do modelo 9	62
Figura 26 - Gráficos de comparação entre modelos 6 a 9	63
Figura 27 - Visão geral da aplicação de mapa.....	69
Figura 28 - Tweet de evento	70
Figura 29 - Avaliação de precisão da API.....	70

Figura 30 - Evento ocorrido na região do bairro da Freguesia.....	71
Figura 31 - Evento ocorrido na região próximo a rodoviária	72

1 Introdução

1.1 Motivação

Uma das grandes revoluções deste século foi a introdução da tecnologia em nossas vidas diárias. Desde o final da década de 90 até os dias atuais, a vida pessoal e profissional mudou muito. A tecnologia tomou conta de cada aspecto da vida humana tornando-se algo essencial e a cada dia evoluindo mais. A criação da internet em 1989 foi o começo da nova era conhecida como Era da Informação. Após um período de alto desenvolvimento tecnológico, em que destaca a invenção da internet banda larga, as redes sociais e a revolução do *mobile*, hoje há uma dependência tanto da internet quanto da informação proporcionada tal como se depende de energia e água.

A evolução nos aspectos mencionados permitiu que qualquer pessoa tenha toda a informação que queira na palma de sua mão em quase tempo praticamente real, ou por vezes em tempo real. Isso proporciona muitos benefícios, mas também gera desafios com a grande carga de informação recebida. A sobrecarga de informação pode fazer com que os usuários não cheguem no resultado que esperam. Dessa forma, ainda há muitos avanços a serem realizados e, neste sentido, a área de Big Data e extração de conhecimento estão em seu melhor momento abordando estes desafios.

A tecnologia não só beneficiou indivíduos, mas também cidades inteiras. O conceito de cidades inteligentes está crescendo cada vez mais. Governos, principalmente em países desenvolvidos, estão se voltando para a tecnologia para obter dados nunca antes pensados e observados sobre as cidades e estão realizando melhorias na vida de seus cidadãos. Muitas pesquisas têm sido desenvolvidas neste contexto, que é amplo e envolve diferentes áreas do conhecimento. A tecnologia da informação também está presente sendo *Big Data* e extração de conhecimento dois campos presentes nas pesquisas sobre cidades inteligentes. Muitas das pesquisas mais atuais que unem cidades inteligentes e extração do conhecimento utilizam as redes sociais como protagonista na pesquisa, em que o Twitter uma das mais utilizadas devido à sua característica de atualização em tempo real.

Dito isto, a motivação deste trabalho é utilizar o que há de mais novo na área de extração de conhecimento para promover o desenvolvimento da pesquisa na área de

cidades inteligentes no Brasil, utilizando a língua portuguesa como língua principal, algo ainda escasso, mas que traz grande potencial no auxílio ao desenvolvimento das cidades brasileiras e no aumento da qualidade de vida do cidadão, de modo a prover a cada um a informação que necessita, quando necessita.

1.2 Objetivos

Este trabalho tem como objetivo o desenvolvimento de um *framework* de ferramentas que poderá ser utilizado para extrair os dados do Twitter, tratar estes dados e abordar o problema de Reconhecimento de Entidades Nomeadas (*Named Entity Recognition – NER*). Para que este objetivo seja alcançado é necessário que vários objetivos intermediários sejam cumpridos.

Em primeiro lugar é necessário compreender o estado-da-arte realizando uma revisão bibliográfica sobre o tema, em particular focando nas ferramentas e técnicas utilizadas na extração de conhecimento do Twitter no contexto de cidades inteligentes. Este passo é importante para garantir que as melhores técnicas possam ser avaliadas e testadas para o português, de modo a enriquecer o conhecimento sobre elas na abordagem à língua portuguesa, que é ainda um conhecimento incipiente.

O segundo objetivo intermediário é a adaptação de algoritmos de extração, tratamento e identificação para a língua portuguesa. Muitos destes estão preparados para utilização na língua inglesa, que, apesar de ter algumas características em comum com o português, também possuem muitas diferenças que impactam nos passos mencionados. Além disso, o Twitter e a linguagem coloquial utilizada nele também impõem outros desafios que eventuais modelos criados para o português não resolvem, sendo necessário realizar um profundo trabalho de adaptação e criação de modelos próprios para a utilização no Twitter em língua portuguesa.

Estes dois objetivos intermediários fazem parte do desenvolvimento do *toolkit* para identificação de entidades nomeadas (NER) no Twitter. Este toolkit terá como objetivo facilitar a utilização do Twitter para extração de conhecimento, principalmente no contexto de cidades inteligentes, tendo ferramentas específicas para este objetivo como, por exemplo, ferramentas de geolocalização utilizando os dados do Twitter para identificar onde eventos estão ocorrendo independentemente das coordenadas fornecidas pelo Twitter em *tweets* com esta funcionalidade habilitada. Estas coordenadas podem não

estar disponíveis ou podem ser incompatíveis com o local exato onde o evento está ocorrendo. Assim, retirar esta informação do corpo pode, em tese, melhorar a precisão da informação. A precisão e eficácia do método utilizado pela ferramenta serão avaliados para determinar o grau de utilidade do *toolkit*.

1.3 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo II: neste capítulo realiza-se uma revisão bibliográfica sobre cidades inteligentes, bem como aprendizagem de máquina, trazendo trabalhos similares com comentários sobre as diferenças com o presente trabalho.
- Capítulo III: neste capítulo o framework desenvolvido é apresentado, revelando seu funcionamento, recursos e componentes. Também são discutidos decisões de implementação fundamentados em argumentos técnicos.
- Capítulo IV: neste capítulo são apresentados os resultados da etapa de treinamento do modelo de *Conditional Random Fields (CRF)* utilizado no trabalho.
- Capítulo V: neste capítulo demonstra-se um caso de uso do framework através de uma simples aplicação de mapa de eventos.
- Capítulo VI: neste capítulo são expostas as conclusões e sugestões de trabalhos futuros.

2 Revisão Bibliográfica

2.1 Cidades Inteligentes & Internet of Things (IoT)

Desde a Revolução Industrial em 1760, o ser humano deixou o modo de vida rural e passou cada vez mais a buscar as cidades para morar, trabalhar e construir suas vidas. Hoje 54% das pessoas moram em cidades no mundo, no Brasil são mais de 85% e em países de primeiro mundo o percentual está sempre entre 75% até aproximadamente 90% [CIA 2017] e este grande número tem se tornado um desafio para governos e os próprios cidadãos, que estão sempre em busca de uma qualidade de vida adequada. No entanto, como obtê-la em grandes cidades onde problemas surgem com o acúmulo de pessoas? Poluição, engarrafamentos constantes, acidentes, crimes e outros problemas só aumentam com o crescimento populacional das cidades. Hoje, com o advento da tecnologia e a era da informação, todos têm se voltado para a tecnologia para buscar soluções para problemas dos mais variados tipos e as cidades são grandes protagonistas com o conceito de cidades inteligentes.

Cidades inteligentes são um conceito muito abrangente que envolve vários aspectos de uma cidade, desde o transporte e trânsito, à saúde. Desta forma, existem diferentes definições para cidades inteligentes que focam em diferentes aspectos.

Sustentabilidade e eficiência são duas características muito valorizadas entre as definições de cidades inteligentes. Em [Hancke and De Silva 2013] se define cidades inteligentes como cidades que utilizam dispositivos inteligentes para monitorar e controlar a infraestrutura e serviços, garantindo eficiência e sustentabilidade. Esta definição é mais voltada ao monitoramento e controle, que em [Hancke and De Silva 2013] é utilizada por meio de sensores, que desempenham várias funções. Na Figura 1 adaptada de [Hancke and De Silva 2013] podemos ver as diferentes áreas onde cidades inteligentes podem ser usadas, de acordo com o artigo.

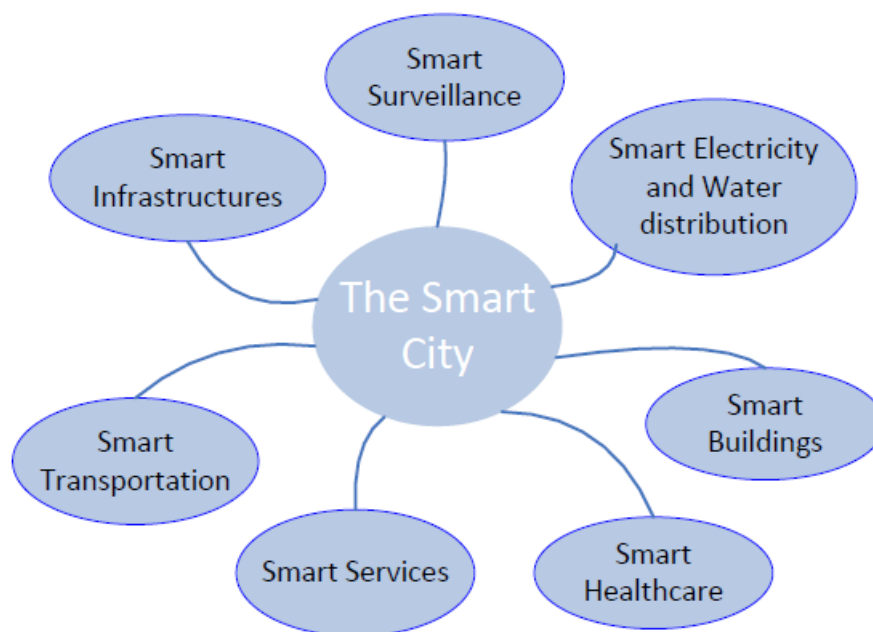


Figura 1 - Cidades inteligentes e suas diferentes áreas (adaptada de [Hancke and De Silva 2013])

O artigo descreve vários exemplos em que sensores podem ser utilizados em várias das áreas ilustradas, como por exemplo em *Smart Transportation* sensores podem ser utilizados para determinar a intensidade e velocidade do trânsito contribuindo para o planejamento de novas vias e aperfeiçoamento de semáforos, que também podem ser inteligentes. Em *Smart Electricity and Water distribution* (Distribuição de eletricidade e água inteligente) é possível utilizar sensores para identificar vazamentos de água ou cortes na eletricidade. Também é possível melhorar o consumo desses recursos utilizando sensores para coletar dados, utilizando medidores inteligentes.

Em [Neirotti et al. 2014] cidades inteligentes também são divididas em diferentes domínios que englobam diferentes aspectos de uma cidade. O artigo também separa estes domínios em *hard domains* e *soft domains* no que diz respeito a como a tecnologia é utilizada em cada domínio. *Hard domains* como transportes, energia e segurança pública utilizam a tecnologia por meio de sensores que monitoram e controlam estes domínios, similar ao que é afirmado em [Hancke and De Silva 2013]. Em “*soft domains*” como educação, inclusão social e administração pública a tecnologia tem uma influência mais limitada possuindo um papel mais auxiliador. Muitas das áreas ilustradas em [Hancke and De Silva 2013] na Figura 1 podem ser descritas como “*hard domains*” corroborando com a definição de [Hancke and De Silva 2013] a respeito de cidades inteligentes. Porém,

podemos ver que sensores não podem ser utilizados em todas as áreas ou em todos os momentos.

[Zanella et al. 2014] fala sobre a indefinição do significado do conceito de cidades inteligentes, mas também a define de uma forma mais voltada à parte econômica e governamental ao também fazer alusão à sustentabilidade e à eficiência, com o objetivo de reduzir custos para a administração e aumentando a qualidade. Esse artigo fala também sobre *Internet of Things* (IoT), assunto também abordado em [Hancke and De Silva 2013] e [Neirotti et al. 2014], dando vários exemplos de como IoT pode auxiliar no desenvolvimento de soluções para diferentes problemas, entre eles: gerenciamento de lixo, congestionamentos, consumo de energia, monitoramento de poluição sonora, etc. [Zanella et al. 2014], em particular, define IoT como um novo paradigma que colocará dispositivos do dia-a-dia na internet através de componentes de rede e protocolos próprios. Muitos destes problemas fazem parte tanto das áreas apresentadas por [Hancke and De Silva 2013], quanto os domínios apresentados por [Neirotti et al. 2014]. Assim, embora haja certa diferença entre as definições, é possível perceber que há um acordo entre pesquisadores que cidades inteligentes e IoT tem papel importante em áreas como trânsito, segurança, consumo de energia e água, entre outras áreas.

O artigo também lista várias características que IoT em cidades inteligentes deve possuir. Como dito, cidades inteligentes englobam diversos aspectos de uma cidade, muitos deles sem qualquer relação um com o outro à primeira vista e, quando se trata de tecnologia, possuem diferentes necessidades e demandam diferentes dispositivos. No entanto, cidades inteligentes também necessitam possuir uma arquitetura centralizada, de acordo com [Zanella et al. 2014]. De fato, governos e entidades com esta responsabilidade serão o centro administrativo desta infraestrutura. É preciso, o máximo possível, utilizar a arquitetura já existente e interconectar esta rede contendo dispositivos heterogêneos em uma entidade centralizadora e isto demanda uma série de desafios.

A grande variedade de domínios, muitos sem qualquer relação, leva a soluções que somente abordam um domínio, com características específicas para um determinado problema, geralmente feitas do zero e sem reuso levando a incompatibilidades [Santana et al. 2016]. Isto vai totalmente contra as definições de cidades inteligentes que advogam por eficiência e sustentabilidade e contradiz especialmente a [Zanella et al. 2014], que aborda, de uma forma técnica, a necessidade de fazer interconexão entre sistemas heterogêneos. [Santana et al. 2016] também destaca a necessidade de se obter interoperabilidade entre os variados sistemas, além de abordar a garantia de privacidade

dos cidadãos, e o desafio da grande carga de dados, também abordado em [Ang and Seng 2016], e variedade de sensores, além de escalabilidade para lidar com o crescimento rápido das cidades. Para isto o artigo apresenta uma plataforma de software que em tese será capaz de auxiliar na solução destes desafios se tornando um *middleware* que irá auxiliar no *design*, implementação, implantação e administração de aplicações para cidades inteligentes. [Wang et al. 2012] propõe outra solução baseado no uso de uma ontologia que represente o conhecimento e seja um ponto unificador entre os sistemas heterogêneos, garantindo a interoperabilidade entre eles.

O problema de interoperabilidade é um dos principais problemas de pesquisa relacionados à utilização de sensores no contexto de cidades inteligentes. No entanto, existem outros problemas que fazem com que, em certos casos seja mais eficiente ou econômico outras soluções. Em [Doran et al. 2015] são enumerados motivos sólidos para a utilização de redes sociais em vez de sensores físicos. Além do problema de interoperabilidade já mencionado, o artigo alerta também sobre problemas de confiabilidade. Tanto o desafio de interoperabilidade como a confiabilidade geram custos de busca de soluções e manutenção, respectivamente, e o custo só aumenta com o grande número de sensores que devem ser implantados. No entanto, o motivo mais interessante que [Doran et al. 2015] traz é o fato de que os sensores trazem informações importantes sobre o que aconteceu, mas são incapazes de identificar as razões dos eventos detectados. [Anantharam et al. 2015] corrobora com as razões de [Doran et al. 2015] afirmando que sensores que oferecem a granularidade de informação necessária nem sempre estão disponíveis. De fato, sensores mais sofisticados geram mais custos que nem sempre estão disponíveis. Assim, as redes sociais são vistas como uma alternativa interessante para países como o Brasil que não possuem uma infraestrutura ou economia fortes como países desenvolvidos, onde o conceito de cidades inteligentes está mais desenvolvido.

2.2 Sistemas de Recomendação

Os sistemas de recomendação são também de grande interesse para a comunidade de pesquisadores em cidades inteligentes já que estes organizam e trazem a informação necessária pelo usuário no contexto apropriado. Esta organização é necessária nos dias de

hoje já que não só o ser humano vive em grandes cidades onde muitos eventos acontecem em paralelo, mas também por causa da grande quantidade de informação gerada no mundo inteiro, não só localmente como era até duas décadas atrás. A solução apresentada neste documento tem como objetivo auxiliar também em sistemas de recomendação que podem ser desenvolvidos utilizando as ferramentas aqui descritas. Deste modo, uma revisão bibliográfica sobre sistemas de recomendação também se torna necessária para entender o que esses sistemas requerem e como as ferramentas desenvolvidas podem auxiliar em seu objetivo.

Existem diferentes tipos de sistemas de recomendação, sendo alguns clássicos na literatura sobre o assunto. [Burke 2002] lista técnicas de recomendação fundamentais:

- Colaborativo: avaliações de outros usuários em itens são utilizadas para recomendar itens para outro usuário.
- Baseado em conteúdo: ver as características de um item e verificar as avaliações do usuário para itens com características similares para criar recomendações.
- Demográfico: compara informações demográficas do usuário com outros usuários e verifica quais itens os outros usuários avaliaram melhor para formar recomendações.
- Baseado em utilidade: Baseia-se nas características do item para determinar a utilidade daquele item para um usuário de acordo com suas preferências.
- Baseado em conhecimento: utiliza o conhecimento sobre como aquele item preenche as necessidades do usuário e juntamente com a descrição das necessidades e interesses do usuário forma recomendações.

O artigo destaca os pontos positivos e negativos de cada técnica, sendo dois problemas os mais conhecidos:

- Problema do novo usuário: técnicas que dependem de avaliações do usuário sofrem com novos usuários, que possui, naturalmente, poucas avaliações, sendo difícil classificá-lo.
- Problema do novo item: novos itens que não possuem muitas avaliações também são difíceis de recomendar.

Para resolver estes e outros problemas relacionados as técnicas descritas, o artigo propõe um sistema de recomendação híbrido que misture diferentes técnicas que se complementam, diminuindo o número de desvantagens de cada técnica.

[Adomavicius et al. 2005] e [Adomavicius and Tuzhilin 2015] propõem uma nova abordagem de sistemas de recomendação baseados em contexto. Os artigos afirmam que em vários contextos como na recomendação de pacotes de viagem, recomendação de produtos e filmes, não é suficiente somente levar em consideração usuários e itens. O

contexto em que um usuário está inserido pode mudar dramaticamente o produto que irá comprar ou o filme que irá assistir. Por exemplo, os filmes que um determinado usuário assiste com a família ou com a namorada podem ser drasticamente diferentes. Todas as técnicas mencionadas em [Burke 2002] também foram mencionadas em [Adomavicius et al. 2005] e [Adomavicius and Tuzhilin 2015] para ilustrar as diferenças entre os diferentes sistemas e um sistema de recomendação baseado em contexto. Para que este sistema seja possível os artigos utilizam uma abordagem multidimensional que pode ser melhor entendida levando em consideração a figura de um cubo como mostrado na Figura 2 retirada de [Adomavicius et al. 2005].

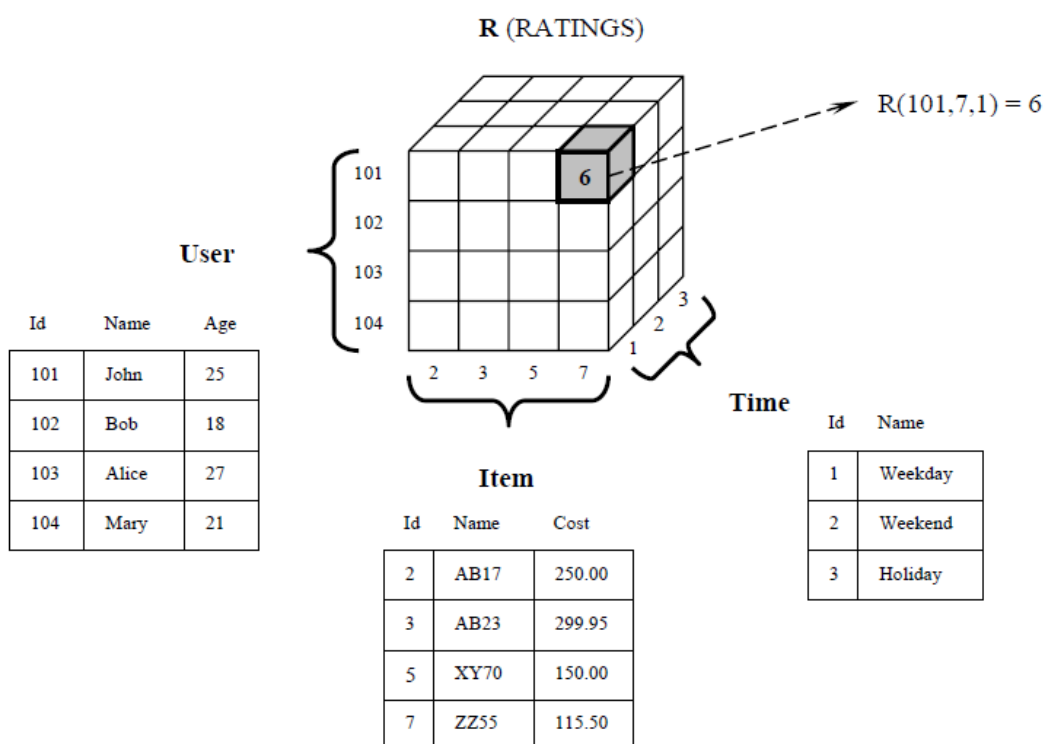


Figura 2 - Sistema de recomendação em três dimensões (adaptada de [Adomavicius et al. 2005])

O exemplo da Figura 2 mostra um sistema de recomendação que leva em consideração um espaço de tempo (dia de semana, fim de semana ou feriado), que é um tipo de contexto, no sistema de recomendação. Assim, um determinado usuário terá diferentes avaliações sobre um item baseado no contexto (neste caso, o tempo) em que está inserido.

Este tipo de sistema de recomendação é especialmente interessante no contexto de cidades inteligentes onde o contexto é bastante importante para diferentes tipos de recomendação. Por exemplo, uma rota entre dois pontos em uma cidade não pode somente levar em consideração a distância e tempo médio de rota, mas também o que pode estar por ventura estar acontecendo naquele momento em dada rota. Esta é uma das preocupações abordadas por [Bajaj et al. 2016] em seu sistema de recomendação de rotas para transportes públicos. O artigo propõe um sistema de recomendação de rotas para transportes públicos que levam em conta a conveniência do usuário e o fato de que as recomendações devem ser feitas em tempo real devido ao alto dinamismo do trânsito nas grandes cidades.

Embora o objetivo deste trabalho não seja utilizar sistemas de recomendação diretamente, a revisão bibliográfica sobre sistemas de recomendação se mostra extremamente importante para que se possa identificar as necessidades destes sistemas que serão clientes do *framework* proposto por este trabalho. Uma das funcionalidades do *framework* será prover contexto para sistemas de recomendação indicando eventos que estão acontecendo em determinadas localidades na cidade, podendo esta informação ser facilmente inserida num processo de recomendação de rotas, por exemplo.

2.3 Extração de conhecimento de redes sociais em cidades inteligentes

As redes sociais têm se tornado parte da vida diária das pessoas durante a década atual. De acordo com pesquisa da Statista [Statista 2017], 2.46 bilhões de pessoas utilizam redes sociais no mundo em 2017 e a expectativa é que este número chegue a 3 bilhões até 2021. Entre as redes sociais mais populares, o Facebook domina o mercado com mais de 2 bilhões de usuários, seguido por YouTube, WhatsApp e Facebook Messenger. Twitter aparece em 11º lugar com 328 milhões de usuários. Porém, o Twitter é a rede social mais utilizada em pesquisas de extração de conhecimento, principalmente no campo de cidades inteligentes devido à sua natureza de publicação em tempo real. Ainda de acordo com a Statista [Statista 2017], 456 mil *tweets* são enviados por minuto e apesar de ser a 11ª rede social mais utilizada o Twitter figura em 2º lugar em número de conteúdo compartilhado. O fato de que 78% dos acessos a redes sociais seja por dispositivos móveis também ajuda

na reação imediata de usuários a eventos que ocorrem ao seu redor e muitos recorrem ao Twitter para um comentário rápido a respeito de tais eventos.

Apesar desse potencial e da grande quantidade de informação disponível, [Farajidavar et al. 2017] afirma que este tipo de técnica é geralmente ignorada como uma técnica que auxilia e complementa a informação retirada de sensores tradicionais. Ainda assim, pesquisas neste sentido surgiram como em [Crooks et al. 2013], [Itoh et al. 2014] e [Miranda et al. 2016].

[Crooks et al. 2013] compara os sensores tradicionais com redes sociais afirmando que de certo modo seres humanos se comportam como sensores ao comentar em redes sociais eventos ocorridos à sua volta. Porém, existem certas dificuldades já que as informações em redes sociais não são facilmente extraídas e precisam ser mineradas e tratadas. O tema principal do artigo é a utilização do Twitter como ferramenta de informação em situações de desastre natural, mais especificamente um terremoto. O artigo conclui que o Twitter pode ser utilizado para este fim, sendo até mesmo mais eficiente do que outras ferramentas oficiais. Como os usuários de redes sociais estão aglomerados nos locais mais importantes em questão de monitoramento: onde há pessoas [Crooks et al. 2013], como cidades, essa abordagem constitui uma excelente alternativa em relação aos métodos tradicionais.

[Itoh et al. 2014] utiliza dados dos cartões de embarque do metrô de Tóquio e o Twitter para observar o fluxo de passageiros e as mudanças neste fluxo em razão de eventos ocorridos. Neste caso, o Twitter é utilizado para extrair as razões da anormalidade no fluxo de passageiros, algo que corrobora com o pensamento de [Doran et al. 2015] a respeito de que redes sociais são capazes de fornecer as justificativas para a ocorrência de um determinado evento. Estudos de caso mostrados concluem que o Twitter mostra as razões por trás de um determinado evento, como uma tempestade, por exemplo, de forma bastante sólida. Estes dados podem ser utilizados para o aperfeiçoamento do serviço em situações extraordinárias.

[Miranda et al. 2016] utiliza o Flickr como a rede social principal de seu estudo, mas também utiliza o Twitter para obter dados culturais de Manhattan. O Flickr é utilizado para obter a taxa de atividade em determinados pontos da cidade em diferentes partes do dia, semana, mês ou ano. O objetivo principal do estudo com o levantamento destes dados é auxiliar arquitetos e pessoas responsáveis por planejamento urbano na criação de novos espaços urbanos como praças e parques ao informar os hábitos e tendências dos cidadãos em determinados locais da cidade. O método foi elogiado por

estes profissionais em estudos de caso. Porém, hoje seria mais adequado utilizar o Instagram para um estudo similar já que esta é a rede social de publicação de fotos mais popular no mundo hoje [Statista 2017]. O Twitter foi utilizado para observar o perfil comportamental de diferentes etnias que vivem em Manhattan, mostrando que a utilização do Twitter vai além da notificação de eventos como mostrado em [Crooks et al. 2013] e [Itoh et al. 2014].

Existem diferentes formas de extrair o conteúdo desejado do Twitter. É possível utilizar-se dos recursos de pesquisa da própria API do Twitter, como realizado por [Crooks et al. 2013] e [Miranda et al. 2016]. O primeiro utilizou *hashtags* (palavras que começam com “#” que indica o tópico do *tweet*) para descobrir *tweets* relacionados ao terremoto estudado pelos autores. Já o segundo utilizou os dados trazidos pela API para filtrar os *tweets* por idioma e assim possuir uma visão cultural das diferentes etnias presentes em Manhattan. Porém, a utilização dos recursos da API é limitada a buscas bem específicas, em que se deve determinar precisamente quais tópicos, palavras ou usuários se deseja pesquisar.

Como dito anteriormente, o objetivo deste trabalho é criar um conjunto de ferramentas que utilize os dados do Twitter para detectar eventos e sua localização. Uma observação importante é que a localização do evento não será retirada dos dados de geolocalização do *tweet*, mas sim do corpo do *tweet*. Essa decisão foi tomada devido ao fato de que a maioria dos *tweets* não oferece uma geolocalização precisa composta por coordenadas, apenas 4.10% dos *tweets* coletados para este projeto entre 24/08/2017 e 04/10/2017 haviam coordenadas disponíveis. Outra razão é que nem sempre a localização do usuário naquele momento é a mesma do evento que está ocorrendo, já que o mesmo pode estar comentando um evento a partir de um outro local. Assim, as ferramentas desenvolvidas vão se propor a extrair do corpo dos *tweets* tanto o evento quanto a localidade do evento, quando disponível na mensagem. Devido ao grande número de tipos de eventos e localidades, e as limitações da API, faz-se necessário utilizar métodos mais inteligentes e automáticos para extração dessa informação utilizando aprendizado de máquina.

2.4 Aprendizado de máquina

Existem três técnicas de aprendizado de máquina que são apresentadas na literatura recente para a extração de conteúdo do Twitter:

- *n-grams*
- *Conditional Random Fields* (CRF)
- CRF com *Convolutional Neural Network* (CNN)

N-grams é a técnica mais simples, enquanto que CRF com CNN é a mais complexa dentre essas. Todas elas se apoiam na relação entre as palavras e sua semântica conjuntamente, realizando isto de formas diferentes. Abaixo serão descritas as técnicas, bem como a literatura vista sobre elas.

2.4.1 N-Gram

Um N-gram é uma sequência ordenada de palavras com N palavras. A probabilidade dessa sequência aparecer em um corpo de texto qualquer, por exemplo, em *tweets* é dada por:

$$P_{\ell_i}(w_1, \dots, w_n) = \frac{c(w_1, \dots, w_n)}{c(w_1, \dots, w_{n-1})}$$

A função $c(x)$ retorna o número de vezes que aquela sequência apareceu no corpo. Outra probabilidade importante é a probabilidade de uma sub-região, que seria um conjunto de N-grams, gerar uma dada frase $T = (w_1, \dots, w_n)$ é dada por:

$$P(T|\ell_i) = \prod_{j=n}^k P_{\ell}(w_{j-n+1}, \dots, w_j)$$

Essa probabilidade nada mais é do que o produto das probabilidades de todos os N-grams contidos na frase T . [Doran et al. 2013]

O N , que indica o tamanho da sequência, pode ser de qualquer tamanho, porém, [Doran et al. 2013] afirma que na maioria dos trabalhos relacionados a redes sociais que utilizaram N-gram se limitou a utilizar $N = 1$, 1-gram, ou unigrama, dada a complexidade computacional envolvida para valores maiores de N . A consequência de utilizar unigrama

é a limitação de isolar todas as palavras e não buscar relações entre elas. Apesar de *tweets* serem fontes curtas de informação, ainda assim existem relações entre as palavras que devem ser consideradas, por exemplo, uma frase que contenha a sequência de palavras “acidente de carro”, cada palavra seria um unigrama e seria tratada separadamente, não sendo possível associar “acidente” e “carro”.

[Doran et al. 2013] fala que uma estratégia seria utilizar $N = 2$, 2-gram, ou bigrama, para capturar relações. Os autores utilizam um exemplo em inglês: “I love driving” e “I hate driving” para mostrar a falta de conexão do unigrama e a melhoria que o bigrama proporciona. Porém, em uma linguagem como o português, onde preposições são muito presentes, como em “acidente de carro”, talvez bigrama ainda não fosse suficiente para descrever o relacionamento desta frase, haveria dois bigramas: (acidente, de) e (de, carro). Assim, a possibilidade de uso de modelos baseados em trigramas para o português poderia ser estudada, mas haveria a introdução de maior complexidade computacional.

[Anantharam et al. 2015] utiliza CRF em seu estudo e faz uma comparação entre os dois mostrando mais limitações de N-gram, especificamente quanto à descoberta de eventos em *tweets* ou qualquer outro corpo de texto utilizado. Naturalmente, eventos não acontecem todo tempo e quando se trata do Twitter, a maioria dos *tweets* não tratam sobre eventos. Assim, os eventos são esparsos e serão obscurecidos pela técnica N-gram. De fato, como é possível perceber pelas fórmulas mostradas acima, a probabilidade é dada pela quantidade de vezes que uma determinada sequência aparece. Sequências relacionadas a eventos, que são esparsas no corpo, terão uma probabilidade extremamente baixa e serão totalmente escondidas por outras sequências, maior parte delas irrelevante.

Outra desvantagem do N-gram, e que torna o trabalho de descoberta de eventos e localidade praticamente impossível, é a incapacidade da técnica de distinguir as diferentes entidades. Em outras palavras, N-gram não é uma técnica apropriada para realização da tarefa de Reconhecimento de Entidades Nomeadas (do inglês *Named Entity Recognition* ou NER), que é exatamente o que este trabalho se propõe a fazer. A seguir será falado mais sobre NER e a utilização de CRF para a realização para solucionar este problema de aprendizagem de máquina.

2.4.2 Named Entity Recognition

O reconhecimento de entidades nomeadas é um problema de aprendizado de máquina que, como o nome diz, tem como objetivo reconhecer entidades que possuem nome próprio em textos escritos em linguagem natural e indicar estas entidades, geralmente em diferentes categorias dependendo do uso, por meio de etiquetas. O exemplo da Figura 3 mostra como um *tweet* pode ser etiquetado.

```
"Text": "RT @LinhaAmarelaRJ: 18:36 Sentido Fundão trânsito bom",  
"taggedText": "RT\t0\r\n@LinhaAmarelaRJ\t0\r\n:\t0\r\n18:36\t0\r\nSentido\t0\r\nFundão\tB-LOCATION\r\ntrânsito\t0\r\nbom\t0\r\n\r\n",
```

Figura 3 - Exemplo de tweet etiquetado

A etiqueta é colocada logo após a palavra, assim, é possível ver que a palavra “Fundão”, que denota uma ilha na cidade do Rio de Janeiro foi etiquetada como “B-LOCATION”. Esta notação de etiquetamento se chama notação BIO e é muito utilizada no campo de linguística computacional [Anantharam et al. 2015]. Ela é aparentemente simples, mas poderosa ao agregar significado à relação entre as palavras. As palavras são etiquetadas respeitando as seguintes regras:

- B-{tipo de entidade}, por exemplo B-LOCATION, quando esta é a primeira palavra de um nome;
- I-{tipo de entidade}, por exemplo I-LOCATION, para todas as palavras após a primeira que pertencem ao nome de uma entidade; e
- “O”, que significa *Other* (outro), é atribuída a toda palavra que não faz parte do nome de uma entidade.

Por exemplo, em uma frase como “O Rio de Janeiro é bonito” a frase seria etiquetagem esperada seria: “O\O Rio\B-LOCATION de\I-LOCATION Janeiro\I-LOCATION é\O lindo\O”.

Com esta notação é possível delinear a relação entre as palavras mais facilmente, além de ser mais fácil de conseguir extrair o nome completo de uma entidade já que o início e o fim de um nome estão claramente definidos.

Além de [Anantharam et al. 2015], outros trabalhos relacionados à extração de entidades nomeadas de -também utilizam esta notação, como [Farajidavar et al. 2017] e [Puiu et al. 2016]. Todos os três trabalhos utilizam esta notação juntamente com *Conditional Random Fields* (CRF).

2.4.3 *Conditional Random Fields* (CRF)

Conditional Random Fields (CRF) é um modelo estatístico geralmente representado por um grafo não-direcionado que mostra a relação entre entidades. No contexto deste trabalho ele representará a relação entre palavras (chamadas de *tokens*) e as etiquetas que são atribuídas aos *tokens*. A Figura 4 mostra um exemplo de um grafo não-direcionado gerado por um CRF.

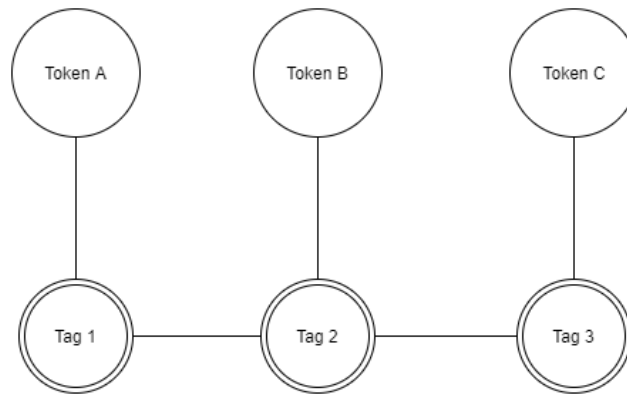


Figura 4 - Grafo básico gerado por um modelo CRF

O modelo estatístico trabalha com valores que representam a taxa de ocorrência de determinadas sequências no corpo de texto. As arestas no grafo não só representam relações entre *tokens* e *tags* e *tags* e outras *tags*, mas também representam funções que retornam justamente o valor dado aquela relação, que geralmente é a probabilidade daquela relação existir. Por exemplo, utilizando a frase de exemplo anterior e extraíndo somente “Rio de Janeiro” para simplificar o grafo podemos representar em CRF esta relação conforme expresso na Figura 5.

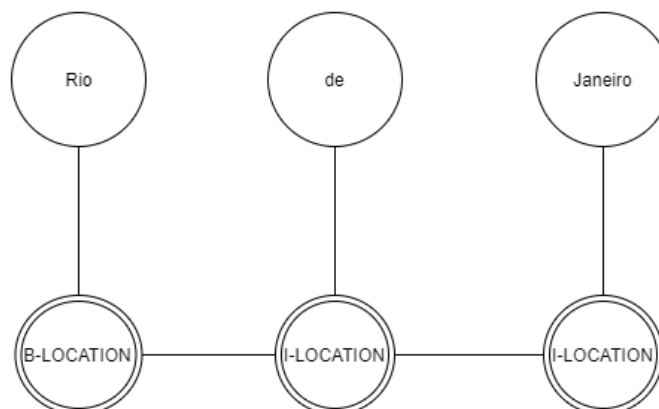


Figura 5 - Exemplo de grafo com tokens e tags

Assim, podemos dizer que temos as seguintes funções (ou fatores, como é normalmente nomeado na literatura sobre CRF) [Sutton 2012] que descrevem as relações e seus valores:

- $\Phi(\text{Rio}, \text{B-LOCATION})$
- $\Phi(\text{de}, \text{I-LOCATION})$
- $\Phi(\text{Janeiro}, \text{I-LOCATION})$
- $\Phi(\text{B-LOCATION}, \text{I-LOCATION})$
- $\Phi(\text{I-LOCATION}, \text{I-LOCATION})$

Esses fatores podem representar a probabilidade de tais relações aparecerem em um *corpus*: qual a probabilidade de Rio ser etiquetada como B-LOCATION? Qual a probabilidade de B-LOCATION ser seguida de I-LOCATION? É fácil entender que esta relação é forte quanto se trata da palavra Rio de Janeiro. Ainda assim, é possível deixar o CRF ainda mais complexo e inteligente adicionando mais fatores. Como por exemplo, levando em consideração a *tag* atribuída aos *tokens* adjacentes ou levar em consideração o *token* anterior para atribuir uma *tag* a uma nova palavra como mostrado na Figura 6.

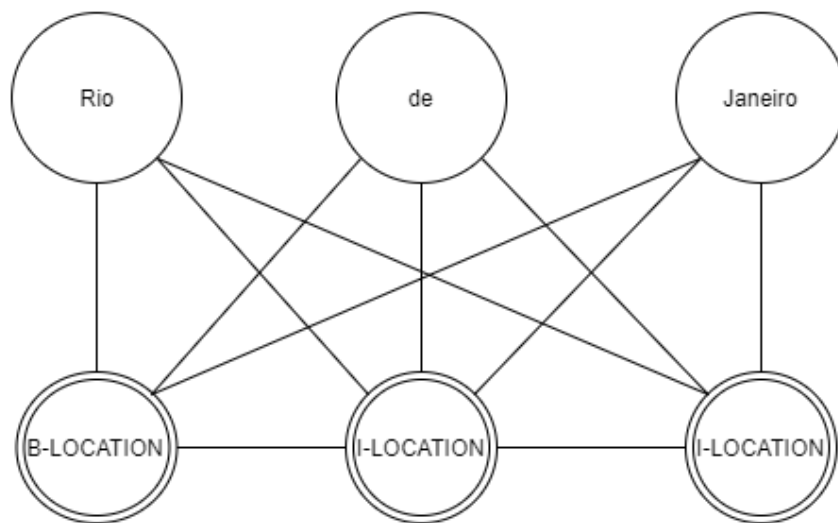


Figura 6 - Exemplo mais complexo de um grafo CRF

Porém, é necessário salientar que não há nunca relações entre tokens, como é possível perceber nos grafos acima. Isto se dá devido ao tamanho do conjunto de tokens, todas as palavras que aparecem em um corpo de texto. Representar todas as relações, dependências, etc. podem gerar problemas de performance. Esta é uma das vantagens do modelo CRF [Sutton 2012]. Porém, características dos tokens também possuem grande influência no modelo. As características são chamadas de *features* na literatura sobre CRF. Alguns exemplos de *features* são:

- A palavra começa com letra maiúscula.
- A palavra é toda em letra maiúscula.
- A palavra possui um tamanho X.
- A palavra começa/termina com um padrão.
- Entre outras...

Utilizando o exemplo anterior, dependendo do corpo de treinamento utilizado para treinar o modelo CRF a palavra “Rio” com letra maiúscula pode ter maior probabilidade de ser etiquetada como “B-LOCATION” do que somente “rio”. No contexto de extração de conhecimento de *tweets*, no entanto, com a linguagem mais informal, pode não ser incomum “Rio de Janeiro” ser escrito totalmente em letra minúscula. Portanto, é necessário sempre treinar o modelo no contexto em que ele será utilizado.

Os resultados alcançados pelos estudos que utilizaram *Conditional Random Fields* são bastantes animadores. [Anantharam et al. 2015] utilizou como base de validação dos eventos extraídos do Twitter uma base de dados de eventos de tráfego do governo da cidade de São Francisco, Califórnia, onde os experimentos foram realizados: 511.org. Do total de eventos detectados (1.042), 40% deles (454) coexistiram em tempo e localidade com eventos reportados no 511.org. Com isso, os autores acreditam que ainda há muitos eventos a descobrir, mas o 511.org não oferece a granularidade necessária para extrair todo o potencial da ferramenta. O etiquetamento em si obteve uma precisão média de 71.5% para localizações e 59% para eventos e recall de 76% e 32%, respectivamente. Porém, ao avaliar a implementação da ferramenta de extração e classificação disponibilizada pelos autores¹ é possível notar que não houve um uso considerável de *features* para aumentar a capacidade de classificação do CRF, assim é certamente possível obter resultados melhores de precisão.

[Puiu et al. 2016] utiliza CRF juntamente com outra técnica de aprendizagem de máquina chamada *Convolutional Neural Network* (CNN) com o objetivo de obter resultados melhores que uma aplicação que somente utiliza CRF. O trabalho de [Puiu et al. 2016] foi utilizado em [Farajidavar et al. 2017] na criação de um framework para cidades inteligentes. A seguir a técnica CNN será comentada, bem como os resultados de [Puiu et al. 2016] e as razões pelas quais este projeto apenas utilizará CRF na classificação dos *tweets* em português.

¹ <https://osf.io/b4q2t/>

2.4.4 Convolutional Neural Network (CNN)

Convolutional Neural Network é geralmente utilizado na área de aprendizagem de máquina em processamento de imagens, por exemplo na identificação de elementos em uma foto, alcançando nesta área resultados muito satisfatórios. Somente recentemente começou a ser utilizado para a classificação de textos, alcançando também bons resultados segundo [Britz 2015]. Convolutional Neural Network é uma técnica que se utiliza de redes neurais para fazer a sua classificação.

Ela funciona utilizando matrizes e convoluções (*convolutions*) que são funções aplicadas a toda uma matriz, como se esta estivesse percorrendo a matriz. Essas funções são chamadas de filtros ou “descobridores de características” (*feature detector*). A matriz, geralmente, é a representação de uma imagem, cada entrada representando um pixel. Assim, é fácil imaginar a sua utilização na descoberta de padrões para reconhecimento em imagens. Pode haver várias convoluções, funções e camadas de matrizes que agregam maior poder ao modelo. A Figura 7 retirada de [Britz 2015] ilustra um exemplo do funcionamento do CNN.

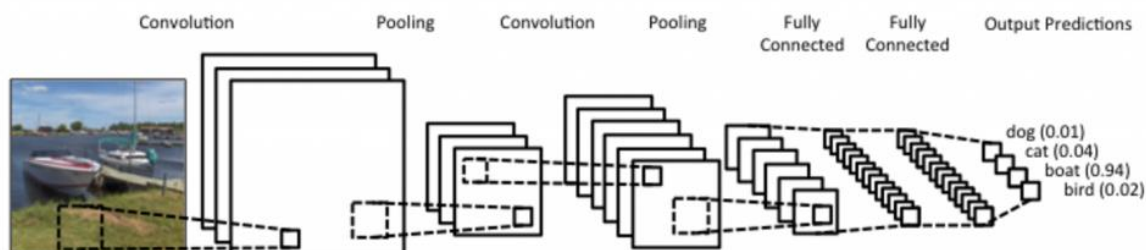


Figura 7 - Funcionamento de um CNN [Britz 2015]

A imagem é representada em uma matriz onde várias funções são utilizadas. As funções, por exemplo, podem transformar a imagem em uma imagem em preto e branco ou com somente os contornos dos objetos aparecendo, ficando assim mais fácil de identificar formas e outros elementos. Após isso, outras funções podem ser utilizadas em outras etapas. Na imagem também há etapas de *pooling* que pode ser descrito como uma amostragem, por exemplo, extrair o valor máximo resultante de uma função numa etapa de convolução. *Pooling* pode ter várias vantagens como diminuição de dimensão da matriz e uniformização para que seja mais fácil chegar a um resultado [Britz 2015]. Após

várias etapas os resultados são conectados e no final a classificação é realizada dentre as categorias desejadas.

Mais especificamente em classificação de texto, as funções são basicamente como as *features* em CRF, por exemplo, se uma determinada palavra existe ou se começa com letra maiúscula. Assim, uma função pode retornar um alto valor se encontrar determinada palavra e um baixo valor quando não encontrar e assim por diante.

[Farajidavar et al. 2017] utiliza em seu trabalho uma junção de Conditional Random Fields e Convolutional Neural Network para atingir resultados melhores do que modelos como o utilizado por [Anantharam et al. 2015]. Ele afirma que modelos como o apresentado em [Anantharam et al. 2015] não obtém bons resultados em dados futuros já que foram treinados de forma muito específica para o problema que está sendo tratado. Assim, ele utiliza CRF para realizar reconhecimento de entidade nomeadas e CNN para POS Tagging, que é a classificação das palavras em suas devidas classes gramaticais, utilizando como treinamento dados mais genéricos retirados da Wikipedia em língua inglesa.

Os resultados foram controversos como mostrado na Tabela 1 abaixo retirada do artigo:

Location tagging performance	CRF-dictionary Tagging			CNN-enhanced LOCATION Tagging		
	Recall	Precision	F-measure	Recall	Precision	F-measure
<i>San Francisco</i> data	0.96	0.93	0.94	0.99	0.87	0.93
<i>London</i> ₁ data	0.49	0.83	0.61	1.00	0.43	0.59

Tabela 1 - Resultados retirados de [Farajidavar et al. 2017]

O trabalho dá destaque ao alto recall do modelo proposto nas duas amostragens de dados utilizados, um em São Francisco e o outro em Londres. Porém, a precisão não foi tão alta no modelo proposto sendo o modelo com CRF apenas um pouco mais consistente entre precisão e recall. É difícil ignorar uma das medidas e somente se focar na outra já que uma precisão alta significa baixo número de falsos positivos. Um alto número de falsos positivos pode acarretar em trabalho desnecessário das aplicações que estarão utilizando a ferramenta de classificação que terão que buscar por localizações inexistentes. O trabalho não conseguiu explicar esta falta de precisão e se limitou a dar alguns comentários sobre a diferenças entre os dois datasets. Quanto ao etiquetamento de eventos o artigo não deixou clara uma comparação entre os dois métodos não sendo possível, pelo melhor de nosso conhecimento, avaliar o método proposto neste contexto.

Portanto, foi concluído que a utilização de ambos CRF e CNN agregaria uma complexidade maior ao trabalho sem uma probabilidade razoável de um resultado melhor do que utilizando apenas o CRF, que também é mais estabelecido na área de reconhecimento de entidades nomeadas, sendo o CNN ainda um modelo ainda a ser estudado mais a fundo nesta área. A utilização de CNN juntamente com CRF pode ser relacionada como trabalho futuro.

2.4.5 Outras técnicas de *Named Entity Recognition*

Além das técnicas mencionadas acima, outras técnicas também foram utilizadas na literatura sobre o tema. Em [Moura 2009], por exemplo, são realizados experimentos na área de NER com utilizando *Hidden Markov Model* além de realizar uma revisão histórica e bibliográfica da área de aprendizagem de máquina. Segundo [Sutton 2012] Conditional Random Fields pode ser entendido como um modelo análogo ao Hidden Markov Model, sendo CRF um modelo discriminativo e o HMM um modelo generativo. Outra técnica utilizada em NER é a Maximização de Entropia [Medeiros and Ribeiro 2012]. Ambas as técnicas possuem eficiência inferior à Conditional Random Fields e Convolutional Neural Networks, sendo estas as técnicas mais estudadas neste campo atualmente.

Após a rever o estado-da-arte e verificar vários exemplos de aplicações similares ao que este projeto se propõe, é possível concluir que uma das grandes contribuições deste trabalho será a introdução de aplicações deste tipo para a língua portuguesa. O modelo CRF será utilizado para a extração de conhecimento do Twitter sendo o mais utilizado em trabalhos recentes, seja sozinho ou com outro modelo como CNN. A junção com CNN, embora promissora, ainda precisa de mais desenvolvimento, sendo algo interessante como trabalho futuro.

A seguir será apresentado o framework desenvolvido por este trabalho e seus componentes, indicando as linguagens e ferramentas utilizadas bem com comentários sobre decisões de performance, entre outras questões que podem surgir no desenvolvimento de um conjunto de aplicações, principalmente quando estas são voltadas para um ambiente em tempo real.

3 O Framework

3.1 Visão geral

Este framework tem como principal objetivo tornar mais fácil o processo de extração de conhecimento do Twitter, neste trabalho, mais especificamente, conhecimento sobre eventos de trânsito, em tempo real sendo possível receber *tweets* através da API do Twitter, trata-los para retirar possíveis ruídos que possam atrapalhar o etiquetamento posterior, armazená-los e entregar a ferramenta de etiquetamento onde será feito o reconhecimento de palavras que indica a localização do evento e o próprio evento. Após a etiquetagem será realizado o processo de extração de conhecimento, identificando as palavras que indicam localização ou evento, alimentando uma aplicação com esta informação. Neste trabalho será criado uma aplicação web simples onde os eventos serão indicados em um mapa. A localização de todos os eventos será indicada somente pelo o que está escrito no *tweet*, as coordenadas atribuídas a um *tweet* pelo Twitter, quando disponíveis, não serão utilizadas em nenhum momento para localizar o evento. A Figura 8 descreve uma visão geral do framework.

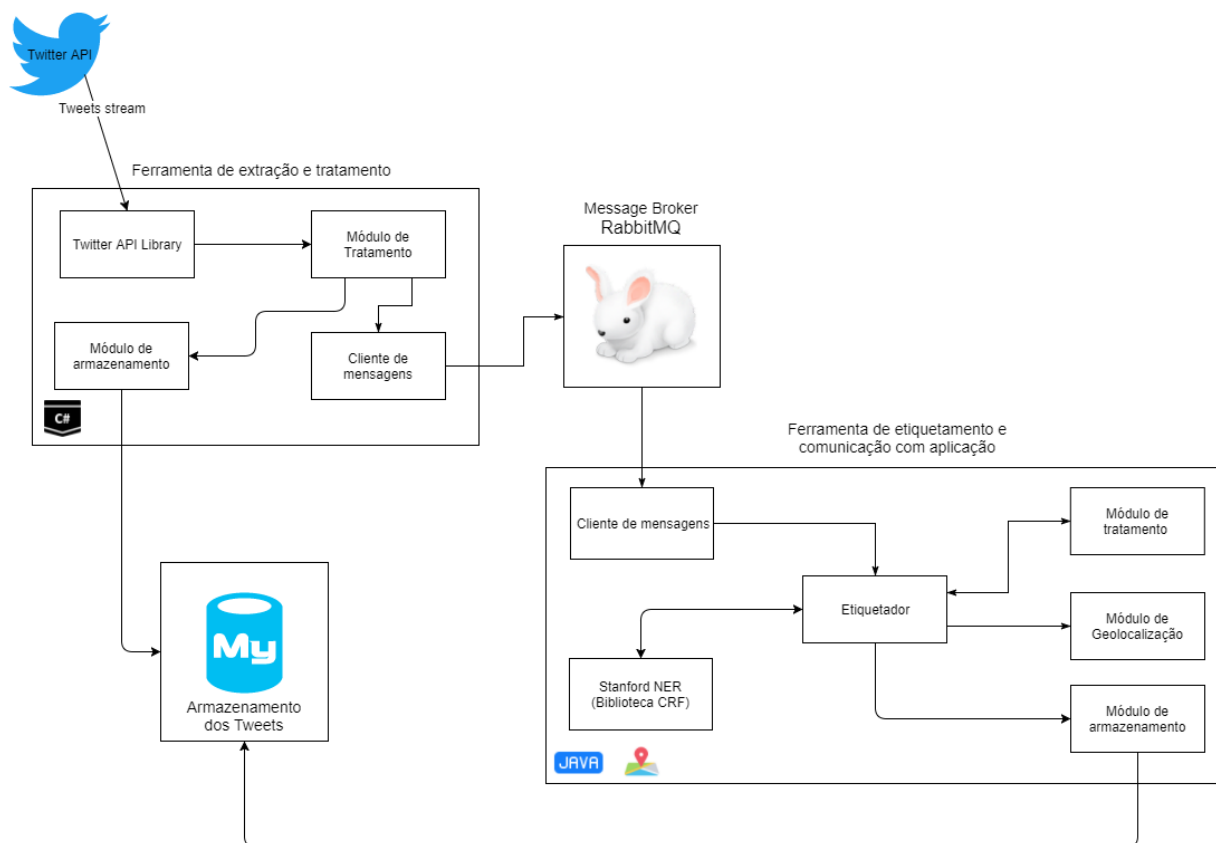


Figura 8 - Visão geral do framework

O framework em si é composto de quatro componentes principais:

- Ferramenta de extração, tratamento e armazenamento
- Módulo de Armazenamento
- Serviço de *message broker* RabbitMQ
- Ferramenta de etiquetamento e comunicação com a aplicação

As setas no diagrama indicam o fluxo de dados dentro do framework, ou seja, elas indicam basicamente o caminho que cada *tweet* percorre até estar pronto para ser utilizado por uma aplicação cliente. Os Algoritmos 1 e 2 descrevem com mais detalhes o funcionamento em alto nível do framework, desde o recebimento do *tweet*, até o armazenamento final no banco de dados. As duas aplicações possuem modos manuais e automáticos, assim, os algoritmos descreverão o modo automático, onde é possível a utilização do framework em tempo real.

ALGORITMO 1 | FERRAMENTA DE EXTRAÇÃO E TRATAMENTO

- 1 Aplicação de Extração é iniciada.
- 2 Enquanto tempo limite ou número de *tweets* não for atingido:
- 3 Aplicação de Extração recebe *TWEET*:
- 4 *TWEET* é armazenado em Lista *TWEETS*
- 5 *TWEET* é enviado para o Módulo de Tratamento
- 6 *TWEET* é enviado para o Cliente de Mensagem
- 7 Se *TWEETS* atingir tamanho estipulado:
- 8 Criar JSON com *TWEETS* para Módulo de Armazenamento
- 9 *TWEETS* são enviados para o Cliente de Mensagem
- 10 Módulo de armazenamento recebe *TWEETS*
- 11 *TWEETS* são inseridos no banco de dados
- 12 Finalizar Aplicação de Extração

O Algoritmo 1 apresenta o funcionamento da Ferramenta de Extração e Tratamento em alto nível. Na linha 1, a aplicação é iniciada. Na linha 2 um *loop* é definido sendo a condição de parada um tempo limite ou número de tweets estipulado pelo usuário sendo um parâmetro configurável da aplicação. Dentro do *loop* a aplicação espera por *tweets* serem enviados pela API do Twitter. Ao receber um *tweet* na linha 3, o *tweet* é armazenado em uma lista de *tweets* na linha 4. O *tweet* passar por um processo de tratamento na linha 5 e, após isto, é enviado ao Cliente de Mensagens na linha 6, sendo enviados a Ferramenta de Etiquetamento. Na linha 7 a aplicação testa se a lista de *tweets* possui um tamanho estipulado pelo usuário, este tamanho também é um parâmetro configurável pelo usuário. Caso este limite tenha sido atingido, a lista é convertido em JSON na linha 8 e enviada para o Cliente de Mensagens na linha 9. Desta vez o Cliente de Mensagens irá enviar os *tweets* para outra fila destinada ao Módulo de Armazenamento. Este módulo recebe os *tweets* na linha 10 e este insere os mesmos no banco de dados na linha 11. A linha 12 finaliza a aplicação após o *loop*.

ALGORITMO 2 | ETIQUETAMENTO E COMUNICAÇÃO

- 1 Aplicação de Etiquetamento é iniciada.

```

2      Aguarda receber TWEET.
3      Enquanto tempo limite não for atingido
4          Se TWEET recebido:
5              TWEET é etiquetado pelo Etiketador
6          Se TWEET ETIQUETADO possuir entidades de localização & evento:
7              Entidades de localização são extraídas
8              Consulta é enviada a API de geolocalização
9              Ao receber o resultado:
10                 Armazenar no banco de dados
11      Finalizar Aplicação e Etiketamento

```

O Algoritmo 2 descreve o funcionamento em alto nível da Ferramenta de Etiketamento e Comunicação. Na linha 1 a aplicação é iniciada. A ferramenta é baseada no conceito de consumidor, assim na linha 2 a aplicação aguarda o recebimento de *tweets* iniciando um loop na linha 3 até o tempo limite for atingido, assim como na Ferramenta de Extração e Tratamento. Caso um *tweet* seja recebido na linha 4, o mesmo é etiquetado pelo módulo Etiketador na linha 5. Na linha 6, caso o *tweet* já etiquetado possua ambas entidades de localização e evento, na linha 7 as entidades de localização são extraídas do *tweet*. Na linha 8 uma consulta a API de geolocalização é enviada com o objetivo de obter as coordenadas da localização extraída. Ao receber o resultado da consulta na linha 9, o *tweet* é atualizado no banco de dados com as coordenadas encontradas. Ao final do *loop* a aplicação é finalizada na linha 11.

Cada um dos componentes mencionados nos algoritmos possui detalhes que serão minuciados nas seções seguintes.

3.2 Componentes

Nesta seção serão detalhados os componentes do *framework* desenvolvido. Entende-se como componente neste contexto as diferentes aplicações que compõe o

framework e seus subcomponentes. Os componentes principais mostrados na Figura 9 são: Ferramenta de Extração e Tratamento, RabbitMQ *Message Broker Server*, Ferramenta de Etiquetamento e Comunicação e o banco de dados.

3.2.1 Ferramenta de extração e tratamento

O primeiro componente do *framework* é a ferramenta de extração, tratamento e armazenamento. Esta é uma aplicação simples que tem como função receber os *tweets* do Twitter utilizando a API² disponibilizada pela própria rede social. Após isto os *tweets* são tratados retirando links e emoticons do texto. Por fim, os *tweets* são armazenados pela aplicação em um banco de dados. A aplicação armazena temporariamente em memória os *tweets* por motivos de performance que serão descritos mais adiante neste trabalho.

A ferramenta foi construída utilizando os paradigmas de orientação a objetos e a linguagem C# .NET Core 2. A escolha pela linguagem C# se deveu à afinidade com a linguagem e também pelo teor inovador da utilização do framework .NET Core 2.0, um *framework* multiplataforma que possibilita a utilização de uma aplicação em diferentes sistemas operacionais sem a necessidade de instalar qualquer software no computador alvo. A linguagem C# sempre foi dada como uma linguagem para desenvolvimento de aplicações para a plataforma Windows, porém isto passou a ser uma visão ultrapassada. Assim, esta ferramenta poderá ser utilizada em diferentes sistemas, incluindo Linux e MacOSX [Microsoft 2017a].

Esta aplicação tem como único dado de interesse os *tweets* recebidos do Twitter, assim sendo a única classe de dados utilizada nesta ferramenta é a classe *Tweet* representada pelo diagrama de classes mostrado na Figura 9.

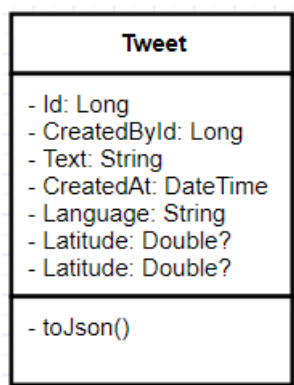


Figura 9 - Diagrama de classe de dados da aplicação

² <https://developer.twitter.com/en/docs>

Outras classes dizem respeito a parte operacional da aplicação: *TweetExtractor*, *TweetFormatter*, *DBConnection*, *Messenger* e *Properties* sendo responsáveis pela extração dos *tweets*, tratamento dos *tweets*, armazenagem em banco de dados, envio de mensagens e configuração de propriedades, respectivamente.

A extração dos *tweets* é realizada utilizando uma biblioteca de comunicação com a API do Twitter desenvolvida para aplicações em .NET, sendo esta recomendada pelo Twitter [Twitter 2017a], chamada *Tweetinvi* [Linvi 2016]. Como o Twitter tem grande importância neste trabalho se faz necessário fazer uma breve discussão sobre o funcionamento de sua API, assim como suas características, recursos e limitações.

3.2.1.1 Twitter API

A API do Twitter é responsável por prover acesso para outras aplicações a recursos do Twitter como os *tweets*, postagem de *tweets*, administração de contas, entre outros recursos. Para utilizar a API é necessário utilizar OAuth como método de autenticação, cadastrando a aplicação junto ao Twitter e gerando chaves exclusivas para aquela aplicação. A autenticação OAuth é sempre utilizada também quando um usuário concede permissão a uma aplicação para acessar dados de sua conta no Twitter. Como este método de autenticação utiliza chaves para credenciamento, a aplicação terceira nunca tem acesso a senha real do usuário dando assim mais segurança para o processo de autenticação e concessão de permissões.

Neste trabalho serão utilizados os recursos de pesquisa e streaming de *tweets*. A pesquisa de *tweets* é realizada utilizando REST API através de uma linguagem de consulta básica, podendo ser configurados vários parâmetros para refinar a busca. Por exemplo, a consulta a seguir procura por *tweets* com o texto “@Twitter” na língua inglesa: <https://api.twitter.com/1.1/search/tweets.json?q=@Twitter&lang=en>

Outros parâmetros podem indicar o autor do *tweet* (parâmetro “from”), destinatário do *tweet* (parâmetro “to”) e limitar a pesquisa para uma determinada área (parâmetro “geocode”). O Twitter impõe uma limitação ao recurso de busca sendo possível ter acesso apenas a *tweets* dos últimos 7 dias. O Twitter oferece opções pagas que concedem acesso aos *tweets* dos últimos 30 dias ou acesso total aos *tweets*, porém neste trabalho será utilizada a opção gratuita, com acesso a até 7 dias de *tweets*.

O recurso de busca é utilizado neste trabalho em situações em que é necessário obter *tweets* de perfis específicos, especialmente de perfis oficiais da cidade do Rio de Janeiro, que informam sobre eventos de trânsito, entre outros eventos e alertas. Os *tweets* destes perfis oficiais serão utilizados para treinamento e testes, como será visto em seções subsequentes.

Porém, como um dos objetivos deste framework não é só extrair eventos de trânsito de perfis oficiais, mas também de perfis de cidadãos comuns que comentam fatos vividos por eles, o recurso mais utilizado da API Twitter é a Streaming API, responsável por enviar a aplicação requisitante *tweets* em tempo real que respeitam os filtros configurados pela aplicação. A Streaming API é uma ferramenta extremamente poderosa mesmo com as limitações impostas pelo Twitter, tal como acontece com a função de busca, sendo a principal delas a limitação da Streaming API básica de enviar somente 1% de todos os *tweets*. Apesar disso é possível extrair uma quantidade considerável de informação do Twitter.

Alguns dos parâmetros mais interessantes da Streaming API são:

- *Language*: delimita a linguagem dos *tweets* que serão enviados.
- *Follow*: delimita uma lista de perfis dos quais *tweets* devem ser enviados
- *Track*: delimita palavras ou frases que devem estar contidas no corpo do *tweet*.
- *Locations*: delimita áreas (*bounding-boxes*) de onde os *tweets* devem ter sido enviados.

Esses quatros parâmetros já garantem um alto grau de granularidade na stream que será recebida pela aplicação. O parâmetro “locations” é o mais importante, não sendo possível realizar a filtragem necessária dos *tweets* para este trabalho em tempo hábil sem a utilização do mesmo.

Streaming API utiliza requisições GET ou POST, porém requisições POST são recomendadas para evitar URLs de tamanho excessivamente grande, que podem ser rejeitadas pela API. De qualquer forma, existem, como já dito, bibliotecas exclusivamente criadas para tornar a comunicação com a API do Twitter mais rápida e fácil. A Figura 9 mostra um exemplo de código que utiliza a Streaming API para receber os *tweets* e adicionar a uma lista de *tweets* utilizando a biblioteca *Tweetinvi* para C#.


```

var stream = Tweetinvi.Stream.CreateFilteredStream();
stream.AddLocation(new Coordinates(-23.076889, -43.761292), new Coordinates(-22.742306, -43.091125));
stream.FilterLevel = StreamFilterLevel.None;

stream.MatchingTweetReceived += (sender, args) =>
{
    Tweet tweet = new Tweet();
    tweet.Id = args.Tweet.Id;
    tweet.CreatedById = args.Tweet.CreatedBy.Id;
    tweet.Text = args.Tweet.FullText;
    tweet.CreatedAt = args.Tweet.CreatedAt;
    tweet.Language = args.Tweet.Language.ToString();
    tweet.Longitude = args.Tweet.Coordinates != null ? (double?)args.Tweet.Coordinates.Longitude : null;
    tweet.Latitude = args.Tweet.Coordinates != null ? (double?)args.Tweet.Coordinates.Latitude : null;
    tweets.Add(tweet);
}

```

Figura 10 - Exemplo de utilização da biblioteca Tweetinvi em C#

No exemplo da Figura 10 a primeira linha cria uma nova *stream* filtrada, na segunda linha é adicionado um filtro de localização sendo configurado duas coordenadas que representam a *bounding box* onde os autores do *tweet* devem estar localizados para que ele seja enviado pela *stream*. Na terceira linha é configurado o *FilterLevel*, que indica a quantidade de *tweets* a ser enviado, já que em alguns casos é mais conveniente obter uma taxa menor de *tweets*. Neste caso o filtro está configurado para *None*, não havendo restrições e, portanto, todos os *tweets* serão enviados. O restante do código demonstra a utilização do que é chamado de *event handler* em C#, sendo o evento neste caso a recepção de um *tweet* que possui as características configuradas. Quando um *tweet* é recebido os dados importantes dele como ID, ID do autor, o próprio texto, data de criação, linguagem e localização (latitude e longitude) são atribuídos a um objeto da classe *Tweet* e em seguida adicionado a uma lista.

Como é possível ver no exemplo, a comunicação com a API do Twitter se torna extremamente fácil com a utilização de uma biblioteca própria para esta função cabendo apenas o trabalho de tratar e armazenar os *tweets* de uma conveniente. O processo de tratamento e armazenamento serão descritos a seguir.

3.2.1.2 Tratamento e armazenamento

O processo de tratamento é bastante simples, envolvendo apenas algumas expressões regulares que serão comparadas ao texto do *tweet* para retirar caracteres que representem emoticons ou padrões que representam URLs (uma sequência de caracteres que começa com “http://”, por exemplo). Por sua vez, o processo de armazenagem em um

banco de dados relacional, no entanto, envolve algumas decisões que afetam a performance do processo sendo algo mais complexo e que necessita ter seus detalhes discutidos com mais cuidado.

Existem algumas situações em que a API do Twitter pode desconectar um streaming de *tweets* [Twitter 2017b]:

- O cliente estabelece mais de uma conexão com as mesmas credenciais.
- O cliente para de ler dados subitamente.
- O cliente ler dados muito lentamente. O streaming é baseado em uma fila de onde o cliente lê os *tweets*. A conexão é cortada caso a fila fique cheia.
- Um servidor de streaming do Twitter é reiniciado.
- Mudanças da configuração de rede do Twitter.

Três delas dizem respeito ao cliente e duas ao próprio Twitter, sendo estas raras de ocorrer. Quanto as três situações relacionadas ao cliente, a primeira é facilmente tratada pela aplicação já que somente uma conexão é necessária de qualquer forma. A segunda pode ser causada por uma parada súbita da aplicação ou queda de conexão com a internet. O Twitter pode negar clientes que se desconectam e reconectam muitas vezes em um curto espaço de tempo. A biblioteca utilizada no trabalho, no entanto, possui recursos para tratar este problema. O terceiro cenário mostrado na lista: a leitura muito lenta dos dados, é o cenário mais complexo de tratar porque é afetado diretamente pela performance da aplicação cliente, incluindo o banco de dados.

Na fase inicial de desenvolvimento da aplicação tanto a extração quanto a armazenagem do *tweet* no banco de dados eram feitos na mesma função, *tweet a tweet*. Este método estava causando muitas quedas de conexão com o Twitter já que a fila do streaming ficava cheia rapidamente devido ao tempo de comunicação entre a aplicação e o banco de dados, especialmente quando o banco de dados não está hospedado no mesmo servidor onde a aplicação está sendo executada.

Assim, fez-se necessário procurar uma forma de realizar ambas as tarefas separadamente e em paralelo, utilizando paralelismo. Muitos computadores hoje em dia possuem processadores com vários núcleos, sendo possível executar várias tarefas ao mesmo tempo. Estas tarefas são geralmente chamadas de *threads*. C# também possui recursos para realizar tarefas em paralelo [Microsoft 2017b]. O exemplo de código na Figura 11 mostra como a ferramenta de extração, tratamento e armazenamento lida com a execução das tarefas em paralelo.

```
Task tweetStream = Task.Factory.StartNew(() => te.FilteredStreamFeatures());
Task insertToDB = Task.Factory.StartNew(() => dc.InsertToMySQLFromFile());
Task.WaitAll(tweetStream, insertToDB);
```

Figura 11 - Trecho de código que mostra a utilização de paralelismo

Existem formas mais complexas de lidar com paralelismo em C#, porém esta forma, apesar de simples, realiza o desejado pela aplicação. As duas primeiras linhas iniciam a extração e o armazenamento em tarefas separadas, a terceira linha configura a aplicação para continuar apenas quando as duas tarefas tiverem sido completadas. O processo de extração lê os *tweets* e os guarda em memória até um certo número configurável na aplicação, ao atingir este número os *tweets* são convertidos em JSON e enviados utilizando um cliente RabbitMQ para ser consumido pelo outro processo que trabalha com o armazenamento no banco de dados.

Porém, o processo de conexão e inserção no banco de dados ainda surge como um problema, os *tweets* eram inseridos separadamente e cada inserção gerava uma nova conexão e uma nova inserção, o que somava tempo que no final totalizava horas de diferença entre o processo de extração e o processo de armazenamento. De acordo com a documentação do MySQL Server [Oracle 2017], a velocidade de uma inserção pode ser impactada pelos fatores abaixo, sendo o número entre parênteses o nível de impacto.

- Conexão (3)
- Envio de query ao servidor (2)
- Tratamento da query (2)
- Inserção da tupla (1 x tamanho da tupla)
- Inserção nos índices (1 x tamanho dos índices)
- Fechamento (1)

A documentação recomenda que ao inserir múltiplas tuplas da mesma fonte um único comando INSERT com o parâmetro VALUES listando múltiplas tuplas seja utilizado. Isto poupa a aplicação de ter que conectar e enviar consultas várias vezes ao banco de dados, uma para cada tupla inserida. Testes realizados com ambas as formas: inserção de cada tupla em separado e inserção em um único INSERT foram realizados com 100 *tweets* alcançando os resultados abaixo:

Método	Tempo
INSERTs separados	2m 5s 486ms
INSERT único	0m 5s 778ms

Tabela 2 - Resultado do teste de desempenho do banco de dados

A tabela mostra resultados muitos superiores para o método onde um só comando INSERT é utilizado corroborando com a recomendação da documentação do banco de dados. Utilizando este método foi possível conseguir a performance desejável para uma aplicação em (quase) tempo real ao solucionar o seu maior gargalo.

3.2.2 Banco de dados

O sistema de gerenciamento de banco de dados (SGBD) utilizado no framework por padrão é o MySQL, da Oracle, sendo o banco de dados *open-source* mais popular do mundo segundo sua desenvolvedora³. O fato do MySQL ser *open-source* proporciona um bom custo-benefício. Além disso, ele é compatível com Windows, Linux e outros sistemas operacionais baseados em UNIX continuando com o compromisso do framework de ser possível o seu uso ao menos em Windows e Linux. Outros bancos de dados podem ser utilizados, mas será necessário realizar modificações no código fonte da aplicação de extração para ser possível a conexão.

O banco de dados, ao espelhar as aplicações a qual serve também possui apenas uma tabela de dados, a tabela de *tweets*, com uma pequena diferença em relação ao diagrama de classes para a aplicação de extração: a adição de um atributo onde será guardado o texto etiquetado que será utilizado pela ferramenta de etiquetamento e extração de informação. O diagrama mostrado na Figura 12 ilustra a tabela do banco de dados.

³ <https://www.oracle.com/mysql/index.html>

Tweet
+TweetId: bigint
+CreatedById: bigint
+Body: nvarchar(500)
+CreatedAt: datetime
+Latitude: double
+Longitude: double
+TaggedBody: nvarchar(500)

Figura 12 - Tabela do banco de dados

3.2.3 Message Broker (RabbitMQ)

Para realizar a comunicação entre as duas aplicações que compõe o framework primeiramente se pensou em utilizar o banco de dados como intermediário, assim a aplicação de extração dos *tweets* gravaria as informações no banco e a ferramenta de classificação leria estas informações e posteriormente atualizaria os dados com o *tweet* etiquetado. Porém, dado o custo de comunicação entre aplicação e banco de dados discutida anteriormente, foi necessário buscar uma solução mais eficiente, que auxiliasse o framework a manipular os dados em (quase) tempo real, objetivo do projeto. A solução encontrada para satisfazer esses requisitos foi a implantação de *message broker*. Um *message broker* é uma aplicação que age como intermediária, traduzindo mensagens de um formato reconhecido pelo remetente para um formato entendível pelo destinatário [Wikipedia 2017a]. A solução de *message broker* é baseada na forma como o *CityPulse Framework* [Puiu et al. 2016] [Puiu et al. 2013] realiza a comunicação entre seus diferentes módulos, inclusive neste trabalho utilizamos a mesma solução de *message broker* utilizada pelo citado *framework*: o RabbitMQ, que também é utilizado por [Bajaj et al. 2016] como o mesmo objetivo.

O RabbitMQ⁴ é uma solução *open-source* de *message broker* desenvolvido inicialmente pela Rabbit Technologies Ltd. e passou a ser desenvolvida pela Pivotal a partir de 2013. A linguagem de desenvolvimento do RabbitMQ é a Erlang, sendo também *cross-platform* [Wikipedia 2017b]. O RabbitMQ possui várias funcionalidades, sendo a mais simples dela o envio de mensagem entre aplicações. A ferramenta pode administrar filas de trabalho onde dois *workers* consomem da mesma fila e dividem o trabalho.

⁴ <https://www.rabbitmq.com>

Também é possível a mesma mensagem para várias aplicações, criar roteamentos, enviar mensagens de acordo com um tópico (padrão), enfim, o RabbitMQ possui diferentes capacidades como *message broker*. Neste trabalho, porém, será apenas utilizado o envio de mensagem simples, ilustrado na Figura 13.



Figura 13 - Envio de mensagem com conceito de produtor-consumidor (adaptado de [Pivotal 2017])

Este tipo de envio de mensagem utiliza o conceito de produtor-consumidor. Neste trabalho o RabbitMQ foi utilizado em duas oportunidades: na comunicação entre os threads utilizados na ferramenta de extração e na intercomunicação entre as duas aplicações que constituem o framework. Em ambas as situações o framework se beneficia da velocidade com que as mensagens são passadas do produtor para o consumidor.

3.2.4 Ferramenta de Etiquetamento e Comunicação

A ferramenta de etiquetamento e comunicação é responsável por receber os *tweets* extraídos pela ferramenta de extração e etiquetá-los utilizando *Conditional Random Fields* como técnica de etiquetamento de entidades nomeadas. Neste trabalho, o modelo utilizado pela ferramenta somente etiqueta localizações e eventos de trânsito, mas com o devido treinamento a ferramenta pode ser usada para etiquetar qualquer tipo de entidades que podem ser encontradas em um *tweet*. Como mencionado, para que a ferramenta seja utilizada é necessário um modelo CRF já treinado com um número suficientemente grande de entidades das quais se deseja identificar. O processo de treinamento e testes do modelo serão expostos em seções posteriores. Aqui serão descritos o desenvolvimento e o funcionamento da aplicação de etiquetamento.

Esta ferramenta também possui funcionalidades básicas que permitem que dados sejam disponibilizados para um website utilizá-las no desenho de um mapa com os eventos ocorridos, bem como o texto do *tweet*, marcados no mesmo. Uma aplicação web com um mapa deste tipo foi desenvolvido para testar a aplicação por completo

demonstrando sua capacidade como um agregador de informações sobre eventos de trânsito.

Como visto na Figura 7 a ferramenta de etiquetamento tem como ponto de entrada um cliente de mensagens do RabbitMQ, tendo papel de consumidor. Esta classe será responsável por receber o *tweet* e entregar a classe de etiquetamento, onde será feito todo o processo de etiquetagem e identificação de informações geográficas. Após isso o *tweet* será enviado para armazenamento e ficará disponível para aplicações usufruírem das informações coletadas. A seguir será descrito com mais detalhes os diferentes componentes desta aplicação.

3.2.4.1 Etiquetamento

A classe responsável pelo etiquetamento dos *tweets* utiliza a biblioteca do *Stanford Natural Language Processing Group*⁵ para realizar NER utilizando CRF. Na literatura, como por exemplo em [Anantharam et al. 2015], é utilizada uma outra biblioteca para CRF NER chamada LingPipe⁶, porém por causa da maior facilidade de implementação e utilização da biblioteca Stanford NER, ela foi escolhida ao invés da biblioteca LingPipe. Como dito anteriormente, em CRF os tokens também influenciam no resultado do etiquetamento através da utilização de características (*features*). A biblioteca Stanford NER possui várias *features* mais utilizadas já implementadas sendo o uso delas somente uma questão de configuração. Já LingPipe, no melhor de nosso conhecimento, não possui uma maneira similarmente conveniente de utilização de *features* sendo necessária à sua implementação. Além disso, Stanford NER é uma biblioteca totalmente livre. Apesar de ser open-source, LingPipe é uma biblioteca comercial que possui limitações e condições caso seja utilizada de forma gratuita para pesquisas e outros fins.

O Stanford NLP Group possui vários estudos na área de processamento de linguagem natural, machine learning, deep learning e análise de dados sendo a biblioteca criada pelo grupo não só para NER, mas também para várias outras funções como PoS Tagging, Tokenization, classificação, entre outras. Mais detalhes sobre os softwares mantidos pelo grupo podem ser encontrados em [StanfordNLP 2017].

⁵ <https://nlp.stanford.edu>

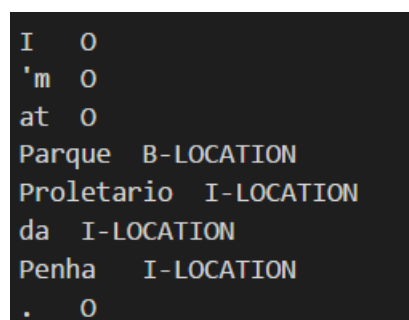
⁶ <http://alias-i.com/lingpipe/>

Para que o etiquetamento seja possível, é necessário apenas um arquivo contendo o modelo CRF treinado, este arquivo já conterá as configurações necessárias, como por exemplo as *features* utilizadas no processo de treinamento, entre outras configurações. Parâmetros de configuração do CRF foram mantidas com seus valores padrão por questão de simplicidade. Mais detalhes serão expostos na seção que detalha o treinamento posteriormente no trabalho. Assim, após o treinamento é relativamente simples utilizar a biblioteca para que novos *tweets* sejam etiquetados. Uma amostra de código demonstrando o etiquetamento é mostrada na Figura 14.

```
File model = new File(modelPath);
Properties props = new Properties();
props.setProperty("inputEncoding", "UTF-8");
props.setProperty("outputEncoding", "UTF-8");
SeqClassifierFlags flags = new SeqClassifierFlags(props);
CRFClassifier classifier = new CRFClassifier(flags);
classifier.loadClassifier(model);
String tweet = classifier.classifyToString(tweets.get(0).Text, outputFormat: "tsv", preserveSpacing: false);
```

Figura 14 - Exemplo de utilização da biblioteca StanfordNER

A linha 1 carrega o modelo de um arquivo, linhas 2 a 5 realizam configurações de codificação para o português, linha 6 cria um novo classificador, linha 7 carrega o modelo no classificador e, finalmente, linha 8 classifica um *tweet* e o retorna em um formato de *string*. O formato de saída utilizada é denotado como “TSV” que possui a aparência mostrada na Figura 15, para facilitar a leitura humana. Também facilita na leitura linha a linha do *tweet* para extrair as entidades relevantes em etapas futuras.



```
I      O
'm     O
at     O
Parque B-LOCATION
Proletario I-LOCATION
da     I-LOCATION
Penha  I-LOCATION
.      O
```

Figura 15 - Exemplo de tweet etiquetado

Após a etapa de etiquetamento o *tweet* estará pronto para ser utilizado em uma aplicação. A extração das informações de localização e evento, bem como o processo de geolocalização serão expostos na seção seguinte.

3.2.4.2 Extração e Geolocalização

A extração da informação se torna simples graças ao formato TSV introduzido pela biblioteca Stanford NER, bastando ler cada *tweet* linha a linha e identificar aquelas palavras que possuem etiquetas de localização e evento. As localizações extraídas do *tweet* serão utilizadas no módulo de geolocalização da ferramenta. É assumido que *tweets* onde há ambas palavras que representam localização e palavras que representam um evento, a localização é relacionada ao evento sendo comentado pelo autor do *tweet*. Assim, acredita-se que uma localização mais precisa do evento possa ser extraída em comparação com as coordenadas trazidas no *tweet* fornecidas pelo Twitter a uma pequena parcela dos *tweets*.

Com os dados de localização e tipo de evento acessíveis a localização em texto retirada do *tweet* pode ser transformada em coordenadas e marcadas em um mapa utilizando uma biblioteca de geolocalização. Neste trabalho foram utilizadas duas bibliotecas para este fim: Google Places API Web Service⁷ e Nominatim⁸ do Open Street Maps API. Ambos foram escolhidos por serem as ferramentas mais conhecidas em relação a geolocalização, o Open Street Maps em particular foi utilizado em vários artigos relacionados [9,15,30] como base de dados de localizações, como por exemplo na construção de um dicionário para reconhecimento de entidades como feito por [Anantharam et al. 2015] e [Farajidavar et al. 2017] e para geolocalização como feito por [Sbodio et al. 2014]. Já o Google Places API Web Service é uma das APIs relacionadas ao Google Maps que são oferecidas comercialmente a empresas e desenvolvedores sendo altamente renomada no mercado.

Ambas as APIs possuem limitações quanto ao número de requisições que podem ser enviadas. O Google Places API Web Service possui um limite de 150.000 requisições diárias em seu plano gratuito, o que é considerado o suficiente para um dia inteiro de *tweets*, onde foi possível extrair em média 150.000 *tweets* por dia no total, destes apenas uma pequena parcela possui localizações e eventos. O Nominatim API requer, em sua política de uso⁹, que haja no máximo uma requisição por segundo. Assim, respeitando os

⁷ <https://developers.google.com/places/web-service/intro>

⁸ <https://github.com/jeremiehuchet/nominatim-java-api>

⁹ <https://operations.osmfoundation.org/policies/nominatim/>

limites de ambas as APIs, foi estipulado que as APIs só serão utilizadas caso o *tweet* possua ambos localização e evento extraídos. Assim, a probabilidade de haver excesso de utilização é praticamente nula. Ambas as APIs são de fácil utilização bastando apenas algumas linhas de código para obter as informações necessárias. Nas Figuras 16 e 17 são mostrados exemplos das informações obtidas nas APIs no formato JSON.

```
"formattedAddress": "Brazil",
"geometry": {
  "location": {
    "lat": -23.006859,
    "lng": -43.3109429
  },
  "viewport": {
    "northeast": {
      "lat": -23.00545156970849,
      "lng": -43.30973451970851
    },
    "southwest": {
      "lat": -23.0081495302915,
      "lng": -43.31243248029151
    }
  }
},
"name": "Jardim Oceânico Station",
"icon": "https://maps.gstatic.com/mapfiles/place_api/icons/generic_business-71.png",
"placeId": "ChIJs6nAfrjQmwAR1j_hTUAQ1aw",
"rating": 4.4,
"types": [
  "subway_station",
  "transit_station",
  "point_of_interest",
  "establishment"
]
```

Figura 16 - Resultado do Google Places API Web Service

```
"place_id": 52962754,
"licence": "Data © OpenStreetMap contributors, ODbL 1.0. http://www.openstreetmap.org/copyright",
"osm_type": "node",
"osm_id": "4272784714",
"boundingbox": {
  "north": -23.0018286,
  "west": -43.3159313,
  "east": -43.3059313,
  "south": -23.0118286
},
"lon": -43.3109313,
"lat": -23.0068286,
"display_name": "Jardim Oceânico, Acesso C, Barra da Tijuca, Zona Oeste do Rio de Janeiro, Rio de Janeiro",
"type": "station",
"place_rank": 30,
"importance": 0.45089190172262
```

Figura 17 - Resultado da Nominatim API

Ambos os resultados são bastante parecidos em termos de coordenadas, nome do local e tipo de local, porém não foram realizados testes aprofundados para comparar a precisão de ambas. De qualquer forma, com as coordenadas é possível marcar os dados,

juntamente com o *tweet* de onde a informação foi retirada, em um mapa sendo este um dos resultados do trabalho. A aplicação de mapa será introduzida em uma seção posterior no trabalho. Porém, a informação pode ser utilizada em qualquer aplicação já que está em formato JSON.

Como dito anteriormente, a aplicação de etiquetamento necessita de um modelo CRF treinado para que a mesma funcione com a precisão necessária para realizar o trabalho de etiquetamento de forma eficaz e que seja de ajuda ao usuário, portanto, a seguir será descrito o processo de treinamento, bem como mais detalhes da biblioteca Stanford NER e do modelo CRF em relação a treinamento.

3.2.4.3 Treinamento

O conceito de treinamento em aprendizagem de máquina é utilizado em algoritmos de aprendizado supervisionado para denotar a etapa onde é utilizado um conjunto de dados propriamente classificado com o objetivo de calibrar parâmetros do modelo utilizado no experimento de acordo com os dados do conjunto de treinamento. Neste trabalho será treinado um modelo de CRF NER, assim o conjunto de dados utilizado é um corpo de texto previamente etiquetado. É importante que o corpo de texto utilizado para treinamento esteja corretamente etiquetado para não confundir o modelo e causar erros em novos etiquetamentos após o treinamento.

No caso do CRF, como visto anteriormente, este baseia-se em tokens, tags e as relações entre eles, assim o treinamento de um modelo CRF pode ser resumido como a construção dessas relações, bem como a probabilidade de estas acontecerem, junto com os atributos (*features*) de cada token.

Durante o período de experimentos foram extraídos utilizando a API do Twitter 4.499.313 *tweets* no período de 24 de Agosto de 2017 até 4 de Outubro de 2017. Destes *tweets*, 11.080 foram utilizados durante o processo de treinamento. Devido ao grande número de *tweets* e ao fato de que a base de dados de *tweet* mencionado foi acumulada durante quase dois meses, o processo de treinamento também foi realizado gradualmente, porém sem prejuízo sobre os resultados. A Figura 18 mostra como o treinamento foi realizado, dando origem a modelos intermediários e pôr fim a um modelo final.

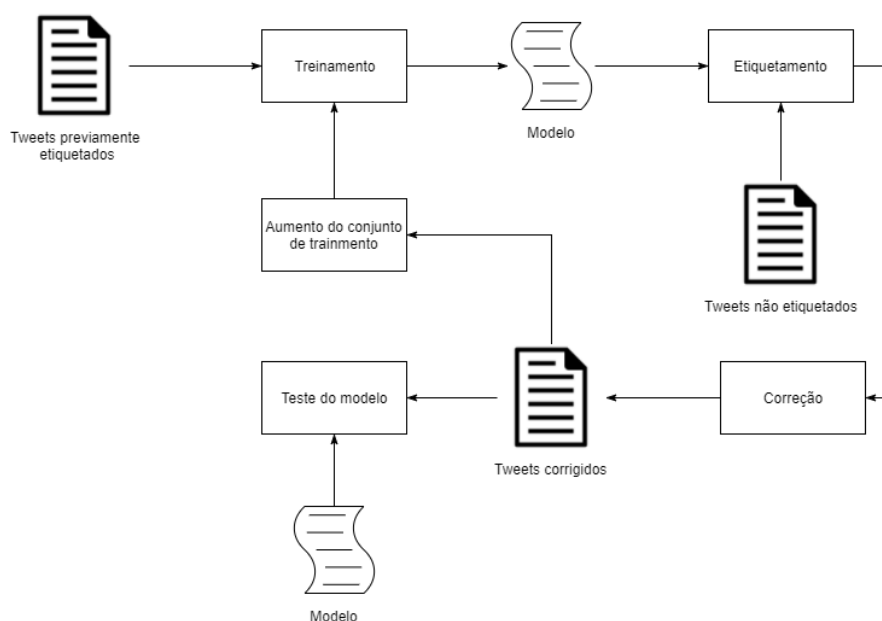


Figura 18 - Processo de treinamento

Em um primeiro momento, para obtermos um conjunto de *tweets* já etiquetados, utilizamos a técnica utilizada por [Anantharam et al. 2015] onde é utilizado um dicionário de ambos localizações e eventos para etiquetar as palavras dos *tweets* extraídos. O dicionário de localizações foi retirado utilizando a base do Open Street Maps¹⁰, enquanto que o dicionário de eventos foi retirado de *tweets* do perfil do Centro de Operações do Rio de Janeiro¹¹ de forma manual. Esta etapa foi meramente para se obter *tweets* já etiquetados, poupando o trabalho de etiquetar todos os *tweets* do início. Com o corpo etiquetado pela técnica de etiquetamento por dicionário, bastou corrigir as falhas encontradas transformando essa amostra em um gabarito que pode ser utilizado para treinar o modelo CRF. Como o intuito não era avaliar a precisão do método baseado em dicionário, não houve registros de precisão, recall e F-measure nesta etapa. Esta primeira amostragem deu origem ao primeiro modelo CRF treinado.

Uma outra amostragem de *tweets* foi retirada da base e etiquetada utilizando o modelo CRF treinado anteriormente, a partir desta etapa somente o modelo CRF foi utilizado, o dicionário foi utilizado apenas como ponto de partida, como mencionado anteriormente. O corpo etiquetado pelo modelo passou por um processo de correção manual onde as palavras erroneamente etiquetadas foram corrigidas. Após isto, para se

¹⁰ <http://wiki.openstreetmap.org/wiki/Xapi>

¹¹ <https://twitter.com/OperacoesRio>

obter um resultado preliminar da precisão, recall e F-measure do modelo, a amostra corrigida foi utilizada para testar o modelo. O StanfordNER proporciona este mecanismo de teste onde uma amostra corrigida (gabarito) é dada como entrada e o modelo é utilizado para etiquetar aquela amostra e comparar com o gabarito, retornando as métricas de precisão, recall e F-measure separadas por entidade, neste caso localização e evento. Este é um método válido de testes já que o modelo não tem conhecimento daquela amostra. Os testes foram feitos para obter dados do avanço gradual da qualidade do etiquetamento, enriquecendo os resultados.

Após a correção e a avaliação preliminar os *tweets* corrigidos são somados aos *tweets* utilizados para treinar o modelo anterior aumentando assim o conjunto de treinamento. Com este novo conjunto de treinamento um novo modelo foi treinado, em teoria com mais conhecimento que o anterior.

Este processo foi repetido no total de 9 vezes, gerando 10 modelos que foram gradualmente sendo enriquecidos. Em um dado momento do processo de treinamento, após o treinamento do modelo 6, foi percebido que o modelo não estava conseguindo identificar muitos eventos nas amostras utilizadas. Acredita-se que isto se deva à natureza esparsa dos *tweets* em si, poucos usuários relatam eventos de interesse comparado ao grande número de *tweets* onde somente comentam eventos pessoais, entre outros assuntos. Assim, foram coletados *tweets* de perfis do Twitter conhecidos por relatar eventos de trânsito, o perfil do Centro de Operações Integradas do Rio de Janeiro e do da Linha Amarela, ambos administrados por entidades oficiais. Foram utilizadas amostras compostas exclusivamente de *tweets* destes perfis para preparar o modelo CRF para também identificar eventos e localizações em *tweets* com a estrutura utilizada por entidades oficiais, utilizando uma linguagem mais formal. Também, vários termos utilizados por estes perfis também podem ser utilizados por pessoas comuns, sendo assim era esperado que a utilização destes *tweets* fosse aumentar a qualidade do etiquetamento.

A Tabela 3 resume a quantidade de *tweets* utilizadas em cada amostra e o total de *tweets* utilizado no processo de treinamento.

Amostra 1 (Dicionário)	4200
Amostra 2 (Geral)	5510
Amostra 3 (Oficial)	1370
Total	11080

Tabela 3 - Números das amostras de treinamento

Embora toda a base de *tweets* possua mais de quatro milhões de *tweets*, apenas uma pequena parcela destes possui ou alguma localização, ou algum evento, ou ambos. Para efeito de experimento a base de *tweets* como um todo foi etiquetada utilizando CRF e os *tweets* com alguma entidade foram contados totalizando apenas 173.012 *tweets*. Ainda assim, o total de *tweets* de treinamento pode ser considerado pequeno, porém, observando os bons resultados conquistados durante a análise dos resultados preliminares, foi possível concluir que este número de *tweets* no conjunto de treinamento já é suficiente para conseguir excelentes resultados. Os resultados preliminares e finais serão discutidos a seguir.

4 Resultados

4.1 Resultados preliminares

Os resultados preliminares foram obtidos de testes realizados durante o processo de treinamento, como discutido anteriormente. As três amostras na tabela acima foram corrigidas de forma gradual e periodicamente um teste preliminar foi realizado para indicar a performance do modelo até aquele momento. No total 10 modelos foram treinados, as três amostras indicadas na tabela acima foram divididas também em 10 partes de acordo com o andamento do treinamento.

A Amostra 1 (dicionário) foi separada em três, dando origem a três versões do modelo, assim, a versão 3 do modelo possui todos os *tweets* da amostra 1. A amostra 2 também foi dividida em três partes dando origem a mais 3 versões do modelo. Portanto, a versão 6 do modelo possui todos os *tweets* da amostra 1 e da amostra 2. A amostra 3, por fim, foi dividida em 4 partes e ao final, o décimo e último modelo contém todos os *tweets* das três amostras.

A Figuras 19, 20 e 21 mostram os resultados dos testes preliminares a partir de Amostra 2, onde os testes começaram a serem realizados. Para facilitar o entendimento, as partes da amostra 2 são denotadas como 2.1, 2.2 e 2.3.

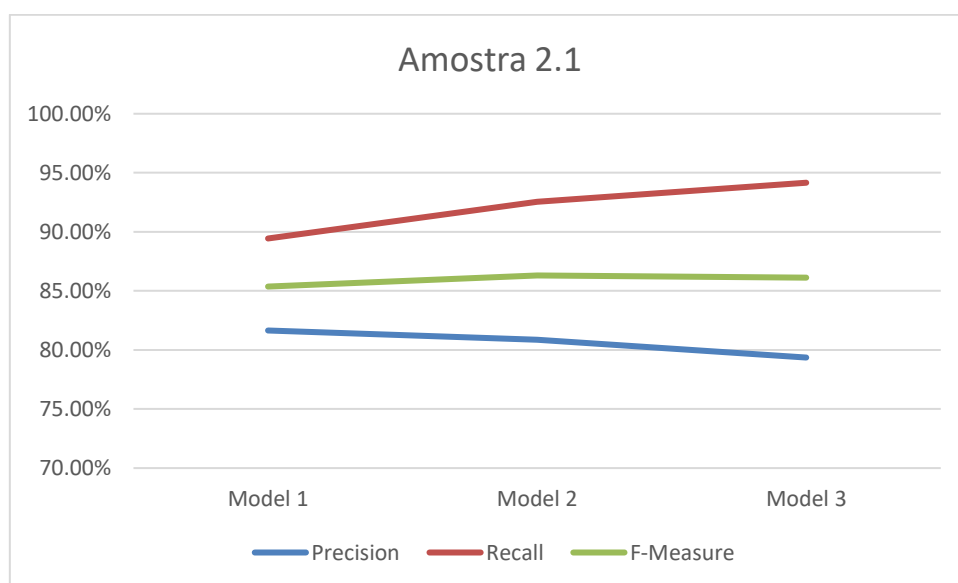


Figura 19 - Gráfico comparativo entre modelos (Amostra 2.1)

A primeira parte da Amostra 2 continha 2335 *tweets* dos 5510 totais da amostra. Foram etiquetadas 27679 palavras em 2878 sentenças. Neste momento do treinamento, havia três modelos, frutos da amostra 1, todos os três foram testados utilizando a amostra 2.1 para ser possível identificar a melhora nos resultados de um modelo para outro. O *recall* teve um aumento considerável do modelo 1 até 3 obtendo 89.44% no modelo 1 e 94.16% no Modelo 3. A precisão, no entanto, sofreu queda. Com o aumento do conjunto de treinamento o modelo passou a identificar mais entidades como é possível ver nas tabelas abaixo, porém apesar da cobertura boa, também passou a obter muitos falsos positivos. Acredita-se que a causa para isto seja o ainda pouco conhecimento do modelo, que não possui ainda fortes relações e proteções contra ambiguidades.

Modelo 1						
Entity	P	R	F1	TP	FP	FN
EVENT	0	0	0	0	0	1
LOCATION	0.8165	0.8948	0.8538	2389	537	281
Totals	0.8165	0.8944	0.8537	2389	537	282

Modelo 2						
Entity	P	R	F1	TP	FP	FN
EVENT	0	0	0	0	0	1
LOCATION	0.8086	0.9258	0.8633	2472	585	198
Totals	0.8086	0.9255	0.8631	2472	585	199

Modelo 3						
Entity	P	R	F1	TP	FP	FN
EVENT	0	0	0	0	0	1
LOCATION	0.7936	0.9419	0.8614	2515	654	155
Totals	0.7936	0.9416	0.8613	2515	654	156

Tabela 4 - Precisão, Recall e F-Measure dos modelos 1, 2 e 3

A amostra 2.1 foi adicionada ao conjunto de treinamento e o modelo 4 foi treinado. Após isto o modelo foi testado utilizando a amostra 2.2 que não faz parte do conjunto de treinamento. Os resultados melhoraram consideravelmente como é possível verificar na Figura 20. No entanto, nesta amostra os resultados se invertem mostrando a precisão maior enquanto que o *recall* se mantém mais abaixo. O Modelo 4, no entanto, inverte este resultado com o *recall* um pouco acima como é possível verificar na Tabela 5.

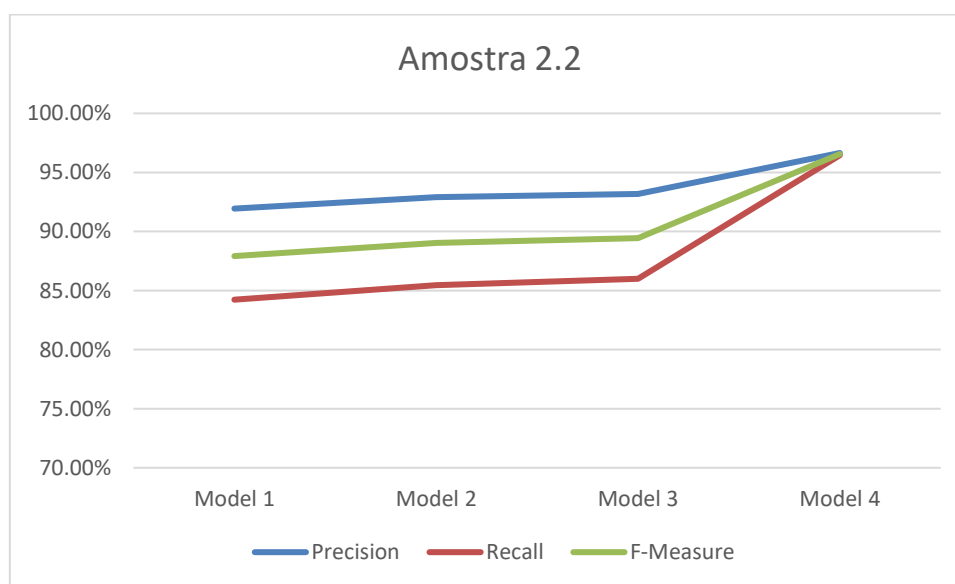


Figura 20 - Gráfico comparativo entre modelos

Modelo 4						
Entity	P	R	F1	TP	FP	FN
EVENT	0	0	0	0	0	18
LOCATION	0.9665	0.9704	0.9685	2887	100	88
Totals	0.9665	0.9646	0.9656	2887	100	106

Tabela 5 - Precisão, Recall e F-Measure do modelo 4

Com a Amostra 2.3 o Modelo 5 continua a tendência havendo aumento do recall e uma pequena queda na precisão. Apesar dos resultados serem muito satisfatórios quando diz respeito a identificação de localizações, são muito inconclusivos em relação a eventos.

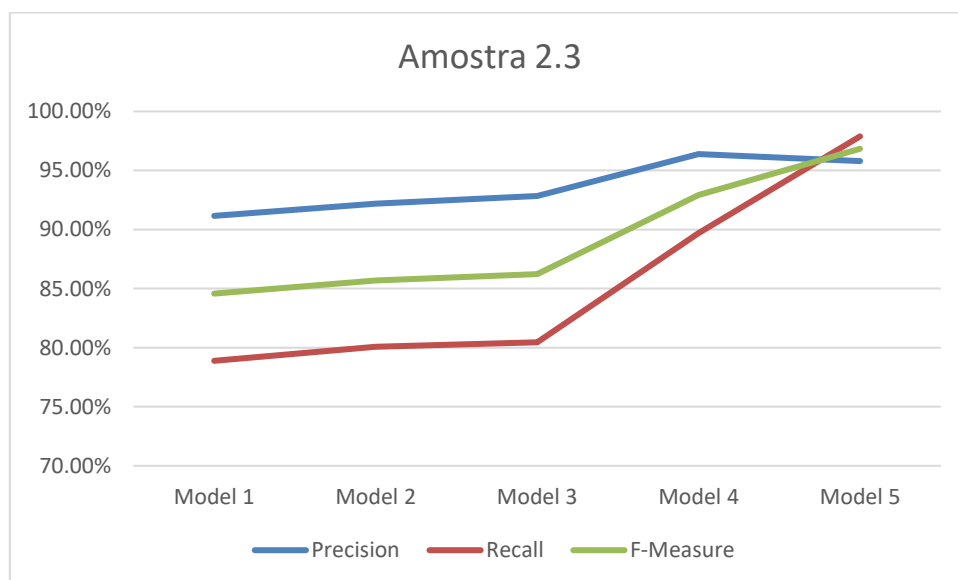


Figura 21 - Gráfico comparativo dos modelos 1 a 5

Modelo 5						
Entity	P	R	F1	TP	FP	FN
EVENT	0	0	0	0	0	6
LOCATION	0.958	0.9828	0.9703	1484	65	26
Totals	0.958	0.9789	0.9684	1484	65	32

Tabela 6 - Precisão, Recall e F-Measure do modelo 5

Não houve eventos suficientes para treinar o modelo e nem para testá-lo. Como dito anteriormente, a maioria dos *tweets* não são sobre eventos de trânsito, naturalmente. Assim, foi decidido testar o modelo utilizando uma outra base mais focada em eventos de trânsito, uma base de *tweets* extraídos de dois perfis oficiais que gerenciam o trânsito da cidade do Rio de Janeiro: o perfil do Centro de Operações do Rio de Janeiro¹² e o da

¹² <https://twitter.com/OperacoesRio>

concessionária da Linha Amarela¹³, importante via expressa da cidade. Assim, o modelo 6, criado a partir da soma das amostras 1 e 2 foi testado utilizando *tweets* da Amostra 3. O resultado pode ser visto na Figura 22 e Tabela 7.

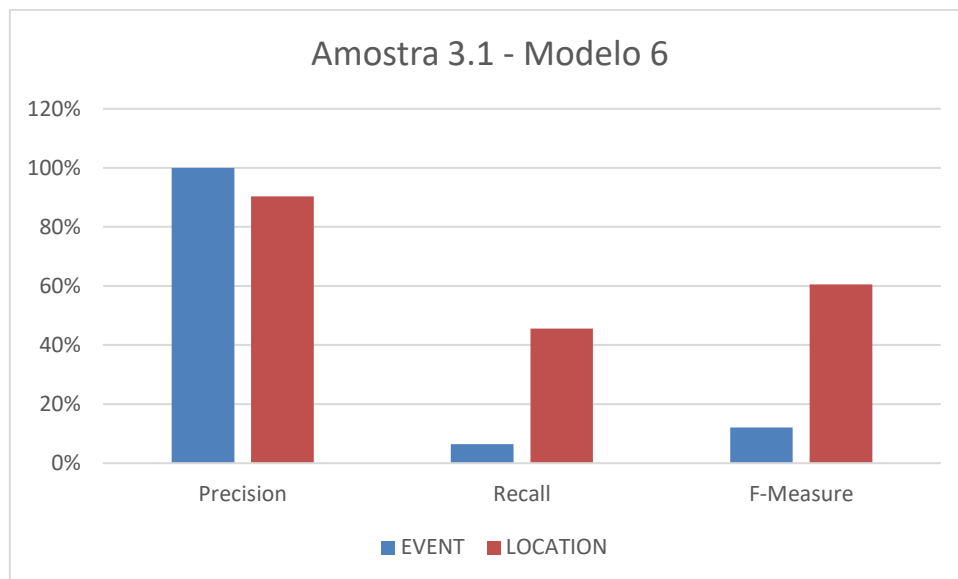


Figura 22 - Resultados do Modelo 6

Modelo 6						
Entity	P	R	F1	TP	FP	FN
EVENT	1	0.0641	0.1205	5	0	73
LOCATION	0.9035	0.4558	0.6059	103	11	123
Totals	0.9076	0.3553	0.5106	108	11	196

Tabela 7 - Precisão, Recall e F-Measure do Modelo 6

O Modelo 6 não foi treinado para identificar eventos e localizações em *tweets* de entidades oficiais, onde o vocabulário é mais técnico, formal e preciso, portanto o modelo teve um desempenho pobre, o que era esperado. Este resultado também indica que a precisão deve ser analisada com cautela como métrica já que a mesma foi ótima apesar do baixo desempenho no geral, principalmente em *recall*. A fim de melhorar os resultados para *tweets* oficiais um modelo 7 foi criado utilizando todo o conjunto de treinamento, mais a Amostra 3.1. O resultado pode ser observado na Figura 24 e na Tabela 8.

¹³ <https://twitter.com/LinhaAmarelaRJ>

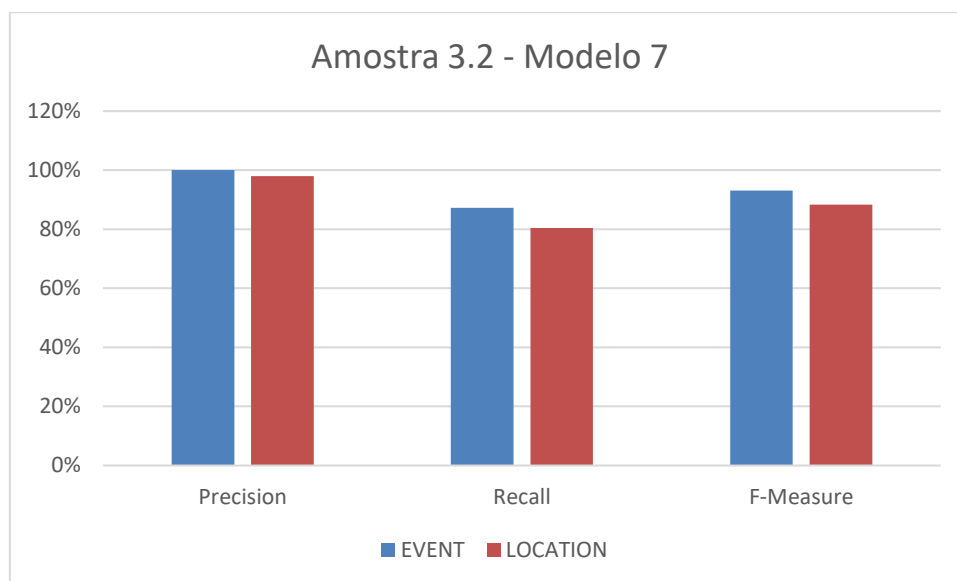


Figura 23 - Resultados do modelo 7

Modelo 7						
Entity	P	R	F1	TP	FP	FN
EVENT	1	0.8721	0.9317	75	0	11
LOCATION	0.9797	0.8042	0.8833	193	4	47
Totals	0.9853	0.8221	0.8963	268	4	58

Tabela 8 - Precisão, Recall e F-Measure do modelo 7

Após treinar um novo modelo, desta vez incluindo *tweets* dos dois perfis mencionados foi notado uma grande melhora tanto na identificação de eventos como em localizações. A identificação de eventos se tornou mais eficiente que a identificação de localizações, que sofreu uma queda se comparado aos resultados anteriores do modelo 5, por exemplo. Isto se deve a mudanças no vocabulário utilizado junto com nomes de localizações, fazendo o modelo reaprender e realizar novas conexões. O fato de que a identificação de eventos foi a mais eficiente é interessante já que este tipo de entidade tem mais tendência a sofrer com ambiguidades do que nomes de locais, que são nomes próprios por natureza.

O Modelo 8 sofreu degradações na identificação de eventos quando testado com a Amostra 3.3. A identificação de localizações voltou a ser a mais eficiente, mas ainda não no patamar anterior onde era utilizado somente *tweets* de público geral. A degradação nesta amostra pode ser estudada mais a fundo analisando os resultados dos testes, porém

isso não foi feito por se tratar apenas de um teste preliminar. Os resultados podem ser vistos na Figura 24 e Tabela 9.

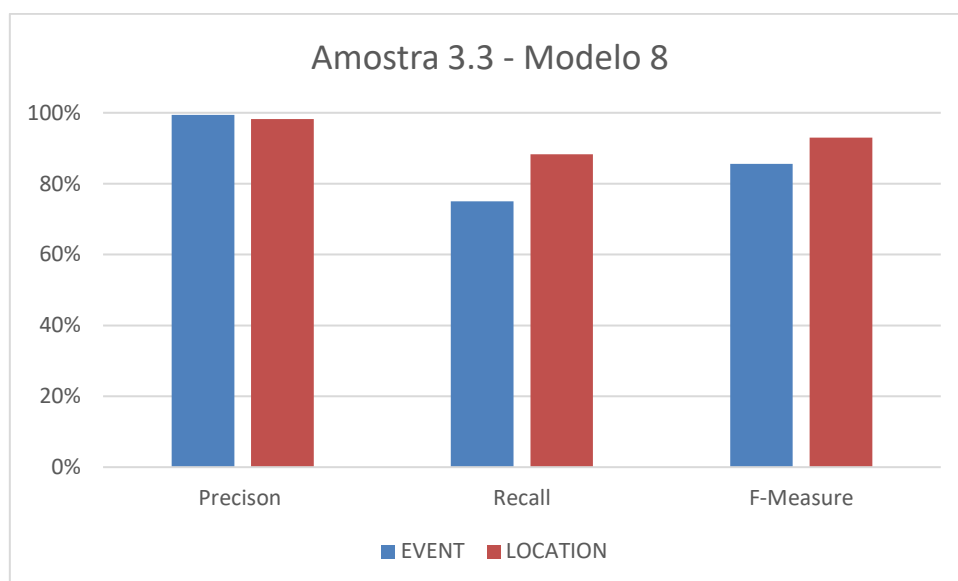


Figura 24 - Resultados do modelo 8

Modelo 8						
Entity	Precision	Recall	F-Measure	TP	FP	FN
EVENT	0.9945	0.7505	0.8555	361	2	120
LOCATION	0.9823	0.8834	0.9302	1220	22	161
Totals	0.985	0.8491	0.912	1581	24	281

Tabela 9 - Precision, Recall e F-Mesure do modelo 8

A degradação no teste anterior surtiu um efeito benéfico nos resultados do Modelo 9, último modelo testado preliminarmente, havendo um aumento de precisão em ambos os tipos de entidades. Os resultados podem ser vistos na Figura 25 e Tabela 10.

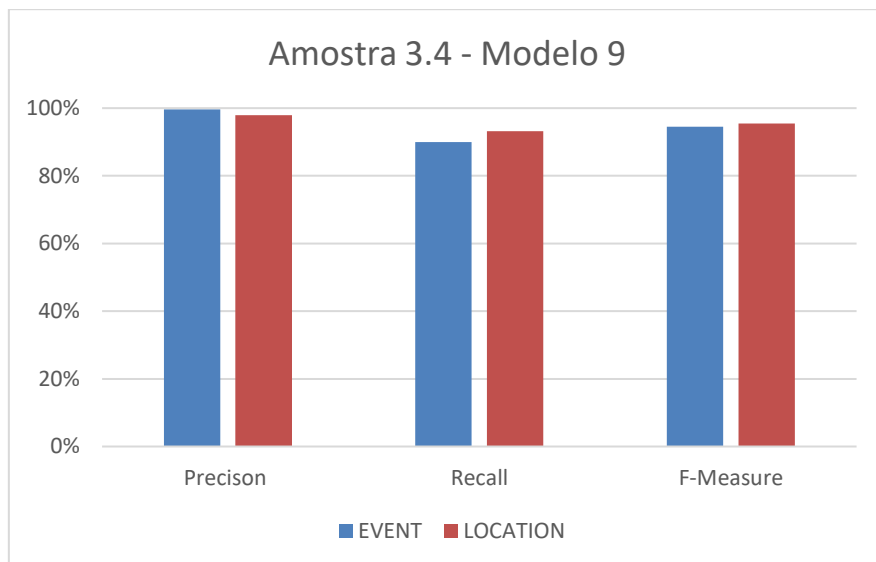


Figura 25 - Resultados do modelo 9

Modelo 9						
Entity	Precision	Recall	F-Measure	TP	FP	FN
EVENT	99.62%	89.98%	94.56%	521	2	58
LOCATION	97.89%	93.17%	95.47%	1528	33	112
Totals	98.32%	92.34%	95.24%	2049	35	170

Tabela 10 - Precisão, Recall e F-Measure do modelo 9

Como os testes foram feitos utilizando amostras diferentes em todos os modelos, uma comparação entre os resultados é inconclusiva. Portanto, esta última amostra foi utilizada em teste para os modelos 6 a 9. Os gráficos com a comparação se encontram na Figura 26.

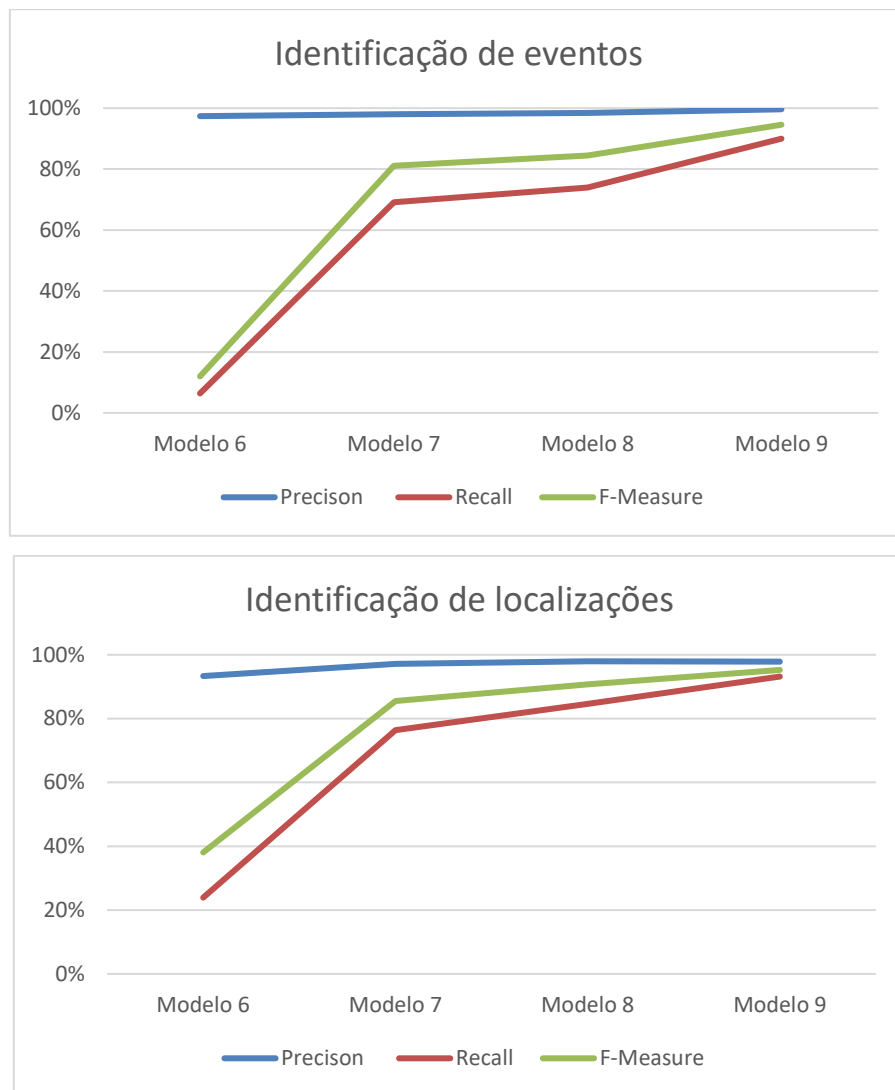


Figura 26 - Gráficos de comparação entre modelos 6 a 9

O aperfeiçoamento do modelo fica claro, concluindo que ao aumentar o conjunto de treinamento, os resultados também seguem o mesmo ritmo, melhorando em qualidade. Também pode-se concluir que a precisão não é o grande desafio no treinamento de modelos CRF já que a métrica apresentou comportamento quase constante entre os modelos. Já o recall teve acentuado aumento sendo de fato a métrica mais interessante no treinamento. De fato, o aumento do recall, que denota a cobertura do modelo, ou seja, quantas entidades verdadeiras foram realmente descobertas, é o objetivo em um conjunto de entidades vasto como o de eventos e localizações, almejando identificar entidades que até mesmo não aparecem no conjunto de treinamento explicitamente.

Com os resultados em ambas entidades acima ou próximo a 90%, concluiu-se que o modelo já apresentava um resultado bastante satisfatório e, portanto, seria possível

entrar na fase final do treinamento do modelo. Assim, a amostra 3.4 utilizada nos últimos testes preliminares foi adicionada ao conjunto de treinamento e o modelo 10 foi criado, com o intuito de ser o modelo final utilizado no framework para casos de uso.

4.2 Resultados finais

Os testes finais foram realizados utilizando três amostras diferentes para avaliar vários aspectos do etiquetamento. Foi percebido durante os testes preliminares que a identificação de eventos em *tweets* do público geral não é uma tarefa fácil devido ao conjunto de dados esparsos, ou seja, há poucos *tweets* onde algum evento é mencionado em comparação ao grande número de *tweets* onde nenhum evento é mencionado. A identificação de localidades conseguiu resultados notáveis já que localidades são mencionadas mais frequentemente e houve oportunidade de o modelo aprender a identificá-las com eficácia. Com o objetivo de obter resultados relevantes na identificação de eventos, foram extraídos *tweets* de perfis próprios para notificação de eventos de trânsito, obtendo sólidos resultados nos testes preliminares.

Os testes finais serão realizados com o objetivo de responder as seguintes questões:

1. O modelo 10 obterá resultados superiores ao modelo 9 quando testado em *tweets* de perfis oficiais?
2. O treinamento com *tweets* de perfis oficiais melhorará a performance de identificação de eventos em *tweets* do público geral?
3. Os resultados sofrem degradação quando *tweets* do público geral e de perfis oficiais são postos juntos em um mesmo conjunto de teste?

Responder estas perguntas é importante para decidir o melhor caminho em casos de uso: se somente *tweets* de perfis oficiais serão usados, por exemplo, bem como concluir sobre a eficácia do etiquetamento em ambos os métodos: utilizando perfis públicos e oficiais. Assim, foram construídas três novos conjuntos de *tweets* para o treinamento final como dispostos abaixo:

- 1) Conjunto com *tweets* do público geral: 1993 *tweets*
- 2) Conjunto com *tweets* de perfis oficiais: 1000 *tweets*
- 3) Conjunto com *tweets* misto: 1018 *tweets* oficiais e 936 de perfis públicos, 1954 no total.

Os três conjuntos foram confeccionados com o objetivo de responderem as três perguntas acima. Os *tweets* advindos de perfis públicos foram retirados da base de

tweets anteriormente mencionada, já os *tweets* de perfis oficiais foram extraídos separadamente, entre o período de 10 de outubro de 2017 a 17 de outubro de 2017 para o segundo conjunto e entre 2 de novembro de 2017 e 10 de novembro de 2017 para o terceiro conjunto. Os resultados dos testes podem ser analisados na Tabela 11.

Amostra 1						
Entity	Precision	Recall	F-Measure	TP	FP	FN
EVENT	0	0	0	0	0	4
LOCATION	0.9505	0.9275	0.9389	192	10	15
Totals	0.9505	0.91	0.9298	192	10	19

Amostra 2						
Entity	Precision	Recall	F-Measure	TP	FP	FN
EVENT	1	0.9645	0.9819	624	0	23
LOCATION	0.9797	0.9445	0.9618	1499	31	88
Totals	0.9856	0.9503	0.9676	2123	31	111

Amostra 3						
Entity	Precision	Recall	F-Measure	TP	FP	FN
EVENT	0.9829	0.9198	0.9503	516	9	45
LOCATION	0.9735	0.9674	0.9704	1395	38	47
Totals	0.976	0.9541	0.9649	1911	47	92

Tabela 11 - Resultados dos testes finais

Os resultados obtidos nos três testes foram em sua maioria excelentes, exceto o resultado da identificação de eventos na amostra 1, composta por *tweets* de perfis do público geral. Este resultado responde à pergunta 2 no qual os testes foram baseados. De fato, em artigos onde foram utilizadas técnicas similares, resultados semelhantes foram obtidos. Em [Anantharam et al. 2015] foi obtido 68% e 50% de precisão para as etiquetas B-EVENT e I-EVENT e recall de 57% e 7%, respectivamente. É importante destacar que ao contrário dos testes realizados em [Anantharam et al. 2015], neste trabalho os testes foram realizados levando em consideração as entidades como um todo, não separando por etiquetas, assim, para comparar os resultados, foram desenhadas as matrizes de confusão expostas nas Figuras 12, 13 e 14 com o resultado de cada etiqueta separadamente, como em [Anantharam et al. 2015].

	B-LOCATION	I-LOCATION	B-EVENT	I-EVENT	O	Total	Precision
B-LOCATION	194	0	0	0	13	207	93.72%
I-LOCATION	0	303	0	0	20	323	93.81%

B-EVENT	0	0	0	0	4	4	0.00%
I-EVENT	0	0	0	0	0	0	N/A
O	8	15	0	0	18613	18636	99.88%
Total	202	318	0	0	18650	19170	
Recall	96.04%	95.28%	N/A	N/A	99.80%		

Tabela 12 - Matriz de confusão referente à Amostra Final 1

	B-LOCATION	I-LOCATION	B-EVENT	I-EVENT	O	Total	Precision
B-LOCATION	1503	5	0	0	79	1587	94.71%
I-LOCATION	7	1144	0	0	53	1204	95.02%
B-EVENT	1	0	622	0	22	645	96.43%
I-EVENT	0	0	0	637	28	665	95.79%
O	65	73	1	2	11317	11458	98.77%
Total	1576	1222	623	639	11499	15559	
Recall	95.37%	93.62%	99.84%	99.69%	98.42%		

Tabela 13 - Matriz de confusão referente à Amostra Final 2

	B-LOCATION	I-LOCATION	B-EVENT	I-EVENT	O	Total	Precision
B-LOCATION	1398	3	0	0	41	1442	96.95%
I-LOCATION	4	1084	0	0	35	1123	96.53%
B-EVENT	1	0	513	0	44	558	91.94%
I-EVENT	0	0	0	530	17	547	96.89%
O	73	79	28	9	19607	19796	99.05%
Total	1476	1166	541	539	19744	23466	
Recall	94.72%	92.97%	94.82%	98.33%	99.31%		

Tabela 14 - Matriz de Confusão referente à Amostra Final 3

O resultado alcançado por este trabalho foi inferior ao de [Anantharam et al. 2015] na primeira amostra, porém muito superior nas amostras 2 e 3 onde *tweets* de perfis oficiais são utilizados fortalecendo a hipótese de que estes tipos de perfis podem ser uma grande fonte de informação confiável para treinar o modelo de forma eficaz e extrair informação posteriormente.

O método utilizado neste trabalho também alcançou resultados melhores na identificação de localizações. [Farajidavar et al. 2017] obteve melhores resultados em eventos e em localizações, apesar da pouca diferença no segundo. Isto corrobora para a utilização de CNN juntamente com CRF em trabalhos futuros.

Ao comparar os resultados do modelo 9 com o modelo 10 é possível perceber uma melhora respeitável nos resultados, principalmente no que diz respeito a identificação de entidades em *tweets* de perfis oficiais, assim a primeira questão de teste possui uma resposta positiva reforçando a ideia de que a qualidade do modelo é proporcional ao tamanho do conjunto de treinamento. Por fim, a terceira pergunta é respondida analisando

os resultados do teste da amostra 3 onde ambos os tipos de *tweets* foram deliberadamente postos em um único conjunto de testes. Esta pergunta é importante devido ao fato de que, apesar da baixa eficácia do modelo em detectar eventos em *tweets* do público geral, levando a utilização de *tweets* de perfis oficiais, o modelo ainda pode aprender mais sobre os *tweets* do público geral enquanto que em paralelo utiliza *tweets* de perfis oficiais para extrair eventos de uma forma comprovadamente eficiente. Acredita-se que este paralelismo seja benéfico para o aprendizado contínuo do modelo. A resposta para a terceira pergunta, portanto, pode ser verificado visualizando os resultados da amostra 2 e 3. O resultado é relativamente dubio, obtendo resultados ligeiramente inferiores na identificação de eventos, mas superiores na identificação de localizações. Independente disto, os resultados alcançados na amostra 3 foram muito satisfatórios indicando que ambas as abordagens podem ser usadas em paralelo com poucos prejuízos a eficiência do modelo.

Os resultados dos testes finais mostraram que o modelo treinado neste trabalho é competente e obtém resultados similares ou melhores que trabalhos anteriores, porém também repete os mesmos problemas com relação a identificação de eventos. Assim, foi proposta uma abordagem onde *tweets* de perfis oficiais são utilizados obtendo resultados extremamente satisfatórios. Portanto, para a demonstração da utilidade do framework como ferramenta para obtenção de dados de eventos, uma aplicação web foi desenvolvida onde, utilizando um mapa, os eventos, juntamente com o *tweet* de origem, serão marcados com a sua localização em coordenadas proporcionadas pela Google Places API Web Service utilizando a localização retirada dos *tweets*. Para esta demonstração serão utilizados apenas *tweets* de perfis oficiais visto que obtiveram resultado superior com menos esforço.

5 Caso de Uso

A aplicação web construída para demonstrar as capacidades do framework desenvolvido consiste em uma simples aplicação de mapa que tem como funcionalidade mostrar os eventos e sua localidade em um mapa de fácil entendimento e interação para o usuário. O objetivo é utilizar somente as informações contidas no *tweet* para criar o mapa, ou seja, tanto a localização quanto o evento serão aqueles descritos no *tweet*. Como mencionado anteriormente, o framework possui funções de geolocalização que utilizam as APIs do Google Places ou Nominatim do Open Street Maps para obter as coordenadas do local realizando uma busca utilizando o nome do local extraído do *tweet* pelo modelo CRF.

A aplicação de mapeamento utiliza HTML, PHP e JavaScript como tecnologias básicas, bem como a Google Maps JavaScript API para renderizar o mapa. Foram utilizadas as instruções em [Google 2017] para desenvolver o site da web. Os passos a seguir são realizados pela aplicação para obter os dados e renderizá-los:

1. Script em PHP é utilizado para conectar ao banco de dados.
2. Uma consulta é realizada para obter os dados.
3. As informações são convertidas para formato XML.
4. Na página HTML em um script em Java Script um mapa é inicializado.
5. Os dados em formato XML são carregados utilizando o script em PHP.
6. Para cada entrada, um marcador é criado e as informações, incluindo latitude e longitude são adicionadas.
7. O marcador é adicionado ao mapa.

A Figura 27 mostra a aplicação em funcionamento.

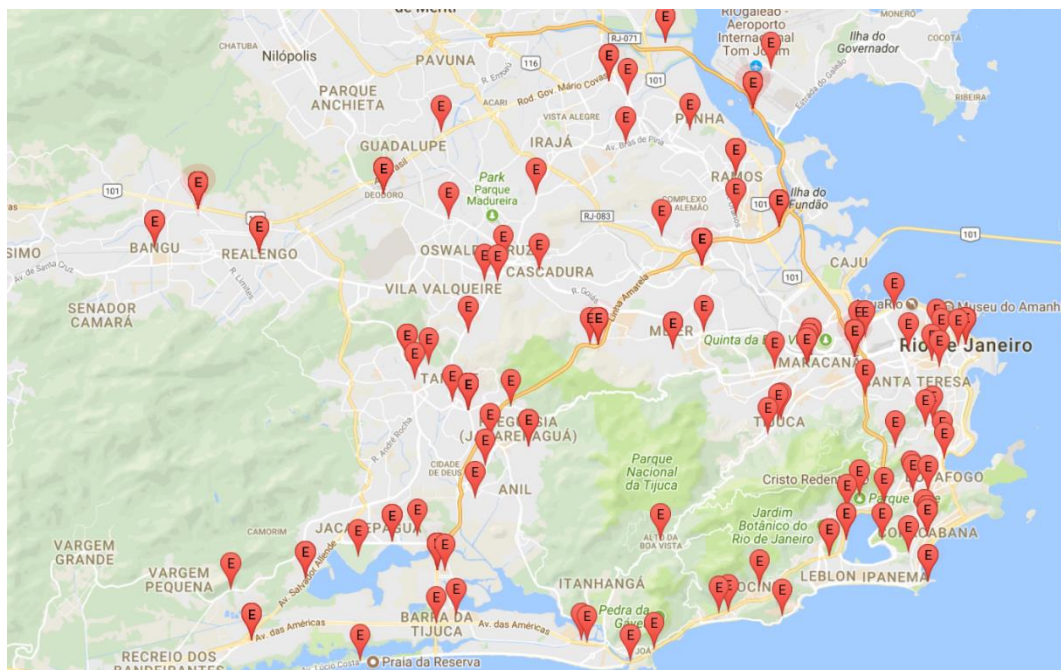


Figura 27 - Visão geral da aplicação de mapa

Na Figura 27 é possível ver o mapa com os pontos de evento espalhados por todo o território da cidade do Rio de Janeiro e região metropolitana. Para esta demonstração foram utilizados parte dos *tweets* utilizados no teste do modelo CRF extraídos entre 2 de novembro de 2017 a 10 de novembro de 2017, não sendo uma demonstração em tempo real, porém é possível utilizar a ferramenta em tempo real. É possível ver que eventos ocorreram em toda a cidade neste período de tempo com uma concentração deles na região do Centro e Zona Sul, como esperado já que o trânsito é mais denso nesta região, principalmente em dias de semana.

Na Figura 28 é mostrado um exemplo de *tweet* que aparece ao clicar no marcador correspondente. Como dito anteriormente, as coordenadas dos marcadores são resultado de uma busca utilizando a Google Places API Web Service utilizando a informação retirada do *tweet*. No exemplo abaixo a API retornou a localização correta com sucesso com uma imprecisão tolerável. A Av. Abrahão Jabour (nomeada na Figura 27 como Rua Abrahão Jabour) está alguns quilômetros afastada da localização retornada pela API como é possível ver na Figura 29. Porém, como eventos de trânsito costumam impactar o trânsito de uma região, esta imprecisão pode ser considerada aceitável.



Figura 28 - Tweet de evento

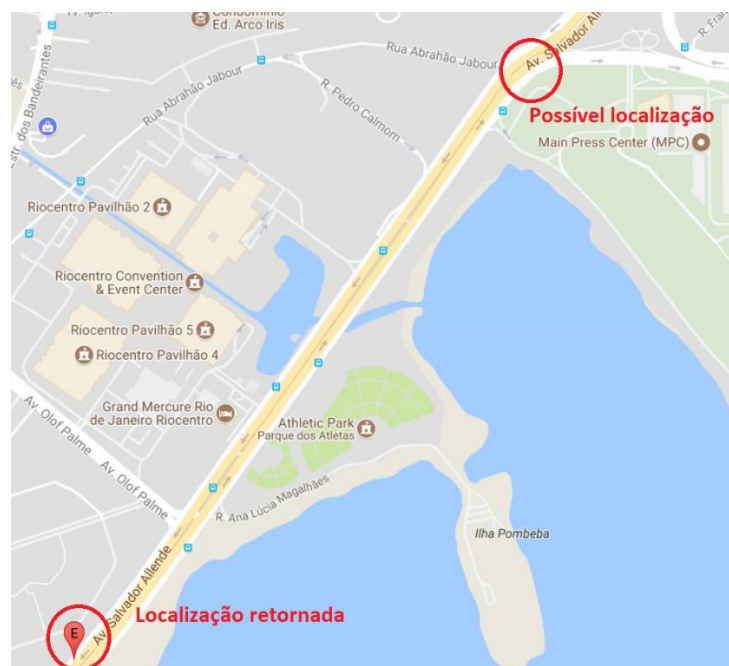


Figura 29 - Avaliação de precisão da API

Porém, é possível obter uma precisão maior utilizando todas as informações que o *tweet* proporciona. No exemplo, a Av. Abraão Jabour é mencionada no *tweet*. Por questão de simplicidade, o framework utilizou a primeira localização encontrada para

obter as coordenadas, porém isto pode ser melhorado utilizando as diferentes localizações no *tweet*, se houver mais de uma localização mencionada, e aumentar com isto a precisão. No entanto, devido à complexidade desta tarefa, a adição deste recurso será contemplada em trabalhos futuros.

Nas Figuras 30 e 31 são mostrados outros exemplos para reforçar a eficácia do *framework*. A utilização da ferramenta em tempo real pode proporcionar uma visão muito satisfatória da condição de trânsito da cidade sem a necessidade de procurar nos perfis oficiais sobre algum evento em sua localidade.



Figura 30 - Evento ocorrido na região do bairro da Freguesia



Figura 31 - Evento ocorrido na região próximo a rodoviária

A Figura 30 mostra um exemplo onde a aplicação obteve um resultado ótimo onde a localização foi bastante precisa, enquanto que na Figura 31 houve certo desvio, a Rodoviária Novo Rio está mais a esquerda do local apontado, porém, como dito, esta precisão é tolerável, o usuário terá condições de tomar sua decisão com esta informação.

Este caso de uso demonstra que o *framework* pode fornecer informações importantes para diversos tipos de informação onde o ponto central está na tomada de decisão por um usuário. Esta simples aplicação de mapas já é capaz de proporcionar informações ao usuário que antes ele precisaria procurar no Twitter, enquanto no mapa se torna mais fácil localizar eventos em sua região. Os testes realizados demonstraram que a informação é extraída com uma precisão confiável. Outras aplicações, como uma aplicação de recomendação de rotas baseado na informação disponibilizada pelo *framework* pode ser desenvolvida no futuro.

6 Conclusão e trabalhos futuros

Este trabalho teve como objetivo construir um framework que proporciona a base para aplicações informativas sobre eventos ocorridos em cidades, neste caso no Rio de Janeiro. O framework possui funcionalidades que permitem ao desenvolvedor configurar e extrair *tweets* utilizando a API do Twitter de forma fácil e sem esforço de desenvolvimento. Outro módulo do framework classifica os *tweets* utilizando *Conditional Random Fields* (CRF), sendo possível realizar a classificação utilizando dados de qualquer cidade, desde que um modelo apropriado seja treinado para obter resultados mais precisos. A comunicação entre os módulos é feita por um *message broker*, o RabbitMQ, garantindo eficiência na troca de dados. Todos os dados são gravados em um banco de dados para futuras análises. Uma aplicação de mapa também foi desenvolvida como exemplo de caso de uso do framework obtendo resultados que comprovam a utilidade da aplicação em proporcionar ao usuário a informação que ele precisa de uma forma muito mais rápida e fácil, sem a necessidade de procurar no Twitter por eventos em sua localidade.

No que se refere à utilização de *Conditional Random Fields* (CRF) para classificar e extrair as informações de eventos e localidades os testes realizados comprovaram que o modelo treinado funciona muito bem na identificação de localizações em ambos *tweets* do público geral e em *tweets* oficiais, com o devido treinamento. Na classificação de eventos, no entanto, o modelo obteve excelentes resultados com *tweets* oficiais, onde o número de eventos é naturalmente maior, enquanto que em *tweets* do público geral, onde a quantidade de eventos é muito pequena em relação ao número de *tweets* sem eventos, não foi considerado eficiente. No entanto, é possível que este problema seja resolvido com uma amostra maior de *tweets*. Outros trabalhos como [Anantharam et al. 2015] e [Farajidavar et al. 2017] também obtiveram poucos eventos e eficiência menor na identificação de eventos em relação a localizações demonstrando que o problema é mais profundo e precisa de mais estudos para ser solucionado. Apesar do resultado não favorável, os *tweets* originados de perfis oficiais podem ser utilizados para extrair informação relevante e proporcionar ao usuário acesso facilitado a esta informação.

Ainda é possível melhorar o framework, no entanto, com novas funcionalidades e melhoramento de código visando a maior eficiência no processo de extração,

classificação e entrega da informação a aplicações, portanto o framework estará disponível no GitHub¹⁴ com o intuito de receber melhoramentos da comunidade. Como mencionado anteriormente, o conjunto de treinamento pode ser aumentado em trabalhos futuros para verificar se a eficiência do modelo em *tweets* do público geral pode ser aperfeiçoada. Além disso, a extração de *hashtags*, elementos importantes no Twitter, pode ser utilizada em experimentos para verificar possível melhoria em resultados, assim como a utilização de distância de edição e lematização, entre outras técnicas de PLN. Além disso, a estratégia de treinamento também pode ser aperfeiçoada, como por exemplo, atribuindo mais valor a *tweets* recentes e penalizando *tweets* antigos com o passar do tempo. Outro possível experimento é a utilização de *Convolutional Neural Networks* (CNN) em novos experimentos. A técnica não foi utilizada neste trabalho pelos argumentos expostos anteriormente, mas é importante utilizar diferentes técnicas para se tentar obter melhores resultados sempre que possível. Além disso, é um tópico atual que está começando a ser muito abordado pela comunidade interessada no tema.

¹⁴ <https://github.com/Leoat12/TwitterNLP>

Referências Bibliográficas

- Adomavicius, G., Sankaranarayanan, R., Sciences, M., Sen, S. and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, v. 23, n. 1, p. 103–145.
- Adomavicius, G. and Tuzhilin, A. (2015). Context-aware recommender systems. *Recommender systems handbook*, p. 191–226.
- Anantharam, P., Barnaghi, P., Thirunarayan, K. and Sheth, A. (2015). Extracting City Traffic Events from Social Streams. *ACM Trans. on Intellig. Syst. & Tech. V, N, Article A*, v. 6, n. 4, p. 2–27.
- Ang, L.-M. and Seng, K. P. (2016). Big Sensor Data Applications in Urban Environments. *Big Data Research*, v. 4, p. 1–12.
- Bajaj, G., Agarwal, R., Bouloukakis, G., et al. (2016). Towards building real-time, convenient route recommendation system for public transit. *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings*,
- Britz, D. (2015). Understanding Convolutional Neural Networks for NLP. <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>, [accessed on Nov 28].
- Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and UserAdapted Interaction*, v. 12, n. 4, p. 331–370.
- CIA (2017). The World Factbook. <https://www.cia.gov/library/publications/the-world-factbook/fields/2212.html>, [accessed on Nov 28].
- Crooks, A., Croitoru, A., Stefanidis, A. and Radzikowski, J. (2013). #Earthquake: Twitter as a Distributed Sensor System. *Transactions in GIS*, v. 17, n. 1, p. 124–147.
- Dan Puiu, Daniel Kümper, Marten Fischer, Sefki Kolozali, Nazli Farajidavar, Feng Gao, Thorben Iggena, Thu-Le Pham, Daniel Puschmann, Joao Fernandes, Bogdan Serbanescu, C. M. (2013). Real-Time IoT Stream Processing and Large-scale Data Analytics for Smart City Applications. *Eu Fp7*
- Doran, D., Gokhale, S. and Dagnino, A. (2013). Human sensing for smart cities. *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*, p. 1323–1330.
- Doran, D., Severin, K., Gokhale, S. and Dagnino, A. (2015). Social media enabled human sensing for smart cities. *AI Communications*, v. 29, n. 1, p. 57–75.
- Farajidavar, N., Kolozali, S. and Barnaghi, P. (2017). A Deep Multi-View Learning Framework for City Event Extraction from Twitter Data Streams.
- Google (2017). Using MySQL and PHP with Google Maps. <https://developers.google.com/maps/documentation/javascript/mysql-to-maps>, [accessed on Nov 28].
- Hancke, G. P. and De Silva, B. de C. (2013). *The role of advanced sensing in smart cities*. v. 13

- Itoh, M., Yokoyama, D., Toyoda, M., et al. (2014). Visual fusion of mega-city big data: An application to traffic and tweets data analysis of Metro passengers. *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*, p. 431–440.
- Linvi (2016). Tweetinvi Library Documentation. <https://github.com/linvi/tweetinvi/wiki>, [accessed on Nov 28].
- Medeiros, R. P. and Ribeiro, P. B. (2012). Extração de Entidades Nomeadas Com Maximização de Entropia (Opennlp). p. 10.
- Microsoft (2017a). .NET Core RID Catalog. <https://docs.microsoft.com/en-us/dotnet/core/rid-catalog>, [accessed on Nov 28].
- Microsoft (2017b). Parallel Programming in .NET. <https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/index>, [accessed on Nov 28].
- Miranda, F., Doraiswamy, H., Lage, M., et al. (2016). Urban Pulse: Capturing the Rhythm of Cities. *IEEE Transactions on Visualization and Computer Graphics*, v. 23, n. 1, p. 791–800.
- Moura, P. N. de S. (2009). Reconhecimento de entidades mencionadas utilizando uma abordagem de Hidden Markov Model. *Trabalho de Conclusão de Curso (UNIRIO)*,
- Neirotti, P., De Marco, A., Cagliano, A. C., Mangano, G. and Scorrano, F. (2014). Current trends in smart city initiatives: Some stylised facts. *Cities*, v. 38, p. 25–36.
- Oracle (2017). Optimizing INSERT Statements. <https://dev.mysql.com/doc/refman/5.7/en/insert-optimization.html>, [accessed on Nov 28].
- Pivotal (2017). RabbitMQ Tutorial for C#. <https://www.rabbitmq.com/tutorials/tutorial-one-dotnet.html>, [accessed on Nov 28].
- Puiu, D., Barnaghi, P., Tönjes, R., et al. (2016). CityPulse: Large Scale Data Analytics Framework for Smart Cities. *IEEE Access*, v. 4, p. 1086–1108.
- Santana, E. F. Z., Chaves, A. P., Gerosa, M. A., Kon, F. and Milojicic, D. (2016). Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *arXiv preprint arXiv:1609.08089*,
- Sbodio, M. L., Calabrese, F., Lorenzo, G. Di, et al. (2014). AllAboard: Visual Exploration of Cellphone Mobility Data to Optimise Public Transport AllAboard: Visual Exploration of Cellphone Mobility Data to Optimise Public Transport. v. 22, n. May 2015, p. 1036–1050.
- StanfordNLP (2017). StanfordNLP Softwares. <https://nlp.stanford.edu/software/>, [accessed on Nov 28].
- Statista (2017). Social Media Statistics & Facts. <https://www.statista.com/topics/1164/social-networks/>, [accessed on Nov 28].
- Sutton, C. (2012). An Introduction to Conditional Random Fields. *Foundations and Trends® in Machine Learning*, v. 4, n. 4, p. 267–373.
- Twitter (2017a). Twitter Libraries. <https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries>, [accessed on Nov 28].
- Twitter (2017b). Twitter Streaming API Connection Guidelines.
- Wang, W., De, S., Toenjes, R., Reetz, E. and Moessner, K. (2012). A comprehensive

ontology for knowledge representation in the internet of things. *Proc. of the 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012*, p. 1793–1798.

Wikipedia (2017a). Message Broker. https://en.wikipedia.org/wiki/Message_broker, [accessed on Nov 28].

Wikipedia (2017b). RabbitMQ. <https://en.wikipedia.org/wiki/RabbitMQ>, [accessed on Nov 28].

Zanella, A., Bui, N., Castellani, A., Vangelista, L. and Zorzi, M. (2014). Internet of things for smart cities. *Internet of Things Journal, IEEE*, v. 1, n. 1, p. 22–32.