Mestrado em Informática e Sistemas

What's Happening? – Uma plataforma de eventos

Trabalho de Projeto apresentado para a obtenção do grau de Mestre em Informática e Sistemas

Autor

Marcelo Aguiar Rodrigues

Orientador

Jorge Fernandes Rodrigues Bernardino

Professor do Departamento de Engenharia Informática e Sistemas

Instituto Superior de Engenharia de Coimbra

Rodrigo Rocha Silva

Professor do Curso de Análise e Desenvolvimento de Sistemas

Faculdade de Tecnologia de Mogi das Cruzes no Centro Paula Souza

Coimbra, Abril, 2018

Abstract

The events are constantly present in our days and are spread through the media, advertisement or through conversations between friends. The digital era has brought with it a number of significant changes in the world of communications and, as result, the dissemination of events is facilitated by social networks that play an important role in the proliferation of events. However, searching events, whether in time or space, is not an easy task these days. Although the technological evolution has allowed the creation of new social event platforms, it is still difficult to know what is happening around our location. Currently, a large number of social events are created and promoted on social networks. With the amount of data that these networks generate, the experience of searching for events is not the most consistent to the user because the results obtained do not always reflect the interests of the user. This work proposes a new concept for a social event platform, named *What's Happening?* aiming to improve the user experience in the search and recommendation of events. In particular, an architecture is proposed for the platform that uses machine learning techniques to classify events obtained from popular social networks, e.g. Facebook, aiming to categorize the events. Given that these services used to obtain events present different data models, it is proposed the LODSE (*Linking Open Descriptions of Social Events*) ontology, based on LODE (*Linking Open Descriptions of Events*) ontology, aiming to improve the integration of the obtained data from these external services, model a social event to be classified and improve the classifications of events. As a form of validation of the machine learning techniques in the proposed architecture as well as the data model created from the LODSE ontology, two experimental evaluations were performed. The first experimental evaluation demonstrated that the best algorithm to classify the event datasets created is Random Forest obtaining 83,33% of correctly classified events. The second experimental evaluation demonstrated that the data model based on the LODSE ontology brings benefits in the classification of events where the results demonstrate an increment of 12.4% of correctly classified events as well as an average runtime gain of 5.9% compared with the data model based on LODE ontology.

Keywords: social events; social events platforms; social event classification; ontologies; machine learning.

Resumo

Os eventos são uma presença constante no nosso dia a dia e são divulgados através dos meios de comunicação, anúncios publicitários ou através de conversas entre amigos. A era digital trouxe consigo uma data de mudanças significativas no mundo das comunicações e consequentemente, a divulgação de eventos é facilitada devido às redes sociais que desempenham um papel importante na proliferação destes. No entanto, a procura de eventos, quer no tempo ou no espaço, não é uma tarefa fácil nos dias que correm. Apesar da evolução tecnológica ter permitido a criação de novas plataformas para divulgação de eventos, ainda existe dificuldade em saber o que está a acontecer em redor da nossa localização. Atualmente, um grande número de eventos sociais é criado e promovido nas redes sociais. Com o aglomerado de informação que estas redes geram, a experiência de procurar eventos não é a mais consistente para o utilizador porque os resultados obtidos nem sempre refletem os interesses do utilizador. Este trabalho propõe um novo conceito para uma plataforma de divulgação de eventos, intitulada de *What's Happening?* com o objetivo de melhorar a experiência do utilizador na procura e recomendação destes. Em particular, é proposta uma arquitetura para a plataforma que utiliza técnicas de *machine learning* para classificar eventos provenientes de redes sociais populares, e.g. Facebook, com o objetivo de os categorizar. Tendo em conta que os serviços utilizados para a obtenção de eventos apresentam modelos de dados diferentes, é proposta a ontologia LODSE (*Linking Open Descriptions of Social Events*), tendo como base a ontologia LODE (*Linking Open Descriptions of Events*), com o objetivo de facilitar a integração dos dados obtidos dos serviços externos, modelar um evento social para ser posteriormente classificado e melhorar a classificação de eventos. Como forma de validação das técnicas de *machine learning* na arquitetura proposta bem como o modelo de dados criado a partir da ontologia LODSE, foram realizadas duas avaliações experimentais. A primeira avaliação experimental demonstrou que o melhor algoritmo para classificar os *datasets* de eventos criados é o *Random Forest* obtendo 83,33% de eventos corretamente classificados. A segunda avaliação experimental demonstrou que o modelo de dados baseado na ontologia LODSE traz benefícios na classificação de eventos demonstrando uma melhoria de 12,4% de eventos corretamente classificados bem como uma melhoria de 5,9% no tempo de processamento quando comparado com o modelo de dados baseado na ontologia LODE.

Palavras Chave: eventos sociais; plataformas de eventos; classificação de eventos sociais; ontologias; machine learning.

Agradecimentos

Quero expressar o meu agradecimento aos dois Professores que me orientaram ao longo desta etapa, Professor Jorge Bernardino e Professor Rodrigo Rocha, pelo apoio prestado, conhecimento e disponibilidade durante todo o projeto.

Aos meus pais pelo apoio incondicional dado durante o meu percurso académico.

A todos os meus colegas pelo apoio, ajuda e ideias partilhadas para o sucesso do projeto.

Índice

Lista de Figuras

Lista de Tabelas

Definições e Acrónimos

API – *Application Programming Interface* – Conjunto de métodos, objetos e protocolos que os programadores podem utilizar para criar aplicações de software ou interagir com sistemas externos.

IMV – *Impute missing values*– técnica de *machine learning* para substituir os dados em falta num conjunto de dados específico por uma média de distribuição genérica.

LODE – *Linking Open Descriptions of Events*– ontologia para publicar descrições de eventos históricos como dados vinculados e para mapeamento entre outros vocabulários e ontologias relacionadas com eventos.

LODSE – *Linking Open Descriptions of Social Events* – ontologia para descrever um evento social.

MMV – *Mark missing values*– técnica de *machine learning* para marcar os dados em falta num conjunto de dados específico.

RMV – *Remove missing values*– técnica de *machine learning* para remover dados em falta num conjunto de dados específico.

1 Introdução

Ao olharmos ao nosso redor, verificamos que a vida está a ficar cada vez mais simples. Pagar as contas de serviços regulares, como por exemplo dos transportes públicos, reservar um hotel ou fazer *check-in* para uma viagem de avião eram tarefas que exigiam tempo e esforço. Hoje, há uma alternativa digital para qualquer tarefa que possamos imaginar, facilitando assim a vida das pessoas.

Vivemos num mundo cada vez mais dependente das novas tecnologias e dos serviços que esta disponibiliza. Como resultado destas inovações, a maioria das pessoas tem mais tempo para descobrir novos lugares, obter novas experiências e participar em eventos que nunca tiveram oportunidade de ir.

Hoje em dia, a procura de eventos é facilitada com os inúmeros serviços existentes, onde facilmente podemos obter recomendações de eventos, baseados na nossa localização ou simplesmente procurar por um determinado evento, num determinado local, para obter mais informações sobre este, como por exemplo o preço. No entanto, a procura de eventos, quer no tempo ou no espaço, não é uma tarefa fácil nos dias que correm.

O acesso facilitado à informação tem permitido a criação de novos serviços para resolver os problemas diários da vida das pessoas, quer seja no trabalho ou no lazer. Na área dos eventos, existe uma necessidade na procura destes, que apesar dos serviços existentes, estes apresentam lacunas na procura, recomendação e notificação de eventos, existindo espaço para melhorias e criação de novos paradigmas e conceitos.

Em suma, este capítulo apresenta em detalhe o tema do trabalho realizado, descrevendo a motivação, o seu enquadramento e as principais contribuições. Por fim, é apresentada a estrutura do presente documento.

1.1 Motivação

Hoje em dia, existe um novo paradigma no mercado empresarial que é caracterizado por empresas que fogem ao modelo das empresas tradicionais, denominadas por *Startups* (Serrasqueiro, Nunes, Leitao, & Armada, 2010). Estas distinguem-se por apresentar características inovadoras, conceitos de negócio diferenciadores e rejuvenescem a indústria com tecnologias disruptivas, principalmente na área das tecnologias de informação e comunicação.

Uma *Startup* parte quase sempre de uma ideia ou de uma necessidade pessoal, que de uma forma simples, pode ser desenvolvida e disponibilizada na Internet. Essa mesma necessidade pessoal, no âmbito dos eventos sociais, é o motivo para a realização deste trabalho, isto porque, existe dificuldades em procurar eventos e/ou obter eventos recomendados para uma determinada localização (Hendrickson, 2012). Na tentativa de resolver estas dificuldades, foi idealizado um novo conceito para uma plataforma de divulgação de eventos que tem como objetivo melhorar a experiência do utilizador no que diz respeito a estas plataformas.

1.2 Enquadramento

Os eventos são uma maneira natural de demonstrar uma ocorrência observável, agrupando pessoas, lugares, tempo e atividades (Shaw, Troncy, & Hardman, 2009). Além disso, eles podem ser considerados como experiências observáveis que são frequentemente documentados através de fotos ou vídeos (Troncy, Fialho, Hardman, & Saathoff, 2010).

Trabalhar com eventos implica lidar com variáveis como data, localização e tempo (Costa-Dasilva, Gomez-Rodriguez, Moreno, & Valcarcel, 2015). Com o aparecimento das redes sociais, a componente social de um evento começou a ganhar algum relevo, onde as listas de utilizadores interessados no evento ou as listas de utilizadores que irão participar no evento tornam-se essenciais para este. Na literatura, é geralmente mencionado que os utilizadores gostam de partilhar as suas histórias, opiniões, fotos e vídeos nas redes sociais, criando assim uma interação direta e social entre os participantes de um determinado evento (Baruah, 2012).

As redes sociais são amplamente utilizadas hoje em dia porque estas têm um papel preponderante que ultrapassa os meios convencionais na divulgação e promoção de conteúdo (Conley, s.d.). Os eventos não são exceção e com a popularidade das redes sociais, cada vez mais se cria e divulga eventos nestes sistemas. O Facebook (Facebook, Facebook, s.d.) é atualmente a maior rede social com 1.79 mil milhões de utilizadores ativos mensalmente (Barrigas, Barrigas, Barata, Bernardino, & Furtado, 2015). Esta rede social contém vários serviços, dentro dos quais o Facebook Events (Facebook, Facebook Events, s.d.), um serviço que permite criar todo o tipo de eventos e promovê-los dentro do nosso círculo de amigos e restantes utilizadores da plataforma.

Quando os utilizadores procuram por um determinado evento, geralmente desejam filtrar a sua pesquisa por categoria, hora ou local e a sua pesquisa não deve retornar mais do que cinco a dez resultados personalizados (Malkin, 2015). No entanto, a personalização é fácil quando um utilizador tem muitos amigos e gosta de muitas páginas no Facebook, mas é difícil quando estas variáveis são o reverso, por exemplo, quando um utilizador não segue determinadas páginas de locais que habitualmente promovem eventos. Além disso, os eventos têm uma vida útil curta e o seu valor diminui quanto mais longe estes estiverem das localizações frequentes do utilizador. Estes cenários dificultam a manutenção e o dimensionamento de uma experiência consistente para

os utilizadores que não tenham um contacto frequente com as redes sociais (Malkin, 2015). Além disso, os eventos acabam por ser aglomerados com outros tipos de informação e os utilizadores raramente os veem no seu *News Feed* (Nguyen & Le, 2016).

Uma aplicação focada apenas na disseminação e recomendação de eventos poderia resolver este problema efetivamente (Nguyen & Le, 2016). É difícil procurar eventos do nosso interesse (Girolami, Chessa, & Caruso, 2015) e quando nos mudamos para uma nova cidade ou país, esta necessidade é particularmente importante porque é difícil saber o que está a acontecer à nossa volta. Além disso, é difícil obter informações fidedignas sobre um determinado evento ou obter recomendações de eventos com base nos nossos interesses. Encontrar conteúdo digital relacionado com eventos é desafiante, exigindo a procura em diferentes serviços (Girolami, Chessa, & Caruso, 2015) e geralmente, a maioria dos eventos contêm informação ambígua e/ou incompleta.

Apesar de existir uma centralização cada vez maior de eventos nas redes sociais, novas plataformas têm surgido no sentido de colmatar alguns dos problemas acima mencionados. Estas plataformas aproveitam-se dos eventos criados nas redes sociais, obtendo-os através de *Application Programming Interfaces (APIs)* públicas e apresentam-nos de acordo com as suas regras de negócio. Focam-se sobretudo em nichos de mercado específicos e apresentam funcionalidades mais simples para melhorar a experiência do utilizador na procura e recomendação de eventos.

No entanto, a experiência do utilizador relativamente às plataformas de eventos nem sempre é agradável porque toda a informação recebida, geralmente é de pouco interesse e muitas vezes acaba por ser considerada *spam* dado o seu grau de fiabilidade (Troncy, Fialho, Hardman, & Saathoff, 2010). Para além da desagradável experiência do utilizador, a notificação sobre eventos, e.g. para uma determinada localização, é também algo essencial neste tipo de plataformas. Esta funcionalidade nem sempre está presente e é bastante importante para prevenir o caso em que sabemos da existência de um evento quando este já ocorreu. Além disso, as funcionalidades mais comuns como procura e recomendação de eventos nem sempre funcionam como o esperado.

Este trabalho propõe um novo conceito para uma plataforma de divulgação de eventos, intitulada de *What's Happening?,* com o objetivo de resolver os problemas acima mencionados e melhorar a experiência do utilizador neste tipo de plataformas. Em particular, este trabalho apresenta as características e funcionalidades da plataforma bem como uma proposta de arquitetura que utiliza técnicas de *machine learning* para classificar eventos provenientes das redes sociais mais populares, e.g. Facebook, com o objetivo de categorizar os eventos.

Tendo em conta que os serviços integrados, na plataforma, para a obtenção de eventos apresentam modelos de dados diferentes, é proposta a ontologia LODSE (*Linking Open Descriptions of Social Events*) tendo como base a ontologia LODE (*Linking Open Descriptions of Events*) (Shaw, Troncy, & Hardman, 2009). Esta nova ontologia tem como objetivo facilitar a integração dos dados obtidos

das *APIs* públicas de serviços externos, modelar os dados de um evento social para ser posteriormente classificado e melhorar a classificação de eventos.

Como forma de validação das técnicas de *machine learning* na arquitetura proposta, realizaram-se duas avaliações experimentais de classificação de eventos. A primeira avaliação experimental teve como objetivo comparar diferentes tipos de algoritmos de três métodos de aprendizagem automática (*Decision Trees*, *Lazy Classifiers* e *Function Classifiers)* e perceber qual o algoritmo que tem uma melhor percentagem de eventos corretamente classificados. A segunda avaliação experimental teve como objetivo comparar os modelos de dados baseados nas ontologias, acima descritas, no processo de classificação. Foi analisada a percentagem de eventos corretamente classificados, memória consumida e o tempo de processamento a fim de verificar se o modelo de dados baseado na ontologia LODSE traz vantagens no processo de classificação de eventos.

1.3 Principais Contribuições

Em resumo, as principais contribuições deste trabalho são:

- Apresentação de um novo conceito de uma plataforma de divulgação de eventos que usa a tecnologia de *machine learning* na sua arquitetura;
- Modelação dos dados de eventos sociais com a ontologia LODE;
- Realização de uma avaliação experimental de classificação de eventos, com três diferentes tipos de algoritmos (*Decision Trees*, *Lazy Classifiers* e *Function Classifiers*) e cerca de 101 121 eventos obtendo 83.33% de eventos corretamente classificados;
- Proposta da ontologia LODSE para modelar os dados de eventos sociais;
- Proposta de uma nova abordagem no âmbito da classificação de eventos;
- Realização de uma avaliação experimental onde a ontologia LODSE incrementa a percentagem de eventos corretamente classificados e demora menos tempo a classificar um conjunto de eventos comparativamente com a ontologia LODE.

Como resultado do trabalho efetuado, foi publicado um artigo científico com o título "An Event Search Platform Using Machine Learning", na "Twenty-Ninth International Conference on Software Engineering and Knowledge Engineering", SEKE 2017, Pittsburgh, USA. O artigo encontra-se no Anexo A. Foi também escrito um artigo científico com o título "LODSE: A new ontology for social event classification" que se encontra em fase de revisão para ser posteriormente submetido. O artigo encontra-se no Anexo B.

1.4 Estrutura

O resto deste documento está organizado da seguinte forma.

No capítulo 0 é apresentado o estado da arte para este trabalho, dividido em duas partes. A primeira parte é dedicada às plataformas de eventos onde vários trabalhos realizados são apresentados com o intuito de resolver os principais problemas deste tipo de plataformas. A segunda parte aborda o tema da classificação de eventos descrevendo vários trabalhos que apresentam diferentes abordagens na tarefa de classificação.

No capítulo 3 é descrito o conceito da plataforma e as suas funcionalidades. Além disso, é explicada a razão da integração da tecnologia de *machine learning* na arquitetura da plataforma, apresentando o desenho geral da arquitetura e descrito todos os seus componentes. Por fim, são apresentados vários protótipos funcionais para demonstrar as funcionalidades nas aplicações cliente.

No capítulo 4 é apresentada a ontologia LODE, as suas classes e propriedades.

No capítulo 5 é apresentada a primeira avaliação experimental. É descrita a estrutura de dados baseada na ontologia LODE, apresentados os vários testes de classificação, os seus resultados e uma análise sobre estes.

No capítulo 6 é descrita a ontologia LODSE para representar o modelo de domínio dos eventos sociais. O capítulo descreve as classes da ontologia, as relações entre ambas e as suas propriedades.

No capítulo 0 é apresentada a segunda avaliação experimental. São descritas as estruturas de dados utilizadas com base nas ontologias LODE e LODSE, apresentados os vários testes de classificação, os seus resultados e uma análise sobre estes.

Por último, o capítulo 8 apresenta as principais conclusões do trabalho realizado bem como propostas para trabalho futuro.

2 Estado da Arte

Este capítulo apresenta e analisa alguns conteúdos e trabalhos de investigação nas áreas de relevo deste trabalho, nomeadamente plataformas de eventos e classificação de eventos.

2.1 Plataformas de Eventos

Segundo João Miguel Branquinho, Professor de Filosofia na Universidade de Letras de Lisboa, o conceito de um acontecimento – ou, num registo talvez mais formal, um evento – é algo que ocorre, toma lugar ou sucede numa determinada região do espaço ao longo de um determinado período de tempo (Branquinho, 2005). Deste modo, podemos considerar vários tipos de eventos como por exemplo, um concerto no Convento São Francisco da Orquestra Filarmónica de Berlim, a partida de xadrez entre Garry Kasparov e o computador Deep Blue ou até mesmo uma operação policial. Os eventos tanto podem ser instantâneos ou de curta duração como também podem ser de longa duração.

A Eventbrite (Eventbrite, s.d.) publicou uma pesquisa revelando que a participação dos gastos do consumidor face a eventos ao vivo em relação ao ano de 1987 aumentou 70% (Malkin, 2015). Os eventos são uma excelente forma de criar conteúdo, seja ele vídeos, fotos ou estados para serem partilhados nas redes sociais. Tendo em conta que estas se alimentam da nossa vaidade, hoje em dia acabamos por participar em mais eventos, mas a maioria dos utilizadores ainda confia no passa a palavra para encontrar novos eventos. Ainda nenhum serviço resolveu eficazmente o problema da descoberta de novos eventos (Malkin, 2015).

Acredita-se que a maioria das pessoas já teve uma experiência em que perdeu um evento apenas por descobri-lo quando este já ocorreu. Este sentimento recebeu o nome de FOMO, *Fear Of Missing Out* (Malkin, 2015). O FOMO levou muitos empreendedores a arranjar soluções para resolver este problema. Tudo o que será necessário fazer é criar um bom serviço, ter uma quantidade significativa de utilizadores e monetizar através de publicidade ou venda de bilhetes.

Com o avanço da tecnologia, novas plataformas foram surgindo, com diferentes modelos de negócio, para melhorar a experiência do utilizador. Estas podem ser categorizadas em diferentes tipos, tais como:

- Plataformas que divulgam uma lista curada de eventos (listas que contêm um conteúdo pré-selecionado por parte dos seus criadores), como por exemplo, pequenos blogs que monetizam o seu conteúdo através de publicidade ou *websites* de divulgação de eventos específicos com inserção manual de eventos, e.g. Agenda UC (UC, s.d.);

- Canais próprios de divulgação por parte de entidades organizadoras, tais como o *website* oficial da entidade, e.g. Everything is new (New, s.d.).
- Redes sociais que contêm serviços de criação e promoção de eventos, e.g. Facebook Events (Facebook, Facebook Events, s.d.), sendo uma das escolhas principais por partes dos organizadores de eventos;
- Plataformas que organizam o seu conteúdo proveniente da informação pública de outras plataformas, e.g. redes sociais, modelando a plataforma de acordo com as suas regras de negócio, como por exemplo a Viral Agenda (Agenda, s.d.);

As plataformas que organizam o seu conteúdo proveniente da informação pública das redes sociais vão de encontro ao conceito que se pretende na plataforma *What's Happening?*. As redes sociais abriram um novo espaço para troca de informações e de expressão da opinião pública (Dong, Liang, & Xu, 2017). Elas não só trouxeram significativas mudanças no paradigma da opinião pública, mas também se tornaram uma força motriz para promover a mudança de conceito na área social.

As redes sociais são dos serviços mais utilizados juntamente com os motores de pesquisa. Dados gerados a partir destes sistemas têm um grande valor porque refletem vários aspetos da sociedade de hoje em dia (Panagiotou, Katakis, & Gunopulos, 2016). Além disso, os dados são facilmente acedidos através de *Application Programming Interfaces* (APIs) públicas permitindo a criação de sistemas mais personalizados e direcionados para uma área específica.

De acordo com (Kaplan & Haenlein, 2010), as redes sociais são aplicações executadas na Internet permitindo aos seus utilizadores criar e trocar conteúdos. O seu conceito é aplicado cada vez mais em aplicações distintas que permitem aos utilizadores ligarem-se entre si e partilhar todo o tipo de informação (Kaplan & Haenlein, 2010). Atualmente, o Facebook e o Twitter são as redes sociais mais usadas e populares face a outros tipos de redes sociais (Nurwidyantoro & Winarko, 2013).

Este tipo de plataformas funciona como um centro principal de divulgação de informação para organizações e indivíduos (Chow, Bao, & Mokbel, 2010), (Kwak, Lee, Park, & Moon, 2010). Geralmente, a arquitetura deste tipo de sistemas cria um canal que permite à rede social ser conhecida por um número potencialmente grande de utilizadores de uma forma passiva (Kimura, Saito, Nakano, & Motoda, 2010). No entanto, este tipo de arquiteturas requer que o emissor de conteúdo e os seus utilizadores estejam ligados entre si através de uma relação ou que as informações do emissor sejam colocadas nos canais utilizados pelos utilizadores finais.

A experiência do utilizador relativamente às plataformas de eventos nem sempre é agradável porque toda a informação recebida, geralmente é de pouco interesse e muitas vezes acaba por ser irrelevante para os interesses do utilizador (Troncy, Fialho, Hardman, & Saathoff, 2010). Existem mais problemas que podem ser associados a este tipo de plataformas, mas alguns trabalhos de

investigação têm apresentado novas propostas para uma organização mais eficiente do seu conteúdo e novos mecanismos para melhorar a procura e recomendação de eventos.

Com base em *Ambient Intelligence*, MAPAS é a plataforma criada em (Costa-Dasilva, Gomez-Rodriguez, Moreno, & Valcarcel, 2015) que usa os conceitos e técnicas desta tecnologia com o objetivo de personalizar a disseminação de eventos, ou seja, o sistema fornece ao utilizador um conjunto de eventos com base nos seus interesses. Esta tem como intuito capturar, guardar e avaliar a experiência do utilizador através da descrição do seu perfil bem como analisar o seu contexto, como por exemplo a localização. Estas técnicas são baseadas na *Agent-Oriented Technology* para criar um sistema composto por vários agentes, com o intuito de ser o mais personalizado para os seus utilizadores. Esta plataforma usa também técnicas de realidade aumentada para fornecer dados espaciais sobre os eventos. A única desvantagem desta plataforma é o elevado grau de personalização necessária por parte do utilizador. Para que esta funcione corretamente, é necessário um nível de detalhe preciso para que o sistema sugira eventos de acordo com os interesses dos utilizadores.

A solução apresentada em (Troncy, Fialho, Hardman, & Saathoff, 2010) é baseada em dados gerados pelo utilizador com uma forte componente em multimédia (fotos e vídeos). Tem como objetivo explorar a ligação intrínseca entre este tipo de dados e os utilizadores. No processo de implementação e para resolver a falta de informação de dados de eventos, usaram-se tecnologias de dados vinculadas para criar um modelo de dados unificado com o intuito de extrair a informação mais heterogênea possível. Para representar essa informação, a ontologia LODE (Shaw, Troncy, & Hardman, 2009) foi usada para detalhar e interligar todos os processos feitos e usaram a taxonomia *Simple Knowledge Organization System* para as categorias. A vantagem desta solução é a boa estrutura dos dados, onde facilmente se pode associar dados multimédia a um determinado evento. No entanto, uma vez que a plataforma é baseada em dados gerados pelo utilizador, o problema de dados inconsistentes é elevado.

O uso de sistemas de recomendação são outra das soluções estudadas em (Rokach & Shapira, 2008) que propõe eventos relevantes para o utilizador onde o uso de mais variáveis para classificar o evento pode melhorar a recomendação. Normalmente, estes tipos de sistema são bons na filtragem de informações indesejadas, mas exigem uma subscrição dos utilizadores nas categorias de eventos do seu interesse (Costa-Dasilva, Gomez-Rodriguez, Moreno, & Valcarcel, 2015).

Plataformas que usam unicamente os dados das redes sociais, são um tipo de sistema líder para a disseminação de eventos conforme apresentado em (Chow, Bao, & Mokbel, 2010) e (Kwak, Lee, Park, & Moon, 2010). No entanto, este tipo de sistemas também apresenta desvantagens. Em primeiro lugar, estas plataformas precisam de uma relação entre a rede social e o utilizador. Em segundo lugar, hoje em dia, a experiência do utilizador diz que a informação obtida destes canais, geralmente é de pouco interesse e muitas vezes considerada como *spam* por causa da sua baixa confiabilidade.

Em detalhe, o trabalho descrito em (Chow, Bao, & Mokbel, 2010) aborda as redes sociais. Aplicações baseadas no conteúdo das redes sociais têm ganho algum relevo no que diz respeito a *web services*, como por exemplo, o Facebook e o Twitter. Com base nestas duas plataformas mais conhecidas, novas plataformas têm tirado proveito da componente social e dos seus dados para implementarem o seu próprio conceito. Com a evolução dos sistemas móveis, principalmente na área da geolocalização, *wireless communication technologies* e serviços de mapas, as aplicações nestas áreas também tem vindo a evoluir permitindo a criação de funcionalidades mais interessantes para uma melhor procura e recomendação de eventos com base nos interesses do utilizador.

Em (Chow, Bao, & Mokbel, 2010) é apresentada a plataforma GeoSocialDB, uma plataforma baseada em geolocalização. A implementação do protótipo desta plataforma teve como objetivo estudar várias operações na base de dados para otimizarem a performance de pesquisas. Uma das conclusões tiradas e de interesse para este trabalho, revela que as plataformas deste tipo podem organizar o seu conteúdo através de três funcionalidades:

- *News Feed* - cada informação publicada por parte de um utilizador é criada com uma localização. Sempre que outro utilizador fica *online* ou atualiza a lista de conteúdos, o serviço apresenta os resultados baseados na localização desse utilizador e do seu raio de procura;
- *News Ranking* - dado que o serviço de *News Feed* pode retornar bastantes resultados, o *News Ranking* propõe apenas apresentar um número de informações mais relevantes de acordo com certos domínios, como por exemplo, os interesses do utilizador ou o tempo em que a informação foi partilhada;
- *Recommendations*: um sistema de recomendações baseado nas preferências do utilizador e também através de uma análise das suas preferências sociais.

Este tipo de análise efetuada em (Chow, Bao, & Mokbel, 2010) permite perceber que os conceitos apresentados também podem ser aplicados à plataforma *What's Happening?*, podendo esta apresentar resultados de acordo com uma análise das características sociais de um utilizador, das suas preferências e sobretudo através do contexto em que o utilizador está inserido, nomeadamente a sua localização.

2.2 Social Event Classification

Uma das principais características na arquitetura da plataforma *What's Happening?* é a utilização de técnicas de *machine learning* para classificar eventos provenientes de outros serviços mais populares, e.g. Facebook, permitindo melhorar a recomendação de eventos com base nas características sociais de um utilizador e as suas preferências.

A classificação de eventos sociais é uma das áreas que tem atraído atenção nos últimos anos (Zeppelzauer & Schopfhauser, 2016) devido à quantidade de dados existentes nas redes sociais e à disponibilidade de *datasets* públicos. Alguns métodos foram propostos para a classificação de eventos sociais baseados em dados textuais.

Em (Sutanto & Nayak, 2013), os autores participaram numa tarefa de *clustering* semi-supervisionada bem como numa tarefa de classificação de eventos sociais. Para a tarefa de classificação, eles usaram alguns dos algoritmos mais populares como o k-Nearest Neighbor (kNN), Decision Trees e Random Forest com *Latent Dirichlet Allocation,* uma técnica de *feature selection* para a seleção de atributos. Para proceder aos testes de classificação, usaram o modo de teste *10-fold cross validation* escolhendo 15% dos dados como dados de treino. Tendo em conta que este trabalho teve mais foco na tarefa de *clustering*, os resultados obtidos para a tarefa de classificação apenas indicaram que é necessária uma maior atenção nas distribuições das categorias usadas para classificar um evento.

Ainda relacionado com a classificação de eventos sociais baseados em dados textuais, em (Benson, Haghighi, & Barzilay, 2011), os autores estudaram a deteção de eventos nas redes sociais, particularmente no Twitter, considerando dados textuais para identificar eventos. O Twitter é caracterizado por frases curtas com um discurso fortemente coloquial e para complicar ainda mais o processo de classificação, as mensagens individuais podem não expressar a relação completa do objetivo dos autores, como é frequentemente assumido num processo de extração de tarefas. Para resolver estes problemas, os autores formularam a sua abordagem com um modelo gráfico estruturado que analisa simultaneamente as mensagens individuais obtidas das redes sociais e induzem um valor canônico para cada propriedade do evento. Aplicaram a sua técnica para criar um calendário de uma cidade com eventos de entretenimento e o seu método mostra 85% de precisão avaliada manualmente.

Com a popularidade de redes sociais como o Instagram, a tarefa de classificar e detetar eventos sociais também é feita a partir de informação visuais, como por exemplo fotografias. Em (Gupta, Gautam, & Chandramouli, 2013), os autores propõem um método para classificar imagens de eventos sociais com base numa *framework* que estes desenvolveram. A estrutura consiste em três etapas que incluem pré-processamento, filtragem e por último, classificação e agrupamento (*clustering*) dos dados. A tarefa de classificação foi realizada com a ajuda do WordNet combinado com informações textuais de um determinado *dataset,* onde obtiveram uma pontuação de F1 de 0,4409. O método apresentado foi apenas restrito à análise de objetos multimédia em vez de informações textuais.

A classificação de eventos também pode ser usada para fins de recomendação. Em (Nguyen & Le, 2016), os autores acreditam que o problema da recomendação de eventos é um problema diferente da recomendação de livros, filmes ou outros tipos de multimédia. Em vez de colocar no topo de uma *News Feed* um maior número possível de eventos adequados aos interesses de um utilizador,

é melhor remover aqueles que não são compatíveis com os seus interesses e reordenar os restantes eventos com o objetivo de melhorar a experiência do utilizador. A partir deste conceito, os autores propõem um novo método para a recomendação dividido em duas etapas – classificação e reordenação. Para a primeira fase, usaram vários modelos probabilísticos de classificadores para prever as probabilidades positivas e negativas para cada relação utilizador-evento, eliminando todos os casos negativos antes de passar para a fase seguinte. Para a segunda fase, os autores trabalharam nas probabilidades positivas, compararam os eventos com base em algumas técnicas de reordenação e escolheram os melhores eventos de acordo com os interesses do utilizador para alimentar as listas de recomendação de eventos. Aplicaram a sua técnica em eventos públicos do Facebook, demonstrando eficácia no método proposto.

Todos estes trabalhos baseiam-se em *datasets* públicos de eventos e no seu modelo de dados. A classificação *multilabel* usando informação ontológica é umas das áreas de pesquisa emergente e combina métodos de *machine learning* com modelos de conhecimento baseados em ontologias. Apesar do trabalho realizado não ser focado em classificação *multilabel* é possível verificar em (Nowak & Lukashevich) que as ontologias podem ser usadas no processo de classificação de dados, dado que os resultados são enriquecidos com o uso adicional de informações baseadas nas ontologias.

Estes trabalhos estão focados na classificação e deteção de eventos, com finalidades diferentes. Por um lado, existem abordagens que tentam detetar eventos através dos *feeds* das redes sociais, sejam eles baseados em texto ou imagens. Por outro, existem abordagens que tentam melhorar a classificação e recomendação de eventos, onde aplicam as técnicas de classificação com outras técnicas como agrupamento de dados (*clustering*) ou reordenação.

É também possível observar que todos os trabalhos partem do conteúdo das redes sociais, para realizarem as suas experiências, dada a grande quantidade de dados que é possível obter a partir destas. As avaliações experimentais realizadas neste trabalho partem do mesmo sentido, onde são usados dados públicos para serem modelados e posteriormente classificados. No entanto, este trabalho apresenta uma abordagem diferente para a classificação de eventos sociais onde, uma ontologia é criada para modelar os dados de um evento social e a classificação de eventos é baseada apenas em identificadores únicos das propriedades de um evento diferindo da classificação dos trabalhos apresentados que classificam os eventos com base no seu conteúdo textual ou multimédia.

3 *What's Happening?* – Uma plataforma de eventos

Cada vez mais somos e estamos a ficar dependentes da Internet e dos serviços que esta disponibiliza. Novas plataformas têm surgido para colmatar algumas dificuldades do dia a dia, mas ainda existem áreas onde aplicados novos conceitos e tecnologias podem resultar numa melhor experiência na procura de informação.

A procura de eventos é uma dessas dificuldades sendo por vezes necessário uma consulta em diferentes serviços para obter as informações que desejamos. Encontrar eventos é desafiante e a maioria destes contêm informação ambígua e incompleta o que dificulta a experiência do utilizador na procura e obtenção de eventos recomendados do seu interesse.

A evolução da tecnologia tem permitido o aparecimento de novas plataformas de divulgação de eventos com a finalidade de resolver os problemas comuns deste tipo de plataformas e proporcionar ao utilizador uma experiência mais consistente. Estas plataformas recorrem aos dados de outros serviços similares que albergam milhares de eventos e apresentam-nos de acordo com as suas regras de negócio. No ano de 2014, o Live Nation promoveu 220 mil eventos, o Brown Paper Tickets promoveu 200 mil, o Eventbrite promoveu 1.7 milhões, o Meetup promoveu 6.6 milhões e o Facebook informou que estava a criar 16 milhões de eventos por mês segundo (Malkin, 2015).

Neste capítulo é apresentado o conceito da plataforma de divulgação de eventos, intitulada de *What's Happening?*. São descritas as várias funcionalidades da plataforma e como estas se distribuem pelos diferentes tipos de utilizadores que a plataforma poderá suportar. É explicada a razão da integração de *machine learning* na arquitetura de uma plataforma deste tipo e apresentada a visão geral da arquitetura e os vários componentes que a compõem. Por último, são apresentados vários protótipos da plataforma para demonstrar o conceito da plataforma e as funcionalidades listadas.

De salientar, que a plataforma apresentada é apenas conceptual. Os conceitos descritos neste capítulo podem ser reaproveitados por plataformas similares, que com uma grande base de eventos, podem melhorar a experiência dos seus utilizadores na procura e obtenção de eventos recomendados.

3.1 Conceito

As plataformas de eventos são segmentadas pelo seu tipo (funcionalidades ou conteúdo), mas na generalidade os eventos são limitados a categorias predefinidas de acordo com as regras de negócio destas.

Este trabalho tem como objetivo propor um novo conceito para uma plataforma de divulgação de eventos onde cada evento é categorizado em várias *tags*, diferenciando-se das plataformas atuais que usam categorias predefinidas. Partindo do conceito do WordPress (Automattic, s.d.), as categorias servem para organizar os conteúdos. Idealmente, um *blog* deve ter poucas categorias e estas não se devem sobrepor. A principal função das categorias é permitir a distinção dos diferentes tipos de artigos. As *tags* foram criadas no sentido de não permitir longas listas de categorias, permitindo uma melhor organização do conteúdo de um *website*. São termos específicos que podem ser associados a um determinado artigo. De seguida, é dado o exemplo para um *blog* de viagens:

- Categorias: Viagens
- Tags: Coimbra, Praça da República, Avenida Sá da Bandeira

Esta organização permite vantagens a nível de organização do conteúdo bem como na pesquisa de artigos de um blog. Vários artigos podem estar associados a diferentes categorias, podendo utilizar *tags* idênticas. Facilmente procura-se por artigos referentes a um determinado monumento ou cidade sem ter que mencionar a categoria específica do artigo.

O 8tracks (8tracks, s.d.) é outro exemplo onde o conceito de *tags* é aplicado. Esta plataforma é um serviço de rádio com base em *playlists* criadas pelos utilizadores onde se pode associar no mínimo três *tags* a cada *playlist*. Estas *tags* podem ser referentes ao estilo musical, artistas presentes na *playlist* ou até mesmo estados de espírito que um utilizador procure naquele momento. A pesquisa por *playlists* é feita através de *tags*. Procurar por uma *playlist* que tenha como artistas Chopin e seja calma é facilmente encontrada devido a este tipo de organização.

O conceito da plataforma *What's Happening?* parte desta organização através de *tags,* para melhorar o processo de procura e recomendação de eventos. Um exemplo simples que pode ser aplicado é:

- Categoria: Música
- Tags: Rock, Progressive Rock, Blues Rock, Heavy Metal

Este tipo de categorização do evento pode trazer inúmeras vantagens, tais como:

- Criação de listas personalizadas de acordo com as preferências do utilizador, isto é, numa fase inicial de registo do utilizador na plataforma, este poderá selecionar as *tags* do seu interesse para obter eventos recomendados;
- Criação de um mecanismo de pesquisa e filtragem de eventos mais personalizado para obter os resultados de acordo com os termos de pesquisa baseados em *tags*;
- Possibilidade de categorizar eventos de forma a que as *tags* descrevam as características mais importantes destes eventos;
- Possibilidade de evoluir a plataforma de eventos sociais para uma plataforma global de eventos devido à diversidade de eventos que podem ser criados e categorizados com *tags*;
- Filtragem de notificações apenas para *tags* do interesse do utilizador quando um evento é criado na sua localização;

De forma a demonstrar as vantagens acima citadas, seguem os seguintes eventos como exemplo:

- Concerto de Música - André Rodrigues no Centro Cultural de Belém, (CCB);
    - Tags: André Rodrigues, CCB, Música Clássica, Contemporânea;
- Concerto de Música - Manuel Antunes no Centro Cultural de Belém (CCB);
    - Tags: Manuel Antunes, CCB, Música Clássica, Barroco;
- Peça de Teatro – Violino no Telhado no Teatro Rivoli;
    - Tags: Filipe La Féria, Teatro, Rivoli, Música Clássica, Jazz;

Na procura de eventos, se o utilizador apenas procurar por eventos de música clássica, a sua pesquisa irá retornar os três eventos acima. No entanto, se o utilizador apenas tiver interesse em saber os próximos concertos de música clássica contemporânea, a pesquisa irá apenas retornar um resultado, o concerto de música de André Rodrigues.

O mesmo exemplo pode ser aplicado na recomendação de eventos. Se o utilizador apenas tiver interesse em eventos de música clássica contemporânea, apenas o concerto de André Rodrigues será o recomendado, visto que os outros dois eventos não se enquadram nas preferências do utilizador. Em suma, a procura e recomendação de eventos retorna resultados mais precisos com base nas *tags* dos eventos, existindo uma relação intrínseca entre as *tags*, o evento e o utilizador.

É de salientar que os exemplos dados foram apenas aplicados a estilos musicais, mas também podem ser aplicados ao local do evento, neste caso o CCB ou Rivoli, bem como aos artistas como André Rodrigues ou Manuel Antunes porque estes também são *tags* do evento.

Se ao invés de *tags* fossem utilizadas categorias predefinidas, onde os dois primeiros concertos acima fossem classificados na categoria de música e a peça de teatro fosse classificada na categoria das artes performativas, a procura e recomendação de eventos não retornaria os mesmos resultados. Em primeiro lugar, não haveria uma distinção entre os dois concertos acima. Se o utilizador apenas tiver interesse em música clássica contemporânea, as suas listas de recomendação ou resultados de uma pesquisa iriam sempre retornar os dois eventos. Por outro lado, se o utilizador quisesse saber todos os eventos relacionados com música clássica e tendo em vista que a música clássica poderá estar diretamente relacionada com a categoria de música, a peça de teatro nunca seria um evento recomendado ou um resultado de uma pesquisa.

De seguida, são apresentadas as várias funcionalidades da plataforma *What's Happening?* que visam implementar o conceito acima descrito.

3.2 Funcionalidades

What's Happening? pretende ser uma plataforma simples na procura, divulgação e criação de eventos sociais. Esta terá dois tipos de utilizadores, podendo estes serem utilizadores registados no sistema ou utilizadores não registados. Como funcionalidades principais, para um utilizador não registado, a plataforma permite:

- Consultar eventos a ocorrer de momento;
- Consultar eventos futuros;
- Ver detalhes de um evento;
- Registo;
- Pesquisar eventos por *tags*;
- Pesquisar eventos por localização;
- Filtrar eventos por horário (hora, dia, mês).

Para utilizadores registados, a plataforma irá também permitir:

- Consultar eventos recomendados para uma determinada localização ou *tag* específica;
- Autenticação;
- Criar eventos;
- Guardar eventos favoritos;
- Consultar eventos favoritos;
- Seguir *tags* do seu interesse;
- Receber notificações sobre um novo evento criado para uma determinada localização ou *tag* específica;
- Receber notificações de eventos recomendados para uma determinada localização ou *tag* específica;

- Definir *tags* para receber notificações sobre eventos criados ou recomendados nas mesmas;
- Definir localizações para receber notificações sobre eventos criados ou recomendados nas mesmas.

A Figura 1 apresenta o diagrama de casos de uso, descrevendo as funcionalidades propostas para uma visão geral dos requisitos da plataforma de acordo com os tipos de utilizadores definidos.

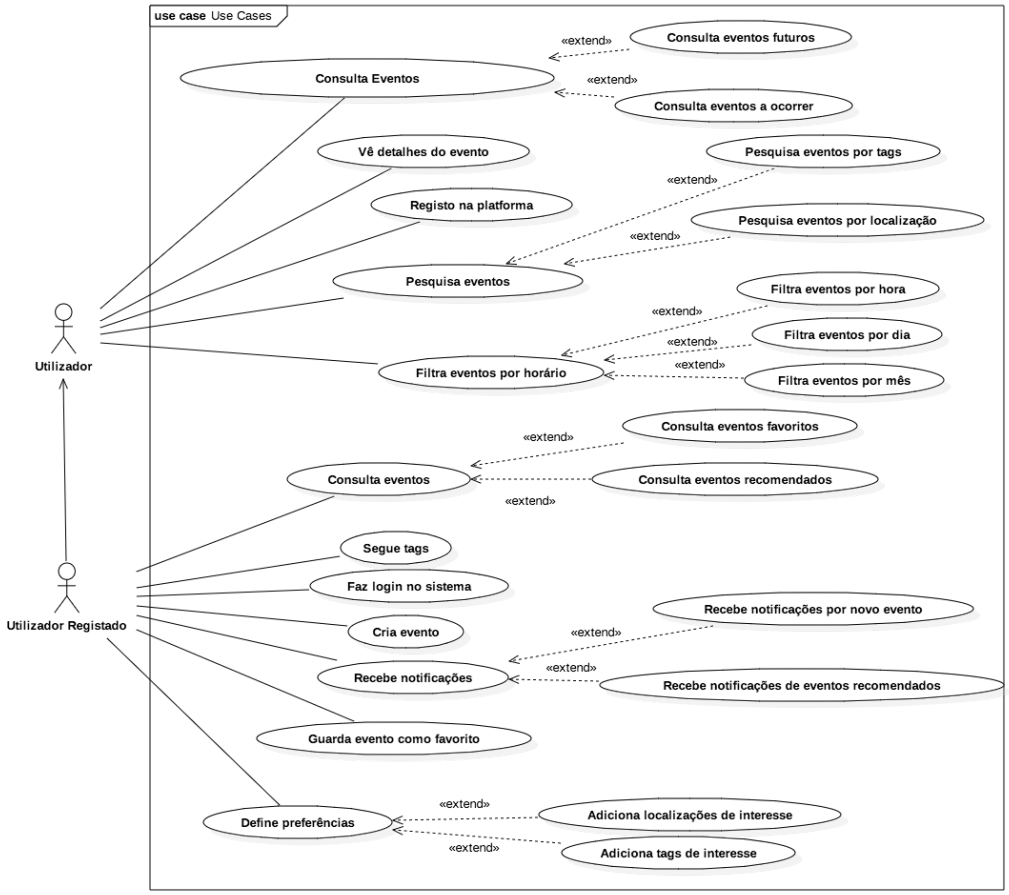


Figura 1: Casos de uso para a plataforma *What's Happening?*

No geral, estas funcionalidades são as funcionalidades básicas para um sistema deste tipo. No entanto, a pesquisa, filtragem e as notificações apresentam diferenças porque se aplicam ao conceito da plataforma. Em suma, é pretendido que com estas funcionalidades a experiência do utilizador seja melhorada e a procura, recomendação e notificação de eventos seja mais precisa e que vá de encontro aos interesses do utilizador.

Destaca-se a criação de eventos por parte dos utilizadores como uma das funcionalidades principais. As vantagens de adição desta funcionalidade estão relacionadas com uma melhor interatividade com os utilizadores, permitindo que estes criem as suas próprias *tags* no momento de criação de um evento. Esta funcionalidade irá permitir que o sistema cubra todos os tipos de

eventos com o *input* do utilizador criando uma relação ativa deste e não apenas passiva. Fazendo a comparação com o serviço 8tracks (8tracks, s.d.), no momento de criação de uma *playlist*, é também possível a criação de *tags*. Isto permite uma grande abrangência de termos de pesquisa na procura por *playlists*, algo que se procura no conceito da plataforma *What's Happening?*.

No entanto, a criação de eventos pode levantar alguns problemas porque leva à criação de conteúdo indesejado e inconsistente. Numa fase inicial, o evento submetido poderá passar por um processo de validação manual com o objetivo de avaliar a veracidade dos dados deste ou editar a sua informação, nomeadamente as *tags*, para que estas caracterizem de melhor forma o evento. No sentido de automatizar este processo, o sistema poderá sugerir ao utilizador várias *tags* a utilizar com base nas informações do evento, como por exemplo local, artista ou horário. Para isso, é necessário a inclusão de um mecanismo de aprendizagem automática que permita classificar um evento com base nos seus dados.

3.3 *Machine learning* numa plataforma de divulgação de eventos

Machine Learning é um método de análise de dados que automatiza o desenvolvimento de modelos analíticos (Mitchell & Hill, 1997). De uma forma simples, esta tecnologia é um conjunto de regras e procedimentos que permite que os computadores possam agir e tomar decisões baseadas num conjunto de dados ao invés de serem explicitamente programados para realizar uma determinada tarefa.

Numa plataforma de eventos, existe uma relação direta entre o conteúdo e os seus utilizadores porque sem conteúdo a plataforma torna-se inútil. Desta forma, a plataforma prevê a integração de vários serviços externos que permitam a obtenção de eventos através das *APIs* públicas dos mesmos.

Cada serviço tem o seu próprio modelo de dados e categoriza os seus eventos com base nas suas regras de negócio. Visto que a plataforma *What's Happening?* categoriza os eventos com base em *tags,* a obtenção e categorização de um evento obtido a partir de um serviço externo teria que ser manual com base na análise dos dados do evento. Este processo traz problemas, porque é um processo lento e pode levar a que o evento já tenha ocorrido no momento da sua categorização.

A fim de evitar a categorização manual dos eventos, é necessário a implementação de um módulo de *machine learning* na arquitetura da plataforma com o objetivo de classificar um evento nas *tags* do sistema de uma forma automatizada.

Outro objetivo de aplicação destas técnicas prende-se com a funcionalidade de criação de eventos. Esta funcionalidade pode permitir a criação de conteúdo indesejado e inconsistente, criando o problema em que a informação de um evento possa ser considerada *spam* dado o grau de pouca confiança. Além disso, um evento mal classificado pode traduzir-se em resultados errados numa

pesquisa, recomendação ou notificação de eventos. No sentido de resolver estes problemas, as técnicas de *machine learning* também podem ser aplicadas nesta funcionalidade. No momento de criação de um evento, o utilizador terá acesso a um conjunto de *tags* sugeridas pelo sistema às quais este é "obrigado" a utilizar para categorizar o seu evento. Assim, o problema de dados inconsistentes poderá ser resolvido com base no mecanismo de aprendizagem automática presente na arquitetura da plataforma. Existe sempre a possibilidade de introdução de novas *tags*, mas quando existir essa necessidade, o evento terá que passar por um processo de validação manual.

Em suma, os objetivos de criação de um mecanismo de *machine learning* na plataforma *What's Happening?* são:

- Automatizar o processo de obtenção de eventos provenientes de serviços externos para que os eventos sejam classificados nas *tags* do sistema;
- Prevenir a criação de conteúdo indesejado e inconsistente na plataforma por parte do utilizador no momento de criação de um novo evento.

3.4 Arquitetura

Este subcapítulo apresenta o desenho da arquitetura do conceito da plataforma de eventos *What's Happening?*. São descritos os vários componentes, o processo de comunicação entre ambos bem como a integração do módulo de *machine learning*. Na Figura 2 é possível observar a visão geral da arquitetura.

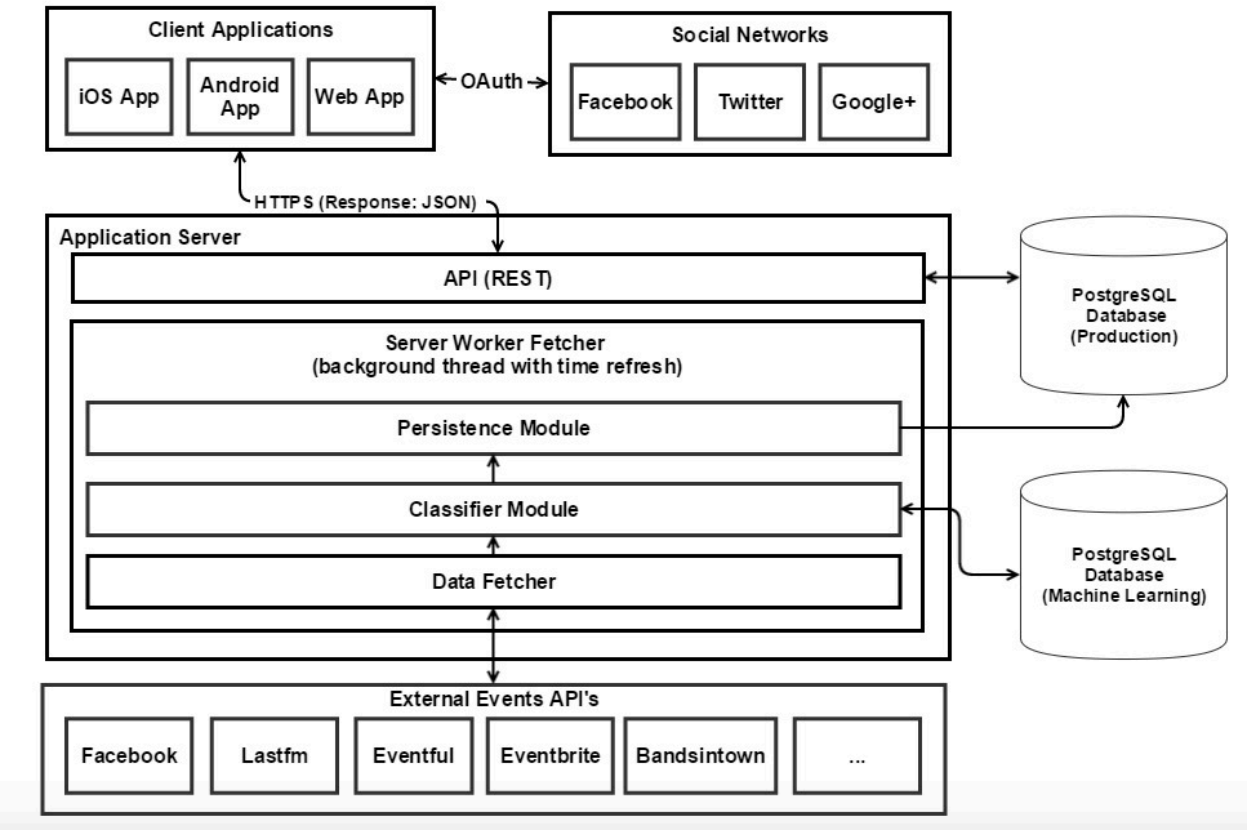


Figura 2: Protótipo da Arquitetura

No geral, a arquitetura da plataforma é baseada num sistema do tipo Cliente-Servidor. Este tipo de sistema tem como base uma estrutura de aplicação distribuída onde o servidor fornece os vários serviços para que as aplicações cliente possam apresentar a informação aos utilizadores. A comunicação entre os clientes e servidor é baseada no protocolo TCP/IP, um dos protocolos mais usados neste tipo de sistemas. É também pretendido que a arquitetura da plataforma seja orientada a micro serviços, permitindo a criação de módulos independentes para a criação de um sistema mais flexível, escalável e com uma manutenção mais simples comparado com sistemas monolíticos (Villamizar, et al., 2015) .

De seguida é descrito cada um dos componentes presentes na arquitetura proposta, nomeadamente as *Client Applications, Social Networks,* o *Application Server,* as bases de dados bem como as *External APIs.*

3.4.1 Client Applications

As *Client Applications* são aplicações que permitem ao utilizador interagir com as funcionalidades da plataforma, descritas no subcapítulo 3.2. Estas aplicações comunicam com o servidor através da *API* desenvolvida através do protocolo *Hypertext Transfer Protocol Secure* (HTTPS). Os dados

enviados estão no formato *JavaScript Object Notation* (JSON) e são compostos por dados de eventos.

Na Figura 2, estão representadas três aplicações cliente:

- Aplicação iOS;
- Aplicação Android;
- Aplicação Web.

Com estas três aplicações, *What's Happening?* pretende ter presença na *Web* e nas duas principais plataformas de distribuição de aplicações móveis, iOS e Android.

3.4.2 Social Networks

São redes sociais que permitem ao utilizador fazer a autenticação no sistema através do protocolo de comunicação OAuth (Auth0, s.d.), que permite a autenticação num sistema terceiro através das suas contas criadas nas redes sociais e é um dos protocolos mais usados atualmente (Yang, Li, Lau, Zhang, & Hu, 2016). As escolhidas são o Facebook, Twitter e Google+ porque são as três mais usadas atualmente (Dreamgrow, s.d.).

As vantagens de permitir apenas a autenticação com base nas redes sociais são:

- Facilitar o registo do utilizador na plataforma *What's Happening?*;
- Obtenção dos dados do utilizador e suas preferências para que a recomendação de eventos, numa fase inicial, seja com base nessa informação;
- Autenticação mais segura, devido à utilização do protocolo OAuth.

3.4.3 Application Server

O *Application Server* é o módulo responsável pelo componente servidor do sistema orientado a micro serviços. Tem como funções principais o suporte às aplicações cliente através de uma *API RESTful* (Kwak, Lee, Park, & Moon, 2010) bem como suportar o módulo responsável pela classificação de eventos, o *Server Worker Fetcher*.

3.4.3.1 Server Worker Fetcher

O *Server Worker Fetcher* é um serviço que tem como função obter e classificar eventos. Este pode ser dividido em três módulos:

- *Data Fetcher* – este módulo é responsável por obter eventos das várias *APIs* externas integradas na plataforma;

- *Classifier module* – este módulo é responsável por integrar os dados provenientes das *APIs* externas para serem posteriormente classificados através de um algoritmo de *machine learning*. A base de dados PostgreSQL (The PostgreSQL Global, 1996) (Machine Learning) serve como base de treino para esta classificação;
- *Persistence module* – este componente é responsável por guardar os eventos já classificados na base de dados PostgreSQL (The PostgreSQL Global, 1996) (Production).

A obtenção de eventos é feita através da técnica de *polling*, ou seja, o módulo *Data Fetcher* faz um *request* aos serviços integrados e recebe os dados, num período de tempo definido pelo sistema. De salientar que a maioria das *APIs* públicas tem restrições no número de *requests* por dia, podendo limitar o número de eventos obtidos diariamente.

3.4.4 Databases

A arquitetura do sistema contempla duas bases de dados. A base de dados PostgreSQL (The PostgreSQL Global, 1996) (Production) é a base de dados responsável por guardar todos os eventos, já classificados para serem posteriormente consultados pelos utilizadores. A PostgreSQL (The PostgreSQL Global, 1996) (Machine Learning) é usada como uma base de treino para o algoritmo de *machine learning* que será usado para classificar os eventos. Esta base de dados será atualizada periodicamente no sentido de melhorar progressivamente a classificação.

A nível dos eventos, ambas as bases de dados têm o mesmo modelo conceptual que pode ser visto na Figura 3.

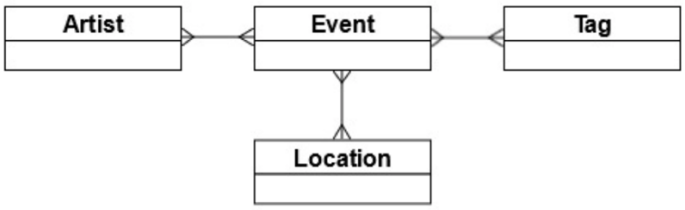


Figura 3: Modelo conceptual da base de dados

O modelo conceptual apresentado na Figura 3 consiste na seguinte estrutura de tabelas:

- **Event** – tabela responsável por armazenar todos os detalhes do evento, tais como nome, descrição e horário, entre outros;
- **Artist** – tabela responsável por armazenar todos os detalhes do artista, tais como nome, descrição e site oficial, entre outros;
- **Location** – tabela responsável por armazenar todos os detalhes da localização, tais como latitude, longitude, nome e site oficial, entre outros;

- **Tag** – tabela responsável por armazenar todos os detalhes das *tags*, nomeadamente o seu nome.

3.4.5 External APIs

As *External APIs* são *APIs* públicas onde é possível obter eventos dos seus sistemas.

3.5 Protótipo de funcionamento

Para as funcionalidades apresentadas no subcapítulo 3.2, são apresentados de seguida vários protótipos para demonstrar a utilização prática do conceito da plataforma *What's Happening?* nas aplicações clientes descritas na secção anterior.

A Figura 4 apresenta o protótipo para a página inicial da aplicação. Esta página é composta por uma caixa de pesquisa onde o utilizador pode procurar por eventos numa determinada localização específica. Se o utilizador não se encontrar autenticado este pode ver uma lista de eventos que estão a acontecer de momento e os futuros eventos que irão ocorrer para a sua localização atual.

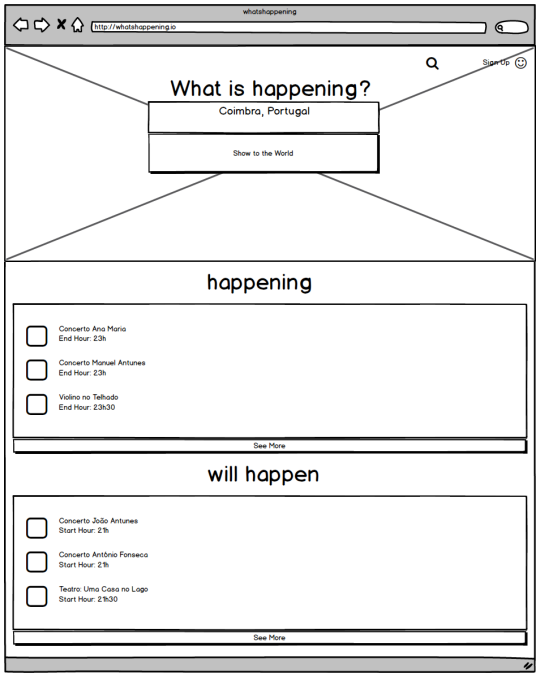


Figura 4: Protótipo da página inicial para um utilizador não autenticado

Caso o utilizador esteja autenticado, este pode consultar também uma lista de eventos recomendados, como demonstrado na Figura 5.

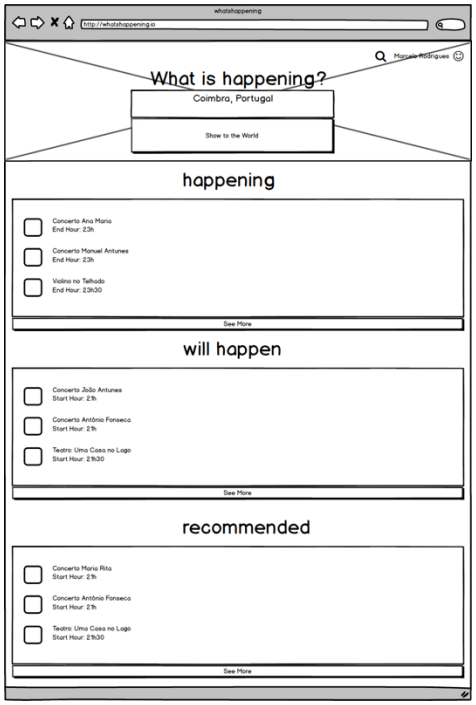


Figura 5: Protótipo da página inicial para um utilizador autenticado

O utilizador pode filtrar as suas listas por *tag* ou horário. Para isso, no topo da página inicial, existe um botão de pesquisa que permite o utilizador colocar as suas *tags* de interesse ou um data e horário específico para que as listas atuais sejam filtradas consoante os parâmetros de pesquisa.

Ao selecionar um evento, o utilizador pode obter mais informações sobre este. A Figura 6 apresenta o protótipo para a página de um evento específico onde o utilizador pode consultar detalhadamente toda a informação sobre este. No caso de o utilizador estar autenticado, este pode também guardar o evento como favorito.

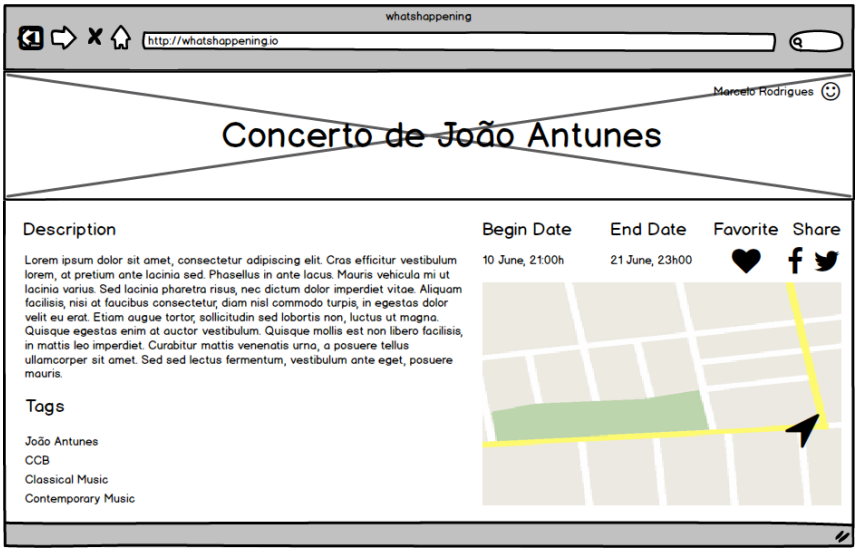


Figura 6. Protótipo da página de um evento

O utilizador também pode adicionar eventos se este estiver autenticado. A Figura 7 apresenta o protótipo para a funcionalidade de adição de eventos, no qual o utilizador pode adicionar as informações mais importantes do evento, nomeadamente a hora, localização, uma descrição e as *tags* do evento. Ainda na Figura 7, é possível observar a secção de *Suggested Tags* que são as *tags* sugeridas para categorizar o evento, baseadas nas informações deste e que são fornecidas pelo sistema de aprendizagem automática da plataforma.

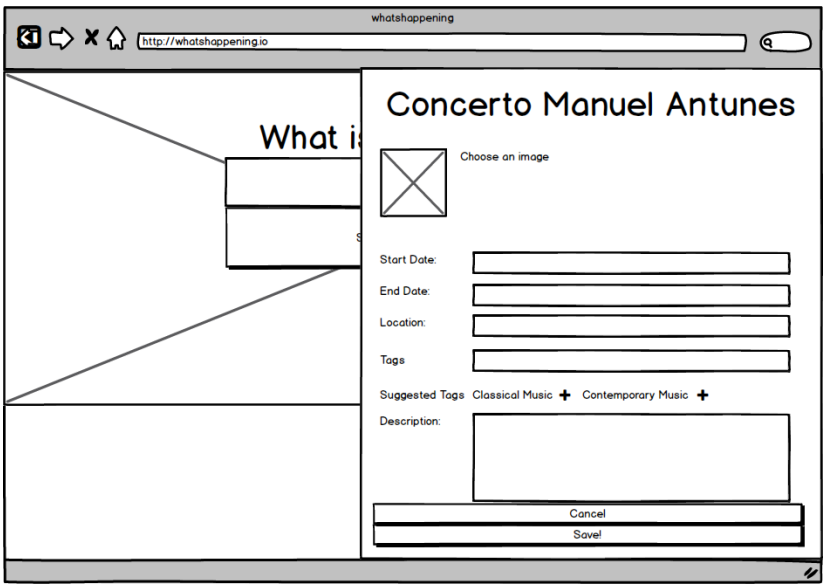


Figura 7: Protótipo da página de adição de um evento

O utilizador também pode aceder às suas preferências podendo fazer um conjunto de operações, tais como, editar a sua informação pessoal, adicionar/remover *tags* do seu interesse, adicionar/remover localizações do seu interesse, consultar os eventos favoritos e ver os eventos

que adicionou no sistema. A Figura 8 apresenta os protótipos para estas funcionalidades, estando estas disponíveis através de um menu lateral.

Figura 8. Protótipo das páginas de preferências de um utilizador

Se o utilizador não se encontrar autenticado, este pode autenticar-se e registar-se através das redes sociais. O protótipo desta funcionalidade é apresentado na Figura 9, onde existem três redes sociais, Facebook, Twitter e Google+, nas quais o utilizador poderá autenticar-se. Caso seja a primeira vez que o utilizador se autentique no sistema, este é redirecionado para a página das preferências para poder editar a sua informação pessoal, efetuando assim o registo na plataforma.

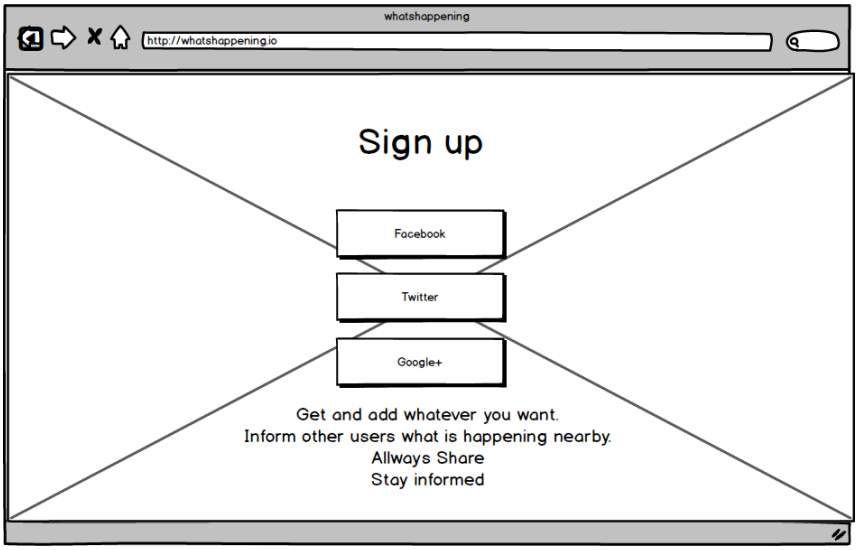


Figura 9: Protótipo da página de autenticação e registo de um utilizador

No sentido de validar a componente de aprendizagem automática na arquitetura proposta, para categorizar os eventos provenientes de serviços externos ou para sugerir *tags* no momento de criação de um evento, realizou-se um estudo e trabalho experimental que é descrito nos próximos capítulos.

4 A ontologia LODE

Como forma de validação das técnicas de *machine learning* na arquitetura proposta, realizaram-se duas avaliações experimentais de classificação de eventos. A primeira avaliação experimental teve como objetivo comparar diferentes tipos de algoritmos de três métodos de aprendizagem automática (*Decision Trees*, *Lazy Classifiers* e *Function Classifiers)* e perceber qual o algoritmo que tem uma melhor percentagem de eventos corretamente classificados. A segunda avaliação experimental realizada teve como objetivo comparar os modelos de dados baseados nas ontologias LODSE e LODE, no processo de classificação de eventos.

Este capítulo apresenta a ontologia LODE, aplicada na primeira avaliação experimental para modelar os dados dos eventos sociais para serem posteriormente classificados. As ontologias são utilizadas em diversas áreas, como inteligência artificial ou engenharia de software (Guarino, 1998), para representarem o conhecimento sobre o mundo ou parte dele.

Uma ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um determinado domínio e é usada para realizar inferência sobre os objetos desse mesmo domínio.

Em (Shaw, Troncy, & Hardman, 2009), os autores estudam vários modelos de dados de eventos e propõem um modelo diferente, que se traduz na ontologia LODE, para encapsular as propriedades mais importantes de um evento com base no seu estudo. O objetivo dos autores é permitir a modelação intemporal dos aspetos factuais dos eventos, podendo estes ser caracterizados em termos com base nos quatro *W's*:

- ***What*** *happened?*
- ***Where*** *did it happened?*
- ***When*** *did it happen?*
- ***Who*** *was involved?*

Os autores excluíram propriedades que permitam categorizar eventos ou que permitam relacionar-se com outros eventos através de relações parciais ou casuais porque estas pertencem a uma dimensão interpretativa que são descritas de melhor forma através de outros modelos apresentados em (Shaw, Troncy, & Hardman, 2009).

4.1 Classes e propriedades da ontologia LODE

Uma ontologia geralmente descreve:

- Indivíduos – os objetos básicos;

- Classes – conjuntos, coleções ou tipos de objetos;
- Atributos – propriedades, características ou parâmetros que os objetos podem ter que partilhar;
- Relacionamentos – entre os vários objetos.

A ontologia LODE é constituída por uma classe e sete propriedades, que são descritas nas secções seguintes:

4.1.1 Classe *Event*

A classe *Event* pretende representar algo que aconteceu que pode estar relacionado com uma notícia ou por algo contado por um historiador. Em (Shaw, Troncy, & Hardman, 2009) os autores explicam que um evento consiste em algumas fronteiras temporais e espaciais, subjetivamente impostas ao fluxo da realidade ou imaginação que desejam tratar como uma entidade, com o propósito de fazer afirmações sobre ela. Em particular, podem ser relacionadas pessoas, lugares ou objetos a um evento.

De salientar que a definição que os autores da ontologia dão à classe *Event,* não especifica que um evento envolva uma mudança de estado, nem tenta distinguir eventos de processos ou estados.

4.1.2 Propriedade *atPlace*

A propriedade *atPlace* refere-se a um local nomeado ou relativamente especificado onde o evento aconteceu. Relaciona um evento a algum lugar significativo, que pode ter um nome, e.g. Coimbra, ou pode ser definido em relação a alguma outra entidade ou entidades, e.g. a área não incorporada entre a Praça do Comércio e a Ponte De Santa Clara. Um evento pode estar relacionado a um ou mais lugares.

4.1.3 Propriedade *atTime*

A propriedade *atTime* representa um instante ou intervalo de tempo abstrato onde o evento aconteceu. Relaciona um evento a alguns limites temporais impostos subjetivamente, isto é, um período de tempo. De salientar que um evento pode estar apenas relacionado a um período de tempo.

4.1.4 Propriedade *circa*

A propriedade *circa* representa um intervalo de tempo que pode ser descrito com precisão usando para isso datas do calendário ou horários de um relógio. Em particular, esta propriedade expressa uma relação temporal expressando proximidade no tempo.

4.1.5 Propriedade *ilustrate*

O valor da propriedade *ilustrate* é um evento ilustrado por alguma coisa, tipicamente um objeto que pode ser representado através de documentos ou comentários.

4.1.6 Propriedade *inSpace*

A propriedade *inSpace* representa uma região abstrata do espaço onde o evento aconteceu. Em particular, a relação existente entre um evento e uma região de espaço apenas afirma que o evento ocorreu num lugar dentro dessa região e não em todos os lugares da região.

4.1.7 Propriedade *involved*

A propriedade *involved* relaciona um evento a qualquer objeto físico, mental ou social envolvido num evento. Não implica qualquer relação casual, de influência ou qualquer outro tipo de relação explicativa como criação ou destruição.

4.1.8 Propriedade *involvedAgent*

A propriedade *involvedAgent* relaciona um evento a um agente seja ele pessoa, grupo, organização ou agente computacional. Não implica qualquer relação causal, de influência ou intencionalidade.

4.2 Aplicação prática da ontologia LODE

Com base no estudo efetuado à ontologia LODE, verificou-se que esta podia ser reutilizada para descrever um evento social. Um evento social possui uma característica distinta: uma relação com as artes diferenciando-se de um evento com base na ontologia LODE. Neste caso, compreende-se por "artes" uma música, pintura, cinema, desporto podendo ainda ser englobadas peças de teatro, festivais, concertos, feiras temáticas entre outros (Duarte, 2009).

Aplicando a ontologia LODE a um evento social, as propriedades da ontologia podem descrever este tipo de eventos. Tendo como exemplo o seguinte evento de Rui Massena, de seguida é demonstrado como é que a ontologia LODE pode modelar os dados de um evento social.

- Evento
    - Concerto de Música de Rui Massena;
    - Local: Convento de São Francisco;
    - Dia: 10 de Maio;
    - Hora: 21h00;
    - Organização: Câmara Municipal de Coimbra.
- Ontologia LODE
    - Classe *Event*

        - Concerto de Música de Rui Massena;
    - Class *atPlace*
        - Convento São Francisco;
    - Classe *atTime*
        - Entre as 21h00 e as 23h00 de segunda-feira, dia 10 de Maio;
    - Classe *involvedAgent*
        - Rui Massena;
        - Câmara Municipal de Coimbra.

Com base neste estudo, os dados dos eventos gerados para a primeira avaliação experimental tiveram em conta as propriedades da ontologia LODE para posteriormente serem submetidos no processo de classificação de eventos.

5 Avaliação dos métodos de aprendizagem automática

Este capítulo apresenta a primeira avaliação experimental que tem como objetivo comparar diferentes tipos de algoritmos de três métodos de aprendizagem automática (*Decision Trees*, *Lazy Classifiers* e *Function Classifiers)* e perceber qual o algoritmo que tem uma melhor percentagem de eventos corretamente classificados. O capítulo começa por explicar como é que os dados foram estruturados através da ontologia LODE. De seguida, é explicado o *setup* experimental, o *hardware* e *software* utilizado para a realização dos testes e os algoritmos usados. Por último, são apresentados os vários testes realizados e os seus resultados.

5.1 Estruturação dos dados com a ontologia LODE

Como explicado no capítulo 4, o propósito da ontologia LODE é criar um modelo que permita encapsular as propriedades mais importantes para descrever eventos permitindo uma modelação intemporal dos aspetos factuais destes. A ontologia é composta por uma classe, a classe *Event* que contêm sete propriedades.

Após o estudo efetuado à ontologia LODE, realizado no capítulo 4, verificou-se que a ontologia podia ser aplicada ao modelo de dados de um evento social, dado que um evento social acaba por ser um subtipo de um evento retratado pela ontologia LODE Não foram utilizadas todas as propriedades da classe *Event*, visto que nem todas se enquadram num evento social. As propriedades escolhidas da ontologia LODE para modelar um evento social são:

- *atPlace* – permite associar um evento social a um local;
- *atTime* - permite associar um evento social a um espaço temporal;
- *involved* –permite associar um evento social a uma pessoa, grupo ou organização.

Dado que o foco deste trabalho são os eventos sociais, foi adicionada uma nova propriedade que permita representar o conteúdo social de um evento, algo que não é contemplado na ontologia LODE. Como mencionado em (Baruah, 2012), os utilizadores gostam de partilhar as suas histórias nas redes sociais e obter conteúdos sociais sobre os eventos. Partindo dos eventos criados no Facebook, geralmente estes necessitam de uma resposta. Esta resposta é traduzida em R.S.V.P. que significa "*Répondez S'ill Vous Plaît*" em francês. Estes dados permitem saber se o utilizador irá participar no evento e podem ser traduzidos nas seguintes variáveis:

- *Attending guests* – representa os utilizadores que irão ao evento;

- *Declined guests* – representa os utilizadores que não irão ao evento;
- *Interested guests* – representa os utilizadores que tem interesse no evento, mas ainda não sabem se poderão ir;
- *No reply guests* – utilizadores que não responderam ao convite do evento;
- *Maybe guests* – utilizadores que poderão ir ao evento.

A inclusão desta nova propriedade tem como objetivo perceber se este tipo de informação traz vantagens no processo de classificação de eventos. Organizando as várias propriedades escolhidas da ontologia LODE com a componente social, foram obtidos os vários atributos listados na Tabela 1 que irão organizar os dados dos *datasets* para serem usados no processo de classificação.

Tabela 1: Atributos organizadas pela ontologia LODE

Propriedades da Ontologia LODE	*Atributos*
atPlace	venue_latitude
	venue_longitude
atTime	event_start_hour
	event_end_hour
	event_start_day_of_month
	event_end_day_of_month
inSpace	venue_id
Involved	artist_id
Social	event_attending_count
	event_declined_count
	event_interested_count
	event_noreply_count
	event_maybe_count

Com o objetivo de perceber se estes atributos são viáveis para o processo de classificação, procedeu-se a um estudo de várias *APIs* para analisar a informação pública que se pode obter a partir destas. Partindo dos atributos definidos na Tabela 1 é possível observar na Tabela 2 que as *APIs* externas retornam dados para todos os atributos definidos. Relativamente aos atributos sociais, estes são apenas obtidos na *API* do Facebook.

Tabela 2: Presença das características por API

Características	*APIs*			
	Facebook	Eventful	EventBrite	Meetup
venue_latitude	X	X	X	X
venue_longitude	X	X	X	X
event_start_hour	X	X	X	X
event_end_hour	X	X	X	X
event_start_day_of_month	X	X	X	X

event_end_day_of_month	X	X	X	X
venue_id	X	X	X	X
artist_id	X	X	X	X
event_attending_count	X			
event_declined_count	X			
event_interested_count	X			
event_noreply_count	X			
event_maybe_count	X			

5.2 Setup Experimental

Este subcapítulo apresenta o *setup* experimental criado para realizar os testes de classificação. É apresentado o *hardware* e *software* utilizado para a realização dos testes e os vários *datasets* com uma breve explicação de como estes foram criados. Por último, são apresentados os algoritmos usados para obter a melhor percentagem de eventos corretamente classificados e o modo de teste utilizado.

5.2.1 Hardware

Para a realização da primeira avaliação experimental, foi utilizada uma máquina com as seguintes características:

- Máquina 1 – Processador 1.4GHz Intel Core i5, 8GB de RAM DD3

5.2.2 Weka

O Waikato Environment for Knowledge Analysis (Weka) (The University of Waikato, s.d.) é um *workbench* de *machine learning* desenvolvido na linguagem Java, atualmente na versão 8 e consiste numa coleção de algoritmos de *machine learning* para tarefas de classificação de dados.

A aplicação *standalone* foi utilizada no processo de classificação de eventos e também na gestão dos dados de eventos, nomeadamente para gerir dados em falta nos *datasets* ou remover atributos não relevantes para a classificação.

O Weka foi o *software* escolhido porque é *open source* e é um dos *softwares* mais usados em *machine learning* (Markov & Russell, 2006).

5.2.3 Datasets

Para realizar a avaliação experimental foi necessário a criação de dois *datasets* de eventos. Na Tabela 3 é possível observar o número de eventos criados para cada *dataset*.

Tabela 3: Datasets do setup experimental

	Facebook Dataset	*Random Dataset*
Número de eventos	1121	100 000

O *Facebook Dataset* é um *dataset* com 1121 eventos, disponível em (https://tinyurl.com/ybhdutxv), selecionados aleatoriamente do Facebook. Após a obtenção dos eventos, estes foram submetidos a um processo de classificação manual para uma determinada *tag* para corresponderem aos resultados esperados já definidos.

O *Random Dataset* é um *dataset* com cerca de 100 000 eventos, disponível em (https://tinyurl.com/y9azre3z) gerados aleatoriamente através de um algoritmo desenvolvido no decorrer deste trabalho.

O objetivo desta avaliação experimental é classificar um evento para um determinado conjunto de *tags*. Ambos os *datasets* tem as mesmas *tags* nas quais o evento pode ser classificado, que são:

- Music
- Art
- Gastronomy
- Sport
- Social
- Conference

5.2.4 Algoritmos

Foram usados três algoritmos de diferentes métodos de aprendizagem automática para obter o melhor resultado de classificação possível. Os algoritmos são:

- **Decision Trees**
  - Random Forest
- **Lazy Classifiers**
  - k-Nearest Neighbor (IBk) ou (KNN)
- **Function Classifiers**
  - Sequential Minimal Optimization (SMO)

O algoritmo *Random Forest* (Breiman, 2001) consiste num conjunto de *n* árvores de decisão construídas, considerando *k* atributos aleatoriamente. Por defeito, $n = 100$ e $k = [\log_2 (\text{número de atributos}) + 1]$. O resultado é obtido por uma votação de todas as árvores que constituem a floresta, ganhando a classe mais votada. É um dos algoritmos de aprendizagem que tem sido amplamente utilizado nas ciências aplicadas e em problemas de classificação de dados (Oshiro, Perez, & Baranauskas, 2012). Tende a apresentar um melhor desempenho do que algoritmos mais simples,

lida bem com grandes conjuntos de dados e com um grande número de atributos e tem um bom nível de acurácia. No entanto, é um algoritmo que necessita de uma elevada capacidade de processamento (Oshiro, Perez, & Baranauskas, 2012).

O algoritmo *k-Nearest Neighbor* (IBk) ou KNN é um algoritmo onde a sua aprendizagem é feita por analogia. Um conjunto de dados de treino é formado por vetores *n-dimensionais* e cada elemento do conjunto representa um ponto no espaço *n-dimensional*. Para determinar a classe de um elemento, o algoritmo procura *k* elementos do conjunto de treino que estejam mais próximos do elemento desconhecido, ou seja, que tenham a menor distância. Os elementos *k* são os *nearest neighbors,* nos quais o algoritmo vai verificar quais as classes desses elementos e a classe mais frequente é o resultado obtido. É um algoritmo simples e flexível, mas que pode ser computacionalmente exaustivo quando lida com grandes conjuntos de dados (Harrington, 2012).

O algoritmo *Sequential Minimal Optimization* (SMO) é um algoritmo baseado em *Support Vector Machines* (SVM). Surge no sentido de resolver o problema da programação quadrática (PQ), problema matemático de otimização com o objetivo de otimizar a função quadrática de várias variáveis ao mesmo tempo (Platt, 1998). Para isso, não usa matrizes de grandes dimensões em memória e decompõe o problema PQ em pequenos problemas a partir do teorema de *Osuna* para assumir a sua convergência. Para resolver o problema PQ nas SVM é necessário a utilização de multiplicadores de *Lagrange* para se conseguir otimizar o problema de forma mais reduzida. A cada iteração, o SMO escolhe dois multiplicadores para otimizar em conjunto e encontra os melhores valores para os multiplicadores (Platt, 1998). Este algoritmo tem a vantagem de ser um dos algoritmos mais rápidos e de também lidar com grandes conjuntos de dados (Platt, 1998).

5.2.5 Modo de Teste

Esta avaliação experimental tem como objetivo avaliar a percentagem de eventos corretamente classificados para os algoritmos acima descritos. Foi usado o modo de teste *10-fold cross validation* em que 90% dos dados são utilizados para treino e 10% dos dados são usados para classificação em cada iteração até 10. O resultado final é a média calculada de todos os resultados para cada iteração produzindo assim uma percentagem final de elementos corretamente classificados.

A vantagem deste método é que todos os dados são usados para treino e teste e cada elemento é usado para teste uma única vez.

5.3 Resultados de Classificação e Discussão

Neste subcapítulo são apresentados os três testes realizados com o objetivo de obter a melhor percentagem de eventos corretamente classificados. Para cada teste, são apresentados os objetivos, resultados e a análise destes.

5.3.1 1º Teste

O primeiro teste teve como objetivo obter a percentagem de eventos corretamente classificados para ambos os *datasets* com o intuito de analisar os resultados para os algoritmos usados. Na Tabela 4 é possível observar a diferença de percentagens obtidas entre os vários algoritmos para cada *dataset* bem como a diferença percentual existente entre os dois *datasets*.

Tabela 4: Resultados do 1º teste experimental

Algoritmos	*% de eventos corretamente classificados*	
	Facebook Dataset	**Random Dataset**
IBk	50,02%	100%
SMO	46,67%	100%
Random Forest	70,28%	100%

Apesar da diferença percentual existente entre os vários algoritmos para o *dataset* do Facebook, torna-se mais importante perceber a diferença percentual existente entre ambos os *datasets.*

Para os algoritmos IBk e SMO, a diferença existente entre *datasets* é de cerca de 50% e para o algoritmo Random Forest é de 30%. Após uma análise aos dados de ambos os *datasets,* esta diferença está relacionada com possíveis valores em falta no *dataset* do Facebook. Uma vez que estes *datasets* são constituídos unicamente por valores numéricos, as *APIs* nem sempre retornam dados para todos os atributos, sendo representados no *dataset* do Facebook com o valor zero. Para confirmar que a existência de valores em falta afeta a classificação de eventos, foi necessário a realização de um segundo teste, que permitisse lidar com estes valores com o objetivo de melhorar os resultados de classificação.

5.3.2 2º Teste

Existem várias técnicas para lidar com os valores em falta que foram consideradas neste segundo teste. Analisando o *dataset* do Facebook ao detalhe, é possível observar que alguns eventos contêm para a *venue_latitude*, *venue_longitude*, *artist_id* e *venue_id* o valor zero.

Existem três técnicas, segundo (Brownlee, 2017), que podem ser utilizadas para gerir os valores em falta, descritas de seguida:

- *Remove missing values* (RMV) – esta técnica permite remover todos os eventos que contêm valores em falta, produzindo um novo *dataset* que contêm apenas eventos com todos os dados;
- *Mark missing values* (MMV) – esta técnica permite marcar todos os dados que se encontram em falta através do caractere "?". É produzido um novo *dataset* onde todos os valores em falta encontram-se assinalados com o caractere acima;

- *Impute missing values* (IMV) – os eventos com dados em falta não necessitam de ser removidos. Esta técnica permite substituir estes dados com uma média de distribuição genérica calculada com base nos dados do *dataset* a ser usado para classificação.

O *Random dataset* não contêm nenhum valor em falta o que explica os excelentes resultados obtidos no 1º teste. A fim de aproximar este *dataset* com o *dataset* do Facebook, procedeu-se à alteração do algoritmo que permitisse a geração de dados aleatórios com a finalidade de criar alguns eventos com valores em falta.

Para este segundo teste foram aplicadas as várias técnicas acima descritas e os resultados encontram-se na Tabela 5.

Tabela 5: Resultados do 2º teste experimental

Algoritmos	*Técnica*	*% de eventos corretamente classificados*	
		Facebook Dataset	**Random Dataset**
IBk	RMV	100%	100%
	MMV	41,60%	62,09%
	IMV	48,12%	68,88%
SMO	RMV	100%	100%
	MMV	43,86%	77,96%
	IMV	43,89%	76,33%
Random Forest	RMV	100%	100%
	MMV	61,54%	70,45%
	IMV	68,89%	79,44%

Numa análise geral, podemos comprovar que utilizando a técnica RMV os resultados melhoram substancialmente obtendo 100% de eventos corretamente classificados em todos os algoritmos. No entanto, num ambiente real, esta técnica não pode ser utilizada visto que os serviços integrados na arquitetura poderão retornar dados em falta.

Para as técnicas MMV e IMV, comparando os resultados da Tabela 5 com a Tabela 4, é possível verificar que estes pioraram. Para a técnica MMV, dado que os valores em falta são marcados com o caractere "?", os algoritmos acabam por não usar estes valores, havendo menos informação para detetar a *tag* do evento que por consequência leva a que exista mais margem para erros. Para a técnica IMV, apesar de ter sido atribuído um valor da média de distribuição genérica, este valor nem sempre corresponde à realidade levando também a maiores margens de erros por parte do algoritmo no processo de classificação. No entanto, com a técnica IMV é possível obter melhores resultados quando comparado com a técnica MMV.

Podemos também concluir que a adição de valores em falta faz com que a performance dos algoritmos e os seus resultados piorem. Torna-se claro que ambas as técnicas MMV e IMV não podem ser levadas em consideração no processo de classificação.

5.3.3 3º Teste

Outra abordagem para lidar com os valores em falta passa por remover os atributos que contenham esses mesmos dados. No entanto, quase todos os atributos podem ter valores em falta, não fazendo sentido ter que removê-los todos.

Para perceber melhor quais os atributos a eliminar, foi usada a técnica de *Feature Selection* para perceber quais os atributos mais relevantes para o cenário de estudo e remover aqueles que causam entropia no processo de classificação. Demasiados atributos podem dificultar o trabalho do algoritmo, tornando-o mais lento e até diminuindo a sua exatidão.

A técnica de *Feature Selection* ajuda a criar um modelo mais preciso possível consoante os dados existentes. Esta técnica ajuda a escolher os atributos que dão uma melhor precisão nos resultados mesmo com menos dados (Guyon & Elisseeff, 2002). Pode ser usada para identificar e remover atributos redundantes que não contribuem para a precisão do modelo de dados ou que podem diminuir a precisão. Menos atributos são desejáveis porque reduzem a complexidade de um modelo e um modelo mais simples é sempre mais fácil de entender e explicar.

O *Weka* suporta várias técnicas de *Feature Selection*. A técnica escolhida foi a *Information Gain Feature Selection*. Esta técnica tem como objetivo calcular o ganho de informação baseada no conceito de entropia (Yang & Pedersen, 1998). Ela é usada como uma medida de atributos mais relevantes numa estratégia de filtragem que avalia cada atributo individualmente (Yang & Pedersen, 1998). O ganho de informação é calculado para cada atributo sendo este o resultado obtido no final. Os valores de entrada variam entre 0 (sem informação) a 1 (informação máxima). Os atributos com mais informação terão um resultado superior aos restantes.

Como o *dataset* do Facebook representa os dados reais sobre eventos, esta técnica foi aplicada apenas neste *dataset* para perceber quais os atributos mais importantes. A Tabela 6 apresenta o contributo de cada atributo para o estudo em causa. É possível verificar que existem quatro atributos que contribuem mais para o processo de classificação. Os atributos são o *artist_id*, *event_start_hour*, *event_end_day_of_month* e o *venue_longitude*. Verifica-se também que os atributos *event_maybe_count*, *event_interested_count* e *event_attending_count*, relativos à componente social do evento, são os mais relevantes para poderem ser usados na classificação de eventos enquanto que os atributos *event_declined_count* e *event_no_reply_count* não tem relevância no modelo preditivo.

Tabela 6: Information Gain para os atributos do dataset do Facebook

Características	*Information Gain*
artist_id	0.8864
event_start_hour	0.4246
event_end_day_of_month	0.4246
venue_longitude	0.3639
event_maybe_count	0.1003

event_interested_count	0.06
event_attending_count	0.0423
venue_id	0.0403
event_declined_count	0
event_no_reply_count	0
event_end_hour	0
event_start_day_of_month	0
event_month	0
venue_longitude	0

Procedeu-se à remoção dos atributos do *dataset* que têm um ganho de informação igual a zero. Realizou-se uma nova sessão de testes de classificação onde os resultados podem ser vistos na Tabela 7.

Tabela 7: Resultados do 3º teste experimental

Algoritmos	*% de eventos corretamente classificados*
IBk	77.74%
SMO	69,74%
Random Forest	83,33%

A Tabela 7 mostra uma melhoria no processo de classificação comparando os resultados com a Tabela 4 e a Tabela 5. O algoritmo *Random Forest* aumentou cerca de 13,05%, o IBk aumentou cerca de 27,02% e o SMO aumentou 23,07%. Esta técnica veio demonstrar que o *dataset* existente continha bastantes atributos irrelevantes confundido os algoritmos, aumentando a complexidade de classificação e produzindo resultados menos satisfatórios.

Em suma, dada a grande diferença nos resultados entre a Tabela 4 e a Tabela 5 em comparação com a Tabela 7, é possível verificar que um dos problemas existentes em ambos os *datasets* estava relacionada com os valores em falta. Além disso, a utilização de uma técnica de *Feature Selection* veio demonstrar que a classificação com as características mais relevantes, mesmo com valores em falta, trouxe enormes melhorias no processo de classificação.

Após a finalização dos testes, conclui-se que numa primeira fase de testes, a técnica de *Feature Selection* deve ser aplicada porque permite perceber a redundância e irrelevância de algumas características. Mesmo para um grande conjunto de dados, no caso da inexistência de valores em falta, os resultados são bastante satisfatórios, atingindo 100% de eventos corretamente classificados, ao contrário do que aconteceu quando estes foram adicionados. Verifica-se também que o *Random Forest* é o melhor algoritmo para classificar os eventos sociais neste modelo de dados, dado que obteve sempre a melhor percentagem de eventos corretamente classificados em todos os testes e o seu melhor resultado foi de 83,3% de eventos corretamente classificados. Conclui-se também que relativamente à componente social adicionada, apenas os atributos

event_maybe_count, *event_interested_count* e *event_attending_count*, são os atributos mais relevantes para o modelo preditivo ajudando o processo de classificação a classificar corretamente o evento para uma determinada *tag*.

6 A ontologia LODSE

A primeira avaliação experimental permitiu perceber que o modelo da ontologia LODE pode ser aplicado num evento social, conseguindo bons resultados no processo de classificação. Dado que a ontologia LODE não possui nenhuma classe ou propriedade que permita categorizar um evento, criou-se uma nova ontologia com o objetivo de modelar um evento social inserindo propriedades que permitam fazer esta categorização e melhorar o processo de classificação de eventos. Além disso, a criação desta nova ontologia tem também como objetivo facilitar a integração dos dados obtidos das *APIs* públicas para um modelo de dados único a ser utilizado na plataforma *What's Happening?*.

Este capítulo apresenta a ontologia criada, com o nome de LODSE (*Linking Open Descriptions of Social Events*) para representar o modelo de domínio dos eventos sociais. O capítulo descreve as classes da ontologia, as relações entre ambas e as suas propriedades.

Existem vários métodos que permitem criar uma ontologia com resultados diferentes. Nos próximos subcapítulos são apresentadas as principais características da ontologia LODSE baseada no método de desenvolvimento definido em (Noy & Mcguinness, 2001) que apresenta conceitos de design para a conceção de uma ontologia.

6.1 O âmbito e domínio da ontologia LODSE

A ontologia LODSE visa cobrir o domínio dos eventos sociais, mais precisamente, eventos musicais, desportivos, artísticos, conferências, entre outros tipos de eventos. O objetivo desta ontologia é criar um modelo que permita definir as propriedades mais importantes para descrever um evento social, para alcançar melhores resultados na tarefa de classificação de eventos e para uma melhor integração dos dados dos eventos provenientes das *APIs* públicas.

Segundo (Grüninger & Fox, 1995), um método para definir o âmbito da ontologia é a criação de uma lista de perguntas que a ontologia deverá ser capaz de responder. Estas perguntas servem como um teste de validação para verificar se a ontologia tem as informações necessárias para representar o domínio dos eventos sociais. As questões definidas são:

- Qual o evento?
- Qual o nome do evento?
- Quem é o artista?
- Quem é o organizador?
- Onde irá decorrer o evento?

- A que horas é o evento?
- Que tipo de evento é?

6.2 Os termos importantes da ontologia

É importante obter uma lista geral dos termos que representam o domínio da ontologia sem nos preocuparmos com a sobreposição entre os conceitos que eles representam, as relações entre os termos ou quaisquer propriedades que os conceitos possam ter. Desenvolver esta lista de termos é uma das vantagens citadas em (Noy & Mcguinness, 2001) porque ajuda a definir as classes e as propriedades da ontologia.

Os termos definidos para a ontologia LODSE são:

- Evento;
- Data;
- Tags;
- Categoria;
- Organização;
- Artista;
- Preço;
- Local;
- Hora de início;
- Hora de fim.

6.3 As classes da ontologia e a sua hierarquia

Existem três métodos para desenvolver as classes e a sua hierarquia de uma ontologia (Uschol & Gruninger, 1996): *top-down*, *bottom-up* e *combination*. Para a ontologia LODSE foi aplicada a metodologia *top-down* que visa desenvolver a ontologia partindo dos conceitos mais genéricos do domínio e posterior especialização dos seus conceitos.

No subcapítulo anterior, foi definida uma lista de termos que podem ajudar a definir as classes da ontologia e responder às questões apresentadas no subcapítulo 6.1. As classes definidas que representam a ontologia LODSE são:

- **Event** – classe que descreve um evento e responde às perguntas *"Qual o evento?"* e *"Qual o nome do evento?"*;
- **Involved** – classe que representa as entidades envolvidas no evento e responde às perguntas *"Quem é o artista?"* e *"Quem é o organizador?"*;
    - **Artist** – subclasse que descreve o artista do evento;

    - **Organization** – subclasse que descreve o organizador do evento;
- **Date** – classe que representa a data do evento e responde à pergunta *"A que horas é o evento?"*;
    - **startDate** – subclasse que representa a data de início do evento;
    - **endDate** – subclasse que representa a data de fim do evento;
- **Venue** – classe que descreve o local onde o evento será realizado e responde à pergunta *"Onde irá decorrer o evento?"*;
    - **City** – subclasse que descreve a cidade onde o evento irá ocorrer;
    - **Country** – subclasse que descreve o país onde o evento irá ocorrer;
- **Taxonomy** – classe que representa a categorização de um evento e responde à pergunta *"Que tipo de evento é?"*;
    - **Tag** – subclasse que representa a *tag* de um evento;
    - **Categoria** – subclasse que representa a categoria do evento.

A Figura 10 mostra a visão geral das classes da ontologia LODSE, as subclasses e as relações entre estas. É também possível observar como esta é criada com base na ontologia LODE, percebendo as diferenças entre ambas as ontologias.

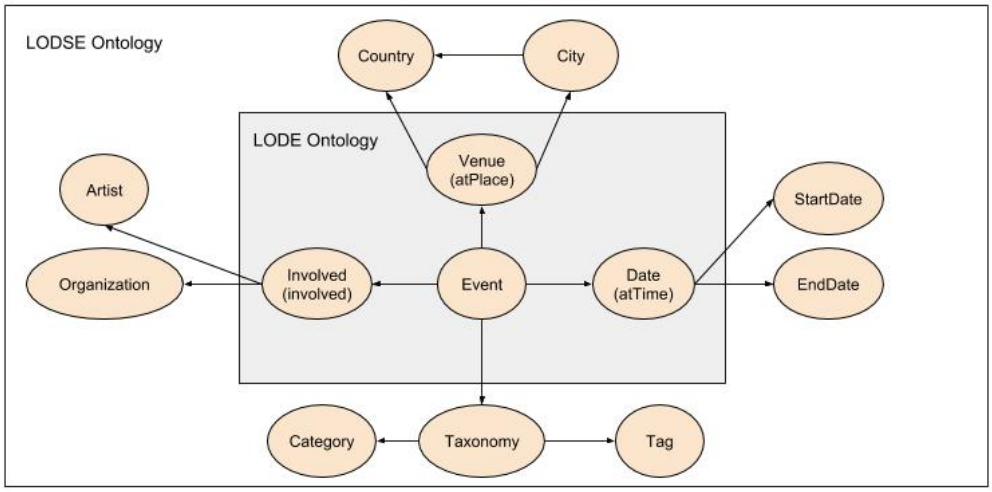


Figura 10. Visão geral das classes da ontologia LODSE

As classes *Event*, *Venue*, *Date* e *Involved* são propriedades importadas da ontologia LODE, que foram transformadas em classes na ontologia LODSE. As restantes são as novas classes pertencentes à ontologia LODSE. A Figura 10 também apresenta as relações existentes entre as várias classes. É possível observar que a classe *Event* é a classe principal da ontologia LODSE, onde da qual partem todas as relações. As relações existentes são explicadas da seguinte forma:

- ***Event – Date***: todos os eventos ocorrem numa determinada data. O evento tem uma hora de início e uma hora de fim;

- **Event – Local:** todos os eventos ocorrem num determinado local (sempre com uma latitude e longitude associados). Este local está localizado numa determinada cidade/país e a cidade pertence a um país;
- **Event – Involved**: todos os eventos têm alguém envolvido. Dependendo do tipo de evento, as entidades que podem estar envolvidas são os artistas ou os organizadores do evento;
- **Event – Taxonomy**: O evento é sempre categorizado através de uma taxonomia específica, isto é, o evento é classificado ou com uma categoria pré-definida ou com uma *tag*.

6.4 As propriedades das classes e os seus tipos

As classes são o foco da maioria das ontologias e descrevem os conceitos de um determinado domínio. Na ontologia LODSE, a classe *Event* representa todos os eventos sociais. Uma classe pode ter subclasses que representam os conceitos mais específicos do que a classe mãe. Por exemplo, podemos dividir a classe de todos os eventos sociais em apenas eventos de música devido à existência da subclasse *Taxonomy* ou dividir todos os eventos sociais apenas organizados por uma determinada entidade devido à existência da classe *Organization*.

No entanto, as classes só por si não fornecem as informações suficientes para responder às questões apresentadas no subcapítulo 6.1. Após a definição das classes, é necessário descrever a sua estrutura interna, mais concretamente definir as suas propriedades.

As propriedades das classes da ontologia LODSE foram escolhidas com base nas classes previamente definidas, nas questões que a ontologia pretende responder listadas no subcapítulo 6.1 e também a partir de uma análise de várias *APIs* públicas, apresentada no subcapítulo 5.1, para perceber as propriedades comuns entre estes serviços.

As propriedades das classes podem ter diferentes *facets* descrevendo o seu tipo ou os valores permitidos. O tipo da propriedade acaba por ser a característica mais importante no desenvolvimento de uma ontologia porque define o tipo de cada propriedade que irá definir as propriedades usadas no processo de classificação. Os tipos que podem ser atribuídos às propriedades são: *number*, *string*, *boolean*, *enumerated* e *instance*.

A Tabela 8 mostra todas as classes, propriedades e seus tipos (*facets*) da ontologia LODSE.

Tabela 8: Classes, propriedades e *facets* da ontologia LODSE

Classe	*Propriedades*	*Facets*
Event	eventID	number
	eventName	string
	eventDescription	string
	eventPrice	number
	eventURL	string
	eventDateCreated	date

	eventDateModified	date
	involvedName	string
Involved	involvedDescription	string
	involvedOfficialWebsite	string
Artist	artistID	number
Organization	organizationID	number
Date	date	date
StartDate	time	date
EndDate	allDay	boolean
	venueID	number
	venueName	string
	venueDescription	string
Venue	venueLatitude	number
	venueLongitude	number
	venueCapacity	number
	venuePostalCode	string
City	cityID	number
Country	countryID	number
Taxonomy	name	string
Category	categoryID	number
Tag	tagID	number

Com base neste estudo, os dados dos eventos gerados para a segunda avaliação experimental têm em conta as classes e propriedades da ontologia LODSE para posteriormente serem submetidos no processo de classificação de eventos.

7 Avaliação da ontologia LODSE no processo de classificação de eventos

Este capítulo apresenta a segunda avaliação experimental realizada que tem como objetivo comparar os modelos de dados baseados nas ontologias LODSE e LODE, no processo de classificação de eventos. Foi analisada a percentagem de eventos corretamente classificados, memória consumida e o tempo de processamento a fim de verificar se a o modelo de dados baseado na ontologia LODSE traz vantagens no processo de classificação de eventos, nomeadamente uma melhor percentagem de eventos corretamente classificados, baixo consumo de memória e tempo de processamento mais rápido face à ontologia LODE. Esta avaliação experimental contou com cerca de 540 testes que demoraram aproximadamente vinte e dois dias e doze horas a serem realizados (tempo de CPU).

O capítulo encontra-se dividido em duas partes. A primeira parte apresenta o *setup* experimental usado para a realização dos testes. A segunda parte apresenta e analisa os resultados obtidos nos testes de classificação.

7.1 Setup Experimental

Este subcapítulo apresenta o *setup* experimental usado na segunda avaliação experimental. É apresentado o *hardware* e *software* utilizado para a realização dos testes, o algoritmo escolhido e a estrutura dos *datasets* de eventos utilizados no processo de classificação.

7.1.1 Hardware

Para a realização da segunda avaliação experimental, foram utilizadas duas máquinas com as seguintes características:

- Máquina 1 – Processador 1.4GHz Intel Core i5, 8GB de RAM DD3
- Máquina 2 – Processador Intel Xeon 2.39 GHz, 40GB de RAM

Como a primeira máquina utilizada não possuía memória suficiente para realizar todos os testes, uma segunda máquina foi adicionada para realizar os testes para *datasets* onde o número de eventos era superior a 10200.

7.1.2 Weka

O Weka, *software* usado na primeira avaliação experimental apresentada no capítulo 5, foi o *software* novamente escolhido para a realização dos testes de classificação.

Para esta segunda avaliação experimental, não foi usada a aplicação *standalone*, mas sim a biblioteca de *machine learning* do Weka, integrada numa aplicação Java desenvolvida no decorrer deste trabalho apresentada na seguinte secção.

7.1.3 Aplicação desenvolvida para a classificação de eventos

Como mencionado anteriormente, foi desenvolvida uma aplicação Java que integra a biblioteca de *machine learning* do Weka para proceder aos testes de classificação de eventos. Foi aplicada esta abordagem para permitir uma melhor recolha dos dados referentes à memória consumida e tempo de processamento dos testes de classificação, sem ser necessário recorrer a ferramentas externas, que introduziam ruído nos parâmetros a ser avaliados e requeriam uma análise manual destes. Para um melhor entendimento da aplicação desenvolvida, a Figura 11 apresenta o fluxo para classificar um determinado *dataset*.

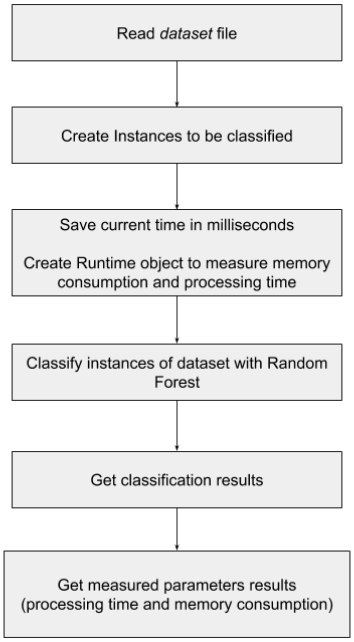


Figura 11. Fluxo da aplicação desenvolvida para classificar um determinado *dataset* de eventos

A aplicação começa por ler um determinado *dataset* de eventos para poder criar o objeto *Instances*, classe da biblioteca *Weka* usada para manipular um conjunto ordenado de instâncias. Após a criação deste objeto, é guardada a hora atual em milissegundos e criado um objeto *Runtime* que permite interagir com o ambiente no qual a aplicação é executada para proceder depois à análise da memória consumida e tempo de execução. De seguida, é feita a classificação das instâncias

criadas com o algoritmo *Random Forest* e após esta estar finalizada são obtidos os resultados de classificação e os resultados relativos aos parâmetros avaliados, nomeadamente tempo de processamento e memória consumida. No anexo C, está disponível o código fonte da aplicação desenvolvida para uma melhor compreensão da lógica da aplicação.

7.1.4 Random Forest

Na primeira avaliação experimental apresentada no capítulo 5, o *Random Forest* foi o algoritmo que obteve a melhor percentagem de eventos corretamente classificados (83,33%) face aos algoritmos IBk e SMO.

Dado os excelentes resultados por parte do algoritmo, a segunda avaliação experimental apenas procedeu aos testes de classificação usando este algoritmo.

7.1.5 Datasets

Os *datasets* criados para esta avaliação experimental são ficheiros ARFF que contêm os dados do evento e um conjunto de *tags* predefinidas para serem usadas nos testes de classificação.

Um ficheiro ARFF possui duas secções distintas. A primeira secção é a informação do cabeçalho que é seguida pelos dados na segunda secção. O cabeçalho contém o nome da relação, a lista de atributos (referentes às colunas dos dados) e os seus tipos. De forma sucinta, esta secção define a estrutura dos dados a serem utilizados no processo de classificação representando as propriedades de um evento social. A segunda secção representa os dados, ou seja, os dados dos eventos para propriedades definidas nas informações do cabeçalho.

Dois tipos de conjuntos de dados foram criados e estes diferem na lista de atributos definidos no cabeçalho de um arquivo ARFF. O primeiro tipo de conjunto de dados é relativo ao modelo de dados baseado na ontologia LODE e o segundo tipo de conjunto de dados é relativo ao modelo de dados baseado na ontologia LODSE.

A Tabela 9 apresenta os atributos do modelo de dados baseado na ontologia LODE. Esses atributos são os mais relevantes e obtiveram 83,33% de eventos corretamente classificados na primeira avaliação experimental.

Tabela 9: Atributos para o modelo de dados baseado na ontologia LODE

Atributos	*Tipo*
artist_id	Numeric
event_start_hour	Numeric
event_end_day_of_month	Numeric
event_maybe_count	Numeric
event_interested_count	Numeric
event_attending_count	Numeric

venue_id	Numeric
venue_longitude	Numeric

A Tabela 10 apresenta os atributos do modelo de dados baseados na ontologia LODSE. Visto que o algoritmo escolhido para realizar os testes é o Random Forest, os atributos do modelo de dados são todas as propriedades das classes da ontologia LODSE com valor numérico, *boolean* ou enumeração porque o Random Forest não suporta valores nominais como *strings.*

Tabela 10: Atributos do modelo de dados baseado na ontologia LODSE

Atributos	*Tipo*
artist_id	Numeric
category_id	Numeric
event_start_hour	Numeric
event_end_hour	Numeric
event_start_day_of_month	Numeric
event_end_day_of_month	Numeric
event_month	Numeric
date_all_day	boolean
event_price	Numeric
organization_id	Numeric
venue_id	Numeric
venue_latitude	Numeric
venue_longitude	Numeric
city_id	Numeric
country_id	Numeric

Os dados dos eventos foram gerados a partir de um algoritmo desenvolvido no decorrer deste trabalho. Como não foram encontrados *datasets* públicos que fossem de encontro ao tipo de dados necessários para a avaliação experimental, o algoritmo desenvolvido gerou todos os dados dos eventos com base nas tabelas Tabela 9 e Tabela 10. Salienta-se que na criação dos dados, existe 30% de eventos que contêm valores em falta. Esta escolha pretende simular um ambiente mais real relativo aos dados que os vários serviços externos podem retornar através das suas *APIs*.

O número de eventos criados varia entre 2040 e 51000. Para cada número específico de eventos, vários ficheiros ARFF foram criados, diferindo no número de *tags*. O número mínimo de *tags* é 6 e o máximo é de 96. No contexto de um ficheiro ARFF, as *tags* representam o valor *@class* e são os valores nos quais um evento pode ser classificado.

Foram definidos 9 conjuntos de eventos (2040, 4080, 6120, 8160, 10200, 20400, 30600, 40800 e 51000) e para cada conjunto, foi criado um *dataset* correspondente a 6, 12, 18, 24, 30 e 96 *tags*. Multiplicando os nove conjuntos de eventos por seis conjuntos de *tags* e duas ontologias,

obtiveram-se 108 *datasets* que serão utilizados no processo de classificação e estão disponíveis para consulta em (https://tinyurl.com/y838r2d6).

O número de *tags* foi criado com base em múltiplos de 6. De 6 a 30, o número de *tags* é linear, sendo aumentado depois para 96. A razão deste salto é para provar se existe um comportamento linear no processo de classificação ao aumentar o número de *tags* e conhecer a variação no desempenho entre os dois tipos de conjuntos de dados das ontologias para *datasets* com um elevado número de *tags*.

Para cada conjunto de dados específico, cinco testes foram realizados. Dado que se obteve 108 *datasets,* a avaliação experimental contou com 540 testes de classificação (108 *datasets* * 5 testes para cada). A realização dos cinco testes para o mesmo *dataset* tem como objetivo obter uma média mais precisa da percentagem de eventos corretamente classificados, consumo de memória e tempo de processamento.

7.2 Resultados

Como referido anteriormente, o objetivo da segunda avaliação experimental é comparar os modelos de dados baseados nas ontologias LODSE e LODE, no processo de classificação, analisando a percentagem de eventos corretamente classificados, memória consumida e o tempo de processamento a fim de verificar se o modelo de dados baseado na ontologia LODSE traz vantagens no processo de classificação de eventos face à ontologia LODE.

A percentagem de eventos corretamente classificados refere-se à quantidade de eventos que foram corretamente classificados num ficheiro ARFF. Num *dataset* com 100 eventos, se 50 eventos foram classificados corretamente, a percentagem de eventos corretamente classificados seria de 50%. O consumo de memória, medido em *megabytes* durante o processo de classificação, refere-se ao consumo de memória necessário para construir o modelo de dados por parte do algoritmo mais o consumo de memória necessário para classificar os eventos. O tempo de processamento, medido em segundos durante o processo de classificação, refere-se ao tempo necessário para construir o modelo de dados mais o tempo necessário para classificar os eventos.

Após a realização dos 540 testes, todos os resultados podem ser consultados no Anexo D. De seguida são apresentados apenas os resultados para os números de *tags* 6, 30 e 96 que dão uma visão geral dos resultados obtidos em todos os testes, para os parâmetros avaliados, onde estes são analisados e por último discutidos.

7.2.1 Percentagem de eventos corretamente classificados

A percentagem de eventos corretamente classificados é a percentagem de eventos que foram classificados corretamente num ficheiro ARFF.

A Figura 12 apresenta a evolução percentual dos resultados dos testes de classificação onde o número de *tags* foi de 6, 30 e 96.

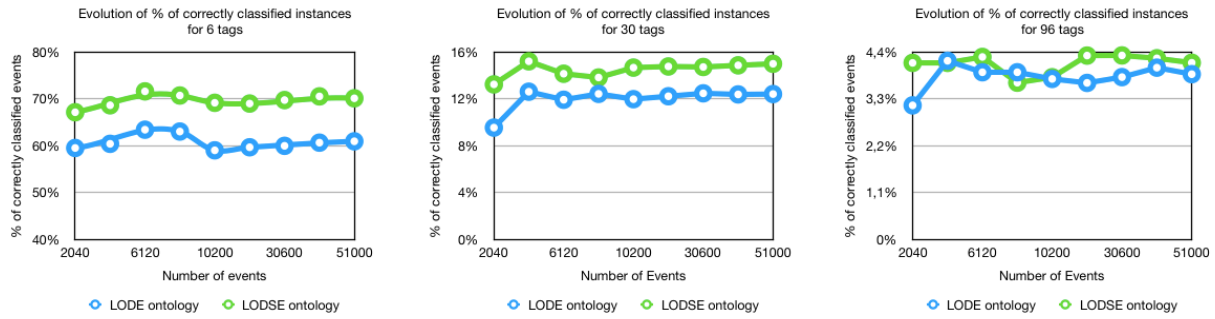


Figura 12: Evolução da percentagem de eventos corretamente classificados para 6, 30 e 96 tags

No geral, verifica-se que a ontologia LODSE apresenta uma melhor percentagem de eventos corretamente classificados em comparação com a ontologia LODE. Em média, a ontologia LODSE obteve mais de:

- 12,78% de eventos corretamente classificados quando o número de *tags* é igual a 6;
- 17,31% de eventos corretamente classificados quando o número de *tags* é igual a 30;
- 7,12% de eventos corretamente classificados quando o número de *tags* é igual a 96.

Na Figura 12, é também possível observar que a percentagem de eventos classificados diminui à medida que o número de *tags* aumenta. Esta redução percentual é justificada com o aumento do número de *tags* para o mesmo número de eventos, ou seja, o número de possibilidades que um evento pode ser classificado aumentou, enquanto que o número de eventos se manteve sempre o mesmo. Desta forma, a percentagem de eventos por *tag* diminuiu, justificando a baixa percentagem para os testes de classificação onde o número de *tags* é de 96.

7.2.2 Memória Consumida

O consumo de memória, medido em *megabytes* durante o processo de classificação, refere-se ao consumo de memória necessário para construir o modelo de dados por parte do algoritmo mais o consumo de memória necessário para classificar os eventos.

A Figura 13 apresenta a evolução da memória consumida para os testes de classificação onde o número de *tags* foi de 6, 30 e 96.

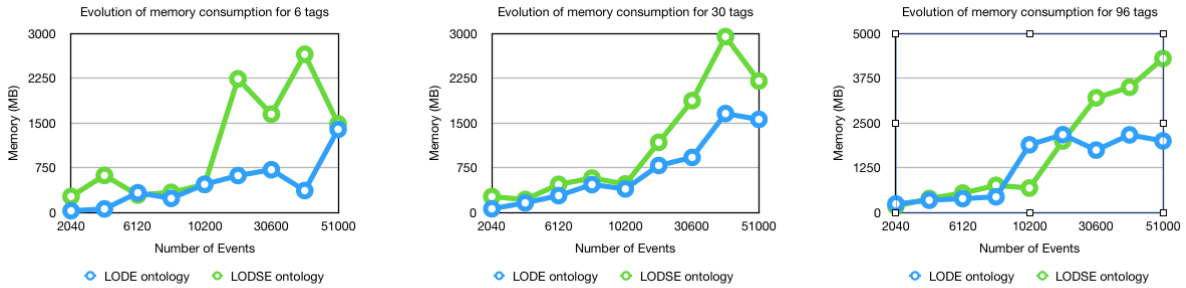


Figura 13: Evolução da memória consumida para 6, 30 e 96 tags

No geral, a ontologia LODSE teve um maior consumo de memória face à ontologia LODE. Em média, a ontologia LODSE consome:

- 46,34% mais memória do que a ontologia LODE quando o número de *tags* é igual a 6;
- 37,20% mais memória do que a ontologia LODE quando o número de *tags* é igual a 30;
- 0,44% menos memória do que a ontologia LODE quando o número de *tags* é igual a 96.

A razão de um maior consumo de memória por parte da ontologia LODSE é devido ao número de árvores necessárias que o algoritmo Random Forest precisa de criar para serem usadas no processo de classificação. No entanto, quando o número de *tags* aumenta, a média de consumo vai sendo reduzida.

O algoritmo *Random Forest* cria as suas árvores com base nos atributos, classes e instâncias definidas num ficheiro ARFF. Segundo (Oshiro, Perez, & Baranauskas, 2012), à medida que o número de árvores aumenta, nem sempre significa um melhor desempenho em comparação com menos árvores. Desta forma, o consumo de memória é menor para a ontologia LODE porque, dado que esta tem menos atributos do que a ontologia LODSE, o algoritmo cria menos árvores para poder classificar os eventos, fazendo com que o consumo de memória seja inferior comparado com a ontologia LODSE.

Através da Figura 13, verifica-se também que o consumo de memória não é linear à medida que o número de eventos aumenta. Tendo em conta que a aplicação desenvolvida para proceder aos testes de classificação foi desenvolvida na linguagem Java, a gestão de memória é automática através do *Garbage Collector*. O *Garbage Collector* é uma abordagem dinâmica para fazer a gestão automática de memória que processa e identifica blocos de memória inativos realocando o armazenamento de memória para reutilização.

7.2.3 Tempo de processamento

O tempo de processamento, medido em segundos durante o processo de classificação, refere-se ao tempo necessário para construir o modelo de dados mais o tempo necessário para classificar os eventos.

A Figura 14 apresenta a evolução do tempo de processamento para os testes de classificação onde o número de *tags* foi de 6, 30 e 96.

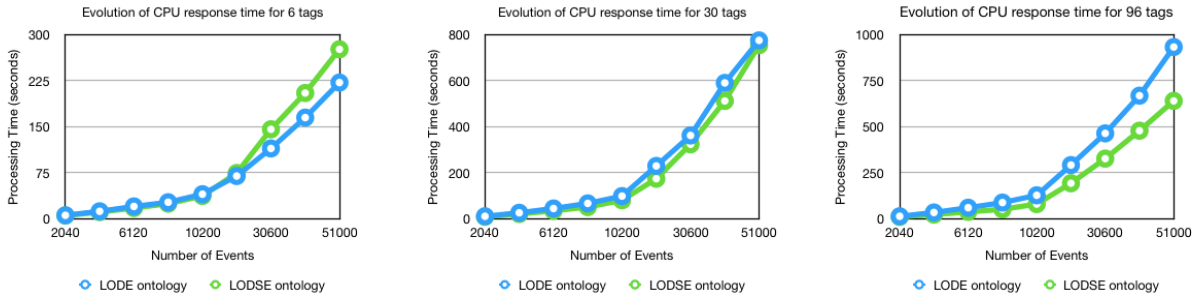


Figura 14: Evolução do tempo de processamento para 6, 30 e 96 tags

Em média, a ontologia LODSE levou:

- 1,64% mais tempo de processamento quando o número de *tags* é igual a 6;
- 7,05% menos tempo de processamento quando o número de *tags* é igual a 30;
- 12,28% menos tempo de processamento quando o número de *tags* é igual a 96;

No geral, a ontologia LODSE demorou menos tempo para classificar *dataset* do que a ontologia LODE. É possível afirmar que à medida que o número de eventos aumenta, o tempo de processamento também aumenta, porque existem mais eventos para serem classificados. A ontologia LODSE teve um melhor tempo de processamento quando o número de *tags* foi igual a 30 e 96. Para um número de *tags* igual a 6, a ontologia LODE teve um menor tempo de processamento por uma percentagem mínima de diferença.

7.2.4 Discussão dos resultados

Em suma, o modelo de dados criado com base na ontologia LODSE traz melhorias no processo de classificação de eventos sociais, obtendo uma melhor percentagem de eventos corretamente classificados e levando menos tempo para classificar os eventos, particularmente para um elevado número de *tags* e eventos. Para a percentagem de eventos corretamente classificados, o melhor resultado obtido pela ontologia LODSE foi para um número de *tags* igual a 30, obtendo mais 17,31% eventos corretamente classificados. Para o tempo de processamento, o melhor resultado obtido pela ontologia LODSE foi para um número de *tags* igual a 96, levando 12,28% menos tempo a classificar eventos do que a ontologia LODE.

No entanto, a ontologia LODE teve um menor consumo de memória face à ontologia LODSE, mas à medida que o número de *tags* ia aumentando, a diferença percentual existente foi diminuindo, onde a ontologia LODSE consegue consumir menos 0,44% de memória para um número de tags igual a 96.

Estes resultados validam com sucesso o modelo de dados criado pela ontologia LODSE para ser usado no processo de classificação de eventos. Embora os dados tenham sido gerados aleatoriamente, o uso de mais atributos para determinar a *tag* de um evento demonstrou melhorias na obtenção de mais eventos corretamente classificados.

É de salientar também a escolha das propriedades da ontologia que fizeram parte do modelo de dados no processo de classificação. Como apenas foram escolhidas as propriedades numéricas da ontologia LODSE baseados em identificadores numéricos das classes entre outras propriedades, pressupõe-se que a classificação de eventos seja menos complexa ao invés de utilizar as propriedades da ontologia LODE. Um exemplo prático pode ser a associação da *tag* "*rock-music*" a um local específico que só tem concertos de música rock. Para o algoritmo, esta associação pode ser realizada porque o modelo de dados contempla um atributo, o *venueID*, que identifica o local em questão. Em suma, a abordagem aplicada na classificação de eventos trouxe também bons resultados no processo de classificação.

Em termos de desempenho, a ontologia LODSE apenas fracassou no consumo de memória apresentado resultados superiores à ontologia LODE. Como explicado anteriormente, este maior consumo está relacionado com a memória necessária para criar o modelo preditivo do algoritmo *Random Forest*. Um maior número de atributos no modelo de dados leva à criação de mais árvores do modelo preditivo. Como a ontologia LODSE possui quase o dobro de atributos que a ontologia LODE, os resultados dos testes de classificação comprovam este maior consumo de memória.

Relativo ao tempo de processamento, as melhorias obtidas podem estar relacionadas apenas ao uso de atributos numéricos, dado que o algoritmo *Random Forest* tem um melhor desempenho quando o modelo de dados é baseado neste tipo de atributos. De salientar que o algoritmo é conhecido por ter bons desempenhos em conjuntos com muitos dados e atributos, confirmando assim o bom desempenho para um número de *tags* igual a 96 por parte da ontologia LODSE.

Esta segunda avaliação experimental demonstra que a ontologia LODSE trouxe vantagens no processo de classificação de eventos sociais, alcançando assim uma melhor percentagem de eventos corretamente classificados e um tempo de processamento mais rápido quando comparado com a ontologia LODE. Em particular, para um elevado número de *tags*, a ontologia LODSE também apresentou bons resultados, demonstrado que é possível classificar eventos para um elevado número de *tags* com um bom desempenho. No entanto, o consumo de memória foi pior dado o número de atributos usados na ontologia LODSE.

As duas avaliações experimentais realizadas tinham como objetivo validar o módulo de *machine learning* na arquitetura da plataforma apresentada em 3.4. Com base nos resultados obtidos, o módulo de classificação da arquitetura, *Classifier Module*, que tem como objetivo classificar os eventos para serem posteriormente guardados na base de dados do sistema, irá usar o algoritmo *Random Forest* para proceder à classificação dos eventos dado que este foi o algoritmo que obteve a melhor percentagem de eventos corretamente classificados. O módulo *Data Fetcher*, que tem como objetivo obter eventos provenientes das *APIs* públicas irá utilizar a ontologia LODSE para modelar os eventos obtidos para depois serem posteriormente classificados pelo módulo *Classifier Module*. Os resultados permitiram também perceber que com o uso da ontologia LODSE e o algoritmo *Random Forest,* obtêm-se uma melhor percentagem de eventos corretamente classificados permitindo que as funcionalidades de procura e recomendação de eventos retornem os resultados esperados consoante o *input* do utilizador. Salienta-se também a as melhorias na velocidade de classificação (tempo de processamento), principalmente para um número elevado de *tags* para que após a obtenção de eventos, estes possam ser imediatamente classificados e guardados na base de dados a fim de estarem disponíveis para serem consultados pelos utilizadores.

8 Conclusões e Trabalho Futuro

A procura de conteúdo digital relacionado com eventos, a obtenção de eventos recomendados do nosso interesse ou ser notificado a tempo e horas de um evento é difícil nos dias de hoje, exigindo o uso de diferentes serviços e geralmente, a maioria dos eventos contêm informação ambígua e incompleta.

A crescente evolução das tecnologias de comunicação e informação tem permitido um acesso à informação facilitado com inúmeras plataformas a surgir para tentar melhorar a experiência do utilizador no âmbito dos eventos sociais.

Este trabalho propõe um novo conceito para uma plataforma de divulgação de eventos, intitulada de *What's Happening?,* com o objetivo de melhorar a experiência do utilizador na procura, recomendação e notificação de eventos. A categorização de eventos é baseada em *tags,* prevendo-se que o conceito aplicado traga melhorias no processo de procura e recomendação de eventos baseado nos interesses dos utilizadores.

Na arquitetura da plataforma, são utilizadas técnicas de *machine learning* que tem como objetivo classificar os eventos, provenientes de *APIs* públicas de outros serviços mais populares, e.g. Facebook, nas *tags* guardadas no sistema da plataforma *What's Happening?*.

Tendo em conta que os serviços utilizados para a obtenção de eventos apresentam modelos de dados diferentes, foi proposta a ontologia LODSE (*Linking Open Descriptions of Social Events*) tendo como base a ontologia LODE (*Linking Open Descriptions of Events*). Esta nova ontologia teve como objetivo facilitar a integração dos dados obtidos das *APIs* públicas dos serviços utilizados, modelar os dados de um evento social para ser posteriormente classificado e melhorar a classificação de eventos. Como forma de validação das técnicas de *machine learning* na arquitetura proposta, realizaram-se duas avaliações experimentais de classificação de eventos.

A primeira avaliação experimental teve como objetivo comparar diferentes tipos de algoritmos de três métodos de aprendizagem automática (*Decision Trees*, *Lazy Classifiers* e *Function Classifiers)* e perceber qual o algoritmo que tem uma melhor percentagem de eventos corretamente classificados. A avaliação experimental contou com três testes distintos e após a realização destes, conclui-se que o melhor algoritmo para classificar eventos é o *Random Forest,* algoritmo que obteve sempre a melhor percentagem de eventos corretamente classificados.

Com base nesta avaliação experimental, conclui-se também que as *APIs* de serviços externos podem retornar valores em falta que podem levar a maus resultados de classificação. No sentido de resolver este problema, podem ser aplicadas várias técnicas que permitem fazer a gestão destes

valores. Além disso, o uso de técnicas de *Feature Selection* é também essencial neste tipo de avaliações, permitindo perceber quais os atributos mais importantes num conjunto de dados com a finalidade de remover os atributos irrelevantes. Após a aplicação de várias técnicas, o algoritmo Random Forest obteve 83,33% de eventos corretamente classificados, sendo o algoritmo escolhido para realizar uma segunda avaliação experimental.

A segunda avaliação experimental teve como objetivo comparar os modelos de dados baseados nas ontologias LODSE e LODE, no processo de classificação. Foi analisada a percentagem de eventos corretamente classificados, memória consumida e o tempo de processamento a fim de verificar se a o modelo de dados baseado na ontologia LODSE traz vantagens no processo de classificação de eventos. A partir da análise dos resultados, conclui-se que a ontologia LODSE trouxe melhorias no processo de classificação. Os testes realizados demonstraram bons resultados, especialmente em *datasets* onde existia um elevado número de eventos e *tags*. No geral, os testes demonstraram uma melhoria média percentual de 12,40% na percentagem de eventos corretamente classificados e uma melhoria média percentual de 5,89% no tempo de processamento para a ontologia LODSE face à ontologia LODE. No entanto, houve um maior consumo de memória por parte da ontologia LODSE devido à memória necessária para criar o modelo preditivo do algoritmo *Random Forest*. De acordo com os resultados obtidos, conclui-se também que é possível classificar eventos sociais com base em identificadores numéricos em vez de dados textuais. A criação da ontologia LODSE ajudou a perceber quais as propriedades mais importantes de um evento social e escolher apenas as propriedades numéricas para serem usadas no modelo de dados na classificação para um melhor desempenho.

Apesar dos bons resultados, existe a necessidade de validar a segunda avaliação experimental com *datasets* de eventos com dados reais. Como trabalho futuro pretende-se realizar uma nova avaliação experimental com dados do Facebook. Os dados devem estar de acordo com a ontologia LODSE e os mesmos testes devem ser realizados para perceber se os resultados se mantêm ou melhoram face aos resultados obtidos neste trabalho. Tendo em conta que a memória consumida foi o único parâmetro onde os resultados foram piores, deve-se aplicar as técnicas de *feature selection* no sentido de perceber quais os atributos mais importantes no modelo preditivo a fim de melhorar não só a percentagem de eventos corretamente classificados, mas também reduzir a memória necessária para classificar eventos.

Após a realização da avaliação experimental acima, se os resultados se mantiverem ou melhorarem, o módulo de *machine learning* na arquitetura da plataforma *What's Happening?* encontra-se validado para que a plataforma possa ser implementada como trabalho futuro e colocada à disposição dos utilizadores a fim de melhorar a experiência destes na procura e recomendação de eventos.

Referências

8tracks. (n.d.). *8tracks internet radio | Free music playlists | Best app for music*. Retrieved from 8tracks: https://8tracks.com/

Agenda, V. (n.d.). *O que é que se passa?* Retrieved from Viral Agenda: https://www.viralagenda.com

Auth0. (n.d.). *OAuth*. Retrieved from OAuth Community Site: https://oauth.net/

Automattic. (n.d.). *WordPress*. Retrieved from WordPress: https://wordpress.com/

Barrigas, H., Barrigas, D., Barata, M., Bernardino, J., & Furtado, P. (2015). Scalability of Facebook Architecture. *New Contributions in Information Systems and Technologies. Springer Nature*, (pp. 763-772).

Baruah, T. D. (2012, May). Effectiveness of Social Media as a tool of communication and its potential for technology enabled connections: A micro-level study. *International Journal of Scientific and Research Publications, 2*(5).

Benson, E., Haghighi, A., & Barzilay, R. (2011). Event discovery in social media feeds. *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, *1*, pp. 389-398.

Branquinho, J. M. (2005). Acontecimento. In J. Branquinho, D. Murcho, & N. G. Gomes, *Enciclopédia de Termos Lógico-Filosóficos* (pp. 23-28). Retrieved from Q.E.D: http://www.joaomiguelbranquinho.com/uploads/9/5/3/8/9538249/acontecimento.pdf

Breiman, L. (2001, October 1). RandomForests. *Machine Learning, 45*, 5-32.

Brownlee, J. (2017). *How to Handle Missing Data with Python*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/handle-missing-data-python/

Chow, C.-Y., Bao, J., & Mokbel, M. F. (2010). Towards location-based social networking services . *Proceedings of the 2010 International Workshop on Location Based Social Networks.* San Jose.

Conley, M. (n.d.). *How to Drive Demand, Create Loyal Fans, and Grow Your Business*. Retrieved from HubSpot: https://www.hubspot.com/facebook-marketing

Costa-Dasilva, J. I., Gomez-Rodriguez, A., Moreno, J. C., & Valcarcel, D. R. (2015, January). A located and user personalized event's dissemination platform. *Journal of Intelligent and Fuzzy Systems, 28(1)*, 71-81.

Dong, T., Liang, C., & Xu, H. (2017, June). Social media and internet public events . *Telematics and Informatics*, 726-739.

Dreamgrow. (n.d.). *Top 15 Most Popular Social Networking Sites and Apps [February 2018]*. Retrieved from Your Source of Content Marketing & Social Media Information: https://www.dreamgrow.com/top-15-most-popular-social-networking-sites/

Duarte, J. D. (2009). *Organização e Gestão de Eventos – Métodos e Técnicas e a sua Aplicação na Actividade das Empresas de Eventos – Estudo de Caso: Dice Eventos.*

Eventbrite. (n.d.). *Eventbrite*. Retrieved from Eventbrite: https://www.eventbrite.pt/

Facebook. (n.d.). *Facebook*. Retrieved from Facebook: https://www.facebook.com/

Facebook. (n.d.). *Facebook Events*. Retrieved from Facebook Events: https://events.fb.com/

Girolami, M., Chessa, S., & Caruso, A. (2015). On service discovery in mobile social networks: Survey and perspectives. *Computer Networks: The International Journal of Computer and Telecommunications Networking, 88*, 51-71.

Grüninger, M., & Fox, M. S. (1995). MethodologyfortheDesignand Evaluation of Ontologies.

Guarino, N. (1998). Formal Ontology and Information Systems. Formal Ontology in Information Systems. *Proceedings of the 1st International Conference*, (pp. 3-15).

Gupta, I., Gautam, K., & Chandramouli, K. (2013). MediaEval 2013 social event detection task: Semantic structuring of complementary information for clustering events. *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop.*

Guyon, I., & Elisseeff, A. (2002, November). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research 3*.

Harrington, P. (2012). *Machine Learning in Action.*

Hendrickson, M. (2012). *The Uphill Battle Of Social Event Sharing: A Post-Mortem for Plancast*. Retrieved from Tech Crunch: https://techcrunch.com/2012/01/22/post-mortem-for-plancast/

Kaplan, A. M., & Haenlein, M. (2010). Users of the World, Unite! The Challenges and Opportunities of Social Media. *Business Horizons.*

Kimura, M., Saito, K., Nakano, R., & Motoda, H. (2010, October). Extracting influential nodes on a social network for information diffusion. *Data Mining and Knowledge Discovery*, 70-97.

Kwak, H., Lee, C., Park, H., & Moon, S. B. (2010). What Is Twitter, a Social Network or a News Media? . *Proceedings of the 19th international conference on World wide web.*

Malkin, H. (2015, 9 26). *Why no one has solved event discovery*. Retrieved from Hugh Malkin: http://www.hughmalkin.com/blogwriter/2015/9/23/why-no-one-has-solved-event-discovery

Markov , Z., & Russell, I. (2006). An Introduction to the Weka Data Mining System. *11th annual SIGCSE conference on Innovation and technology in computer science education*, (pp. 367-368).

Mitchell, T., & Hill, M. (1997). *Machine Learning.*

New, E. I. (n.d.). *Everything Is New*. Retrieved from Everything Is New: http://www.everythingisnew.pt/

Nguyen, D. L., & Le, T. M. (2016). Recommendation system for Facebook public events based on probabilistic classification and re-ranking. *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, (pp. 133-138). Hanoi.

Nowak, S., & Lukashevich, H. (n.d.). Multilabel Classification Evaluation using Ontology Information. *CEUR Workshop Proceedings*, *474.*

Noy, N., & Mcguinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. *Knowledge Systems Laboratory*(32).

Nurwidyantoro, A., & Winarko, E. (2013). Event detection in social media: A survey . *ICT for Smart Society (ICISS).*

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How Many Trees in a Random Forest? *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, *7376*, pp. 154-168.

Panagiotou, N., Katakis, I., & Gunopulos, D. (2016). Detecting Events in Online Social Networks: Definitions, Trends and Challenges. In *Solving Large Scale Learning Tasks. Challenges and Algorithms* (Vol. 9580, pp. 42-44).

Platt, J. (1998). Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.

Rodrigues, M. (2009). LODE: Linking Open Descriptions of Events . *The Semantic Web: Fourth Asian Conference.*

Rokach, L., & Shapira, B. (2008). *Recommender Systems Handbook.*

Serrasqueiro, Z., Nunes, P. M., Leitao, J., & Armada, M. (2010). Are there non-linearities between SME growth and its determinants? A quantile approach. *Industrial and Corporate Change, 19*, 1071-1108.

Shaw, R., Troncy, R., & Hardman, L. (2009). LODE: Linking Open Descriptions of Events. *The Semantic Web, Fourth Asian Conference.* Shanghai: ASWC 2009.

Sutanto, T., & Nayak, R. (2013). MediaEval 2013 Social Event Detection. *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop.*

The PostgreSQL Global. (1996). *PostgreSQL*. Retrieved from PostgreSQL: The world's most advanced open source database: https://www.postgresql.org/

The University of Waikato. (n.d.). *Weka 3: Data Mining Software in Java* . Retrieved from Weka 3: Data Mining Software in Java : https://www.cs.waikato.ac.nz/ml/weka/

Troncy, R., Fialho, A., Hardman, L., & Saathoff, C. (2010). Experiencing Events through User-Generated Media. *COLD'10 Proceedings of the First International Conference on Consuming Linked Data*, *665*, pp. 61-72. Shanghai.

UC, A. (n.d.). *Eventos na cidade de Coimbra*. Retrieved from Agenda UC: http://agenda.uc.pt/

Uschol, M., & Gruninger, M. (1996, June). OntologiesPrinciplesMethods and Applications. *Knowledge Engineering Revie, 11*(2).

Villamizar, M., Garcés, O., Castro, H., Merino, M. V., Salamanca, L., Casallas, R., & Santiago, G. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. *10th Computing Colombian Conference.*

Yang, R., Li, G., Lau, W. C., Zhang, K., & Hu, P. (2016). Model-based Security Testing: an Empirical Study on OAuth 2.0 Implementations. *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, (pp. 651-662).

Yang, Y., & Pedersen, J. O. (1998, April). A Comparative Study on Feature Selection in Text Categorization.

Zeppelzauer, M., & Schopfhauser, D. (2016). Multimodal classification of events in social media. *Image and Vision Computing.*

Anexo A – An Event Search Platform Using Machine Learning

Artigo publicado na SEKE 2017, The Twenty-Ninth International Conference on Software Engineering and Knowledge Engineering (SEKE 2017), Pittsburgh, USA.

An Event Search Platform Using Machine Learning

Marcelo Aguiar Rodrigues
Polytechnic of Coimbra
ISEC
Coimbra, Portugal
a21180873@isec.pt

Rodrigo Rocha Silva
São Paulo State Technological College
CISUC
Mogi das Cruzes, Brasil
rodrigo.rsilva@fatec.sp.gov.br

Jorge Bernardino
Polytechnic of Coimbra - ISEC
CISUC
Coimbra, Portugal
jorge@isec.pt

***Abstract*—Currently, the evolution of technology allows to find which events occurs around us at any given location. Social networks are one of the reasons of this trend and new applications are emerging aiming at finding and disclosing events. This paper proposes a platform of event searching. In particular, we propose a new architecture that uses machine learning to classify events with tags. An experimental evaluation with different types of algorithms was done using Facebook as a source of dataset events.**

Keywords - events search; data mining; machine learning

I. INTRODUCTION

Nowadays, the quantity of digital information about what happens around us is dispersed in many applications. Usually, working with events, implies dealing with variables like date, location and time [1]. With the emergence of social networks, other types of relevant information should be considered, like a list of users that have an interest in the event or a list of users who will attend the event. In the literature, it is usually mentioned that users like sharing their stories, opinions, photos, and videos on social networks, creating a direct and social interaction between the participants on a certain event [2].

The events are a natural way to show an observable occurrence, grouping people, places, times, and activities [3]. Also, they might be considered as observable experiences that are often documented through photos and videos [4].

This paper presents a new idea of a platform for event searching. In particular, we propose a new architecture using machine learning to provide more accurate information according to the user interests. The main advantage of the platform is to bring a more personalized system where the user can find what s/he needs and get recommend events based on personalized tags that s/he follows.

Our main contributions are: a new approach for an event search platform using machine learning; integration of LODE ontology to structure event data and use it on classification; and classification tests with 101,121 events with 83.33% of classified events.

The remainder of this paper is organized as follows. Section II describes our event search platform and its architecture. Section III describes the process of organization data with the LODE ontology. Section IV describes the algorithms used and the experimental tests for events classification. Finally, Section V concludes the paper and presents future work.

II. THE EVENT SEARCH PLATFORM

Finding digital content related to events is challenging, requiring searching at different sources and sites [5] and sometimes, the data is ambiguous and incomplete.

A. The idea

The goal is to create an event search platform where every event can be classified with several tags. A good similarity is for example the Foursquare application [6]. Each place is associated to multiple tags, e.g., a restaurant can be associated to pasta, cocktails, pizza and, others, depending on their service type.

Our idea is to take advantage of these tags system and apply it on an event search platform. For example, a Bruce Springsteen concert [7] may be associated with tags like rock, hard rock or folk. Merging these two concepts (events and tags) can bring some advantages, such as:

- The platform can accommodate not only predefined events with selected tags, but all kind of events. For example, we can have one event related with music and one event related with a scheduled construction work on a specific street;
- Creation of customized lists according to the user's preferences;
- Creation of a more personalized search engine to return more accurate events to the user;
- Better interactivity with users, allowing them to create and classify events with tags. If a tag does not exist, the user can create the tag at the time of creating the event, allowing the system cover all type of events with the user input. As a business rule, each event should have at least one tag.

Machine learning is used for events classification to bring more improvements in the recommendation and search of events, as well as on the notification of events. Its main goal is to classify events obtained from APIs in several tags, but it can also help make the system more personalized to the user. For the platform, we can add a new feature like the suggestion of tags in the process of creating an event. For example, if a tag is followed by 1000 users, the suggestion of this tag at the time the event is created, can reach a larger number of people who might be interested in participating in it.

Yet, there is some concern about allowing users to create their events as well as classify them. This feature may lead to

DOI Reference Number: 10.18293/SEKE2017-159

inconsistent data and may have repercussions on the events classification. To solve this problem, we can use the recommendation system proposed in [8]: at the time of creating the event, the platform recommends a series of tags that can be used to classify the event depending on its data. If it is necessary to create a new tag, the submitted event must go through an approval process, to verify that the tags created are related with the event. This way, we think that it is possible to solve the issue of data inconsistency generated by the user.

B. Proposed architecture

Figure 1 shows the architecture of our platform and how its components communicate between them. Next, we will describe every component and its main function.

Client Applications are the applications that allow the user interact with the system and view the lists of events as well as create their own events. These applications will communicate with the server through the developed API, which is based on HTTPS (Hypertext Transfer Protocol Secure) protocol. The data sent is in JSON (JavaScript Object Notation) format and consists of event data.

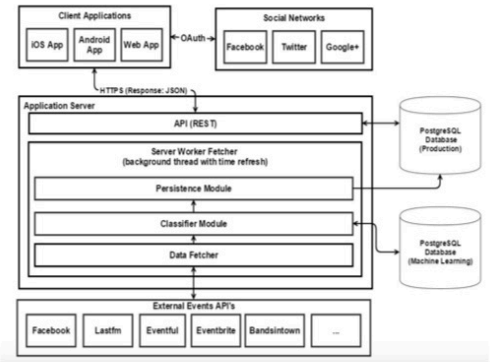


Figure 1. Proposed architecture

Application Server is the first responsible for supporting client applications, containing a RESTful API [9] to handle the requests about events. There is also a module for managing event classification (Server Worker Fetcher).

In order to save the data, the current architecture provides two databases. The PostgreSQL (Production) database will be the database that stores all event data already classified and used by the applications described above. The PostgreSQL (Machine Learning) is a copy of PostgreSQL (Production) database and will only be used as training base of the algorithm to classify the events coming from external APIs. This database will be updated periodically to improve event classification.

Server Worker Fetcher is a server worker whose main function is to get and classify events. It is divided into three modules: Data Fetcher module handles the communication between external APIs and the server to get events data; Classifier module handles the classification of event data through a machine learning algorithm. The PostgreSQL (Machine Learning) database provides the data for classification; Persistence Module stores event data already classified into the PostgreSQL (Production) database.

Finally, the external APIs are the APIs responsible to provide events.

III. EVENT DATA WITH LODE ONTOLOGY

One of the main contributions of this work is to show a different approach of an event search platform using machine learning. This section aims to present the process of integrating the LODE ontology to structure the event data and use them in the classification. A comparative study of several APIs was carried out, to understand which entities are similar between them.

The purpose of LODE ontology is to enable interoperable modelling of factual aspects to encapsulate the most useful properties to describe events [3]. The goal is to give answers about *"What is happening?"*, *"Where it is happening?"*, *"When it is happening?"*, *"Who is involved?"* [3], and organize this information in several properties, which are *Event*, *atPlace*, *atTime* and, *involved*.

Events often need a response from the user. This response is called R.S.V.P, which means "Répondez S'ill Vous Plaît" in French. This data permits to know if whether users will attend the event. This status is represented in several users counts that can be subdivided into the following categories:

- Attending guests: represents the guests that will attend the event;
- Declined guests: represents the guests that won't be attending the event;
- Interested guests: represents the guests that have interest in the event but don't know if they will be attending;
- No reply guests: guests who didn't reply to the invite;
- Maybe guests: guests that maybe will attend the event.

The final attributes of our dataset can be seen on Table I:

TABLE I. *Attributes of our dataset*

Properties	**Attributes**
atPlace	venue_latitude
	venue_longitude
atTime	event_start_hour
	event_end_hour
	event_start_day_of_month
	event_end_day_of_month
inSpace	venue_id
involved	artist_id
social	event_attending_count
	event_declined_count
	event_interested_count
	event_noreply_count
	event_maybe_count

IV. EXPERIMENTAL EVALUATION

In the experimental evaluation, we intend to find the best classification result in order to validate events classification for only one tag.

A. Algorithms

We use the following algorithms provided by *Weka,* corresponding to different classification categories: Decision Trees, was chosen the Random Forest [10], for the lazy classifiers, the K-Nearest Neighbors [11] was chosen, whose implementation in Weka is named IBk and, for function classifiers, Sequential Minimal Optimization (SMO) [12] was chosen, an algorithm for training support vector machines.

B. Classification results and discussion

In order to perform these experiments, it was necessary to create two datasets. The first dataset has about 1,121 events sourced from Facebook. The second dataset is a generated dataset with about 100,000 events.

The tests performed in this work are evaluated using the correctly classified instances. The 10-fold cross validation test mode was used, which means that 90% of the data is used for training and 10% for testing in each fold test.

The first test aimed to get the classification results for both datasets, to understand the first results, without changing the data as well as the algorithms. Table II shows the difference between the results obtained for the dataset with Facebook events in relation to the randomly dataset.

TABLE II. *Results of the first classification test*

	% of correctly classified instances	
Algorithms	***Facebook Dataset***	***Randomly Dataset***
IBk	50.02%	100%
SMO	46.67%	100%
Random Forest	70.28%	100%

For the IBk and SMO algorithms, the difference is around 50% and for the Random Forest algorithm the difference is around 30%. These differences are related to some missing values in the Facebook dataset. Since the dataset is composed by numeric data, the APIs do not always return all data to the attributes, leading to missing values. These same values are represented as zero, which on our view, affects the classification of events. There are three approaches to lead with missing values: mark, impute and remove missing values.

The technique to mark missing values aims to change the missing data that will be represented as "?". Yet, instances with missing values do not have to be removed and we can replace the missing values with other values with the mean of the numerical distribution. In order to have also missing data on our generated dataset, we created another one and we did the same tests for this new dataset. The results for these two techniques described above can be seen on Table III.

Comparing the results of Table II with Table III for the generated dataset, we can conclude that adding values missing also made the results worse. It is clear that both approaches cannot be taken into account in the classification process.

The last approach to deal with missing values is to remove events that contain one or more attributes with missing data. Considering the results of the previous Table II and III, we chose Feature Selection to understand which attributes are the most useful or relevant to our scenario. This is important because the number of attributes used can make the work of the classifier more difficult, making it slower and even diminishing accuracy.

TABLE III. *Classification results for mark and impute missing values techniques*

		% of correctly classified instances	
Algorithms	**Technique**	***Facebook Dataset***	***Randomly Dataset***
IBk	Mark Missing Values	41.60%	62.09%
	Impute Missing Values	48.12%	68.88%
SMO	Mark Missing Values	43.86%	77.96%
	Impute Missing Values	43.89%	76.33%
Random Forest	Mark Missing Values	61.54%	70.45%
	Impute Missing Values	68.89%	79.44%

Feature selection method aids to create an accurate predictive model. They help choose features that will give good or better accuracy whilst requiring less data [13]. They can be used to identify and remove irrelevant or redundant attributes from data that do not contribute to the accuracy of a predictive model or can decrease the accuracy.

Many feature selection techniques are supported in Weka. We choose the Information Gain Based Feature Selection, a popular technique to calculate the information gain based on the entropy concept. It is used as a measure of feature relevance in filter strategies that evaluate a feature individually [14]. We can calculate the information gain for each attribute for the output variable. Entry values vary from 0 (no information) to 1 (maximum information). Those attributes with more information will have a higher information gain than the others. Since the Facebook dataset represents the actual data of our platform, we only applied this technique on this dataset to understand the most relevant attributes. Table IV only shows the attributes that have a contribution for our case.

TABLE IV. *Attributes contribution gain results*

Attributes	**Information Gain**
artist_id	0.8864
event_start_hour	0.4246
event_end_day_of_month	0.4246
venue_longitude	0.3639
event_maybe_count	0.1003
event_interested_count	0.0600
event_attending_count	0.0423
venue_id	0.0403

We used an arbitrary cut-off of 0, which means that the attributes with this value were removed from the dataset. We proceeded again to the classification tests with the changes made on the dataset. The results can be seen on Table V.

Table V shows a great improvement comparing with results of Table II. Random Forest increased 13.05%, IBk increased 27.02%, and SMO increased 23.07%. This feature selection

showed that we have a lot of irrelevant attributes making the classifiers slower and even in some cases diminishing its accuracy.

TABLE V. *Classification results after apply the Information Gain Feature Selection*

Algorithms	% of correctly classified instances
IBk	77.74%
SMO	69.74%
Random Forest	83.33%

In conclusion, given the large difference in the results between Table II and Table III, compared with Table V, it is possible to verify that one of the problems of our dataset and the unsatisfactory results of the first two tests are related with the missing data. However, with the use of Information Gain feature selection technique, when classifying with only the most relevant attributes of our dataset, even with missing data, the results have risen considerably. In addition, within the relevant attributes it is possible to observe that 3 attributes are related with R.S.V.P, confirming that they bring relevant data in the classification of events.

In a first phase, feature selection needs to be applied since it allows to remove immediately the redundancy and irrelevance of some attributes. Even for a large database with 100,000 events, if we don't have missing data, the results are very good, as shown in Table II, but if we add missing data the results are worst. In this case, techniques of remove missing values should be applied, to understand the impact of these missing data in the dataset.

V. Conclusions and Future Work

We propose an event search platform with a new architecture using machine learning. The use of machine learning aims to classify events in a specific tag. Our idea takes advantage of a tags system to agglomerate, not only predefined events on some categories, but all kind of events. Other advantages of our proposal, are to create customized data according to the user's preferences and a better interactivity between the users and events.

The LODE ontology was used to organize the data obtained from external APIs and was made an experimental study to find the best classification result and algorithm to validate the addition of machine learning on the architecture proposed. For performing these experiments, it was necessary to create two datasets: the first dataset has about 1,121 events sourced from Facebook and the second dataset is a generated dataset with about 100,000 events.

From the three algorithms used (Random Forest, IBk, and SMO), the first results weren't satisfactory for the Facebook dataset. The best result was 70.28% for the Random Forest algorithm. But, for the generated dataset, the results were good, reaching 100% of classified instances.

From the experimental tests, it was verified that sometimes the APIs return missing values which leads to a poor classification of the algorithms. Using the feature selection technique, we came to the conclusion that certain attributes of the Facebook dataset were irrelevant. After being eliminated, Random Forest obtained the best classification result, reaching 83.33% of classified instances. Comparing the results of the generated dataset in the beginning of tests with this result, it is possible to conclude that our training data can't have missing values because, the algorithms performance is worst

Although the classification result (83.33%) was good, there are open issues that we will be performed as future work. The experimental dataset has a small event base, so it is necessary to have more events to confirm the results obtained in these tests. With more events, other techniques of feature selection, such as, learner feature selection or correlation feature selection, must be considered, to understand the data generated in the dataset, to find a pattern that allows obtaining the best percentage for the classification of events.

All these results prove that the proposed platform is viable. Yet, allowing users to create and sort their events, the ambiguity and inconsistency of the data may be a problem in the future. Despite the proposals presented in this paper to solve the problem, they must be validated. Also, it is also necessary to find a solution to merge the data coming from external APIs, since each API has its own data structure. These issues lead us to other relevant issues about the performance, such as: the time that takes to build our training base and prepare the data for classifications, the performance of a classification procedure and the combination of data among the external APIs.

References

[1] Costa-Dasilva, Ignacio, J., Gómez-Rodríguez, A., González-Moreno, J.C. and Ramos-Valcárcel, D. (2015) 'A located and user personalized event's dissemination platform', Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 28(1), pp. 71–81.

[2] Baruah, T.D. (2012), "Effectiveness of Social Media as a tool of communication and its potential for technology enabled connections: A micro-level study", International Journal of Scientific and Research Publications, Volume 2, Issue 5.

[3] Shaw, R., Troncy, R. and Hardman, L. (2009) 'LODE: Linking open descriptions of events', in Lecture Notes in Computer Science. Springer Nature, pp. 153–167.

[4] Troncy, R., Fialho, A., EURECOM, Hardman, L. and Saathoff, C. (2010) 'Experiencing events through user-generated media'

[5] Girolami, M., Chessa, S. and Caruso, A. (2015) 'On service discovery in mobile social networks', Computer Networks: The International Journal of Computer and Telecommunications Networking, 88(C), pp. 51–71.

[6] Made, L. and SFfourSq (2016) Food, Nightlife, entertainment. Available at: https://foursquare.com/ (Accessed: 25 November 2016).

[7] Springsteen, B. (no date) Official Website. Available at: http:// brucespringsteen.net (Accessed: 22 November 2016).

[8] Ricci, F., Rokach, L. and Shapira, B. (eds.) (2015) Recommender systems handbook. Springer Nature.

[9] Garriga, M., Mateos, C., Flores, A., Cechich, A. and Zunino, A. (2016) 'RESTful service composition at a glance: A survey', Journal of Network and Computer Applications, 60, pp. 32–53.

[10] Breiman, L. (2001) Machine Learning, 45(1), pp. 5–32.

[11] Aha, D. W., Kibler, D. and Albert, M. K. (1991) 'Instance-based learning algorithms', Machine Learning, 6(1), pp. 37–66.

[12] Cohen, (1995), "Fast Effective Rule Induction," In Proceedings of the Twelfth International Conference on Machine Learning.

[13] Guyon, I. and Elisseeff, A. (2003) 'An introduction to variable and feature selection', The Journal of Machine Learning Research, 3, pp. 1157–1182.

[14] Yang, Y. and Pedersen, J. O. A Comparative Study on Feature Selection in Text Categorization. In Proceedings of the 14th International Conference on Machine Learning. San Francisco, CA, USA, pp. 412–420, 1997

Anexo B – LODSE: A new ontology for social event classification

Artigo em revisão para ser posteriormente ser submetido.

LODSE: a new ontology for social event classification

Marcelo Aguiar Rodrigues
Polytechnic of Coimbra
ISEC
Coimbra, Portugal
a21180873@isec.pt

Rodrigo Rocha Silva
São Paulo State Technological College
CISUC
Mogi das Cruzes, Brazil
rodrigo.rsilva@fatec.sp.gov.br

Jorge Bernardino
Polytechnic of Coimbra
CISUC
Coimbra, Portugal
jorge@isec.pt

***Abstract*—A large number of social events is hosted on platforms like Facebook. With the massive quantity of information created in these systems, finding events is challenging and sometimes the data is ambiguous or incomplete. One of the main challenges in social event classification is related to the incompleteness and ambiguity of metadata created by users. This paper presents a new ontology, named LODSE (Linking Open Descriptions of Social Events) based on the LODE (Linking Open Descriptions of Events) ontology to describe the domain model of social events. The aim of this ontology is to create a data model that allows defining the most important properties to describe a social event. The data model created will be used in an experimental evaluation to compare both ontologies in social event classification. The experimental evaluation for LODSE ontology has shown a better percentage of correctly classified events and also improvements in the performance, namely a low memory consumption and faster runtime. The results demonstrate an increment of 12.4% of correctly classified events as well as an average runtime gain of 5.9%.**

Keywords: social events; social event classification; ontologies; machine learning; random forest.

I. INTRODUCTION

Social networks are extensively used because they have a preponderant role that goes beyond the conventional means in the dissemination and promotion of content. Social events are no exception and with the popularity of social networks, there is an increase of events in these systems. Facebook is the largest social network with 1.79 billion monthly active users [1]. Facebook Event is a powerful tool and it is used to create and promote events, but they are usually agglomerated with other types of information and users rarely notice them [2].

An application focusing only on dissemination and recommendation of events could solve this problem effectively [2]. In fact, it is difficult to search events of our interest [3], and when we move to a new city or country, this necessity is particularly important because it is hard to know what is happening around us. Furthermore, it is difficult to get reliable information about a particular event or to get recommendations of events based on our interests. Finding digital content related to events is challenging, requiring searching at different sources and sites [3] and most of the data is ambiguous and incomplete.

One of the major challenges in social event classification is the incompleteness and ambiguity of metadata created by users [4]. If we have ambiguity in the data, the classification process can lead to a bad categorization of events where the percentage of correctly classified events may be small. Consequently, classifying events for a large number of tags, the classifier can have performance issues, specifically a slow runtime or higher memory consumption.

In this paper, we propose a different approach for social event classification by creating a new ontology to define the data model to be used in the classification process. This ontology is designated as LODSE (Linking Open Descriptions of Social Events) and it is based on LODE ontology. The LODE ontology creates a model that allows for encapsulating the most important properties to describe events [5]. The aim of LODSE ontology is to create a data model that allows defining the most important properties to describe a social event. The LODSE ontology was created to help the process of classifying events because using a greater knowledge of social events domain may bring advantages in structuring a good data model to obtain better results in social event classification.

An experimental evaluation was made, aiming to compare the performance of Random Forest algorithm when using a data model based on LODE ontology and LODSE ontology. An analysis of the results is presented focusing on the percentage of correctly classified events, memory consumption, and runtime as well.

The main contributions of this work are the following:

- A new ontology LODSE is proposed, which deals with social events;
- A new approach for social event classification;
- LODSE ontology increments the percentage of correctly classified events as well as the execution time when compared to LODE ontology.

The rest of this paper is organized as follows. Section II presents related works, focusing on social event classification. Section III describes the LODSE ontology, its classes, properties and presents a comparison with LODE ontology. Section IV describes the experimental setup used in the experimental evaluation. Section V presents the results of the experimental evaluation and discussion. Finally, Section VI concludes the paper and presents future work.

II. RELATED WORK

Social networks have opened up a new space for information exchange and expression of public opinion. Not only did they bring significant changes in the paradigm of public opinion, but they also became a driving force to promote social change [6].

Social networks and search engines are the most used services these days. Data generated from these systems have

a great value because they reflect several aspects of today's society [7]. Additionally, data is easily accessed through public Application Programming Interfaces (APIs) allowing the creation of more customized systems targeted to a specific area.

Social event classification is one of the areas that has attracted more attention in recent years [4] due to the amount of data on social networks and the availability of public datasets. Some methods have been proposed for social event classification based on textual metadata.

In [8], the authors participated in a semi-supervised clustering task as well as the classification of social events. For the classification task, they used popular classifiers such as k-Nearest Neighbour (kNN), Decision Trees and Random Forest with Latent Dirichlet Allocation as feature selection. Taking into account that this work focuses more on the clustering task, the results obtained for the classification task only indicate that is needed attention on the imbalance categories distributions.

Still related to social event classification based on textual metadata, in [9], the authors studied event detection on social networks, particularly in Facebook and Twitter considering textual metadata to identify events. They formulate their approach as a structured graphical model which simultaneously analyses individual messages and induces a canonical value for each event property. They have applied their technique to create a city calendar with entertainment events and their method shows up to 63% recall and up to 85% precision evaluated manually.

With the popularity of social networks like Instagram, the task of social event classification also tries to classify events based on visual information. In [10], the authors propose a method to classify social event images based on a framework that they developed. The framework consists of three stages, which includes pre-processing, filtering, and clustering & classification. The classification task was performed with the help of WordNet which was combined with textual information from a given dataset. They achieved an F1 main score of 0.4409. The presented method was constrained only towards the analysis of media objects rather than textual information.

Event classification can also be used for recommendation purposes. In [2], is presented a method for event recommendation divided into two stages – classification and re-ranking. For the first phase, they used a blending model of probabilistic classifiers to predict positive and negative probabilities for each user-event pair, and then they evaluate on those results to eliminate all bad cases before passing to the next phase. The classification phase showed that most of the bad cases were eliminated by assessing directly on the negative probabilities by an optimized threshold.

In [20], the authors propose an evaluation measure for the performance assessment of multiannotation classification systems incorporating ontology knowledge. A distance-based misclassification cost was extended from the unilabel to the multilabel case and further enriched with ontology information like its hierarchy, an annotation agreement factor and penalties for ignoring relationships. Despite the differences between this work and our proposal, this paper allows us to perceive that an ontology knowledge can bring advantages in the process of classification.

All these papers are focused on the classification of events but with different purposes. It is also noted that the data normally used in the classification process is textual data of nominal type (e.g. string). Our work presents a different approach for social events classification, where an ontology is created to generate the data model and the classification of events is based only on numerical data.

III. THE LODSE ONTOLOGY FOR SOCIAL EVENTS

This section presents LODSE ontology to represent the domain model of social events. It describes the classes of the ontology, their relationships and the properties of the classes.

An ontology is a data model that represents a set of concepts within a domain and it is used to perform inference on the objects of that domain. The ontologies are used in several areas such as artificial intelligence, or software engineering [11] as a way of representing knowledge about the world or part of it. An ontology usually describes:

- Individuals – the basic objects;
- Classes – sets, collections or types of objects;
- Attributes – properties, characteristics, or parameters that objects may have to share;
- Relationships – between objects.

There are many methods to create an ontology with different results. In the next subsections are introduced the main features of LODSE ontology based on the guide [12], which presents ontology-design concepts.

A. *The LODE ontology*

According to [5], the purpose of the LODE ontology is to create a model that allows encapsulating the most important properties to describe events. Their goal is to answer questions such as:

- What is happening?
- Where is it happening?
- When is it happening?
- Who is involved?

These questions provide a data model organization into the following properties: *Event*, *atPlace*, *atTime*, and *involved* and they were reused to be used in LODSE ontology.

The reason for choosing the LODE ontology is due to its structure that has the most important properties of an event [5]. Since it does not have any class that takes into account the categorization of an event, there was a need to create a new ontology that provides a better representation of a social event.

B. *The domain and scope of LODSE ontology*

The LODSE ontology aims to cover the domain of social events, more precisely, music, sports, performing arts, conferences, among other types of events. The purpose of this ontology is to create a model that allows defining the most important properties to describe a social event to achieve better results in the task of social event classification.

This ontology will be used to improve the classification of events and also its recommendation by defining the most important classes and properties of a social event. It will also

allow the creation of a generic data model that can be used by several applications/platforms for an easier integration of data, obtained from different services.

According to [13], one method to define the scope of the ontology is to create a list of questions that it should be able to answer. These questions serve as a validation test to see if the ontology contains enough information to represent the domain of social events. The following questions were defined to understand if the ontology represents the scope of social events:

- What event is it?
- What is the name of the event?
- Who is the artist?
- Who is the organizer?
- Where will the event occur?
- What time is the event?
- What kind of event is it?

C. *The important terms of the ontology*

It is important to get a general list of terms that represent the domain of the ontology without worrying with the overlap between concepts they represent, relations among the terms or any properties that concepts may have.

Developing this list of terms that correspond to the domain is one of the advantages cited in [12] because it will help to define the classes and the properties of the ontology.

The terms defined for the LODSE ontology are Event, Date, Tag, Organization, Artist, Tickets, Venue, Price, Category, and Friends.

D. *The classes and the class hierarchy*

There are three methods to develop the hierarchical classes of an ontology [14] which are: top-down, bottom-up and combination. We used the top-down method that aims to develop the ontology, starting from the most general concepts of the domain and subsequent specialization of the concepts.

In the previous subsection, a list of terms was defined, that can help create the classes of the ontology and answer the questions presented in subsection B. The classes that represent the ontology are the following:

- **Event** – a class that describes an event and answers the questions "*What event is it?*" and "*What is the name of the event?*";
- **Involved** – a class that describes who is involved in the event and answers the questions "*Who is the artist?*" and "*Who is the organizer?*";
  - Artist – a subclass describing the artist of the event;
  - Organization – a subclass describing the organizer of the event;
- **Date** – a class that represents the date of the event and answers the question "*What time is the event?*";
  - startDate – a subclass representing the start date of the event;
  - endDate – a subclass representing the end date of the event;
- **Venue** – a class that describes the place where the event will take place and answers the question "*Where will the event occur?*";
  - City – a subclass describing the city where the event will take place;
  - Country – a subclass describing the country where the event will take place;
- **Taxonomy** – a class that represents the categorization of an event and answers the question "*What kind of event is it?*"
  - Tag – a subclass representing the event tag;
  - Category – a subclass representing the category of the event;

Fig. 1 shows how LODSE ontology was created based on LODE ontology, the classes that compose the LODSE ontology, subclasses and the relationships between them. The classes Event, Venue, Date and Involved are the classes imported from the LODE ontology and the other classes are new and were created for the LODSE ontology.

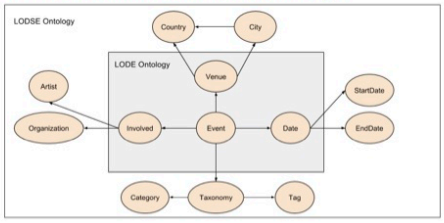


Figure 1. A comparison between LODE ontology and LODSE ontology

Fig. 1 also shows the relations between the classes. It is possible to observe that the Event class is the main class of LODSE ontology, from which all existing relations start. The relationships of the classes are explained as follows:

- *Event – Date:* All events occur on a certain date. The event can have a start and also an end date;
- *Event – Venue*: All events take place at a particular venue. This venue is located in a city/country and the city belongs to a country;
- *Event – Involved*: Every event has someone involved. Depending on the type of the event the entities that may be involved are the artists or the event organizers;
- *Event – Taxonomy*: The event belongs to a certain taxonomy, this is, the event is classified or with a pre-defined category or with a tag.

E. *The properties of classes and their facets*

Classes are the focus of most ontologies and they describe concepts in the domain. In LODSE ontology, the class *Event* represents all social events. A class can have subclasses that represent concepts that are more specific than the superclass. For example, we can divide the class of all social events into music events because of the *Taxonomy* subclass.

Classes by themselves do not provide enough information to answer the questions presented in subsection B. Since

classes are already defined, we must describe the internal structure of concepts, the properties.

The properties of the classes of the LODSE ontology were chosen based on the classes previously defined, on the questions that the ontology intends to answer listed in subsection B and also from an analysis of several event APIs such as Facebook, Eventful, Eventbrite and Meetup to perceive the common properties between these services.

Properties can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the property. The value type turns out to be the most important facet in the development of an ontology because it defines the type of each property that will define the properties used in the classification process. The values that the properties can take are number, string, boolean, enumerated and instance.

Table I shows all classes, properties, and facets of the LODSE ontology.

TABLE I. CLASSES, PROPERTIES AND FACETS OF LODSE ONTOLOGY

Class	Properties	Facets
Event	eventID	number
	eventName	string
	eventDescription	string
	eventPrice	number
	eventURL	string
	eventDateCreated	date
	eventDateModified	date
Involved	involvedName	string
	involvedDescirption	string
	involvedOfficialWebsite	string
Artist	artistID	number
Organization	organizationID	number
Date StartDate EndDate	date	date
	time	date
	allDay	boolean
Venue	venueID	number
	venueName	string
	venueDescription	string
	venueLatitude	number
	venueLongitude	number
	venueCapacity	number
	venuePostalCode	string
City	cityID	number
Country	countryID	number
Taxonomy	name	string
Category	categoryID	number
Tag	tagID	number

After defining the classes, the properties of the classes and their facets, we have all the information needed to be used in the experimental setup, presented in the next section, to create the datasets to be used in the experimental evaluation.

IV. EXPERIMENTAL SETUP

This section presents the experimental setup used in the experimental evaluation that aims to compare the performance of Random Forest algorithm when using a data model based on LODE ontology and LODSE ontology. We intend to get a better percentage of correctly classified events and also improve the performance in the classification process, in order to have low memory consumption and faster runtime with the LODSE ontology comparing with LODE ontology.

We start to present the hardware and software used to run the classification tests. Next, we present the algorithm Random Forest and we describe the structure of our datasets.

A. The hardware

To perform the experimental evaluation, two machines with the following characteristics were used:

- Machine_1 – Processor 1.4GHz Intel Core i5, 8GB RAM DD3
- Machine_2 – Intel Xeon Processor 2.39GHz, 40GB RAM

Since the first machine used did not have enough memory to perform all the tests during the experimental evaluation, a second machine was added to perform the tests for numbers of events greater than 10200.

B. The software – Weka

The Waikito Environment for Knowledge Analysis (Weka) [15] is a data mining software in Java, currently in version 8 and it consists of a collection of machine learning algorithms for data mining tasks.

We integrated the Weka library in a Java application, developed by us, to get the percentage of correctly classified events and also to measure the memory consumption and runtime. We have chosen this software because it is open source and has interesting features that allow to test our ideas easily. Moreover, Weka is one of the most used software in data mining and has made an outstanding contribution to the data mining field [16].

C. The algorithm – Random Forest

Weka allows to choose several algorithms of data mining to perform classification tests in our data. We choose the Random Forest algorithm to classify a dataset of events for a certain tag.

We have chosen the Random Forest because it is a method of ensemble learning widely used in the literature and applied field [18].

The Random Forest [17] algorithm consists of a combination of classification trees, in which each tree is dependent on the values of an independently generated random vector and with the same distribution for all trees in the forest. After a large set of trees is generated, each will vote for a class, and the class that wins will be chosen.

D. The datasets

The dataset files created for the experimental evaluation are ARFF files which contain the event data and the tags to be used in the classification tests. ARFF files have two different sections. The first section is the header information which is

followed by the data in the second section. The header of an ARFF file contains the name of the relation, the list of attributes (the columns in the data) and their types. This section defines the structure of our data and it will represent the properties of a social event. The second section represents the data and has the event data for the properties defined in the header information.

Two types of datasets were created and they differ in the list of attributes defined in the header of an ARFF file. The first type of datasets is relative to the data model based on the LODE ontology and the second type of datasets is relative to the data model based on the LODSE ontology.

Table II presents the attributes of the data model based on the LODE ontology. These attributes are the most relevant ones and they obtained 83.33% of correctly classified events in [19].

TABLE II. ATTRIBUTES FOR THE DATA MODEL BASED ON THE LODE ONTOLOGY

Attributes	Type
artist_id	Numeric
event_start_hour	Numeric
event_end_day_of_month	Numeric
event_maybe_count	Numeric
event_interested_count	Numeric
event_attending_count	Numeric
venue_id	Numeric
venue_longitude	Numeric

Table III presents the attributes of the data model based on the LODSE ontology. Considering that the algorithm chosen to perform the tests is the Random Forest, the attributes of the data model are all the properties of the ontology with value type numeric, boolean or enumeration because Random Forest does not support nominal values like strings.

TABLE III. ATTRIBUTES FOR THE DATA MODEL BASED ON LODSE ONTOLOGY

Attributes	Type
artist_id	Numeric
category_id	Numeric
event_start_hour	Numeric
event_end_hour	Numeric
event_start_day_of_month	Numeric
event_end_day_of_month	Numeric
event_month	Numeric
date_all_day	Boolean
event_price	Numeric
organisation_id	Numeric
venue_id	Numeric
venue_latitude	Numeric
venue_longitude	Numeric
city_id	Numeric
country_id	Numeric

The data of the events were generated from an algorithm developed by us. Since no public datasets were found to meet the type of data that our experimental evaluation needs, this algorithm was developed to generate all event data based on the attributes of Tables II and III and the event tags. It is important to note that in data generation there is 30% of events in all datasets that contain missing values. This choice is intended to simulate a more real-world environment of data that multiple services can return, including events with missing data.

The number of events generated ranges from 2040 events to 51000 events. For each specific number of events several ARFF files were created and they differ in the number of tags. The minimum number of tags is 6 and the maximum is 96. In the context of an ARFF file, the tags represent the *@class* value and are the values at which an event can be classified.

We generated 9 sets of events (2040, 4080, 6120, 8160, 10200, 20400, 30600, 40800, and 51000) and each specific set has a file correspondent to 6, 12, 18, 24, 30 and 96 tags. Multiplied by two ontologies we get the total number of datasets, 108 and they will be used in the experimental evaluation.

The number of tags was created based on multiples of 6. From 6 to 30, the number of tags is linear, but then the number of tags was increased to 96. The reason for this difference is to prove if there is a linear behaviour, increasing the number of tags and to know the variation in performance between the two types of datasets when we have a large number of tags.

For each specific dataset, five tests were performed. With 108 datasets created, the experimental evaluation counted on 540 tests (108 datasets * 5 tests for each). The reason for conducting 5 tests for the same dataset is to obtain a more accurate average of the percentage of correctly classified events, memory consumption and runtime. The results for the measured properties are presented in the next section.

V. EXPERIMENTAL EVALUATION RESULTS

As previously mentioned, with this experimental evaluation we aim to get a better percentage of correctly classified events and also improve the performance, in order to have low memory consumption and faster runtime with the LODSE ontology comparing with the LODE ontology.

The percentage of correctly classified events refers to how many instances were correctly classified in an ARFF file. In a dataset with 100 events, if 50 instances were correctly classified, the percentage of correctly classified events is 50%. The memory consumption, measured in megabytes during the classification process, refers to the memory consumption to build the data model and the classification of the instances of a dataset. The runtime, measured in seconds during the classification process, refers to the time to build the data model and the classification time of the instances of a dataset.

This section shows the results obtained on the classification tests after performing the 540 tests in approximately twenty-two days and twelve hours (CPU time) for the percentage of correctly classified instances, memory consumption, and runtime where the number of tags was 6, 30 and 96.

A. Percentage of correctly classified instances

The percentage of correctly classified instances is the percentage of the instances that were correctly classified in an ARFF file.

Fig. 2 shows the evolution of the results for the percentage of correctly classified instances where the number of tags is equal to 6, 30 and 96. Overall, the LODSE ontology has a better percentage of correctly classified events comparing to the LODE ontology. On average, the LODSE ontology has:

- 12.78% more correctly classified events when the number of tags was 6;
- 17.31% more correctly classified events when the number of tags was 30;
- 7.12% more correctly classified events when the number of tags was 96.

In Fig. 2, we can also see that the percentage of correctly classified events reduces as the number of tags increases. This percentage reduction is justified with the increase of tags for the same number of events because we incremented the number of tags but the number of events was always the same. The percentage of events per tag decreased which worsened the percentage of correctly classified events for a high number of tags, e.g. 96 tags.

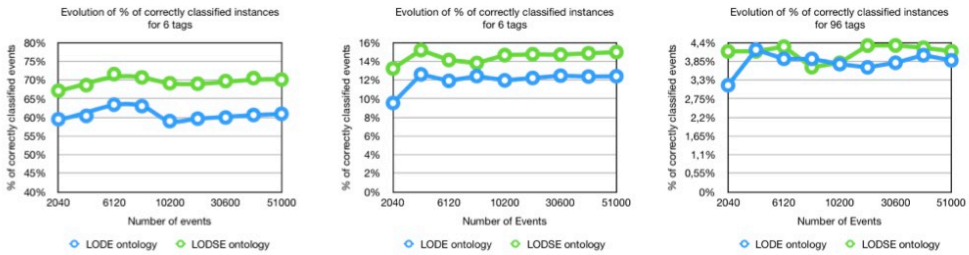


Figure 2. Evolution of % of correctly classified instances for 6, 30, and 96 tags

B. Memory Consumption

The memory consumption is measured in megabytes and analyses the memory used in the classification process, which means, the memory used to build the data model and the classification of the instances of an ARFF file.

Fig. 3 shows the evolution of memory consumption where the number of tags is equal to 6, 30, and 96. Overall, the LODSE ontology has a higher consumption than the LODE ontology. On average, the LODSE ontology consumes:

- 46.34% more memory than the LODE ontology when the number of tags was 6;
- 37.20% more memory than the LODE ontology when the number of tags was 30;
- 0.44% less memory than the LODE ontology when the number of tags was 96.

The LODSE ontology consumes more memory than the LODE ontology because of the number of trees needed to construct the model to be used in the classification process. However, when the number of tags increases, the memory consumption average is reduced.

The Random Forest algorithm constructs its trees based on the defined attributes, classes, and instances. According to [18], as the number of trees grows, it does not always mean better performance compared to fewer trees. Therefore, the memory consumption is lower for the LODE ontology because the algorithm has built fewer trees to be able to classify the instances while the LODSE ontology with more attributes had a higher memory consumption due to the need to build more trees.

It is also worth mentioning that the memory consumption is not linear as the number of events increases as we can see in Fig. [3]. Taking into account that the software used was Weka, the memory management is done by the garbage collector of Java. The garbage collector is a dynamic approach to do automatic memory management and heap allocation that processes and identifies dead memory blocks and reallocates storage for reuse.

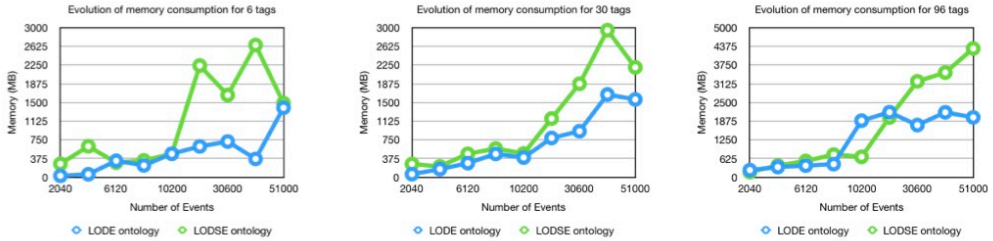


Figure 3. Evolution of memory consumption for 6, 30, and 96 tags

C. *Runtime*

The runtime is measured in seconds and corresponds to the time of a classification test, which means the time to build the data model plus the classification time of the instances of an ARFF file.

Fig. 4 shows the evolution of runtime where the number of tags is 6, 30 and 96. On average, the LODSE ontology took:

- 1.64% more time when the number of tags is 6;
- 7.05% less time when the number of tags is 30;
- 12.28% less time when the number of tags is 96.

Overall, the LODSE ontology takes less time to classify a given dataset than the LODE ontology. We can affirm that as the number of events increases, the runtime also increases because we have more data to be classified. The LODSE ontology had a better runtime when the number of tags was equal to 30 and 96. Otherwise, for a number of tags equal to 6, the LODE ontology had a better runtime but for a smaller difference.

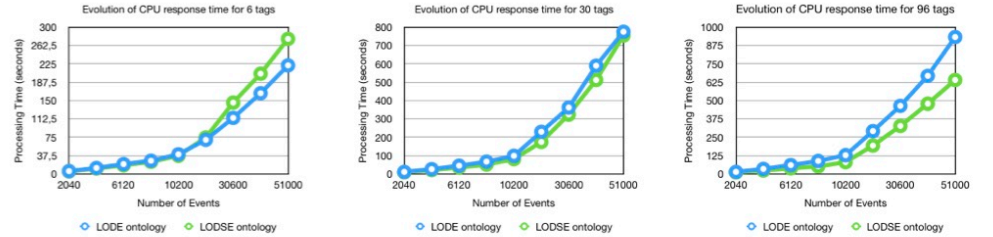


Figure 4. Evolution of runtime for 6, 30, and 96 tags

D. *Discussion of the results*

Overall, the data model created based on the LODSE ontology brings improvements in social event classification obtaining a better percentage of correctly classified instances and taking less time to classify the events, particularly for a higher number of tags and events. For the percentage of correctly classified instances, the best result achieved by LODSE was when the number of tags was equal to 30 obtaining 17.31% more correctly classified instances. For runtime, the best result achieved by LODSE was when the number of tags was equal to 96 taking 12.8% less time to classify events.

However, LODE ontology had a lower memory consumption than the LODSE ontology but, as the number of tags increases, the consumption gap is reduced and for 96 tags, the new ontology used 0.44% less memory than the LODE ontology.

These results validate positively the LODSE ontology and its data model in social event classification. Although the data were randomly generated, the use of more attributes to determine the tag of the event demonstrated improvements in the percentage of correctly classified events. It is also worth mention the choice of attributes of the data model. Since we choose only the numerical properties of the LODSE ontology classes, it becomes easier for the algorithm perceive the tag of an event because one property can be the unique identifier of that class. A practical example can be the association of "rock-music" tag to a specific venue which only has concerts of rock music. For the algorithm, this association can be accomplished because we have an attribute *venueID* that identifies the venue in question.

In terms of performance, LODSE ontology consumed more memory than the LODE ontology. As explained previously, this is related to the memory required to create the predictive model of the Random Forest algorithm. More attributes in the data model can lead to the creation of more trees on the predictive model. Since LODSE ontology has almost the double of attributes than the LODE ontology, the results prove this higher consumption of memory.

When looking at the runtime, the improvements obtained may be related only to the use of numeric attributes, which are the type of attributes in which the Random Forest algorithm performs better.

This experimental evaluation demonstrates that the LODSE ontology brought advantages in social event classification, achieving a better percentage of correctly classified events and a faster time in the classification process compared to LODE ontology. In particular, for a large number of tags, the LODSE ontology also demonstrated better results compared to the LODE ontology. However, the memory consumption was slightly worse because of the higher number of attributes in the LODSE ontology.

VI. Conclusions and Future Work

One of the major challenges in social event classification is related to the incompleteness and ambiguity of metadata generated by users which leads to poor results. Therefore, the classifier can have performance issues for a large number of tags, specifically a slow runtime or higher memory consumption. To solve the mentioned problems, a new ontology was proposed in this paper, named LODSE (Linking Open Descriptions of Social Events) to create a data model that allows defining the most important to describe a social event, based on LODE ontology. The experimental evaluation was performed with 540 tests varying the number of events between 2040 to 51000 and the number of tags between 6 and 96. The aim was to compare the performance between the two data models, based on the presented ontologies, in the classification process. We analysed the percentage of correctly classified events, the memory consumption, as well as the runtime.

From the analysis of results, it is possible to observe that the LODSE ontology brought improvements in social event classification. The tests performed have shown good results, especially in datasets where there is a large number of events and tags. In general, they demonstrated an average gain of 12.40% in the percentage of correctly classified events as well as a mean of gain of 5.89% in runtime. However, memory consumption remained above the LODE ontology due to the memory required to create the predictive model of the Random Forest algorithm. According to the results obtained, we conclude that our approach can be used in social event classification using only numerical data to describe a social event instead of textual metadata. The creation of the LODSE ontology helped to understand what were the most important properties of an event and choose only the numerical ones to be used in the data model resulting in a better percentage of correctly classified events.

Despite the good results, there is a need to validate the same experimental evaluation but with real data. As future work, a new experimental evaluation should be performed with data from Facebook. The data should be in agreement with the data model of LODSE ontology and the same tests should be performed in order to understand if the results are better or not. It should be also analysed the relevance of the attributes and applied the feature selection techniques or performed the classification tests with different algorithms.

References

[1] Barrigas, H., Barrigas, D., Barata, M., Bernardino, J., & Furtado, P. (2015). Scalability of Facebook Architecture . New Contributions in Information Systems and Technologies, 763-772.

[2] Nguyen, D., & Le, T. (2016). Recommendation system for Facebook public events based on probabilistic classification and re-ranking. 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE), (pp. 133-138).

[3] Girolami , M., Chessa, S., & Caruso, A. (September de 2015). On service discovery in mobile social networks. Computer Networks: The International Journal of Computer and Telecommunications Networking, 88(C), 51-71.

[4] Zeppelzauer, M., & Schopfhauser, D. (2016). Multimodal classification of events in social media,. Image and Vision Computing.

[5] Hardman, L., Troncy, R., & Shaw, R. (2010). LODE: An ontology for Linking Open Descriptions of Events. Obtido de LODE: An ontology for Linking Open Descriptions of Events: http://linkedevents.org/ontology/

[6] Dong, T., Liang, C., & Xu, H. (June de 2017). Social media and internet public events . Telematics and Informatics, 726-739.

[7] Panagiotou, N., Katakis, I., & Gunopulos, D. (2016). Detecting Events in Online Social Networks: Definitions, Trends and Challenges. Em Solving Large Scale Learning Tasks. Challenges and Algorithms (Vol. 9580, pp. 42-44).

[8] Sutanto, T., & Nayak, R. (2013). ADMRG @ MediaEval 2013 Social Event Detection. Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop, 1043.

[9] Benson, E., Haghighi, A., & Barzilay, R. (2011). Event discovery in social media feeds. HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 1, pp. 389-398 .

[10] Gupta, I., Gautam,, K., & Chandramouli, K. (2013). Vit @ MediaEval 2013 social event detection task: Semantic structuring of complementary information for clustering events. Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop, 1043.

[11] Guarino, N. (1998). Formal Ontology and Information Systems. Formal Ontology in Information Systems: Proceedings of the 1st International Conference (pp. 3-15). Trento: IOS Press.

[12] Noy, N. F., & Mcguinness, D. (January de 2001). Ontology Development 101: A Guide to Creating Your First Ontology. Knowledge Systems Laboratory, 32.

[13] Grüninger , M., & Fox, M. S. (1995). Methodology for the Design and Evaluation of Ontologies.

[14] Uschold, M., & Gruninger, M. (1996). Ontologies Principles Methods and Applications.

[15] Waikato, M. L. (s.d.). Weka 3: Data Mining Software in Java . Obtido de https://www.cs.waikato.ac.nz/ml/weka/

[16] Russell, I., & Markov, Z. (2006). An Introduction to the Weka Data Mining System. 11th annual SIGCSE conference on Innovation and technology in computer science education, (pp. 367-368).

[17] Breiman, L. (2011). Random Forests. Machine Learning, 45(1), 5-32.

[18] Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (s.d.). How Many Trees in a Random Forest? International Workshop on Machine Learning and Data Mining in Pattern Recognition. 7376, pp. 154-168. Springer.

[19] Rodrigues, M., Silva, R. R., & Bernardino, J. (2017). An Event Search Platform Using Machine Learning. The 29th International Conference on Software Engineering and Knowledge Engineering, (pp. 319-322).

[20] Nowak, Stefanie & Lukashevich, Hanna. (2009). Multilabel Classification Evaluation using Ontology Information. CEUR Workshop Proceedings. 4.

Anexo C – Código fonte da aplicação desenvolvida para classificação de eventos

```
public class EventsClassifier {

	public static void main(String[] args) throws Exception {

		String fileName = "resources/8160 eventos/6 Categorias/Original Dataset - 6 Categories - 
Old.arff";

		System.out.println("FileName: " + fileName);

		BufferedReader reader = new BufferedReader(new FileReader(fileName));

		long startSystemTimeNano = getSystemTime( );
		long startUserTimeNano   = getUserTime( );

		Instances data = new Instances(reader);
		reader.close();

		// Setting class attribute
		data.setClassIndex(data.numAttributes() - 1);

		System.out.println("Num of instances: " + data.numInstances());
		System.out.println("Num of attributes: " + data.numAttributes());
		System.out.println("Num of classes: " + data.numClasses());

		long startTime = System.currentTimeMillis();

		int mb = 1024 * 1024;
		Runtime instance = Runtime.getRuntime();

		classifyWithRandomforest(data);

		System.out.println("\nTime");
		System.out.println("---------------------------\n");

		long estimatedTime = System.currentTimeMillis() - startTime;

		System.out.println("Total Time: " + estimatedTime / 1000);

		System.out.println("\nMemory Results");
		System.out.println("---------------------------\n");

		System.out.println("***** Heap utilization statistics [MB] *****\n");

		// available memory
		System.out.println("Total Memory: " + instance.totalMemory() / mb);

		// free memory
		System.out.println("Free Memory: " + instance.freeMemory() / mb);

		// used memory
		System.out.println("Used Memory: " + (instance.totalMemory() - instance.freeMemory()) / 
mb);

		// Maximum available memory
		System.out.println("Max Memory: " + instance.maxMemory() / mb);

		// Time Results
```

```
                System.out.println("\nTime Results in seconds");
                System.out.println("--------------------------\n");


        //http://nadeausoftware.com/articles/2008/03/java_tip_how_get_cpu_and_user_time_benchmarking#Timin
gasinglethreadedtaskusingCPUsystemandusertime

                long taskUserTimeNano    = getUserTime( ) - startUserTimeNano;
                long taskSystemTimeNano  = getSystemTime( ) - startSystemTimeNano;

                System.out.println("User Time: " + taskUserTimeNano / 1000000000.0);
                System.out.println("System Time: " + taskSystemTimeNano / 1000000000.0);

                System.out.println("------------------------------------------------------------------------
--------------------------");
                System.out.println("------------------------------------------------------------------------
--------------------------\n");
        }

        public static void classifyWithRandomforest(Instances trainInstance) {

                try {

                        System.out.println("\nClassify with Random Forest");
                        System.out.println("--------------------------\n");

                        RandomForest randomForest = new RandomForest();

                        randomForest.setNumIterations(30);

                        System.out.println("Num Iterations: " + randomForest.getNumIterations());

                        // max num iterations by default = 100
                        // max depth = 0 = unlimited

                        randomForest.buildClassifier(trainInstance);
                        showResults(trainInstance, randomForest);

                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        public static void showResults(Instances trainInstance, Classifier classifier) {

                try {

                        System.out.println("\nResults");
                        System.out.println("--------------------------\n");

                        Evaluation evaluation = new Evaluation(trainInstance);
                        evaluation.crossValidateModel(classifier, trainInstance, 10, new Random(1));
                        System.out.println(evaluation.toSummaryString("\nResults\n", true));
                        System.out.println(evaluation.fMeasure(1) + " " + evaluation.precision(1) + " " +
evaluation.recall(1));

                } catch (Exception e) {

                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        /** Get CPU time in nanoseconds. */
        public long getCpuTime( ) {
            ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
            return bean.isCurrentThreadCpuTimeSupported( ) ?
```

```
                bean.getCurrentThreadCpuTime( ) : 0L;
        }

        /** Get user time in nanoseconds. */
        public static long getUserTime( ) {
            ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
            return bean.isCurrentThreadCpuTimeSupported( ) ?
                bean.getCurrentThreadUserTime( ) : 0L;
        }

        /** Get system time in nanoseconds. */
        public static long getSystemTime( ) {
            ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
            return bean.isCurrentThreadCpuTimeSupported( ) ?
                (bean.getCurrentThreadCpuTime( ) - bean.getCurrentThreadUserTime( )) : 0L;
        }
}
```

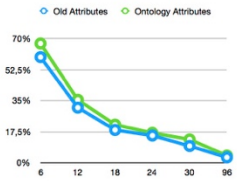
Anexo D - Resultados completos da segunda avaliação experimental

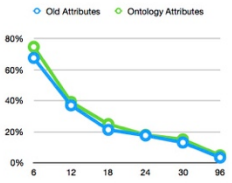
2040 eventos

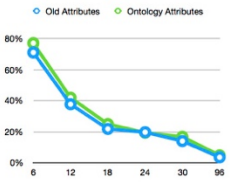
Classificação

% eventos corretamente classificados

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	59,481%	67,1158%	67,4152%	74,5509%	70,9581%	76,9461%
12	31,0878%	35,479%	36,976%	39,1717%	37,7246%	41,9162%
18	18,5629%	21,507%	21,3074%	24,9501%	21,8563%	25%
24	15,4192%	16,9162%	17,6647%	18,014%	19,7605%	19,4112%
30	9,5309%	13,2236%	13,1238%	15,1198%	14,0719%	16,8164%
96	3,1437%	4,1417%	3,493%	4,8403%	3,6427%	4,8902%



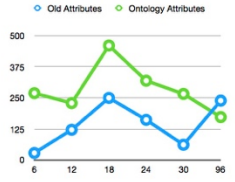


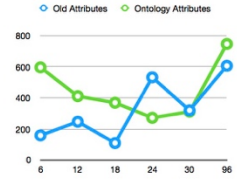


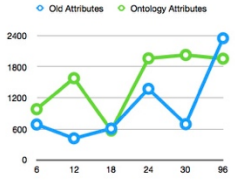
Memória

Used Memory (MB)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	29	269	159	596	687	983
12	122	229	247	411	421	1579
18	250	460	110	369	609	573
24	162	319	532	272	1374	1961
30	62	266	320	311	693	2026
96	239	173	606	746	2348	1955



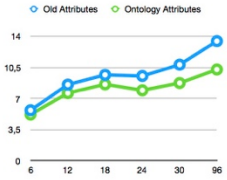


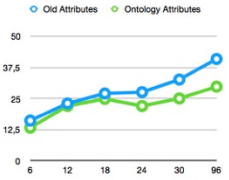


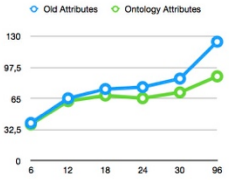
CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	5,69	5,21	16,17	13,35	39,56	38,03
12	8,57	7,62	23,06	22,06	65,19	62,36
18	9,67	8,60	27,06	24,87	74,97	68,20
24	9,54	7,93	27,59	22,03	76,98	65,38
30	10,81	8,75	32,64	25,05	85,83	71,47
96	13,46	10,27	40,90	29,78	124,32	88,18





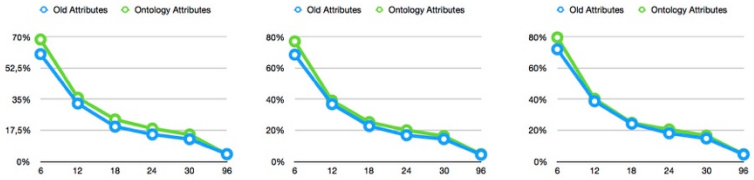


4080 eventos

Classificação

% de eventos corretamente classificados

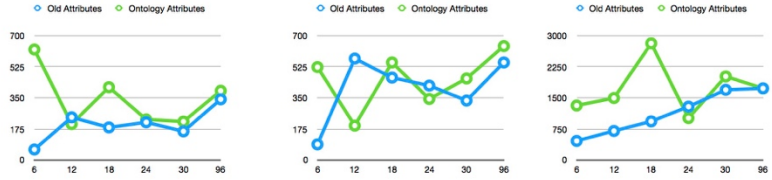
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	60,3676%	68,6029%	68,6275%	77,1814%	72,1324%	79,7304%
12	32,598%	35,8824%	36,8382%	39,1176%	38,7255%	40,2941%
18	19,6324%	23,652%	22,7941%	25,3186%	24,2402%	24,8529%
24	15,2451%	18,5539%	16,9118%	20,1225%	18,1618%	20,6373%
30	12,598%	15,1961%	14,4608%	16,3971%	14,8284%	16,6667%
96	4,1912%	4,1422%	4,4118%	4,7794%	4,5343%	4,7059%



Memória

Used Memory (MB)

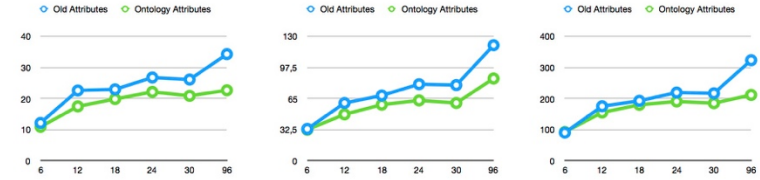
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	60	622	89	523	464	1316
12	241	203	571	194	704	1497
18	184	410	464	549	935	2813
24	213	229	419	344	1289	1014
30	162	217	335	459	1694	2017
96	342	390	549	641	1729	1729



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	12,19	10,95	33,36	32,55	90,58	92,83
12	22,60	17,50	60,37	48,57	175,37	155,40
18	22,96	19,87	68,26	58,64	192,94	180,06
24	26,77	22,15	80,01	63,11	219,45	190,70
30	26,12	20,89	79,10	60,40	216,95	185,20
96	34,27	22,67	120,70	85,95	323,27	211,87

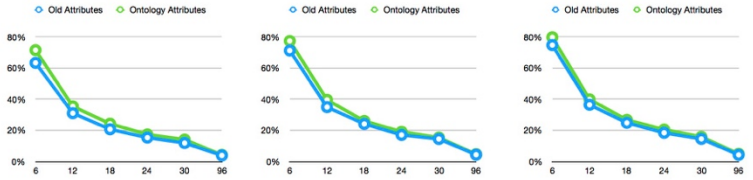


6120 eventos

Classificação

% de eventos corretamente classificados

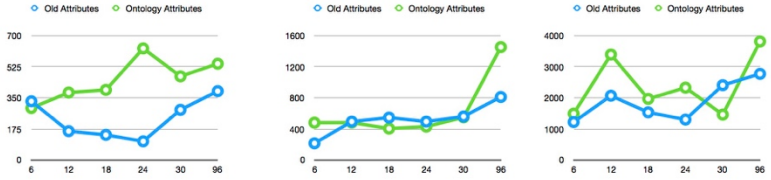
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	63,3824%	71,5686%	71,3072%	77,451%	74,7712%	79,8366%
12	31,0621%	35,4412%	34,9837%	39,6242%	36,4869%	39,9183%
18	20,7516%	24,2974%	24,1667%	26,0621%	24,8693%	26,781%
24	15,3922%	17,402%	17,1078%	19,1503%	18,4641%	20,5719%
30	11,9118%	14,134%	14,5098%	15,4085%	14,5588%	15,8824%
96	3,9216%	4,281%	4,4118%	4,7712%	4,3137%	5,0163%



Memória

Used Memory (MB)

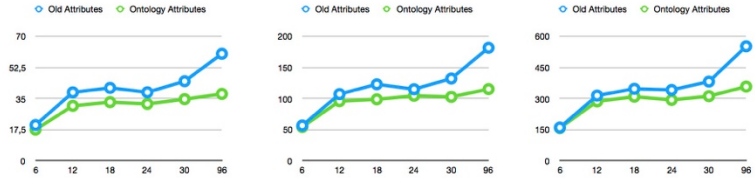
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	331	293	217	482	1224	1493
12	163	380	496	483	2070	3391
18	142	395	547	406	1535	1968
24	106	628	496	431	1304	2329
30	283	471	561	550	2408	1463
96	388	542	812	1452	2772	3812



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	20,14	17,6	56,82	54,46	160,87	158,20
12	36,51	30,93	107,23	95,92	314,65	286,64
18	41,01	33,03	122,96	98,84	347,19	309,26
24	36,54	32	115,02	104,57	341,64	293,75
30	44,76	34,69	132,21	102,77	381,89	311,76
96	60,18	37,64	181,60	115,22	551,78	357,78

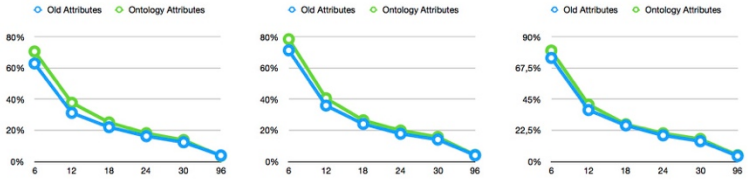


8160 eventos

Classificação

% de eventos corretamente classificados

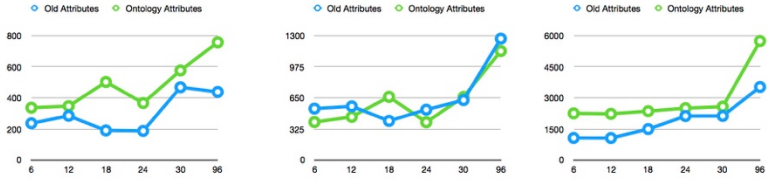
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	62,9902%	70,6863%	71,4338%	78,5784%	74,8407%	80,1593%
12	31,2623%	37,7574%	35,9804%	40,6373%	37,2304%	41,1765%
18	22,1201%	25,1225%	24,1422%	26,5196%	26,0662%	26,9485%
24	16,3113%	18,174%	17,8554%	19,9755%	18,9338%	20,3554%
30	12,4142%	13,8113%	14,1422%	15,7475%	14,7304%	16,3113%
96	3,9216%	3,6765%	4,0319%	4,2525%	3,9828%	4,5711%



Memória

Used Memory (MB)

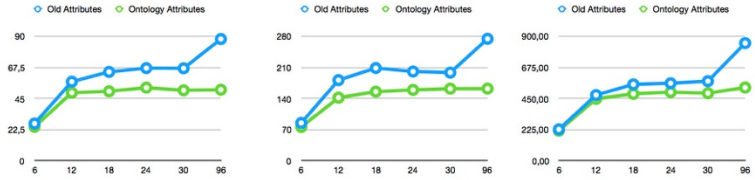
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	237	337	538	398	1070	2254
12	285	347	562	452	1066	2231
18	191	502	410	661	1498	2359
24	188	367	527	396	2126	2505
30	468	576	627	660	2131	2579
96	438	757	1268	1139	3525	5738



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	26,98	24,65	85,34	76,42	228,10	218,22
12	57,23	49,26	181,49	141,96	475,72	448,27
18	64,30	50,27	208,65	155,59	552,85	484,36
24	67,01	52,91	200,91	159,45	560,97	496,76
30	66,88	50,98	198,39	162,30	575,92	488,96
96	88,04	51,37	274,41	162,38	850,60	529,91

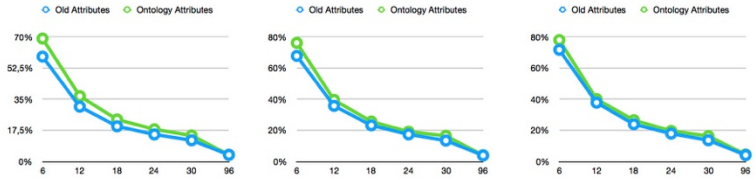


10200 eventos

Classificação

% de eventos corretamente classificados

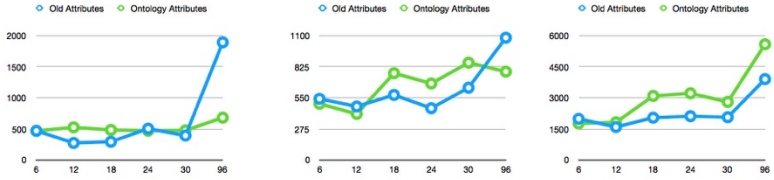
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	58,9804%	69,1275%	67,8529%	76,1961%	71,7745%	78,0294%
12	30,8529%	36,7745%	35,7843%	39,5686%	37,8235%	40,0784%
18	19,8235%	23,6471%	23,2451%	25,7059%	23,951%	26,5588%
24	15,2451%	18,1373%	17,4412%	19,1863%	17,9412%	19,7059%
30	11,9608%	14,6569%	13,5%	16,5294%	13,6961%	16,3529%
96	3,7647%	3,8039%	3,9314%	4,0588%	4,1275%	4,5294%



Memória

Used Memory (MB)

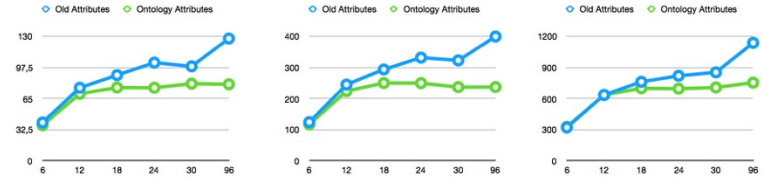
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	472	470	542	498	1994	1769
12	277	527	474	408	1597	1819
18	296	485	576	768	2047	3096
24	506	473	459	677	2118	3221
30	395	478	640	861	2070	2804
96	1892	683	1081	782	3903	5581



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	40,08	37,17	124,44	117,02	321,71	328,44
12	76,27	70,05	244,89	224,19	635,15	634,78
18	89,49	76,53	293,53	249,99	761,80	698,88
24	102,54	76,29	331,54	249,68	819,28	695,41
30	98,66	80,57	322,71	236,75	852,87	706,89
96	127,65	79,96	399,15	236,94	1137,05	753,66

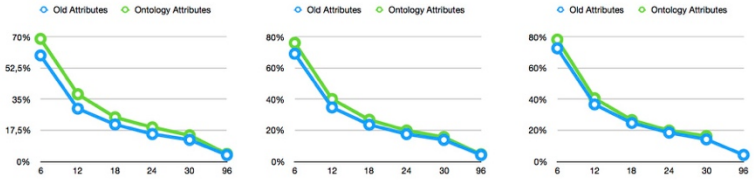


20400 eventos

Classificação

% de eventos corretamente classificados

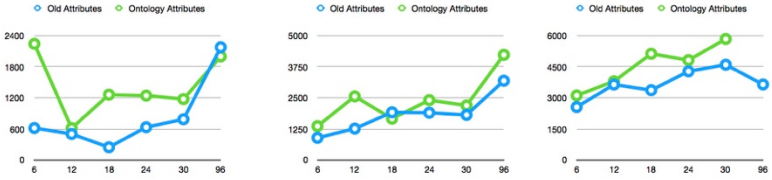
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	59,6324%	68,9657%	69,2206%	76,1814%	72,7108%	78,3333%
12	29,7402%	37,8431%	34,7745%	40,1422%	36,7157%	40,6225%
18	20,848%	24,8578%	23,652%	26,8431%	24,7647%	26,652%
24	15,4559%	19,2402%	17,6324%	19,9902%	18,5637%	19,9363%
30	12,2108%	14,7549%	13,951%	15,6912%	14,2647%	16,3676%
96	3,6716%	4,3137%	4,2206%	4,6814%	4,2549%	OutOfMemory



Memória

Used Memory (MB)

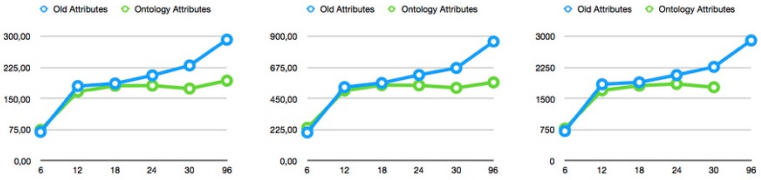
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	619	2241	897	1365	2558	3117
12	504	617	1267	2561	3643	3797
18	249	1262	1922	1666	3370	5127
24	635	1244	1904	2409	4278	4817
30	788	1177	1816	2196	4597	5836
96	2175	1999	3188	4227	3647	OutOfMemory



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	69,70	74,53	204,90	238,37	717,66	776,36
12	180,03	166,68	532,91	508,06	1844,87	1699,23
18	186,34	180,98	562,98	547,43	1892,36	1811,98
24	205,87	181,37	620,79	545,77	2067,51	1853,49
30	229,74	174,13	671,04	527,45	2262,68	1773,12
96	291,90	193,11	861,65	567,41	2901,21	OutOfMemory

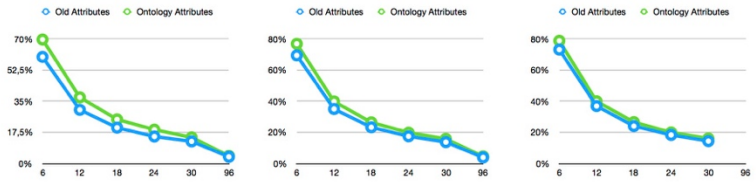


30600 eventos

Classificação

% de eventos corretamente classificados

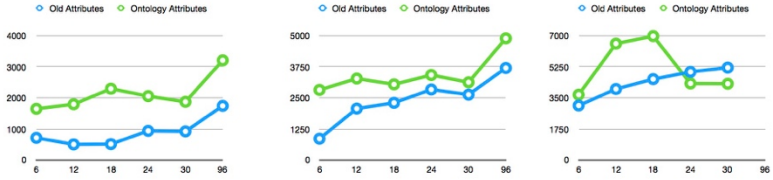
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	59,9379%	69,7092%	69,4804%	76,902%	73,1765%	78,866%
12	30,209%	37,2222%	35,0425%	39,8725%	36,9444%	39,9935%
18	20,183%	24,8203%	23,3072%	26,4575%	24,0752%	26,6307%
24	15,2059%	19,0719%	17,5196%	19,8954%	18,3791%	20,0131%
30	12,4608%	14,7092%	13,7843%	15,8399%	14,4248%	16,1046%
96	3,8072%	4,317%	3,9346%	4,6111%	OutOfMemory	OutOfMemory



Memória

Used Memory (MB)

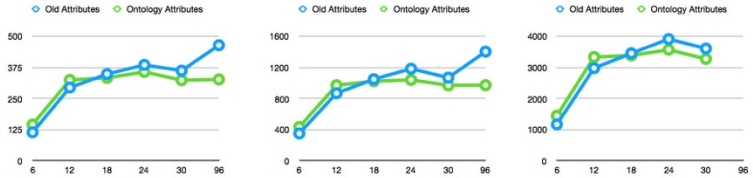
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	717	1649	863	2819	3070	3677
12	508	1796	2070	3276	3998	6546
18	517	2294	2302	3043	4554	6970
24	941	2056	2836	3419	4961	4309
30	924	1875	2630	3126	5200	4294
96	1744	3206	3700	4889	OutOfMemory	OutOfMemory



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	114,66	145,88	352,32	434,89	1170,64	1449,83
12	294,33	324,85	869,24	971,36	2978,70	3332,71
18	348,42	333,37	1048,62	1022,87	3454,66	3384,41
24	385,16	357,02	1183,34	1040,18	3910,55	3569,05
30	362,22	323,88	1068,80	970,48	3607,95	3274,54
96	464,11	326,85	1403,14	972,54	OutOfMemory	OutOfMemory

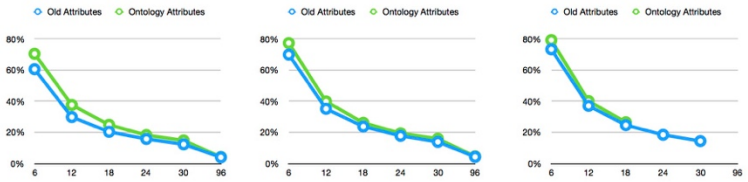


40800 eventos

Classificação

% de eventos corretamente classificados

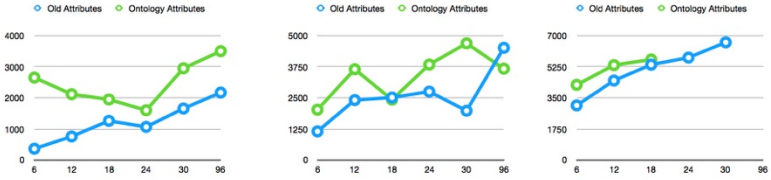
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	60,6152%	70,5%	69,9877%	77,3995%	73,375%	79,3015%
12	29,8725%	37,6495%	35,1765%	39,8897%	37,0833%	40,1373%
18	20,4069%	24,8676%	23,7941%	26,174%	24,6789%	26,5466%
24	15,8088%	18,2917%	17,8235%	19,4657%	18,4902%	OutOfMemory
30	12,3676%	14,8529%	13,9289%	15,9387%	14,4608%	OutOfMemory
96	4,0294%	4,25%	4,3088%	4,652%	OutOfMemory	OutOfMemory



Memória

Used Memory (MB)

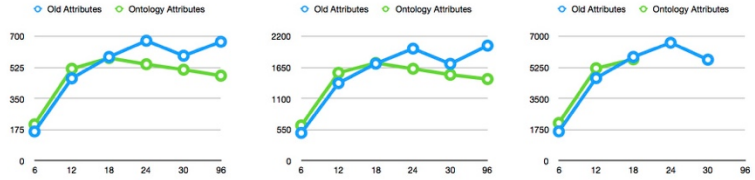
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	368	2652	1156	2027	3077	4232
12	761	2118	2412	3649	4473	5340
18	1263	1949	2515	2430	5367	5655
24	1069	1602	2752	3838	5763	OutOfMemory
30	1659	2949	1983	4690	6619	OutOfMemory
96	2170	3501	4509	3671	OutOfMemory	OutOfMemory



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	164,99	205,05	493,21	625,20	1653,72	2128,27
12	463,57	517,76	1373,26	1555,52	4651,14	5204,00
18	583,87	578,00	1716,95	1729,05	5841,34	5712,23
24	674,99	543,00	1982,55	1626,86	6636,51	OutOfMemory
30	590,70	512,38	1713,63	1520,81	5686,19	OutOfMemory
96	669,07	478,56	2034,47	1444,70	OutOfMemory	OutOfMemory

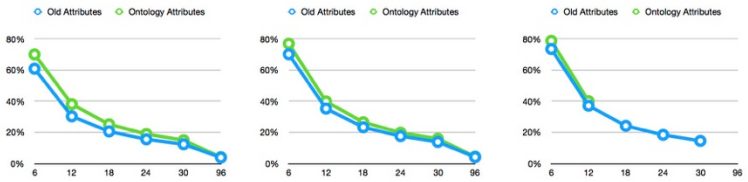


51000 eventos

Classificação

% de eventos corretamente classificados

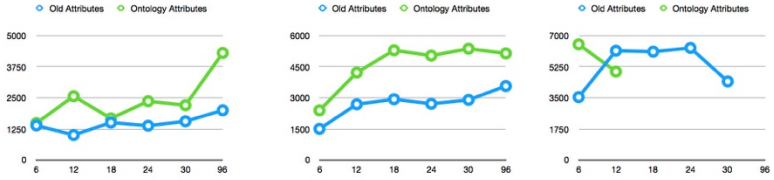
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	60,9059%	70,0941%	70,2196%	76,9922%	73,5667%	78,8118%
12	30,3353%	38,1235%	35,2922%	39,8922%	37,1471%	40,0333%
18	20,6549%	25,1922%	23,3412%	26,5176%	24,2196%	OutOfMemory
24	15,5765%	19,0059%	17,6255%	19,8529%	18,449%	OutOfMemory
30	12,398%	14,9843%	13,8765%	15,9216%	14,5745%	OutOfMemory
96	3,8765%	4,1451%	4,1529%	4,4608%	OutOfMemory	OutOfMemory



Memória

Used Memory (MB)

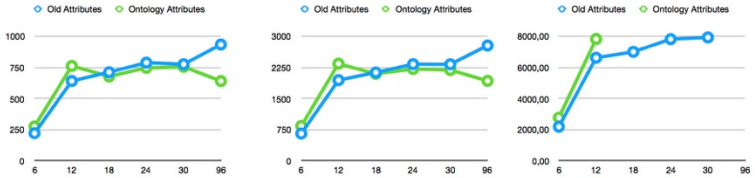
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	1394	1486	1508	2397	3543	6507
12	1009	2567	2692	4219	6151	4977
18	1511	1676	2935	5292	6097	OutOfMemory
24	1383	2364	2716	5036	6307	OutOfMemory
30	1561	2205	2904	5369	4423	OutOfMemory
96	2001	4304	3570	5143	OutOfMemory	OutOfMemory



CPU

CPU User Time (Seconds)

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Categorias	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
6	222,11	276,34	657,63	840,63	2192,70	2775,15
12	641,38	762,09	1943,62	2341,11	6621,04	7825,76
18	711,79	677,82	2129,57	2096,40	7004,66	OutOfMemory
24	788,47	746,59	2328,17	2212,09	7818,16	OutOfMemory
30	775,64	755,11	2325,73	2188,05	7928,01	OutOfMemory
96	933,60	640,77	2777,55	1926,93	OutOfMemory	OutOfMemory

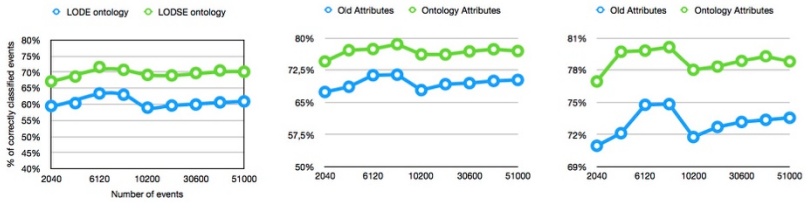


Resultados Finais - Evolução Classificação

Classificação - 6 Categorias

% eventos corretamente classificados - 6 Categorias

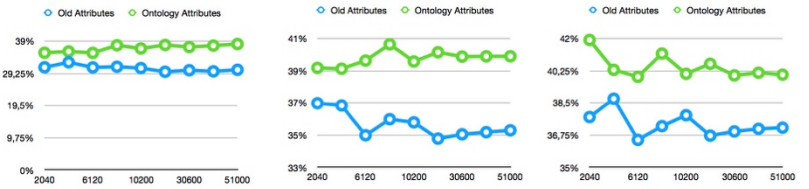
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	59,481%	67,1158%	67,4152%	74,5509%	70,9581%	76,9461%
4080	60,3676%	68,6029%	68,6275%	77,1814%	72,1324%	79,7304%
6120	63,3824%	71,5686%	71,3072%	77,451%	74,7712%	79,8366%
8160	62,9902%	70,6863%	71,4338%	78,5784%	74,8407%	80,1593%
10200	58,9804%	69,1275%	67,8529%	76,1961%	71,7745%	78,0294%
20400	59,6324%	68,9657%	69,2206%	76,1814%	72,7108%	78,3333%
30600	59,9379%	69,7092%	69,4804%	76,902%	73,1765%	78,866%
40800	60,6152%	70,5%	69,9877%	77,3995%	73,375%	79,3015%
51000	60,9059%	70,0941%	70,2196%	76,9922%	73,5667%	78,8118%



Classificação - 12 Categorias

% eventos corretamente classificados - 12 Categorias

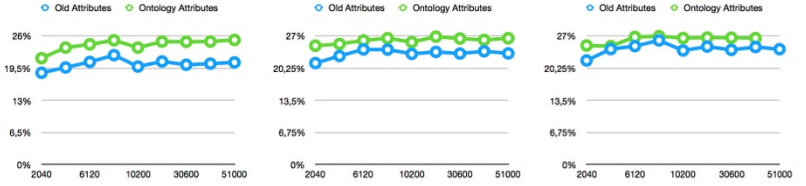
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	31,0878%	35,479%	36,976%	39,1717%	37,7246%	41,9162%
4080	32,598%	35,8824%	36,8382%	39,1176%	38,7255%	40,2941%
6120	31,0621%	35,4412%	34,9837%	39,6242%	36,4869%	39,9183%
8160	31,2623%	37,7574%	35,9804%	40,6373%	37,2304%	41,1765%
10200	30,8529%	36,7745%	35,7843%	39,5686%	37,8235%	40,0784%
20400	29,7402%	37,8431%	34,7745%	40,1422%	36,7157%	40,6225%
30600	30,209%	37,2222%	35,0425%	39,8725%	36,9444%	39,9935%
40800	29,8725%	37,6495%	35,1765%	39,8897%	37,0833%	40,1373%
51000	30,3353%	38,1235%	35,2922%	39,8922%	37,1471%	40,0333%



Classificação - 18 Categorias

% eventos corretamente classificados - 18 Categorias

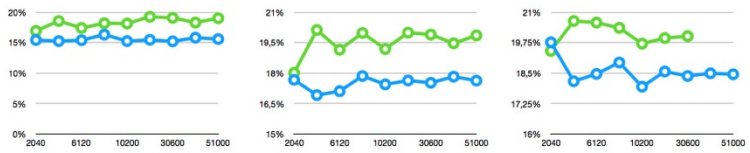
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	18,5629%	21,507%	21,3074%	24,9501%	21,8563%	25%
4080	19,6324%	23,652%	22,7941%	25,3186%	24,2402%	24,8529%
6120	20,7516%	24,2974%	24,1667%	26,0621%	24,8693%	26,781%
8160	22,1201%	25,1225%	24,1422%	26,5196%	26,0662%	26,9485%
10200	19,8235%	23,6471%	23,2451%	25,7059%	23,951%	26,5588%
20400	20,848%	24,8578%	23,652%	26,8431%	24,7647%	26,652%
30600	20,183%	24,8203%	23,3072%	26,4575%	24,0752%	26,6307%
40800	20,4069%	24,8676%	23,7941%	26,174%	24,6789%	26,5466%
51000	20,6549%	25,1922%	23,3412%	26,5176%	24,2196%	OutOfMemory



Classificação - 24 Categorias

% eventos corretamente classificados - 24 Categorias

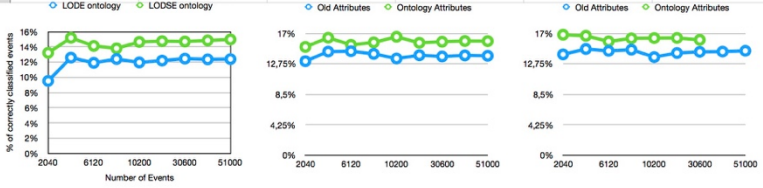
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	15,4192%	16,9162%	17,6647%	18,014%	19,7605%	19,4112%
4080	15,2451%	18,5539%	16,9118%	20,1225%	18,1618%	20,6373%
6120	15,3922%	17,402%	17,1078%	19,1503%	18,4641%	20,5719%
8160	16,3113%	18,174%	17,8554%	19,9755%	18,9338%	20,3554%
10200	15,2451%	18,1373%	17,4412%	19,1863%	17,9412%	19,7059%
20400	15,4559%	19,2402%	17,6324%	19,9902%	18,5637%	19,9363%
30600	15,2059%	19,0719%	17,5196%	19,8954%	18,3791%	20,0131%
40800	15,8088%	18,2917%	17,8235%	19,4657%	18,4902%	OutOfMemory
51000	15,7755%	19,0059%	17,8255%	19,8529%	18,4286%	OutOfMemory



Classificação - 30 Categorias

% eventos corretamente classificados - 30 Categorias

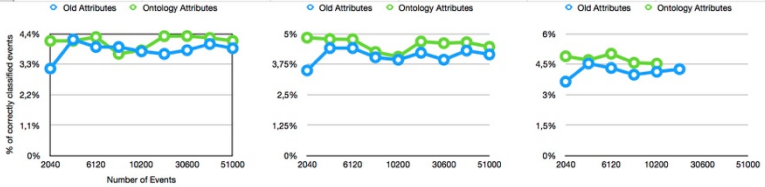
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	9,5309%	13,2236%	13,1238%	15,1198%	14,0719%	16,8164%
4080	12,598%	15,1961%	14,4608%	16,3971%	14,8284%	16,6667%
6120	11,9118%	14,134%	14,5098%	15,4085%	14,5588%	15,8824%
8160	12,4142%	13,8113%	14,1422%	15,7475%	14,7304%	16,3113%
10200	11,9608%	14,6569%	13,5%	16,5294%	13,6961%	16,3529%
20400	12,2108%	14,7549%	13,951%	15,6912%	14,2647%	16,3676%
30600	12,4608%	14,7092%	13,7843%	15,8399%	14,4248%	16,1046%
40800	12,3676%	14,8529%	13,9289%	15,9387%	14,4608%	OutOfMemory
51000	12,398%	14,9843%	13,8765%	15,9216%	14,5745%	OutOfMemory



Classificação - 96 Categorias

% eventos corretamente classificados - 96 Categorias

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	3,1437%	4,1417%	3,493%	4,8403%	3,6427%	4,8902%
4080	4,1912%	4,1422%	4,4118%	4,7794%	4,5343%	4,7059%
6120	3,9216%	4,281%	4,4118%	4,7712%	4,3137%	5,0163%
8160	3,9216%	3,6765%	4,0319%	4,2525%	3,9828%	4,5711%
10200	3,7647%	3,8039%	3,9314%	4,0588%	4,1275%	4,5294%
20400	3,6716%	4,3137%	4,2206%	4,6814%	4,2549%	OutOfMemory
30600	3,8072%	4,317%	3,9346%	4,6111%	OutOfMemory	OutOfMemory
40800	4,0294%	4,25%	4,3088%	4,652%	OutOfMemory	OutOfMemory
51000	3,8765%	4,1451%	4,1529%	4,4608%	OutOfMemory	OutOfMemory

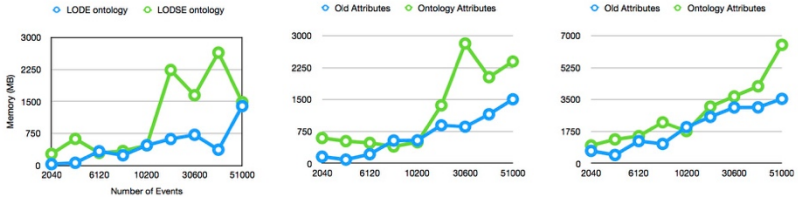


Resultados Finais - Evolução Memória

Classificação - 6 Categorias

Used Memory (MB) - 6 Categorias

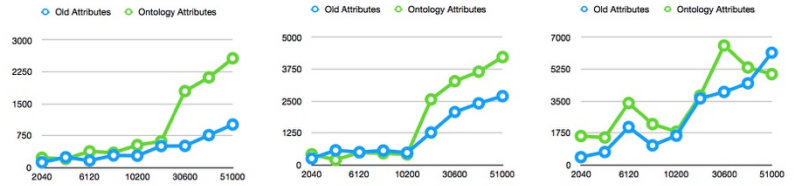
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	29	269	159	596	687	983
4080	60	622	89	523	464	1316
6120	331	293	217	482	1224	1493
8160	237	337	538	398	1070	2254
10200	472	470	542	498	1994	1769
20400	619	2241	897	1365	2558	3117
30600	717	1649	863	2819	3070	3677
40800	368	2652	1156	2027	3077	4232
51000	1394	1486	1508	2397	3543	6507



Classificação - 12 Categorias

Used Memory (MB) - 12 Categorias

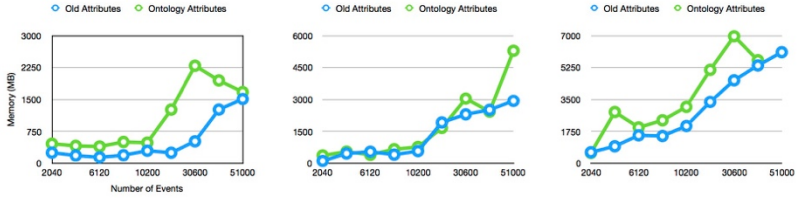
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	122	229	247	411	421	1579
4080	241	203	571	194	704	1497
6120	163	380	496	483	2070	3391
8160	285	347	562	452	1066	2231
10200	277	527	474	408	1597	1819
20400	504	617	1267	2561	3643	3797
30600	508	1796	2070	3276	3998	6546
40800	761	2118	2412	3649	4473	5340
51000	1009	2567	2692	4219	6151	4977



Classificação - 18 Categorias

Used Memory (MB) - 18 Categorias

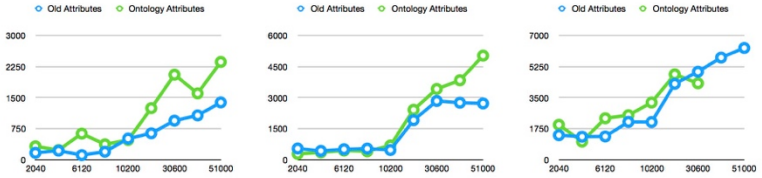
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	250	460	110	369	609	573
4080	184	410	464	549	935	2813
6120	142	395	547	406	1535	1968
8160	191	502	410	661	1498	2359
10200	296	485	576	768	2047	3096
20400	249	1262	1922	1666	3370	5127
30600	517	2294	2302	3043	4554	6970
40800	1263	1949	2515	2430	5367	5655
51000	1511	1676	2935	5292	6097	OutOfMemory



Classificação - 24 Categorias

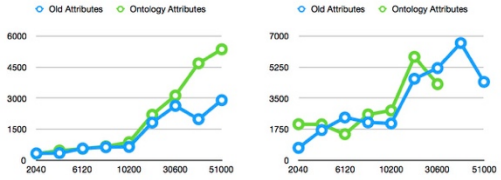
Used Memory (MB) - 24 Categorias

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	162	319	532	272	1374	1961
4080	213	229	419	344	1289	1014
6120	106	628	496	431	1304	2329
8160	188	367	527	396	2126	2505
10200	506	473	459	677	2118	3221
20400	635	1244	1904	2409	4278	4817
30600	941	2056	2836	3419	4961	4309
40800	1069	1602	2752	3838	5763	OutOfMemory
51000	1383	2364	2716	5036	6307	OutOfMemory



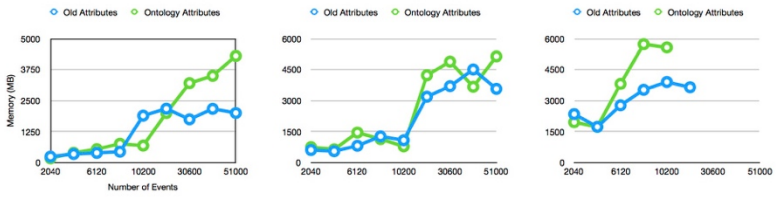
Used Memory (MB) - 30 Categorias

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	62	266	320	311	693	2026
4080	162	217	335	459	1694	2017
6120	283	471	561	550	2408	1463
8160	468	576	627	660	2131	2579
10200	395	478	640	861	2070	2804
20400	788	1177	1816	2196	4597	5836
30600	924	1875	2630	3126	5200	4294
40800	1659	2949	1983	4690	6619	OutOfMemory
51000	1561	2205	2904	5369	4423	OutOfMemory



Used Memory (MB) - 96 Categorias

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	239	173	606	746	2348	1955
4080	342	390	549	641	1729	1729
6120	388	542	812	1452	2772	3812
8160	438	757	1268	1139	3525	5738
10200	1892	683	1081	782	3903	5581
20400	2175	1999	3188	4227	3647	OutOfMemory
30600	1744	3206	3700	4889	OutOfMemory	OutOfMemory
40800	2170	3501	4509	3671	OutOfMemory	OutOfMemory
51000	2001	4304	3570	5143	OutOfMemory	OutOfMemory

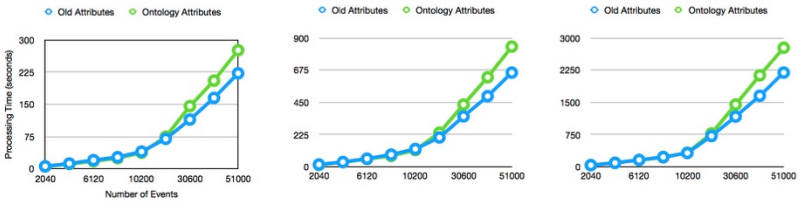


Resultados Finais - Evolução CPU

Classificação - 6 Categorias

CPU User Time (Seconds) - 6 Categorias

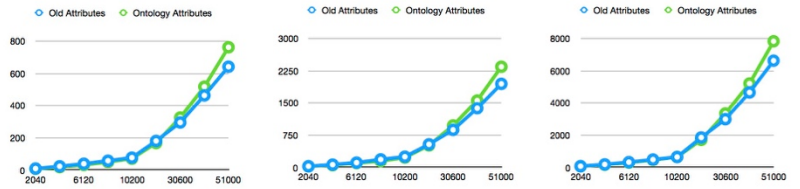
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	5,69	5,21	16,17	13,35	39,56	38,03
4080	12,19	10,95	33,36	32,55	90,58	92,83
6120	20,14	17,6	56,82	54,46	160,87	158,20
8160	26,98	24,65	85,34	76,42	228,10	218,22
10200	40,08	37,17	124,44	117,02	321,71	328,44
20400	69,70	74,53	204,90	238,37	717,66	776,36
30600	114,66	145,88	352,32	434,89	1170,64	1449,83
40800	164,99	205,05	493,21	625,20	1653,72	2128,27
51000	222,11	276,34	657,63	840,63	2192,70	2775,15



Classificação - 12 Categorias

CPU User Time (Seconds) - 12 Categorias

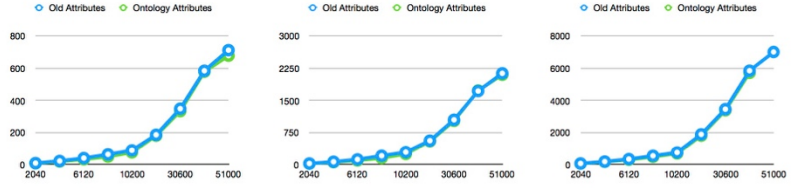
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	8,57	7,62	23,06	22,06	65,19	62,36
4080	22,60	17,50	60,37	48,57	175,37	155,40
6120	38,51	30,93	107,23	95,92	314,65	286,64
8160	57,23	49,26	181,49	141,96	475,72	448,27
10200	76,27	70,05	244,89	224,19	635,15	634,78
20400	180,03	166,68	532,91	508,06	1844,87	1699,23
30600	294,33	324,85	869,24	971,36	2978,70	3332,71
40800	463,57	517,76	1373,26	1555,52	4651,14	5204,00
51000	641,38	762,09	1943,62	2341,11	6621,04	7825,76



Classificação - 18 Categorias

CPU User Time (Seconds) - 18 Categorias

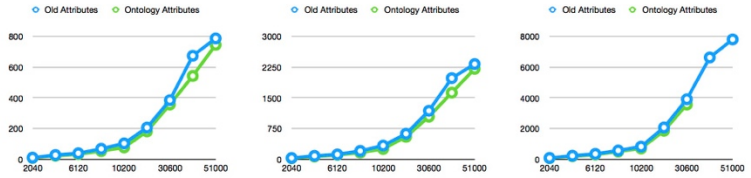
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	9,67	8,60	27,06	24,87	74,97	68,20
4080	22,96	19,87	68,26	58,64	192,94	180,06
6120	41,01	33,03	122,96	98,84	347,19	309,26
8160	64,30	50,27	208,85	155,59	552,85	484,36
10200	89,49	76,53	293,53	249,99	761,80	698,88
20400	186,34	180,98	562,98	547,43	1892,36	1811,98
30600	348,42	333,37	1048,62	1022,87	3454,66	3384,41
40800	583,87	578,00	1716,95	1729,05	5841,34	5712,23
51000	711,79	677,82	2129,57	2096,40	7004,66	OutOfMemory



Classificação - 24 Categorias

CPU User Time (Seconds) - 24 Categorias

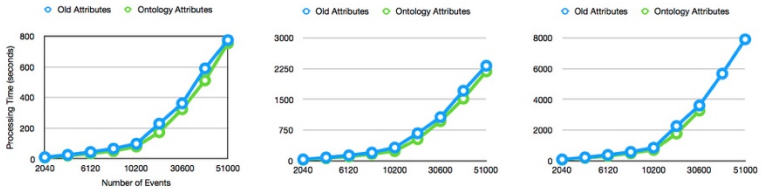
	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	9,54	7,93	27,59	22,03	76,98	65,38
4080	26,77	22,15	80,01	63,11	219,45	190,70
6120	38,54	32	115,02	104,57	341,64	293,75
8160	67,01	52,91	200,91	159,45	560,97	496,76
10200	102,54	76,29	331,54	249,68	819,28	695,41
20400	205,87	181,37	620,79	545,77	2067,51	1853,49
30600	385,16	357,02	1183,34	1040,18	3910,55	3569,05
40800	674,99	543,00	1982,55	1626,86	6636,51	OutOfMemory
51000	788,47	746,59	2328,17	2212,09	7818,16	OutOfMemory



Classificação - 30 Categorias

CPU User Time (Seconds) - 30 Categorias

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes	Old Attributes	Ontology Attributes
2040	10,81	8,75	32,64	25,05	85,83	71,47
4080	26,12	20,89	79,10	60,40	216,95	185,20
6120	44,76	34,69	132,21	102,77	381,89	311,76
8160	66,88	50,98	198,39	162,30	575,92	488,96
10200	98,66	80,57	322,71	236,75	852,87	706,89
20400	229,74	174,13	671,04	527,45	2262,68	1773,12
30600	362,22	323,88	1068,80	970,48	3607,95	3274,54
40800	590,70	512,38	1713,63	1520,81	5686,19	OutOfMemory
51000	775,64	755,11	2325,73	2188,05	7928,01	OutOfMemory



Classificação - 96 Categorias

CPU User Time (Seconds) - 96 Categorias

	numIterations = 10		numIterations = 30		numIterations = 100	
Nº Eventos	numIterations = 10	numIterations = 30	numIterations = 100	Nº Eventos	numIterations = 10	numIterations = 30
2040	13,46	10,27	40,90	29,78	124,32	88,18
4080	34,27	22,67	120,70	85,95	323,27	211,87
6120	60,18	37,64	181,60	115,22	551,78	357,78
8160	88,04	51,37	274,41	162,38	850,60	529,91
10200	127,65	79,96	399,15	236,94	1137,05	753,66
20400	291,90	193,11	861,65	567,41	2901,21	OutOfMemory
30600	464,11	326,85	1403,14	972,54	OutOfMemory	OutOfMemory
40800	669,07	478,56	2034,47	1444,70	OutOfMemory	OutOfMemory
51000	933,60	640,77	2777,55	1926,93	OutOfMemory	OutOfMemory