INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de Recomendação de Tutoriais

RUI FERREIRA DE VASCONCELOS MARTINS
Outubro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de Recomendação de Tutoriais

Rui Ferreira de Vasconcelos Martins

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Tecnologias de Conhecimento de Decisão

Orientador: Professor Nuno Bettencourt

Júri:

Presidente:

Vogais:

Porto, 2016

Dedico esta Dissertação aos meus pais, irmã, Guilherme e em especial a Susana, minha esposa, pela ausência que a mesma me privou de vocês.

Agradecimentos

Aproveito este momento para agradecer ao Prof. Nuno Bettencourt, orientador desta dissertação, por toda a disponibilidade e empenho na orientação.

À Prof. Susana Nicola pela disponibilidade e prontidão em ajudar.

Aos meus colegas com quem fui discutindo, tirando ideias e opiniões.

Aos meus amigos pela paciência e amizade.

E sobretudo à minha esposa, Susana, que sempre esteve do meu lado e me apoiou.

Resumo

Nos dias de hoje são as nossas decisões e a qualidade destas que nos definem. A capacidade de tomar decisões em tempo útil é um desafio a que este trabalho propõe resolver.

Atualmente existem inúmeras aplicações de *software* que têm como objetivo facilitar a experiência dos utilizadores nos mais diversos campos. Esta tese aborda um *software* específico e como adaptar um sistema de recomendações ao mesmo. Este exemplo poderá ser aplicado a outros tipo de *software* no futuro.

Um *software* capaz de gerir de tutoriais de forma a poderem ser vistos e avaliados por outros utilizadores é por si só um sistema que poderá facilitar e elevar pesquisas a outro patamar, mas e se a este *software* fosse acrescentado um sistema de recomendação de tutoriais? E se em vez de o utilizador andar à procura de um tutorial lhe fosse recomendado um que faça parte do seu perfil?

Neste documento estará descrito um conjunto de pesquisas do que já foi feito neste campo bem como as tecnologias existentes. Após uma avaliação cuidada e respetiva discussão é eleita a tecnologia que poderá proporcionar a esta ideia uma solução viável. Será montada uma arquitetura e com base nesta será desenvolvida uma solução para este problema com base em vários coeficientes e algoritmos.

Após a finalização deste processo, a avaliação será feita com base na cobertura de dados e tempo de resposta uma vez que este tipo de recomendações devem ser tomada o mais rápido possível para que este sistema seja eficiente. Será também implantando um conjunto de avaliações para que se possa concluir se esta aplicação acrescentou valor a esta aplicação

Palavras-Chave: Recomendação, tutoriais, *mahout*, sistemas de recomendação.

Abstract

Nowadays our decisions and the quality of these are what define us. The ability to make quick decisions is a challenge that I propose to solve.

Currently there are software that aims to facilitate the experience of users in various fields. This document mentions about a specific software and how it is possible to add a recommendation system to it. This example can be applied to other types of software in the future.

A mentoring management software so that they can be seen and evaluated by others is itself a system that could facilitate and enhance research to another level, but what if this software was an added tutorials recommendation system? What if instead of walking the user looking for a tutorial would you recommend one that is part of the profile?

This document will describe a body of research that has already been done in this field as well as existing technologies. After careful evaluation and its discussion it is chosen which technology can provide this idea a viable solution architecture based on this a solution is developed by using various coefficients and algorithms.

After completion of these cases the temporal assessment is made since such recommendations should be taken as soon as possible for this system to be effective. It will also be deploying a set of assessments so that it can be concluded if this application added value to this application.

Keywords: Recommendation, tutorials, mahout, recommender systems.

Índice

Índice de Imagens

Índice de Tabelas

Acrónimos

AHP - Análise Hierárquica de Processos

API - Application Programming Interface

CPU - Communications Processor Unit

CRUD - Create Read Update Delete

CRM - Customer Relationship Management

FIFO - First In First Out

GNU - General Public License

RAM - Random Access Memory

UML - Unified Modeling Language

1 Introdução

O presente capítulo serve para dar aos leitores uma breve contextualização do problema e das áreas que envolve.

A aplicação de gestão de tutoriais tem uma missão bem clara: ajudar a fazer e a perceber aquelas tarefas que ninguém quer fazer. No meio profissional quantas vezes não se repetem sempre as mesmas coisas às pessoas que entram na empresa pela primeira vez? Empresas com um negócio e um objetivo bem estabelecido têm metas fixas e entregas bem definidas, quando é preciso mais recursos humano é porque há um crescente de trabalho ou porque há risco que as metas não sejam cumpridas. Daí que iniciar uma nova pessoa nas suas tarefas é, a curto prazo um desperdício de dois recursos, um que entra de novo e o recurso que o tem que acompanhar. Nos dias de hoje há que ser competitivos, que mostrar porque se é melhor; há a necessidade de se utilizar, de forma inteligente, todos os recursos. Assim surgiu a aplicação de gestão de tutoriais, para que não haja necessidade de despender tempo a dizer sempre as mesmas coisas. Sendo esta aplicação uma inovação no seu campo de intervenção precisa de elementos novos e capazes, precisa de um sistema de recomendação para que se torne cada vez mais autónomo e eficaz.

Inteligência artificial consiste em dotar máquinas com a capacidade de um ser humano ao nível da aprendizagem e da tomada de decisão. Enquanto seres humanos, somos desde muito novos capazes de captar informação do mundo que nos rodeia e com essas experiências somos capazes de tomar decisões de uma forma ponderada. Com base nas nossas vivências conseguimos distinguir, segundo as nossas premissas, o bem do mal e com base nisso tomamos ações para ultrapassar problemas ou atingir os nossos objetivos.

A inteligência artificial faz parte dos estudos de Ciências da Computação e os programas utilizam a mesma linguagem de sistemas convencionais.

A aprendizagem máquina é um tipo de inteligência artificial que fornece aos computadores a capacidade de aprender, sem serem explicitamente programados com essa informação. Aprendizagem máquina centra-se no desenvolvimento de programas que podem ensinar-se a crescer e mudar, quando expostos a novos dados. De uma

maneira simplificada é habilitar uma máquina para que possa passar por todo o processo de aprendizagem que um ser humano passou adquirindo informação, guardando e utilizando para que todo o seu conhecimento possa evoluir e assim ganhar experiência para adquirir novas capacidades de ultrapassar problemas e atingir objetivos [1].

Os sistemas de recomendação são uma subárea de aprendizagem de máquina (*machine learning*) e são mecanismos de filtragem de informação inteligente que reduzem o processo de tomada de decisão para apenas algumas propostas, tornando-se uma parte integrante da experiência do utilizador dentro de algumas das aplicações favoritas.

1.1 Estrutura do Documento

Este Documento é composto inicialmente por uma fase introdutória onde se disponibiliza o problema e o contexto do que se propõe passando de seguida para uma pesquisa e levantamento do que se têm desenvolvido no ramo de sistemas de recomendação. Após esta pesquisa, e escolhido a *framework,* passa-se à arquitetura da aplicação que vêm imediatamente antes da descrição detalhada de como a solução para este problema foi desenvolvida. Desse desenvolvimento surgem resultados, que serão escortinados na ultima secção, a conclusão, onde também se avalia a qualidade do que foi atingido.

1.2 Problema

Neste projeto o problema é encarado como uma oportunidade de gerar valor a um *software* já existente. Usando uma aplicação já existente, o desafio passa pela projeção e construção de um sistema de recomendação para habilitar e valorizar o *software*. A primeira fase consiste na pesquisa por *software* existente; após essa pesquisa é feita uma avaliação para se retirarem as respetivas conclusões do *software* encontrado. Após finalização desta pesquisa e respetivas conclusões passa-se para a arquitetura deste *software* onde é construído uma solução com base nesta arquitetura idealizada no ponto 4 desenho da solução.

Com este trabalho pretende-se responder a um conjunto de questões, tais como:

- Qual o melhor algoritmo para um sistema de recomendação?
- Qual o algoritmo mais rápido?
- Qual o algoritmo mais eficaz?

- Qual a *framework* ideal para este sistema?
- Qual o valor acrescentado ao software com este sistema de recomendação?

1.3 Análise de Valor

Todo o projeto, produto ou serviço tem determinado valor para o cliente final. São os clientes que percecionam o valor de determinado produto ou serviço.

Valor não é somente um bem adquirido pelo cliente, mas a diferença entre os benefícios e sacrifícios, tangíveis ou intangíveis obtidos por determinado produto/serviço. O valor do *software* de recomendação é um valor que deriva do consumo, no qual o cliente através da experiência com o software, vai conseguir avaliar benefícios funcionais, a customização, a flexibilidade e a utilidade do produto. Deste modo, uma proposta de valor bem definida, na qual explicite a quem se destina o produto, qual o cliente alvo, os benefícios e a razão de o produto ser único, vai permitir ao cliente optar pela empresa em causa e não pela sua concorrência.

O que se pretende com este sistema de recomendação é adicionar valor à aplicação e com isso acrescentar valor para os seus clientes.

2 Contexto e Estado da Arte

Neste capítulo é feita uma análise mais granular do problema bem como uma contextualização mais profunda das áreas envolventes. É também apresentado neste capítulo o resultado de um conjunto de pesquisas sobre trabalhos já desenvolvidos e tecnologias mais usadas. É feita a análise de valor onde, de uma maneira simples, é representado todo o valor que este sistema de recomendação vai trazer para o *software* e seus utilizadores.

2.1 Detalhes sobre o contexto e o problema

Sendo esta aplicação um *software* tendo como principal objetivo auxiliar os utilizadores na criação de tutoriais de apoio a várias plataformas fazia todo o sentido a adoção de um sistema de recomendação para que se pudesse tirar um maior partido da aplicação.

O principal objetivo com este sistema de recomendação passa por criar valor à aplicação, recomendando aos utilizadores tutoriais que possam ser úteis dentro do seu âmbito de pesquisas tendo sempre em conta pesquisas de outros utilizadores.

Com este *software* pretende-se aperfeiçoar o que já existe e integrar num sistema que por si só já está empenhado em facilitar a vida aos utilizadores. A capacidade de recomendar está encapsulada em vários algoritmos, o objetivo é que sejam recomendados tutoriais que se aproximem o mais possível das necessidades reais dos utilizadores.

A finalidade deste processo de integração de um sistema de recomendação com o software já existente vai permitir ao utilizador, escolher outros tutoriais que estejam no âmbito da sua pesquisa. E para um melhor entendimento deste sistema de recomendação é importante que haja alguma contextualização relativamente a inteligência artificial, aprendizagem máquina e sistemas de recomendação que serão abordados nos pontos seguintes.

Será criado um conjunto de dados relativamente a interação dos utilizadores com os tutoriais, nesse documento estará presente três colunas, identificação do utilizador, identificação do tutorial, título e força de ligação.

2.1.1 Inteligência artificial

A inteligência artificial é um ramo da ciência da computação que se compromete a desenvolver produtos capazes de tomar decisões e resolver problemas, ou seja, tornar um produto idêntico ao pensamento humano capaz, através de vários mecanismos, aprender e tomar decisões de uma maneira semelhante à dos humanos.

É uma ciência que já existe há décadas e que têm vindo a ser muito impulsionada com o grande desenvolvimento da informática e da computação, habilitando vários produtos com esta "inteligência" [3].

Os cientistas Hebert Simon e Allen Newell foram os pioneiros em 1950, na área com o primeiro laboratório na Universidade de Carnegie Mellon. Inicialmente o grande objetivo era criar máquinas que conseguissem desenvolver tanto atividades simples como também aquelas atividades extremamente complexas [4].

2.1.2 Aprendizagem máquina

Há diferentes abordagens de inteligência artificial para solucionar uma grande variedade de problemas. Uma dessas abordagens é a aprendizagem máquina, uma das áreas mais relevantes dentro da inteligência artificial.

Os algoritmos de aprendizagem máquina procuram padrões dentro de um conjunto de dados. Esses algoritmos existem há bastante tempo[23], mas só nos últimos anos é que começou a existir uma enorme quantidade de dados graças a informatização em massa e principalmente à Internet, capazes de fornecer a estes algoritmos dados significativos para que se possa tirar mais e melhores conclusões dos mesmos.

Esta área de inteligência artificial só começou a ganhar algum relevo pelos fatores enumerados acima – informatização dos dados e Internet – uma vez que sem esses dados a capacidade de recomendar algo seria muito complicada. Se não houvesse uma plataforma capaz de registar as compras, vendas e ofertas de determinados artigos por utilizador os sistemas de recomendação não seriam eficazes uma vez que não teriam termos de comparação. [5]

2.1.3 Sistema de Recomendação

Os sistemas de recomendação são uma subárea de aprendizagem de máquina (*machine learning*) e são mecanismos de filtragem de informações inteligentes que reduzem o processo de tomada de decisões para apenas algumas propostas, tornando-se uma parte integrante da experiência do utilizador dentro de algumas das suas plataformas favoritas. Podem ser recomendados itens diversos como por exemplo livros, investimentos ou viagens.

É amplamente utilizado como uma estratégia de marketing, já que ao recomendar produtos que estejam alinhados ao interesse do utilizador, é mais provável que ele venha adquirir esse produto. Podemos ver estes sistemas em plataformas bem conhecidas e bem-sucedidas. Como seria se a Netflix não recomendasse filmes para assistir, ou a Amazon não oferecesse produtos alternativos para comprar? E sobre as ligações recomendadas no Facebook e LinkedIn, ou recomendações de notícias no Yahoo? Nenhum destes sites tinham o sucesso que têm agora se não tivessem um sistema de recomendação implementado. [2]

É possível fazer recomendações comparando as preferências de um utilizador ou um grupo de utilizadores. É também possível fazer recomendações através de itens com características idênticas aos que o utilizador já demonstrou interesse no passado.

Os sistemas de recomendação podem ser classificados basicamente em três categorias:

- **Baseada em conteúdo** – Recomenda ao utilizador produtos que sejam semelhantes ao que ele preferia no passado. A recomendação é feita a partir de etiquetas ou classificadores que os produtos contenham e a partir destas podemos recomendar produtos que tenham as mesmas etiquetas ou classificadores. Por exemplo uma pessoa que compre o livro "The Java Programming Language" certamente também estará interessada em livros de programação orientada a objetos e a livros cuja a linguagem de programação seja Java.
- **Filtragem Colaborativa** – Consiste em recomendação de itens que pessoas com o gosto semelhante preferiram no passado. Para isso utiliza-se toda a vizinhança do utilizador com a seguinte premissa "Se um utilizador A gostou do produto X e Y e o utilizador B gostou do produto X existe uma grande probabilidade de o utilizador B também goste do produto Y".

- **Sistemas Híbridos** – Como o próprio nome indica esta abordagem consiste em utilizar as vantagens dos dois sistemas acima descritos reduzindo assim a desvantagem de cada um deles complementando-se.

2.1.4 Medidas de avaliação

Embora exista um capítulo dedicado à avaliação destes sistemas existe uma necessidade de uma pequena contextualização do que existe no campo da avaliação de sistemas de recomendação. Uma forma eficaz para a avaliação de sistemas de recomendação é através da comparação das predições realizadas com as respetivas avaliações reais de utilizadores para as instâncias preditas.

A obtenção de medidas para medir o desempenho de um sistema de recomendação antes da utilização é fundamental para verificar se as recomendações se enquadram no objetivo da solução. De seguida são apresentadas quatro das principais métricas utilizadas na avaliação de sistemas de recomendação:

- Precisão (*Precision*)
- Recuperação (*Recall*)
- Medida F1 (*F1 Measure*)

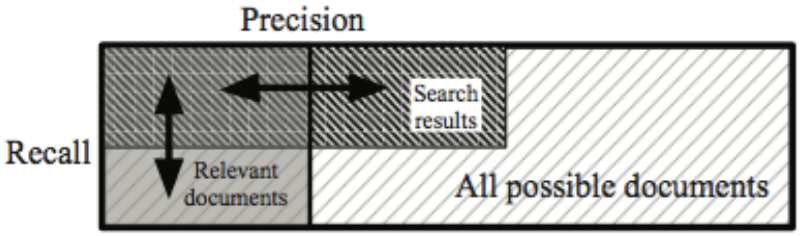

Figura 1 - Esquema exemplificativo de p*recision* e *recall*

No esquema da figura 1, onde pode-se observar a *precision* e o *recall* antes de passarmos para as respectivas formulas.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Figura 2 - Fórmula de Precisão

Na figura 2 esta representada a fórmula da qual se obtém a precisão. Resulta da divisão da união dos documentos relevantes com a lista de documentos produzidos por um

mecanismo de pesquisa com a lista de documentos produzidos por um mecanismo de pesquisa. [24]

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Figura 3 - Fórmula de Recuperação

Na figura 3 esta representada a fórmula da qual se obtém a *recall*. Resulta da divisão da união dos documentos relevantes com a lista de documentos produzidos por um mecanismo de pesquisa com os documentos relevantes. [24]

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figura 4 - Fórmula de Medida F1

Na figura 4 esta ilustrada a formula de *F1 Measure* que é o resultado do cálculo de *Precision* e *Recall.*

2.2 Análise de valor

Neste secção será apresentada a análise de valor bem como o modelo de Canvas deste negócio.

O *software* de recomendação proposto nesta tese é uma ferramenta que pretende melhorar a qualidade do serviço do *software* em que se insere. Para isso vai implementar um conjunto de regras e premissas que possam ser úteis aos utilizadores deste *software* recomendando tutoriais dentro do seu âmbito de pesquisa. Na proposta desta solução tivemos em conta as seguintes definições:

- Valor – Que é basicamente o que o cliente recebe ao adquirir o nosso produto.
- Valor percebido – É definido como a perceção do cliente de custo-benefício de um produto ou serviço.

Assim conseguimos entender a importância da definição de valor de qualquer tipo negócio. Para se ter sucesso num negócio, deve-se conhecer os seus contornos e limitações para que se possam ultrapassar. Deve estar muito bem definido o que o

cliente terá com a aquisição do negócio e as mais valias que deverá recolher face aos negócios disponíveis no mercado.

2.2.1 Valor

Uma vez que este sistema de recomendação irá ser utilizado especificamente para este *software,* os utilizadores que irão usar este sistema de recomendação serão única e exclusivamente os utilizadores do próprio *software* onde o sistema de recomendação estará inserido

O sistema de recomendação a implementar acrescentará valor à aplicação e por isso acrescentará também valor aos clientes. Após a visualização de um determinado tutorial de uma determinada área, este *software* terá a capacidade de recomendar a este utilizador outros tutoriais capazes de satisfazer as suas necessidades. O objetivo está bem claro nesta fase do projeto, pretendendo-se, assim, a satisfação do cliente no uso desta aplicação. Para isso existe a necessidade de implementação de uma componente de inteligência artificial para um resultado mais eficaz e mais rápido.

É possível identificar uma relação negócio-cliente da aplicação onde o *software* de gestão de tutoriais é claramente o negócio uma vez que disponibiliza a ferramenta para uso dos seus clientes. Os seus clientes serão os utilizadores do *software* uma vez que serão eles a tirar o usufruto desta aplicação.

Nesta análise de valor podemos identificar alguns tipos de valor atingidos com este sistema de recomendação na aplicação tais como a inovação, usabilidade e agilidade.

2.2.2 Possíveis cenários de negócio

Apesar do objetivo principal deste sistema de recomendação ser a aplicação direta de um sistema de recomendação, não impossibilita que no futuro não se possa integrar em outros sistemas do género. O sistema de recomendação poderá ser integrado em quase todos os sistemas existentes porque em todos eles há decisões a tomar, e havendo decisões, há uma utilidade das recomendações para que se possa atingir um determinado objetivo mais fácil e rapidamente.

Não invalida a aplicação de um modelo de negócio neste sistema de recomendação. No módulo de competências de análise de valor foi estudado o processo de negociação Win

Win que segundo Jo Joy[25] as partes envolvidas deverão ter em conta os procedimentos seguintes:

Definir claramente as expetativas e objetivos;

- A. Definir claramente as expetativas e objetivos. Neste ponto é necessário saber aquilo que se pretende, se se está a projetar um sistema, é fundamental conhecer o mínimo para o poder passar da idealização para a realização. Pretende-se, desse modo, um sistema de recomendação eficiente capaz de satisfazer o utilizador;
- B. Identificar pontos indiscutíveis. Como indiscutível tem-se o objetivo, o sistema de recomendação etambém o prazo de entrega que foi definido inicialmente;
- C. Prever contra-ofertas que possa fazer ou receber. À medida que o desenvolvimento vai ocorrendo é normal que haja alguns imprevistos e é neste ponto que se pode discutir esses imprevistos. Se o cliente quiser outro tipo de funcionalidade tais como, módulo de estatística de recomendados será negociado mais tempo para o satisfazer.
- D. Conhecer todos os pormenores e todos os assuntos envolvidos. Neste ponto é preciso ter acesso a todos os dados relativamente ao *software* e aos seus utilizadores uma vez que estes mesmos dados serão o ponto de criação para elaborar o conjunto de dados;
- E. Antecipar o que a outra parte deseja. Por vezes, devido à falta de algum conhecimento técnico do cliente, é preciso prever o que a outra parte pretende. Neste caso em concreto não houve muita dificuldade porque os objetivos foram discutidos em conjunto;
- F. Decidir qual é o máximo / mínimo que vai receber ou dar. No inicio deste projeto, e mesmo durante o mesmo, foi sempre definido o mínimo exigido, que era o sistema de recomendação, bem como o máximo esperado, que seria o próprio sistema de recomendação bem como a interligação entre os sistemas;
- G. Estar preparado para explicar o porquê desse máximo / mínimo que vai receber ou dar. Este ponto vai estar explicado no final deste relatório com um capítulo exclusivo;

O ponto A e B foram definidos no início deste projeto. O ponto C, D e E, embora se conhecesse quase todo na totalidade, existia algumas variáveis que podiam alterar-se no desenvolvimento do projeto ficando assim definido na conclusão do mesmo. O ponto F e G E que tinha como finalidade a comparação dos resultados obtidos com os resultados esperados, conclusão essa que foi obtida aquando a elaboração deste documento.

2.2.3 Modelo de Canvas

Na figura seguinte está representado o modelo de Canvas para este negócio.

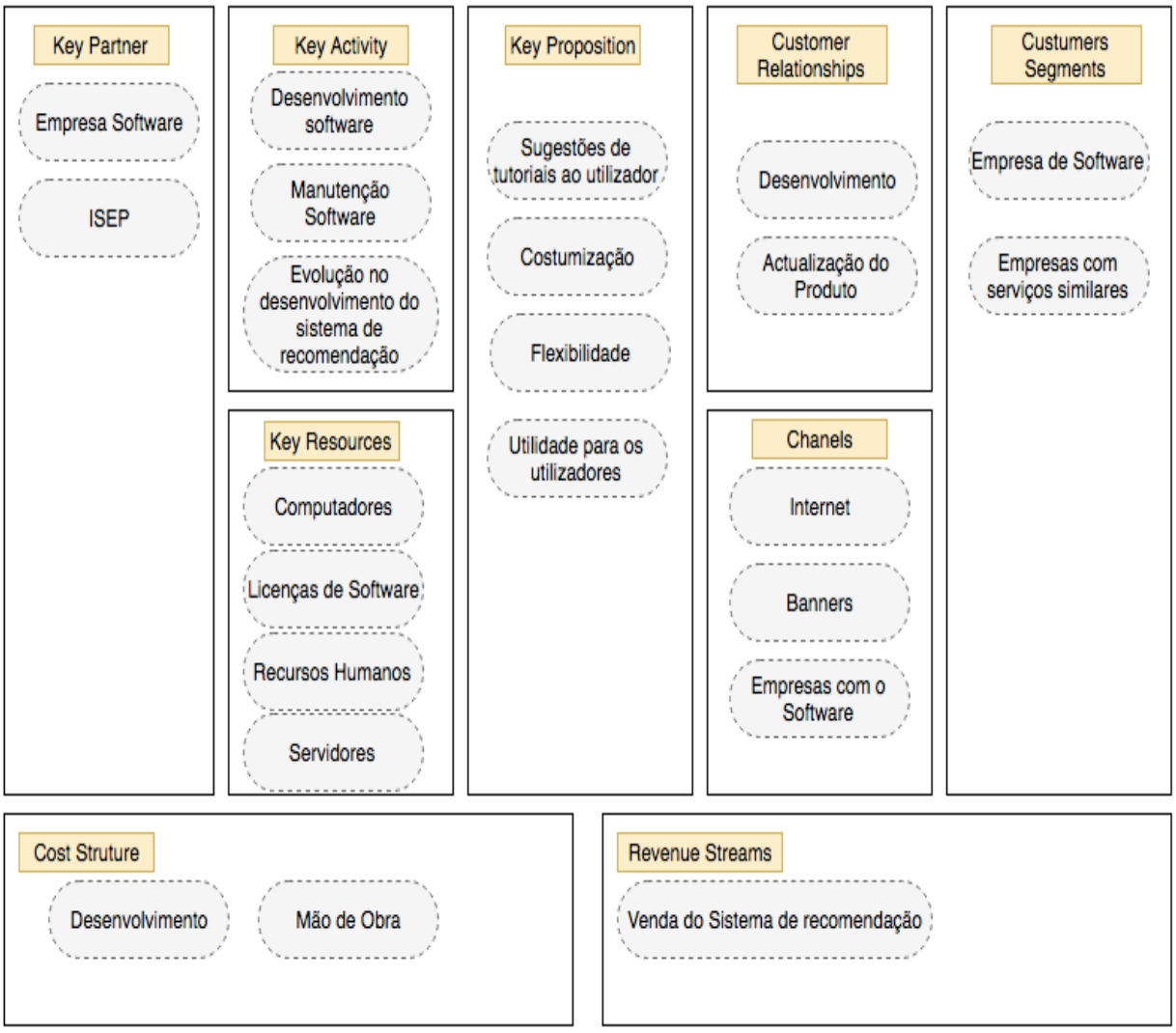

Figura 5 –Modelo de Canvas

2.2.4 De que forma pode analisar/modelar e/ou quantificar a criação de valor

Existem várias formas para se analisar e quantificar a criação de valor. A mais rudimentar e mais simples será a satisfação do cliente com um simples inquérito sabendo se a qualidade de *software* melhorou, piorou ou manteve-se na perspetiva de

utilizador. Nesta abordagem podemos considerar a quantidades de "cliques" que o utilizador fez até conseguir encontrar um tutorial que correspondesse as suas expetativas.

Uma outra forma de analisar e quantificar a criação de valor seria, por exemplo, a teoria de Análise Hierárquica de Processos (AHP) desenvolvida por Thomas Saaty. Dos métodos multicritérios, o método de AHP é o mais popular. A vantagem principal do AHP é a facilidade inerente ao método de manipular fatores intangíveis, fatores esses determinantes no processo de decisões. Também os cálculos matemáticos são mais simplificados e compreensíveis, fazendo desta técnica ideal para ser empregada em processos de avaliação. Segundo Thomas Saaty[22], a aplicação da AHP é divida em três passos:

- 1º) Representar um problema através de uma estrutura hierárquica, onde o primeiro nível da hierarquia representa o objetivo, seguindo de critérios, subcritérios nos níveis intermediários e, finalmente, as alternativas disponíveis;
- 2º) Fazer comparação par a par;
- 3º) Derivar a prioridade ou valor de preferência para as alternativas.

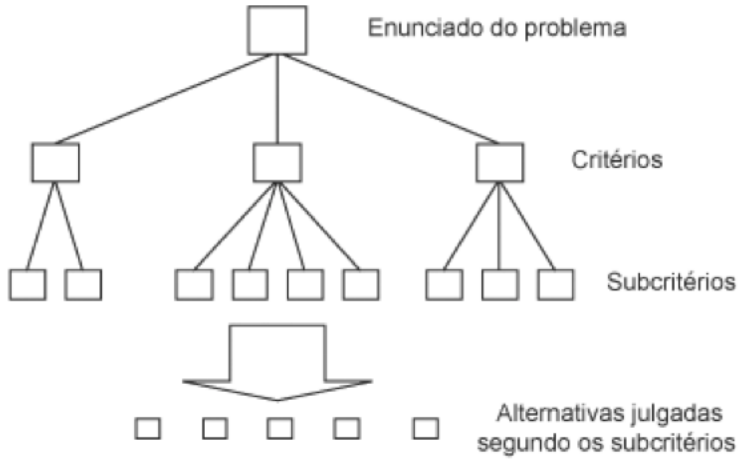

Figura 6 - Análise Hierárquica de Processos

2.3 Estado da Arte (Em soluções/abordagens existentes)

No início dos anos 90, quando a Internet passou a ser utilizada de forma massiva como fonte de informação, essa mesma informação começou a crescer de uma forma exponencial criando um problema que não existia até a data, enorme quantidade de informação que precisava de endereçamento.

Em muitos *sites* que apresentam uma grande variedade de conteúdos (como artigos, notícias, ou itens de compra) descobriu-se que os utilizadores tinham dificuldades em encontrar itens do seu interesse. Os sistemas de recomendação segundo Resnick e Varian, ajudam os utilizadores a limitar a sua pesquisa através do fornecimento de uma lista de itens que pode interessar a um utilizador específico. Diferentes abordagens foram sugeridas para fornecer recomendações significativas para os utilizadores. Abordagens provenientes do campo de recuperação de informação dependem do conteúdo dos itens (como descrição, categoria, título, autor) e, portanto, são conhecidas como recomendações baseadas em conteúdo segundo Mooney e Roy. Estes métodos utilizam alguns pontos de semelhança para combinar itens com base no seu conteúdo. Segundo esses pontos, uma lista de itens semelhantes aos do utilizador selecionado anteriormente pode ser fornecida.

Sistemas de recomendação baseadas no conhecimento segundo Burke, representando um passo mais longe, usando conhecimento mais profundo sobre o utilizador e o seu domínio. Em particular, o utilizador é capaz de introduzir informação explícita sobre as suas preferências. Assim, por exemplo, o utilizador pode especificar o interesse em livros de recomendação em linguagem Java, e o sistema pode sugerir o livro "Mahout in Action".

Outra possibilidade é evitar o uso de informações sobre o conteúdo, mas sim usar o histórico de dados recolhidos de outros utilizadores. Estes métodos são amplamente conhecido como filtragem colaborativa segundo Resnick.

Finalmente, alguns sistemas tentam criar modelos híbridos que combinam filtragem colaborativa e recomendações baseadas em conteúdo segundo Balabanovic e Shoham, 1997 e mais tarde por Burke, 2002[27].

2.3.1 Casos Estudados

O GroupLens, laboratório de pesquisa de interação maquina homem, usou a filtragem colaborativa para filtrar artigos que pudessem ser interessantes para um determinado utilizador. Para isso o utilizador apenas tinha que fazer avaliações de artigos do seu interesse. As combinações do resultado das avaliações de um utilizador com outro eram capazes de fornecer resultados personalizados [6].

A investigação sobre estes algoritmos de recomendação ganhou outra importância quando a Netflix lançou o prémio Netflix para melhorar o sistema de recomendação que é usado internamente. O objetivo era bastante simples, a construção de um algoritmo capaz de ultrapassar o atual para isso tinha que passar nos testes *offline* por 10%. E devido ao elevado valor do prémio para que o conseguisse (um milhão de dólares) levou a que estes sistemas passassem para outro patamar devido a importância que têm nos nossos dias [6].

Uma das abordagens possíveis para aumentar a precisão da previsão é através de funções de similaridade. As funções de similaridade são utilizadas em sistemas de recomendação para combinar e tratar um conjunto elevado de dados e representar esse mesmo conjunto num único valor representativo a semelhança de dois utilizadores acerca de um único item [7].

Num determinado estudo verificou-se que quase metade dos seus dados eram conhecidos mas que pouco mais de metade destes dados possuíam informação sobre testes/treino. É por isso de esperar que algumas organizações não exponham os seus dados fazendo assim uma exposição detalhada de como estes dados foram usados, tornando assim alguns dos testes inviáveis como era de esperar [8].

2.3.2 Medidas de Similaridade

A medida de similaridades é uma função que quantifica a semelhança entre dois objetos. Não existe uma definição única de medidas de similaridade. No entanto, os resultados da similaridade avaliam-se quando se obtém grandes valores para objetos similares e valores zero ou negativos para objetos diferentes.

Existem varias medidas de similaridade para usar neste estudo [9]:

- Distância Euclidiana: uma maneira simples para determinar a similaridade é calcular a distância euclidiana entre dois objetos de dados [15];
- Distância Minkowski;
- Similaridade de coseno ou L2 Norm, medida como o ângulo entre os dois vetores. [15];
- Correlação de Spearman, que se trata de uma medida de correlação não-paramétrica, ou seja, avalia uma função monótona arbitrária que pode ser a descrição da relação entre duas variáveis [26];
- Coeficiente de Tanimoto – é um coeficiente de relação entre o tamanho da interseção entre os itens de dois utilizadores para um total de preferência dos utilizadores [13];
- Similaridade de Log Likelihood – é muito idêntico ao Tanimoto uma vez que também se foca na interseção entre a preferência de dois utilizadores mas ao contrário questiona-se sobre quão improvável é que estes dois utilizadores não estão tendo a interseção dos itens preferidos [13];
- Correlação de Pearson – mede o grau da correlação entre duas variáveis de escala métrica [14];

2.3.3 Métricas

A obtenção de métricas para medir o desempenho de um sistema de recomendação antes da utilização é fundamental para verificar se as recomendações se enquadram no objetivo da solução. As principais métricas são:

- Precisão (Precision): A precisão de um sistema indica a quantidade de itens recomendados que são do interesse do utilizador em relação ao conjunto de todos os itens que lhe são recomendados. Indica também quanto uma predição se aproxima da avaliação real do utilizador;
- Recuperação (Recall): O índice de recuperação indica a quantidade de itens de interesse do utilizador que aparecem na lista de recomendações;
- Cobertura (Coverage): A cobertura é a proporção de itens que são passíveis de serem recomendados em relação ao conjunto de todos os itens conhecidos pelo sistema de recomendação;
- F-measure: Esta é uma métrica que combina a precisão (precision) e a recuperação (recall).

2.3.4 Sumário

Em suma, como se pode verificar, já existe alguma informação sobre a aplicação de sistemas de recomendação, o seu uso e manipulação de dados. Embora existam muitos estudos neste campo a conclusão que se pode retirar, através de vários artigos lidos, é a inexistência de muitos sistemas de recomendação de código aberto e com dados conhecidos existindo apenas a descrição de uma forma pormenorizada e alguns detalhes de como foi implementado.

Existe também nestes artigos uma forte vertente de avaliação do resultado usando sobretudo casos de treino/testes dividindo a fonte de dados e treinando o algoritmo para obter resultados concretos.

2.4 Estado da arte tecnológico

Os sistemas de recomendação já existem há alguns anos e têm ganho alguma notoriedade nesta ultima década devido aos inúmeros usos que podemos dar a estes sistemas.

Nesta fase inicial de pesquisa para encontrar o *software* que irá de encontro às necessidades, foi feita uma pesquisa sobre o *software* existente dando sempre preferência por sistemas de código aberto para que fosse possível adaptar parte das bibliotecas para otimizar o sistema. Optou-se por fazer um estudo mais aprofundado do Apache Mahout, GraphLab, LensKit, MyMediaLite e EasyRec bom como uma contextualização da base de dados usada.

2.4.1 GraphLab

O GraphLab, baseado em C++, foi criado na Universidade de Carnegie Mellon em 2009 e é caracterizado pela capacidade de computação paralela iterativa.

GraphLab é uma estrutura de aprendizagem extensível que permite que programadores sejam capazes de desenvolver e construir mais facilmente aplicações e serviços inteligentes de grande escala. Inclui para isso estruturas distribuídas de dados e bibliotecas ricas para a transformação e manipulação de dados, ferramentas de aprendizado de máquina orientados para tarefas escaláveis para criar, avaliar e melhorar os modelos de aprendizado.

GraphLab possui varias caraterísticas em que as mais relevantes são:

- Capacidade de criar modelos sofisticados
- Inteligência baseada em tarefas
- Capacidade de interagir com uma capacidade enorme de dados
- Capacidade de explicar facilmente o resultado dos seus dados
- Capacidade de estender a API (Application Programming Interface)

Como foi referido nos pontos anteriores, a capacidade de estender a API é sem dúvida um ponto forte deste *software* para além disso, esta API é projetada para ser fácil de usar para iniciantes, mas suficientemente flexível para os utilizadores de dados especializados [10].

GraphLab oferece um grande conjunto de módulos capazes de satisfazer as necessidades da maioria dos utilizadores tais como:

- Recommender - Construção de sistemas de recomendação;
- Data matching - Capacidade de encontrar itens semelhantes;
- Deep learning - Construir e treinar redes profundas;
- Graph analysis - Capacidade de examinar gráficos para encontrar comunidades escondidas;
- Sentiment analysis - Aprendizagem em fóruns, revisões nos Media;
- Frequent pattern mining - Extrair conjuntos frequentes a partir de dados de registo de eventos;
- Churn prediction - Previsão que tipo de produtos o utilizador poderá comprar;
- Image analysis with deep learning - Construção de modelos de rede;
- Regression - Realizar análise de regressão usando modelos lineares ou árvores de regressão;
- Classification - Classificação por meio de regressão logística, impulsionada por árvores de decisão, suportar máquinas de vetores, ou redes neurais;
- Nearest neighbors - Definir e objetos de consulta com base no recurso de similaridade.

2.4.2 Apache Mahout

Apache Mahout é um projeto da fundação Apache Software e é implementado sob o Apache Hadoop. É também usado para criar implementações de algoritmos de

aprendizado de máquina escaláveis e distribuídas que estão focados nas áreas de *clustering*, classificação e filtragem colaborativa para encontrar pontos em comum em grandes grupos de dados ou para marcação de grandes volumes de conteúdo *web* tais como:[11]

- Filtragem colaborativa - Analisa o comportamento do utilizador e faz recomendações de produtos;
- Clustering - Leva itens de uma determinada classe (tais como páginas *web* ou artigos de jornal) e organiza-os em grupos que ocorrem naturalmente, de tal forma que os itens pertencentes a um mesmo grupo são semelhantes entre si;
- Classificação - Aprende com categorizações existentes e, em seguida, atribui itens não classificados para a categoria melhor.

Mahout contém bibliotecas Java para algoritmos matemáticos comuns e operações focada em estatísticas e álgebra linear, bem como coleções de Java primitivos. Apache Mahout suporta os seguintes algoritmos de medida de similaridade [12]:

- Similaridade da distância euclidiana;
- Similaridade City-bloc;
- Similaridade coseno;
- Similaridade da correlação de Pearson;
- Similaridade da correlação de Spearman;
- Coeficiete de similaridade Tanimoto;
- Similaridade de Log likelihood.

Este *software* foca-se, sobretudo, em aprendizagem de máquina e este projeto visa tornar uma ferramenta poderosa para a construção de aplicações inteligentes de uma maneira fácil o mais rapidamente possível.

O Mahout é escalável ao nível:

- Grandes conjuntos de dados - os algoritmos básicos são implementados em grandes sistemas escaláveis e distribuídos;
- Suportar diferentes casos de negócios - distribuído sob Apache Software License;
- comunidade escalável - há uma grande comunidade para facilitar as discussões sobre o projeto e seus potenciais casos de uso.

2.4.3 EasyRec

EasyRec é um sistema de recomendação de código aberto de fácil manuseamento capaz de recomendações de alta qualidade para qualquer *site* com necessidades de personalização. O acesso a este sistema é fornecido através de serviços Web, garantindo uma integração fácil e rápida.

Umas das caraterísticas mais importantes deste *software* é um conjunto de estatísticas de uso e outros negócios de informações relevantes apresentadas por meio de uma interface de administração e gestão. Para além disso o administrador do sistema EasyRec é suportado por uma variedade de funções de administração e configuração, incluindo a importação manual ou adaptação de regras de negócios.

EasyRec oferece também um conjunto de serviços de personalização tais como:

- Recomendações despersonalizado – Os outros utilizadores também compraram/visualizaram;
- Recomendação personalizada de acordo com as preferências individuais;
- Rankings como "top itens comprados", "top mais vistos", etc.;
- Agrupamento manual dos itens;
- Usando tipos de itens (livro, dvd's, MP3) nas recomendações específicas do tipo.

2.4.4 MyMediaLitle

É uma biblioteca documentada bastante leve de algoritmos de recomendação. À semelhança dos anterior é também um *software* de código aberto que pode ser usado e distribuído sob os termos da GNU (General Public License). Esta biblioteca explora dois cenários na filtragem colaborativa tais como [20]:

- Predição de classificação (como por exemplo as avaliações por estrelas que existem em alguns sítios na internet);
- Previsão com base em cliques de utilizadores.

O MyMediaLitle tem à disposição vários métodos de recomendação [21]:

- Inclui rotinas de avaliação para previsão e classificação;
- Portabilidade: Escrito em C #, para a plataforma .NET;
- Serialização: salvar e recarregar modelos de recomendação;

- Atualizações incrementais em tempo real para muitas recomendações;
- Diversificação baseada em atributo de listas de recomendação.

2.4.5 LensKit

LensKit é um *software* de código aberto Java que fornece um conjunto de algoritmos básicos, desenvolvido principalmente por pesquisadores da Universidade do Estado de Texas e GroupLens Research na Universidade de Minnesota, com a contribuição por varias pessoas à volta do mundo. É apoiada por algumas organizações e está disponível também sob os termos da GNU.

2.4.6 MongoDB

De uma forma muito resumida e sucinta para contextualização MongoDB é uma aplicação de código aberto que ao contrário das bases de dados mais utilizadas não é orientada a objetos. Escrita em linguagem de programação C++ é orientada a documentos facilitando o agrupamento de informação, abrindo portas a muitas aplicações que até aqui não tinham sido conseguidas por bases de dados relacionais [18].

2.5 Avaliação de soluções

Uma grandeza a ser testada será o tempo. O tempo nos dias de hoje poderá ser a grandeza com maior impacto. Neste momento as empresas não querem só soluções, querem as soluções em tempo útil. Para se tratar e analisar um conjunto significativo de dados e dar resposta, é preciso otimizar o mais possível o *software* para que se obtenha uma resposta em tempo útil para ajudar o utilizador nas suas pesquisas. Para isso vamos utilizar as medidas do tempo para garantir uma maior satisfação do utilizador.

Para se verificar que o sistema de recomendação está a devolver os resultados pretendidos é necessário fazer uma avaliação do mesmo. Para ter mesmo a certeza da qualidade dos dados seria necessário fazer testes com utilizadores reais. No entanto, para termos uma avaliação aproximada podemos recorrer a testes de validação cruzada que consiste em retirar uma amostra do nosso conjunto de dados e colocar num ficheiro à parte e gerar um novo conjunto de dados para cada utilizador sem essa amostra. Com isto pretende-se avaliar a capacidade que o sistema de recomendação tem em obter da amostra que foi retirada. A obtenção de métricas para medir o desempenho de um

sistema de recomendação antes da utilização é fundamental para verificar se as recomendações se enquadram no objetivo da solução. As métricas que foram utilizadas são:

- Precisão (Precision);
- Recuperação (Recall;
- F-measure.

2.6 Sumário

Um sistema de recomendação tem que ter a capacidade para aumentar a utilidade do produto num longo prazo.

Tabela 1 - *Frameworks* de recomendação

Nome	Licença	Linguagem	Tipo
CofiRank	MPL	C++	Filtragem Colaborativa
EasyRec	GPL v2	Java	Recomendação
GraphLab	Apache 2.0	C++	Computação de alta performance
Lenskit	LGPL v2.1	Java	Recomendação
Mahout	Apache 2.0	Java	General ML
MyMediaLitle	GPL	C#/Java	Recomendação

Como se pode ver na tabela 1, parcialmente retirada do livro "Recommendation System in Software Engineering" existe uma quantidade significativa de *software* de recomendação. Nesta tabela só contém as mais importantes. À frente de cada sistema pode-se ver a informação relativamente à sua licença, linguagem e tipo.

Embora existam mais sistemas de recomendação do que os enumerados acima, grande parte dos sistemas não foram considerados para análise dada a sua fraca documentação. Versões recentes, a pouca estabilidade e em alguns casos com linguagens de programação fora da zona de conforto.

Vai-se usar a comparação usada no artigo [8] para tentar demonstrar o comportamento de algumas destas *frameworks*. Neste exemplo o autor comparou as *frameworks* Mahout, Lenskit e o MyMediaLite.

Foi usado o *dataset* do Movielens 100k para correr a avaliação em cada algoritmo, em baixo estão representados os resultados para as diferentes métricas utilizadas na avaliação do autor em que nas colunas as iniciais correspondem:

- AM = Apache Mahout;
- LK = Lenskit;
- MML = MyMediaLite;
- gl = global;
- pu = por utilizador;
- cv = validação cruzada;
- RT = rácio.

E por cada linha estão presentes as iniciais correspondentes:

- UB = baseada em utilizadores;
- Pea = Correlação de Pearson;
- Cos = semelhança de coseno;
- IB = baseado no item.

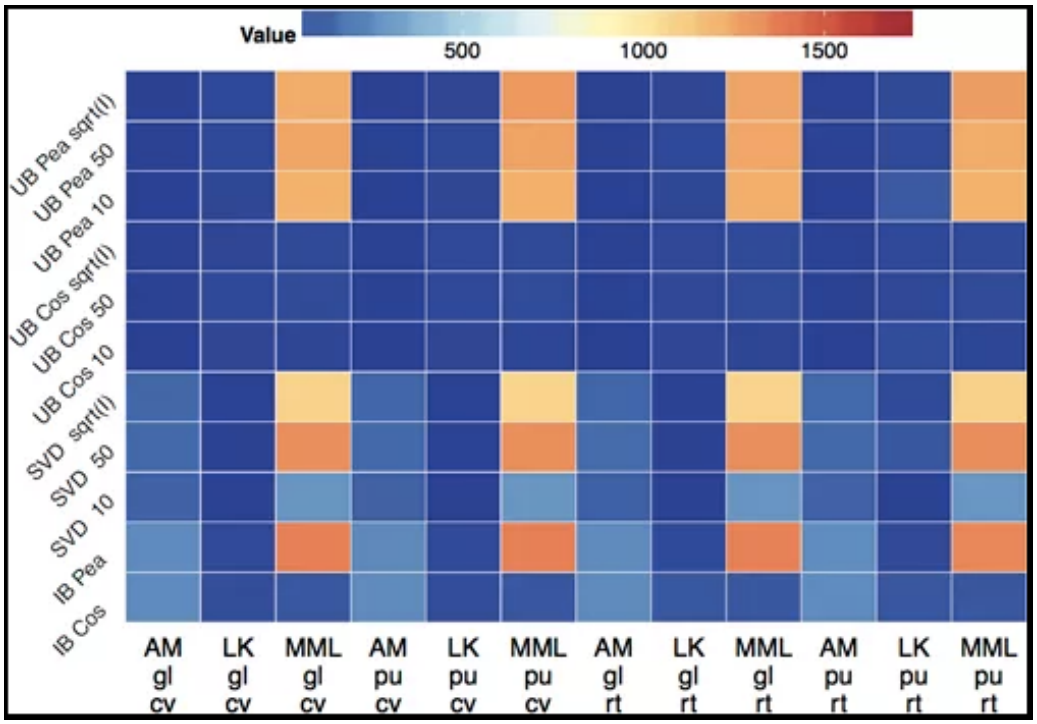

Figura 7 - Tabela de tempos [8]

Podemos observar na figura 7 como diferem os tempos em cada algoritmo. O MyMediaLite parece ser geralmente um pouco mais lento do que os restantes. De notar que todas as experiências foram realizadas em dois terminais de trabalho idênticos executando Linux Fedora. O MyMediaLite como é escrito em C # foi executado usando o framework Mono [8].

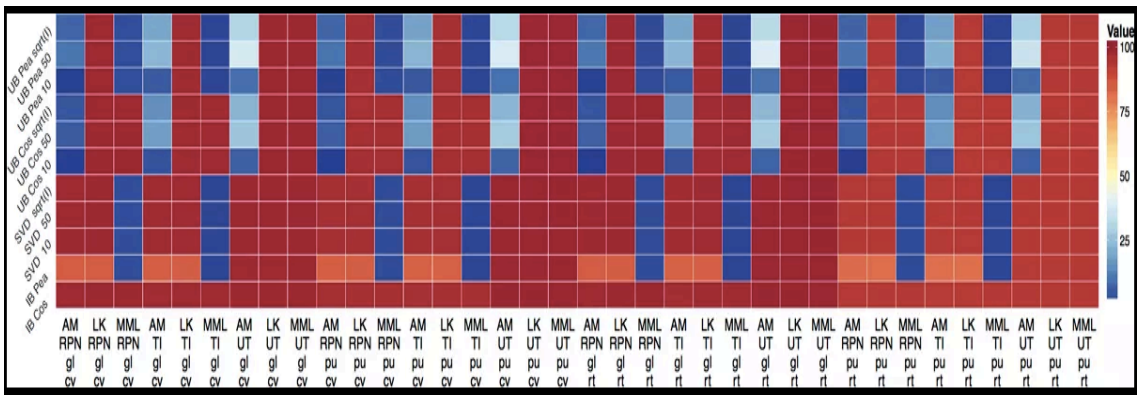

Figura 8 - Tabela de coberturas [8]

Quando se trata de catalogar cobertura, ou seja, a fração de itens no conjunto de treino que está realmente a ser recomendado, vemos novamente os três têm desempenhos diferentes. Mahout e MyMediaLite parecem ter problemas em conseguir uma elevada cobertura (nota: cor azul indica baixa cobertura) [8].

Com estes dados que vêm do trabalho documentado num artigo com o título acima descrito podemos à partida descartar o MyMediaLite devido com as suas limitações ao nível de cobertura de e desempenho em relação os outros dois.

Tanto o Mahout como o Graphlab são dos sistemas com mais documentação e fóruns para esclarecimento de dúvidas. Embora EasyRec forneça um serviço muito viável e muito bom com um módulo de estatística muito completo, carece um bocado de documentação o que me levou a abandonar este sistema de recomendação deixando a minha decisão final entre o Mahout e o GraphLab.

As grandes vantagens do Mahout é o facto de ser escalável, ter uma licença Apache, uma grande e boa comunidade de apoio e uma boa documentação. Mas um dos aspetos que mais pesaram nesta decisão foi mesmo o módulo de avaliação, que é capaz de avaliar uma configuração diferente do mesmo algoritmo, para isso só precisamos de atualizar o parâmetro e correr novamente o nosso sistema.

E são por estas razões que optei por usar o Apache Mahout para desenvolver este sistema de recomendação, o que não me impede no futuro de adaptar o outro sistema e de tirar as minhas próprias conclusões.

A avaliação é, também parte importante deste projeto, toma contornos muito relevantes neste tipo de sistema pois é através deles que podemos concluir o sucesso do nosso sistema de recomendação. Devido à aplicação não ter uma base de dados com a quantidade de dados ideal para este tipo de recomendação, por ser uma aplicação relativamente recente, vai ser necessário injetar mais dados para que se possa obter resultados mais expressivos e para que os métodos de avaliação sejam capazes de obter resultados aceitáveis.

3 Desenho

Neste capítulo é descrito o *design* usado para desenvolver a nossa solução. Foi a arquitetura desenhada no início deste projeto.

3.1 Arquitetura

Neste ponto vai estar demonstrado a arquitetura do nosso sistema. Aqui estão patentes as peças que compõem o sistema.

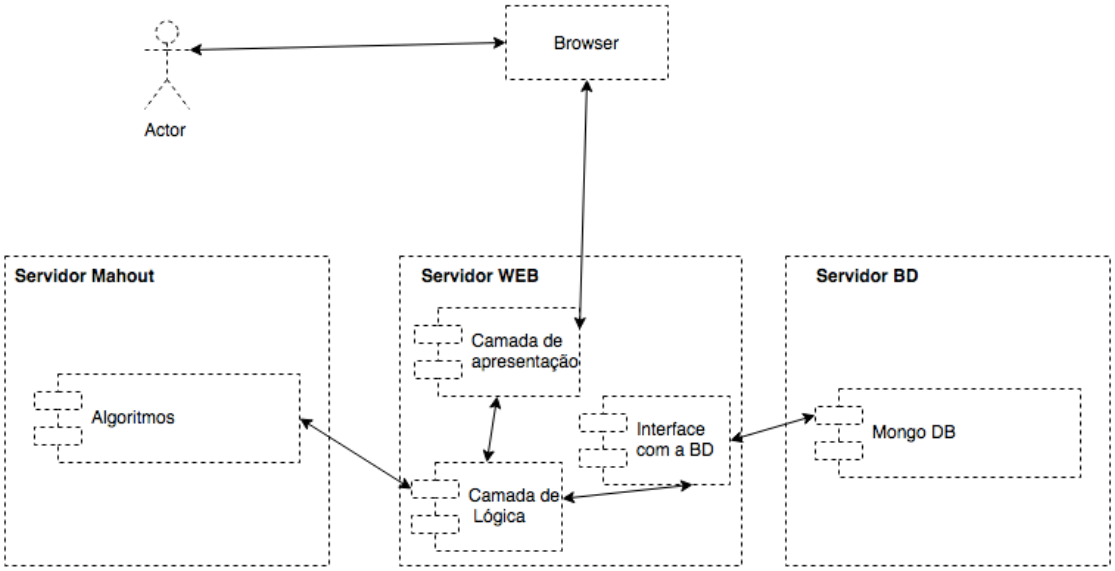

Figura 9 - Diagrama de componentes inicial

Na figura acima está representado o diagrama de componentes simplificado de como vão funcionar e interagir todas a peças do nosso sistema de recomendação.

Após a interação dos utilizadores a aplicação vai recolher toda a informação relativamente a todos os utilizadores; após essa recolha o sistema vai manipular os dados ajustando-os aos parâmetros de entrada do sistema de recomendação. De seguida é enviado ao sistema de recomendação o nosso conjunto de dados e é neste ponto que vai estar concentrado a maior parte do esforço; é aqui que se vai aplicar todas as métricas e cálculos. O motor do nosso sistema de recomendação estará situado no servidor de Mahout e é aqui que se vai aplicar todas as métricas e cálculos. Após cálculos de recomendação, o conjunto de dados é devolvido à aplicação com toda a

informação relativa ao utilizador em questão. Após passagem para a aplicação, a informação é processada e ajustada à necessidade do *software* para que seja mostrada ao utilizador.

O sistema de recomendação apenas aceita números inteiros pelo que é preciso fazer um mapeamento para que se possa transformar id's de pessoas (descritos em hexadecimal) e id's de tutoriais (à semelhança dos utilizadores também descritos em hexadecimal) em números inteiros. Para isso foi necessário implementar uma funcionalidade de mapeamento para que, quando os dados viessem do sistema de recomendação se possa usar esta mesma função para colocar os id's de modo a que sejam legíveis pela aplicação. Toda esta implementação está a cargo da camada lógica.

O sistema de recomendação representado na figura 9 como Servidor Mahout estará num servidor externo e irá usar uma arquitetura REST para efetuar as comunicações entre a aplicação. Esta irá ser uma das soluções a testar uma vez que devido à quantidade significativa de dados, o recurso a uma arquitetura REST poderá não ser a melhor solução. Se através da nossa avaliação não obtivermos resultados satisfatórios poderemos ter que alterar a arquitetura para uns semelhantes representados no diagrama de instalação seguinte:

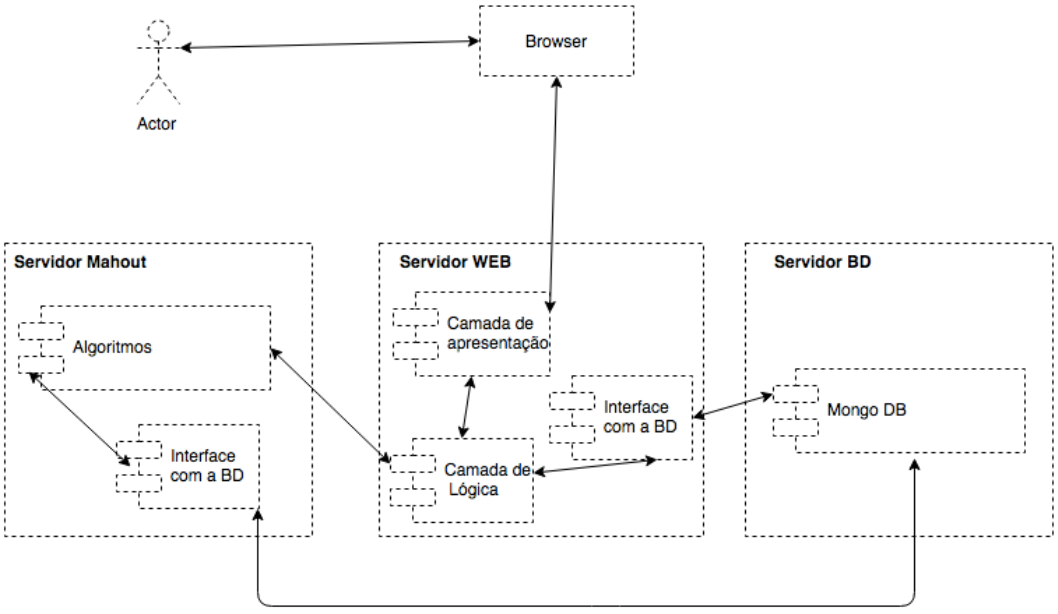

Figura 10 - Diagrama de instalação alternativa

Como podemos verificar na figura 10 este sistema é muito semelhante ao da figura anterior apenas muda o modo com que se obtém os dados. Enquanto no sistema inicial os dados eram lidos e tratados pela aplicação neste segundo esquema, e após o *trigger* disputado pela mesma aplicação, é o nosso sistema de recomendação que vai fazer a recolha e tratamento dos dados podendo assim ser mais eficaz numa perspetiva temporal.

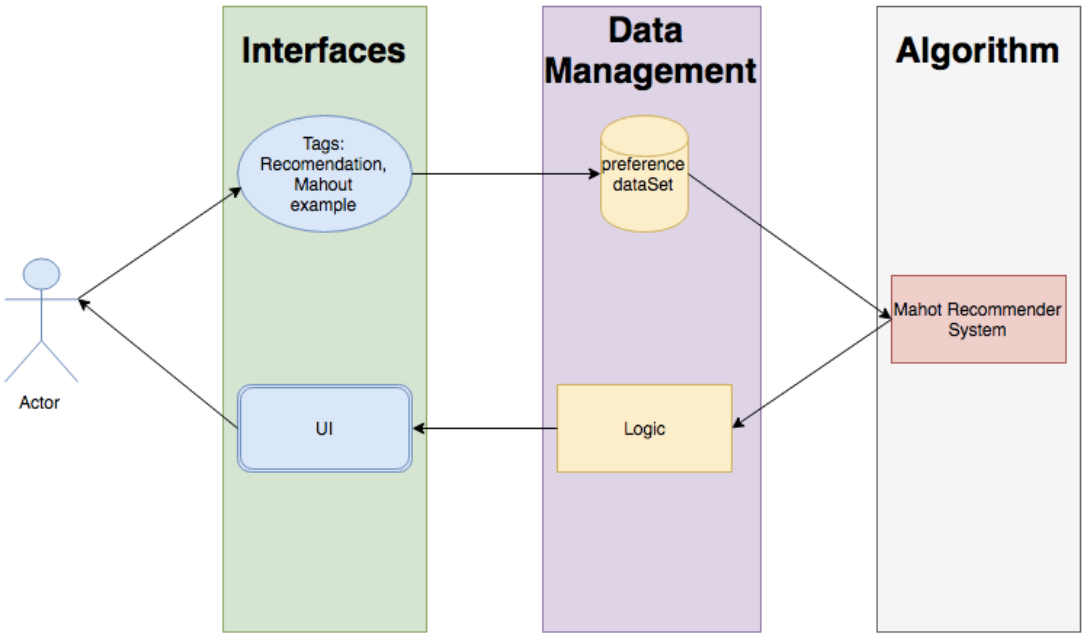

Figura 11 - Sistema de Recomendação (idealização)

Na figura 11 está representeada a idealização deste sistema de recomendação. Esta ilustração é composta por:

- Actor – Os utilizadores têm preferência por elementos, neste caso designados por tutoriais. O sistema de recomendação vai ter como objetivo o auxílio do utilizador na escolha do tutorial mais adequado;
- Interfaces – O sistema de recomendação vai basear o seu sistema com base nas etiquetas ou mesmo tutoriais que este utilizador tenha visto ou recomendado criando assim força nas suas ligações que serão importantes para um sistema de recomendações eficiente;
- Preference DataSet – A medida que o utilizador vai consultando ou recomendado tutoriais, existe uma base de dados que vai gravando e atualizando as preferências deste utilizador;

- Algorithm – A parte mais importante deste sistema que consiste na criação de um ou mais algoritmos que sejam capazes de recomendar tutoriais com base na informação de um determinado utilizador;
- Logic – Embora a parte da lógica esteja presente em todas as etapas deste sistema de recomendação é neste ponto que seria mais importante ter uma lógica presente. Por exemplo, se for recomendado a um utilizador um determinado tutorial que tenha sido apagado ou mesmo um tutorial que tenha sido reportado por outros utilizador como não estando relacionado com o título ou mesmo catalogado corretamente neste ponto da lógica seria descartado para esta recomendação;
- UI – A parte final desta etapa será também importante pois será a parte visível do nosso sistema de recomendação;

3.2 Diagrama de sequência

De seguida é apresentado um possível caso de uso relativamente a um exemplo de interação com o programa de gestão de tutoriais com o sistema de recomendação implementado:

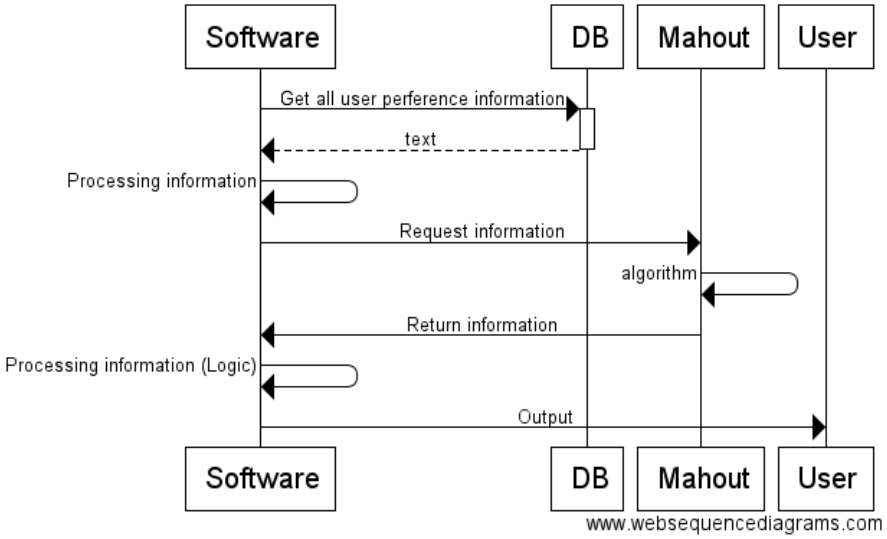

Figura 12 – Diagrama de sequência da aplicação final

Na figura 12 está representado um diagrama de sequência representativo de um ciclo completo de uma recomendação para um determinado utilizador.

Após o *trigger,* a aplicação pede à base de dados de toda a informação relativa aos utilizadores e suas preferências. De seguida processa a informação enviando para o nosso sistema de recomendação. Após a receção dos dados tratados o sistema de recomendação vai aplicar os algoritmos e o resultado desse processamento é retornado à aplicação fazendo as suas respetivas transformações.

3.3 Caso de Uso

Neste subcapítulo será apresentado um caso de uso relativamente à interação do utilizador com o sistema.

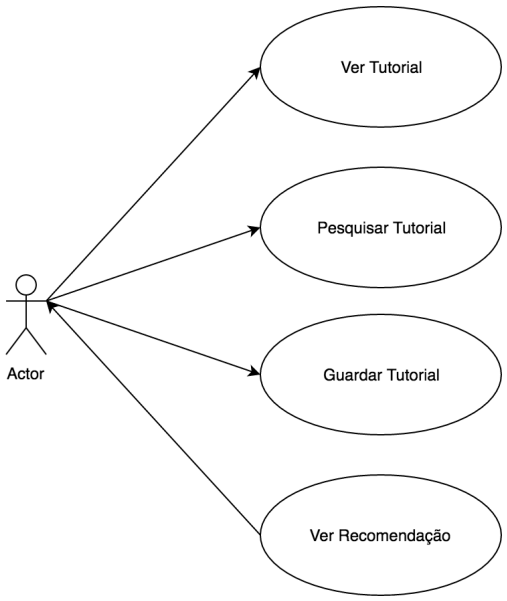

Figura 13 - Case de Uso do sistema de recomendação

Na figura 13 está ilustrada de uma forma simples o caso de uso deste sistema de recomendação bem como a interação com o *software*. Neste sistema o utilizador terá ações como ver tutorial, pesquisar tutorial, e guardar tutorial, informação essa que após ser guardada vai ser tratada de forma que o nosso algoritmo possa gerar recomendações validas com base nos tutoriais visualizados, pesquisados e guardados por este utilizador. Após o algoritmo correr é devolvido ao utilizador os tutoriais recomendados.

3.4 Base de dados

Não será envolvido nenhum esforço a curto prazo a nível de bases de dados uma vez que se irá usar a base de dados da aplicação. Tendo como objetivo usar o sistema de

recomendação como um sistema externo à aplicação, tem todo o sentido que seja a mesma o responsável por todo o tratamento CRUD da informação na base de dados. É também uma mais valia a base de dados usada na aplicação, *mongoDB*, uma vez que é uma base de dados disposta em coleções bastante rápida para um grande conjunto de dados.

3.5 Sumário

É uma arquitetura mantida deste o início deste trabalho, que denota o conhecimento inicial daquilo que se pretende. A base da arquitetura não foge ao que existe nos dias de hoje nos sistemas de recomendação. Esta arquitetura foi a base para o desenvolvimento do trabalho como vai ser explicado no capítulo seguinte.

4 Implementação

O capítulo presente serve para elaborar e guiar todo o processo usado para chegar às conclusões que eram propostas.

Após pesquisa de vários artigos e documentos existentes sobre o assunto, sistemas de recomendação, chegou-se a conclusão que, embora existam inúmeros sistemas de recomendação documentados nenhum deles se aproxima para o objetivo pretendido porque pretende-se adaptar a um sistema único.

Como foi descrito nos capítulos anteriores este sistema de recomendação terá um uso imediato muito específico, a capacidade de recomendar tutoriais a outros utilizadores, baseado nas suas pesquisas.

O facto desta aplicação onde se vai enquadrar este mesmo sistema de recomendação ser muito recente, o conjunto de dados criado não é muito extenso sendo para isso necessário injetar dados, de uma forma rigorosa e com critério, para que se possa avaliar o sistema de recomendação.

De agora em diante o conjunto de dados original, ou seja, o que tem os dados reais com as limitações descritas em cima será chamado de *dataset* original, e o conjunto de dados gerado após a injeção criteriosa de dados será chamado de *super dataset*. Usaremos também uma variável importante que define o numero máximo de itens que devem ser removidos do conjunto de dados que será definida como AT.

4.1 Criação do *super dataset*

Para a criação do *super dataset* foi elaborado um algoritmo que fosse capaz de dividir o *dataset* original em três partes idênticas sem nunca perder a sua identidade. O *dataset* tem três colunas: identificação do utilizador, identificação tutorial e força de ligação.

Pegou-se na primeira coluna da primeira metade e ligou-se de uma forma aleatória à segunda coluna da segunda metade criando forças de ligação num intervalo superior ao número mínimo e inferior ao número máximo de todos os valores presentes na terceira coluna. Repetiu-se o processo da segunda metade para a terceira metade e da terceira metade para a primeira metade tentado-se ser o mais rigoroso possível na criação de um *super dataset*.

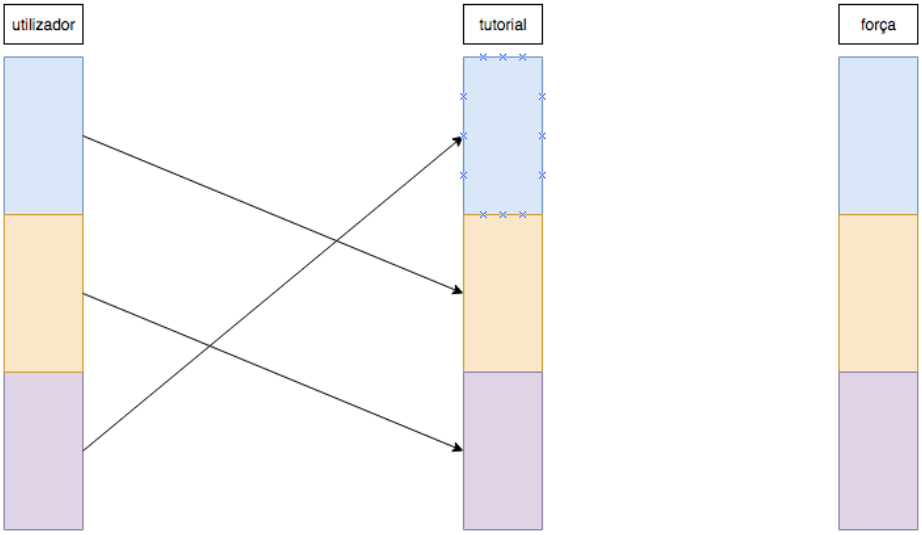

Figura 14 - Esquema de mapeamento de dados na construção do *super dataset*

Para aumentar a complexidade do *super dataset* foram criadas ligações da segunda coluna primeira parte dos dados com a primeira coluna da segunda parte dos dados repetindo o processo da segunda metade para a terceira metade e da terceira metade para a primeira metade. E assim conseguimos enriquecer o nosso *super dataset* para que possamos avaliar o nosso sistema da melhor maneira.

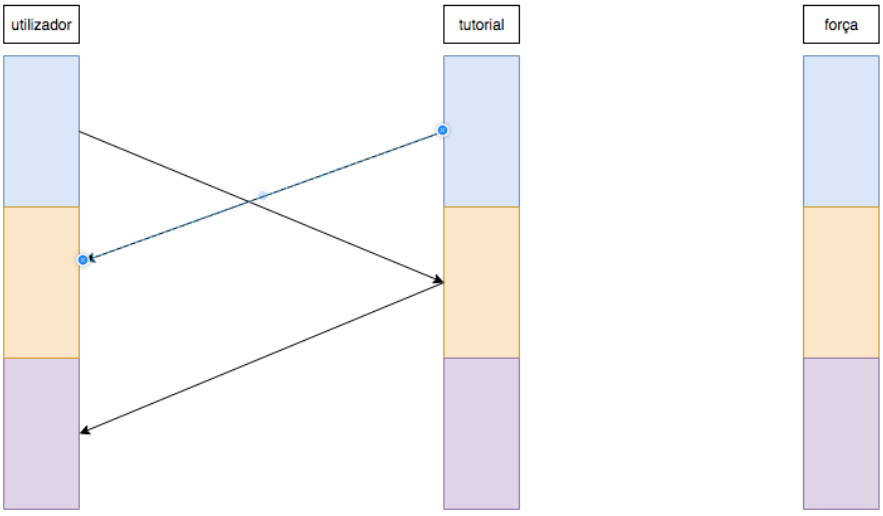

Figura 15 - Esquema do mapeamento de dados, reforçado, na construção do *super dataset*

É de notar que os dados utilizados embora sejam reais foram todos encriptados para números reais positivos num sistema incremental a partir do número um.

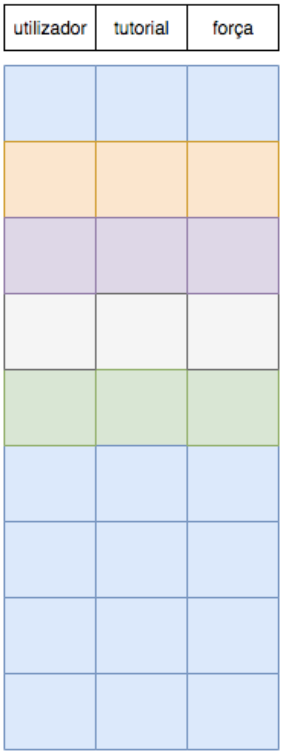

Figura 16 - Esquema final exemplificativo de como fica o *super dataset*

Na figura 16 esta representado o resultado final da criação do super dataset que representa a o cruzamento de dados retratado na figura 14 e 15.

4.2 Sistema de Recomendação

O passo seguinte foi construir o sistema de recomendação propriamente dito. Não foi construído um sistema de raiz, foi usado um dos inúmeros sistemas já existentes uma vez que no início deste trabalho foi referido que este projeto não servia para inventar algo de novo mas sim aplicar algo que já existe a outros propósitos.

A implementação do sistema para avaliação do motor de recomendação exigiu bastante pesquisa e testes de código uma vez que existia alguma complexidade.

Após todo este processo de adaptação, foi-se construindo o sistema de recomendação tentando adaptar-se ao que era pretendido. Inicialmente foi construído todo um processo de avaliação de recomendação e só depois foi elaborado o sistema de recomendação propriamente dito, que está assente essencialmente em três pontos fundamentais: divisão em um modelo de treino e outro de itens relevantes, cálculo de todas as

recomendações dos modelos de treino por utilizadores, comparação do que é gerado dessa recomendação com os itens relevantes que são gerados.

4.2.1 Modelo de Treino e Relevantes

Neste ponto foi aproveitado o algoritmo que já fazia isso em outros *dataSets*, o propósito deste algoritmo era correr todo o *super dataset* e por cada utilizador criar um documento semelhante onde se retiravam algumas ligações com determinados itens.

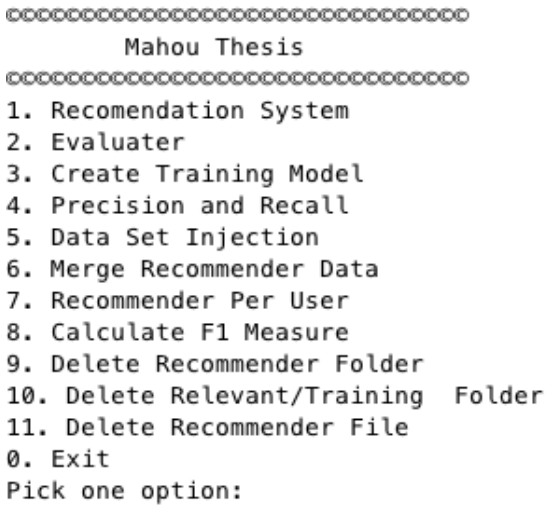

Figura 17 - Menu do sistema de recomendação

Para este primeiro passo era escolhido a opção numero três onde se cria o modelo.

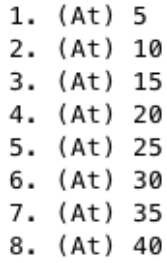

Figura 18 - Submenu do sistema de recomendação

Em seguida, como se pode ver na figura 18, corresponde ao submenu da figura 17, escolheu-se qual o AT ideal para esta solução. Este AT representa o número máximo de elementos que podem estar presentes em cada pasta de cada utilizador no modelo dos relevantes. Num sistema perfeito e a ter em conta em trabalhos futuros esta variável, AT, pode ser calculada pelo próprio sistema encontrando o máximo de elementos presente nas pastas do relevantes e testando este algoritmo em intervalos equivalentes.

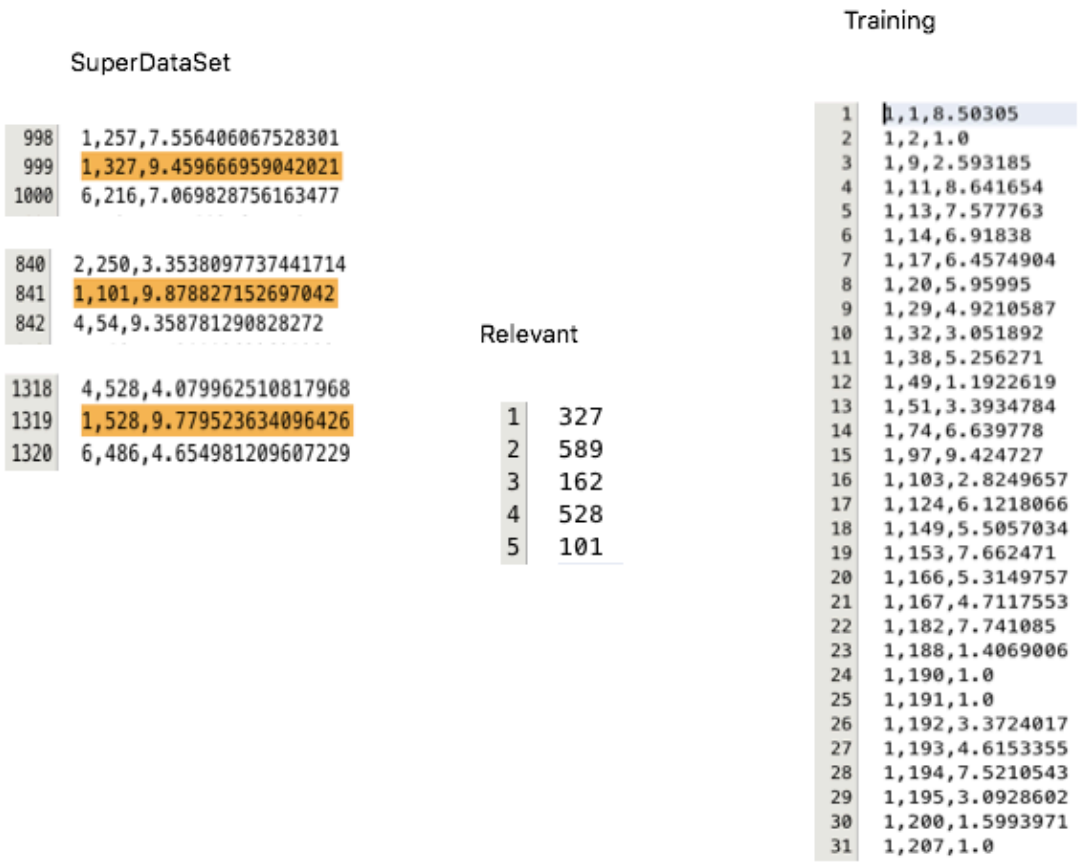

Figura 19 - Exemplo de criação de relevantes e *datasets* de treino

Como podemos ver na figura acima usamos como por exemplo o utilizador 1 para demonstrar o funcionamento da divisão do *super dataset* em modelos de treino e relevantes.

Podemos ver os *id's* presentes na lista dos relevantes que estavam no *super dataset* foram removidos da lista de treinos. Isto serve para que se possa avaliar o nosso sistema segundo a capacidade de recomendar.

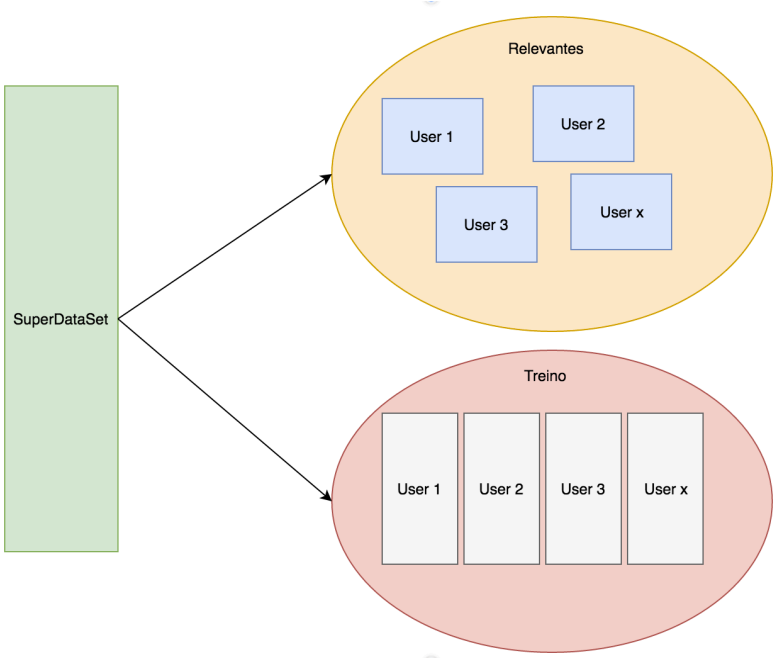

Figura 20 - Ilustração de divisão em relevantes e treino

Este algoritmo cria para cada utilizador uma pasta que contém os elementos relevantes retirados e um ficheiro de treino que é muito idêntico ao *super dataset* exceto que não contenham os elementos presentes na pasta dos relevantes como ilustrado na figura 20. A única variável de entrada para esta parte do algoritmo é o AT e para satisfazer esta condição foi necessário calcular a média da força de ligação de cada utilizador com os seus tutoriais e a partir desta média foram retirados todos os elementos em que a força se aproximasse desta média até satisfazer a nossa condição. Claro que determinado utilizador poderia não ter o número de ligações exigido pelo AT, e neste caso o ficheiro dos relevantes apenas era preenchido com o que existissem ou em caso de o número de ligações ser inferior a dois era descartado uma vez que não teria qualquer relevância para a simulação porque era eliminado.

4.2.2 Recomendação por Relevantes

Para este ponto foi criado um algoritmo capaz de criar recomendações segundo similaridades selecionadas. Após a seleção da opção de recomendação pelo utilizador do programa era pedido ao utilizador para escolher o tipo de similaridade pretendida.

```
1. (User) PearsonCorrelationSimilarity
2. (User) TanimotoCoefficientSimilarity
3. (User) LogLikelihoodSimilarity
4. (Item) PearsonCorrelationSimilarity
5. (Item) TanimotoCoefficientSimilarity
6. (Item) LogLikelihoodSimilarity
```

Figura 21 - Submenu do sistema de recomendação (Recomendação por relevantes)

Como está representado na figura 21 o utilizador poderia escolher o algoritmo pretendido baseado no utilizador ou em elemento e dentro desta opção poderia também escolher a similaridade :

- PearsonCorrelationSimilarity
- TanimotoCoefficientSimilarity
- LogLikelihoodSimilarity

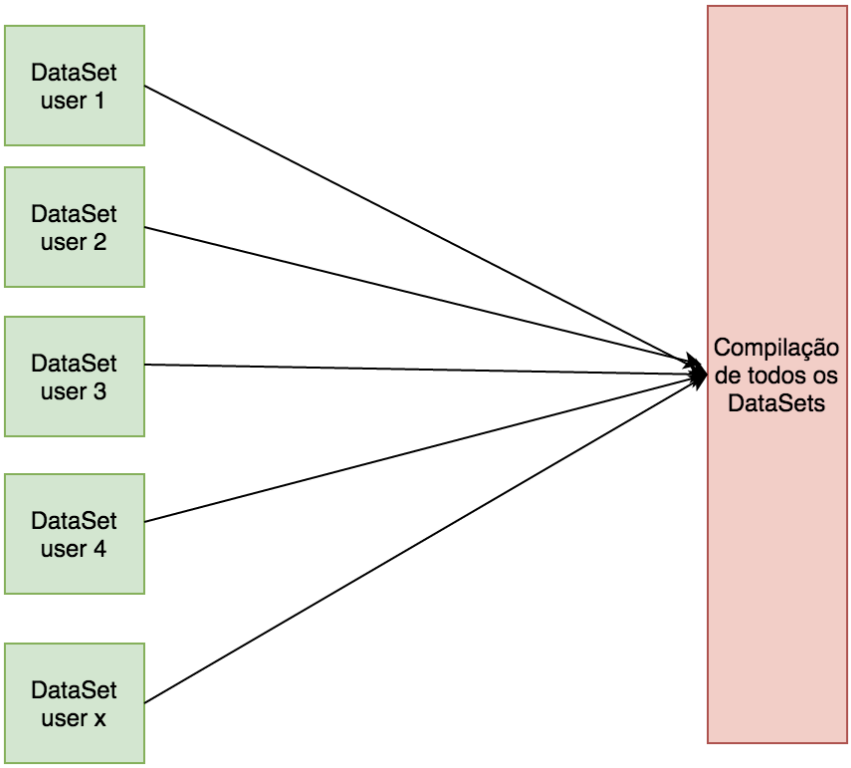

Figura 22 - Ilustração de recomendação de cada *dataset* de treino e junção num único.

O agregado de todos os resultados, fruto da recomendação de todos os utilizadores é concentrado num só ficheiro, como se pode ver na imagem 22, para ser usado pelo último ponto deste sistema de recomendação.

Para esta parte do algoritmo as variáveis de entrada são o tipo de algoritmo usado, se é baseado em utilizador/elemento e que tipo de similaridade escolhido, e o grau de vizinhança, caso se use o algoritmo baseado em utilizador. Neste ponto o Mahout tem funções específicas capazes de calcular e devolver as recomendações baseadas em utilizador ou elemento assim como calcular segundo uma vizinhança, no caso do algoritmo baseado em utilizadores. Embora fossem funções explícitas do *Mahout* usadas nesta parte do algoritmo, foram testadas várias hipóteses tanto ao nível de similaridade como ao nível de vizinhança em que o parâmetro poderia variar. Esta parte do algoritmo seria a mais importante para que pudéssemos chegar ao algoritmo capaz de obter melhor resultados para o *software* em questão.

4.2.3 Comparação

Neste último ponto foi elaborado um algoritmo capaz de comparar o resultado do ponto 4.2.1 com os resultados do 4.2.2. O objectivo seria coletar uma amostra e retirar essa mesma amostra do conjunto de dados do utilizador. O objetivo era assegurar que o sistema de recomendação era capaz de recomendar alguns ou mesmo todos os elementos da amostra.

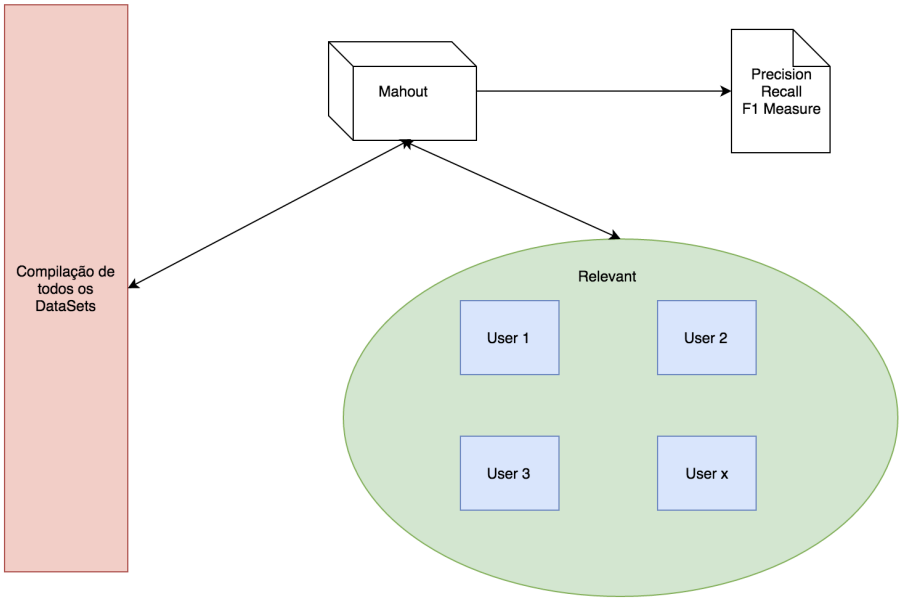

Figura 23 - Ilustração da fase final de recomendação.

Como se pode ver na figura 23 o sistema *Mahout* vai ao s*uper dataset* buscar todas as ligações de um determinado utilizador comparando o tutorial a que este utilizador está ligado e vai, ainda comparar com os elementos que estão na pasta dos relevantes.

Após coletar esta informação é calculado as medidas de *precision* e *recall* e a partir destas calcula-se a F1 *maesure* pela qual a nossa conclusão foi guiada.

Como é óbvio não era possível tirar alguma conclusão se tivéssemos apenas usado o *dataset* original para esta recomendação porque não teríamos nenhum ponto de comparação para avaliarmos esta situação. Por isso usamos este mesmo sistema para outro *dataset* conhecido e documentado para atestar a viabilidade do mesmo.

4.3 Sumário

Neste capitulo conseguiu-se descrever os passos seguidos para a construção deste sistema. De salientar o enriquecimento do conjunto de dados de maneira a poder obter um melhor rendimento do sistema, uma vez que este tipo de sistema só começa a ter resultados satisfatórios com um número elevado de dados. Este processo, que está automatizado, foi corrido algumas vezes para que se pudesse tirar o melhor partido dos resultados. É no próximo capitulo que será demonstrado os resultados obtidos desta implementação,

5 Avaliação

O presente capítulo têm como objetivo apresentar os principais resultados e as conclusões obtidas neste trabalho demonstrando desta forma se o objetivo foi atingido e quais as limitações.

5.1 Resultados

Será neste ponto que será mostrado e discutido o resultado a que chegamos através dos pontos anteriores.

Para avaliar da melhor maneira esta implementação foi necessário efetuar vários testes com valores de entrada diferentes para que fosse possível chegar a uma conclusão. Foram usados vários valores para AT para cada algoritmo baseado em utilizador ou elemento bem como para cada tipo de similaridade como poderemos ver nos pontos seguintes.

5.1.1 At = 5

Para um valor de AT igual a 5 obtivemos os valores representados na tabela seguinte:

Tabela 2 - Recomendação baseada em utilizador para AT=5

	Pearson(User)	Tanimoto(User)	LogLikelihood(User)
Precision	0,0487	0,2021	0,0252
Recall	0,3628	0,3167	0,3718
F1 Measure	0,0858	0,2467	0,0473

Para estes valores foi criado um gráfico para que se pudesse ter uma noção mais clara dos valores obtidos.

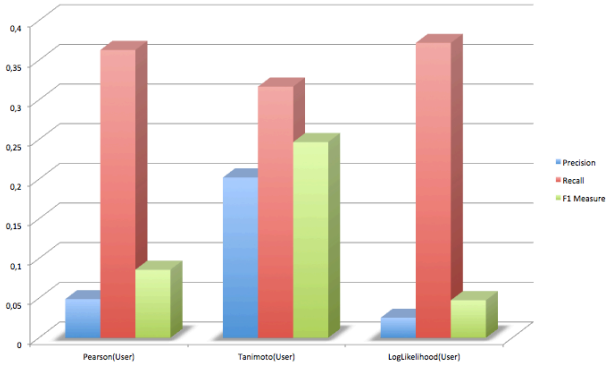

Figura 24 – Gráfico de recomendação baseada em utilizador para AT=5

Como se pode observar no gráfico representado na figura 24, para os três tipos de similaridade, Pearson, Tanimoto e Likelihood, entende-se de forma clara que o melhor algoritmo baseado em utilizadores seria o coeficiente de Tanimoto porque é o que atingiu o *F1 measure* maior. Embora o algoritmo que usa o coeficiente de Likelihood tenha um *recall* maior têm um *precision* muito baixo o que faz com que o seu *F1 measure* seja inferior ao do Tanimoto

Tabela 3 - Recomendação baseada em elemento para AT=5

	Pearson(Item)	Tanimoto(Item)	LogLikelihood(Item)
Precision	0,0463	0,0130	0,0137
Recall	0,4062	0,3747	0,3995
F1 Measure	0,0831	0,0251	0,0265

Para estes valores foi criado um gráfico para que se pudesse ter uma noção mais clara dos valores obtidos.

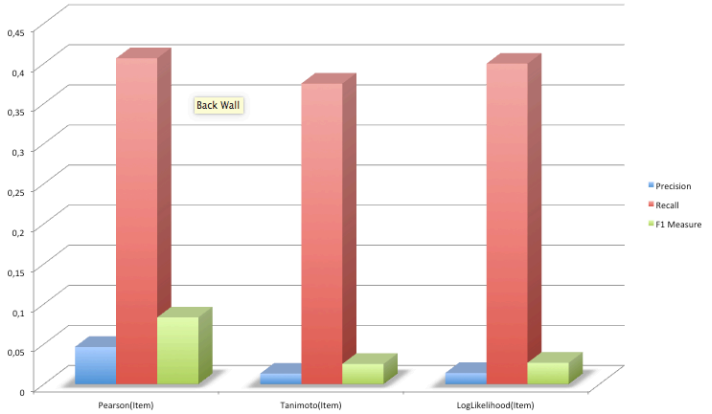

Figura 25 – Gráfico de recomendação baseada em elemento para AT=5

Neste gráfico representado na figura 25, para os mesmos três tipos de similaridade, o melhor algoritmo baseado em elementos seria o coeficiente de Pearson porque é o que atingiu o *F1* measure maior. Este caso deveu-se por ter um *precision* maior relativamente aos outros uma vez que o *recall* são em todos muito semelhantes.

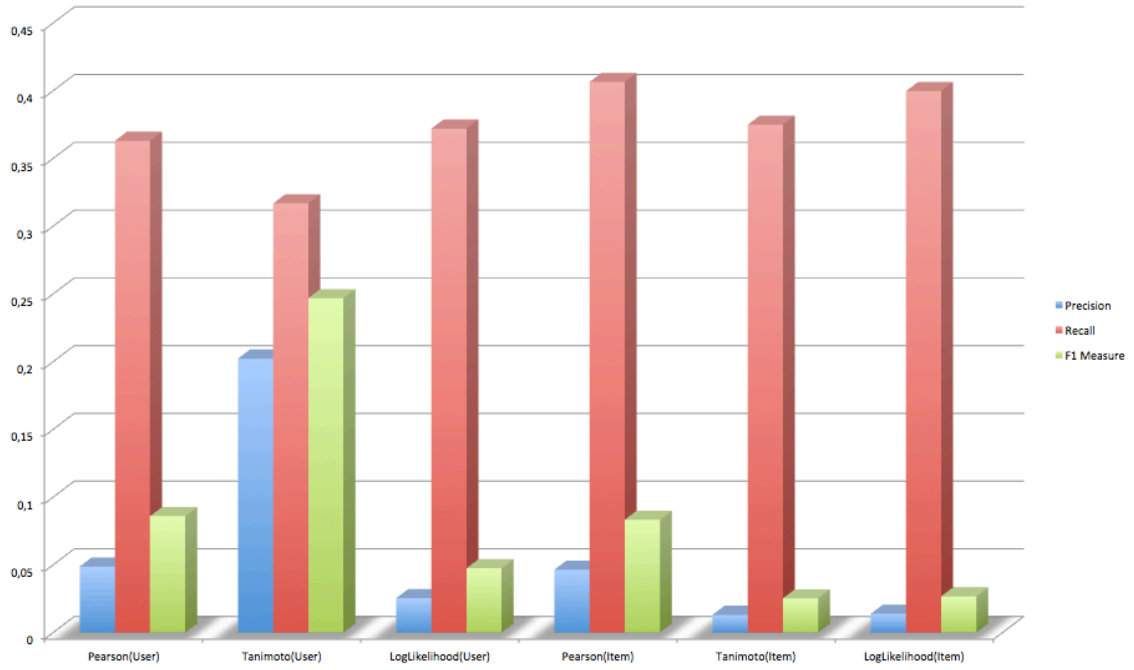

Figura 26 - Gráfico de recomendação para AT=5

No gráfico representado na figura 26, que é somente a visualização dos gráficos baseados em utilizador e elemento lado a lado, podemos ter uma perceção melhor de qual o algoritmo indicado para um AT igual a 5. Pode ver na evolução da barra azul ao longo do gráfico que o melhor algoritmo seria o Tanimoto baseado em utilizador.

5.1.2 At > 5

Neste ponto apresentaremos o resultado para os algoritmos que tomaram os valores de AT igual a 10, 15, 20, 25, 30, 35 e 40 uma vez que para todos eles se obteve sem o mesmo resultado. Apresenta-se de seguida os respetivos valores e a sua representação gráfica.

Tabela 4 - Recomendação baseada em utilizador para AT > 5

	Pearson(User)	Tanimoto(User)	LogLikelihood(User)
Precision	0,0514	0,1552	0,0263
Recall	0,3775	0,3291	0,37566
F1 Measure	0,0906	0,2109	0,04924

Tabela que corresponde ao gráfico:

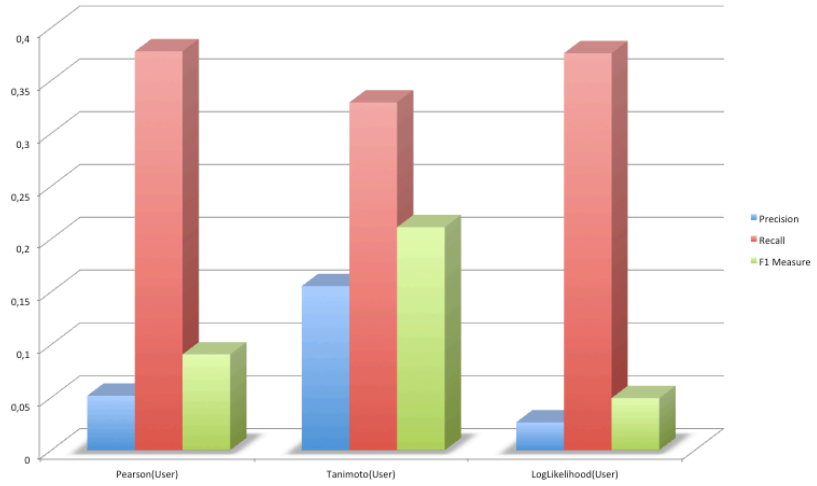

Figura 27 – Gráfico de recomendação baseada em utilizador para AT > 5

Como se pode ver o gráfico representado na figura 27 para os três tipos de similaridade, Pearson, Tanimoto e Likelihood, entende-se de forma clara que o melhor algoritmo baseado em utilizador para este AT´s seria o coeficiente de Tanimoto porque é o que atingiu o *F1 measure* maior.

Tabela 5 - Recomendação baseada em elemento para AT > 5

	Pearson(Item)	Tanimoto(Item)	LogLikelihood(Item)
Precision	0,0465	0,0132	0,0138
Recall	0,4036	0,3774	0,4032
F1 Measure	0,0834	0,0256	0,0267

Com a representação gráfica:

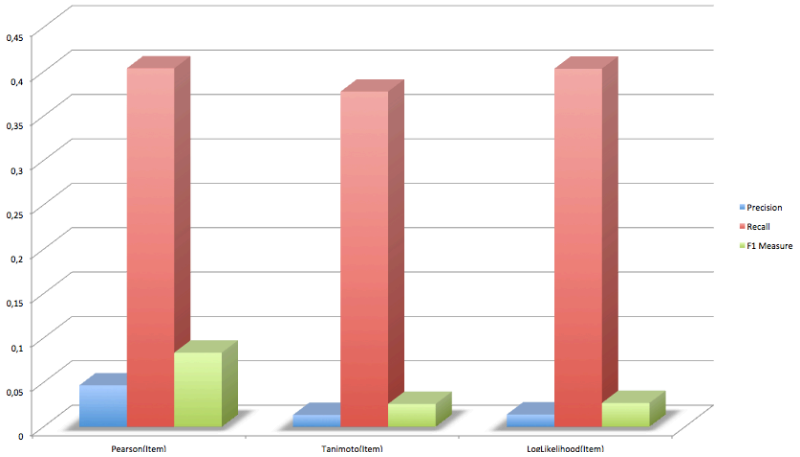

Figura 28 – Gráfico de recomendação baseada em elemento para AT= a 10,15,20,25,30,35 e 40

Neste gráfico representado na figura 28 para os mesmos três tipos de similaridade, o melhor algoritmo baseado em elementos seria o coeficiente de Pearson porque é o que atingiu o *F1 measure* maior. Este caso deveu-se por ter um *precision* maior relativamente aos outros uma vez que o *recall* são em todos muito semelhantes.

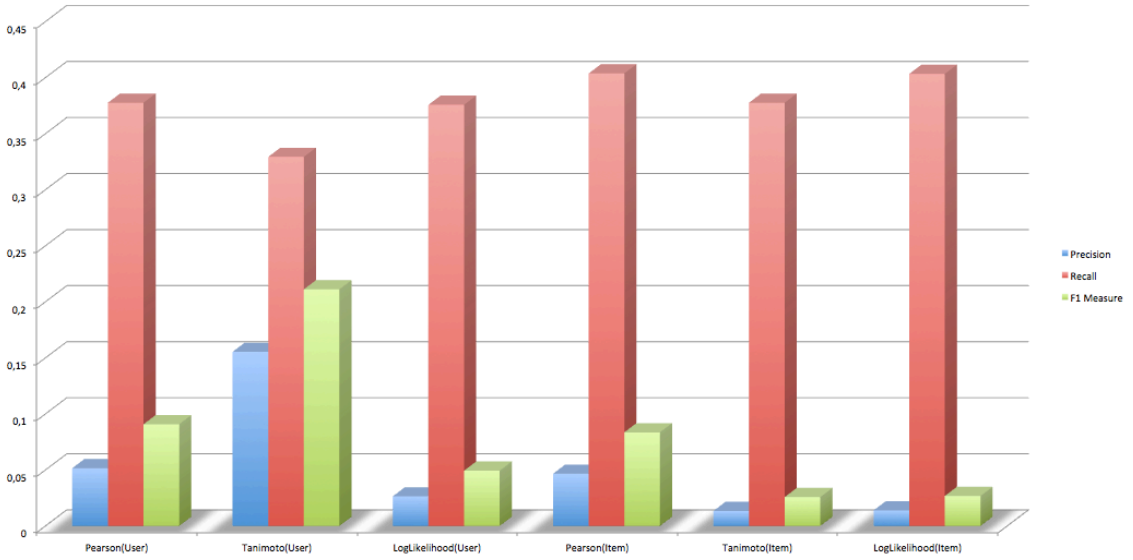

Figura 29 - Gráfico de recomendação para AT > 5

Como se pode ver evolução da barra azul ao longo do gráfico representado na figura 29 o melhor algoritmo seria o Tanimoto baseado em utilizador comparativamente com todos as variáveis para AT igual a 10,15,20,25,30,35 e 40 .

5.1.3 Análise

Através destes resultados obtidos e ilustrados nos pontos anteriores através de tabelas e gráficos podemos concluir que para este caso específico o melhor algoritmo seria o Tanimoto baseado numa recomendação por utilizadores.

Só foi possível chegar a esta conclusão após vários testes com a entrada de várias variáveis tentando cobrir todos os casos possíveis, correndo o mesmo teste várias vezes para que pudéssemos estabilizar num número em concreto. Embora se tenha chegado a um resultado que se possa considerar satisfatório faltou aqui uma entrada de mais uma variável que terei em conta em trabalhos futuros, algoritmo segundo vizinhança no caso de ser baseada em utilizadores. Neste sentido e baseado em utilizadores não podemos concluir que este será com toda a certeza o melhor algoritmo, mas pode-se afirmar que se aproxima de uma forma quase certa do algoritmo ideal para o nosso caso de estudo.

5.2 Resultados (Tagged book marks)

Para garantir a fiabilidade e estabilidade do nosso programa foi necessário testar o mesmo usando um *dataset* conhecido para que se pudesse garantir a veracidade do sistema. Neste caso e após cálculos chegou-se a conclusão que o AT que pudesse devolver os melhores resultados estaria entre 30 e 35.

5.2.1 At = 30

Apresentaremos neste subcapítulo os resultados obtidos para um AT igual a 30.

Tabela 6 - Recomendação baseada em utilizador para AT= 30

	Pearson(User)	Tanimoto(User)	LogLikelihood(User)
Precision	0,322975236	0,599008886	0,107625377
Recall	0,115424686	0,407306397	0,135319197
F1 Measure	0,170069899	0,484898034	0,119893845

Para estes valores foi criado um gráfico para que se pudesse ter uma noção mais clara dos valores obtidos.

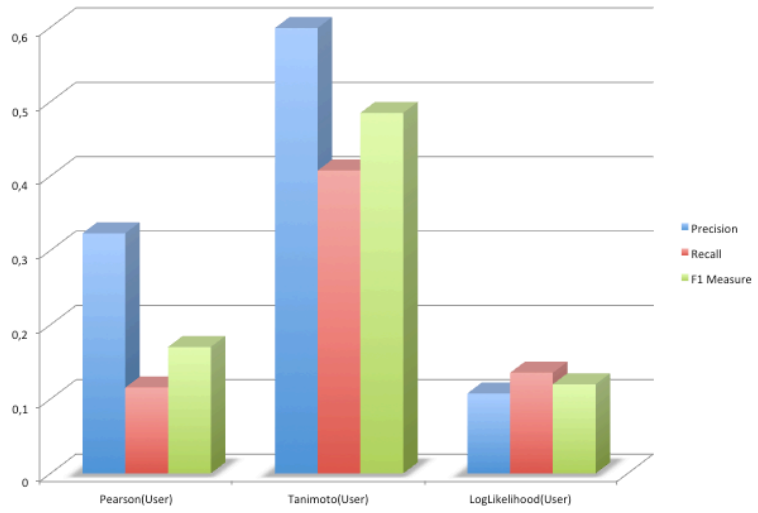

Figura 30 –Gráfico de recomendação baseada em utilizador para AT= 30

Como se pode observar na figura 30 relativamente à tabela 6 que representa os valores de *precision* e *recall* baseado em utilizadores o algoritmo de Tanimoto apresenta melhores resultados comparativamente aos outros algoritmos.

Tabela 7 - Recomendação baseada em elemento para AT= 30

	Pearson(item)	Tanimoto(item)	LogLikelihood(item)
Precision	NaN	NaN	NaN
Recall	NaN	NaN	NaN
F1 Measure	NaN	NaN	NaN

Na tabela 7 estão representados os valores obtidos após correr o algoritmo com base em itens. Como se pode ver não foram obtidos quaisquer valores para qualquer variável. Neste ponto não há nenhuma tese que fundamente este resultado deixando para trabalho futuro a fundamentação do mesmo.

5.2.2 At = 35

Tabela 8 - Recomendação baseada em utilizador para AT= 35

	Pearson(User)	Tanimoto(User)	LogLikelihood(User)
Precision	0,3317	0,5990	0,1076
Recall	0,1183	0,4073	0,1353
F1 Measure	0,1745	0,4848	0,1198

Para estes valores foi criado um gráfico para que se pudesse ter uma noção mais clara dos valores obtidos.

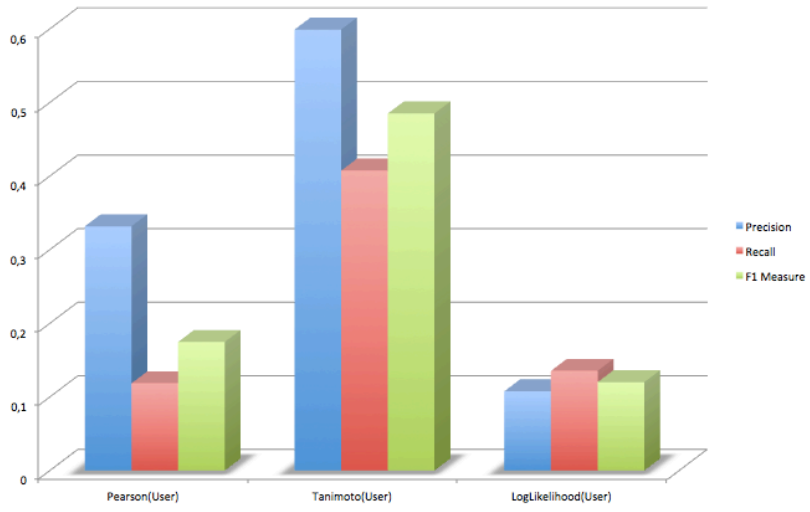

Figura 31 - Gráfico de recomendação baseada em utilizador para AT= 35

Também para este caso os valores que o algoritmo de Tanimoto, baseado em utilizadores, retorna é que obtém melhor resultado em relação aos outros algoritmos.

Tabela 9 - Recomendação baseada em elemento para AT= 35

	Pearson(items)	Tanimoto(items)	LogLikelihood(items)
Precision	NaN	NaN	NaN
Recall	NaN	NaN	NaN
F1 Measure	NaN	NaN	NaN

A semelhança dos resultados obtidos para um AT igual a 30 também nesta simulação o resultado obtido não foi satisfatório não encontrando nenhuma explicação para este facto.

5.3 Avaliação Temporal

Outra das grandezas que foi testada foi o tempo. Como foi apresentado no capítulo da desenho em que a mesma vai variar dependendo do tempo de processamento de cada arquitetura. O objetivo é que o sistema de recomendação seja o mais rápido possível e que seja capaz de processar todos os dados em tempo útil. Para isso o tempo será uma medida que se têm de ter em consideração para que o sistema vá de encontro às necessidades do utilizador.

Para o cálculo desta grandeza foi medido o tempo demorado por cada algoritmo a completar um ciclo de recomendação e avaliação.

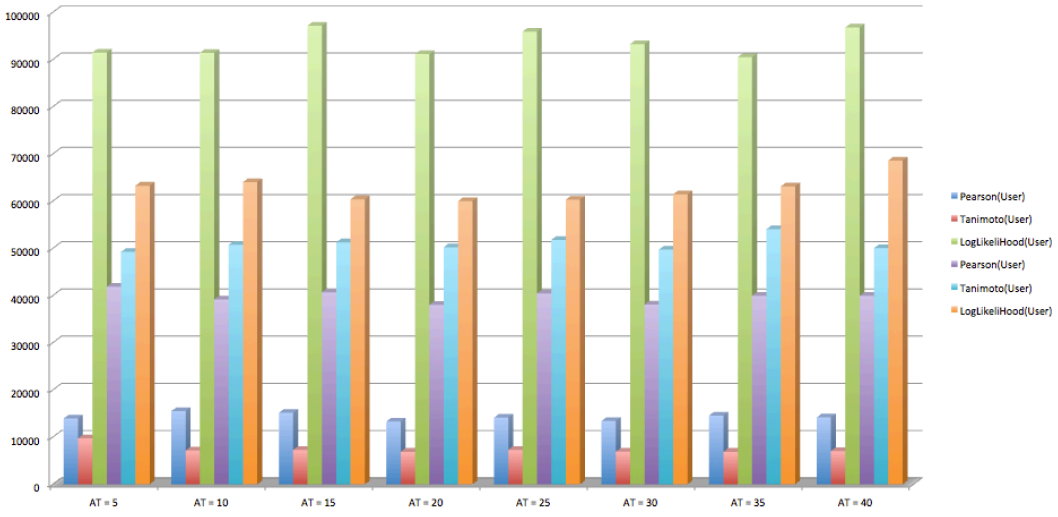

Figura 32 - Gráfico tempo decorrido por algoritmo , recomendação e avaliação (milissegundos)

Segundo a premissa do parágrafo anterior notou-se algumas diferenças com maior impacto no algoritmo de Log Likelihood em relação aos outros dois, tanto nos algoritmos baseados em item como nos baseados em utilizadores. Como se pode

verificar na figura 32, com os tempos em milissegundos, o algoritmo de Tanimoto baseado em utilizador é que obtém melhor comportamento relativamente aos outros que foram testados. Este algoritmo é sempre melhor temporalmente, independentemente do AT utilizado.

Efetuou-se também testes unicamente no algoritmo que faz a recomendação por utilizador para extrair a melhor ou melhores performances neste campo.

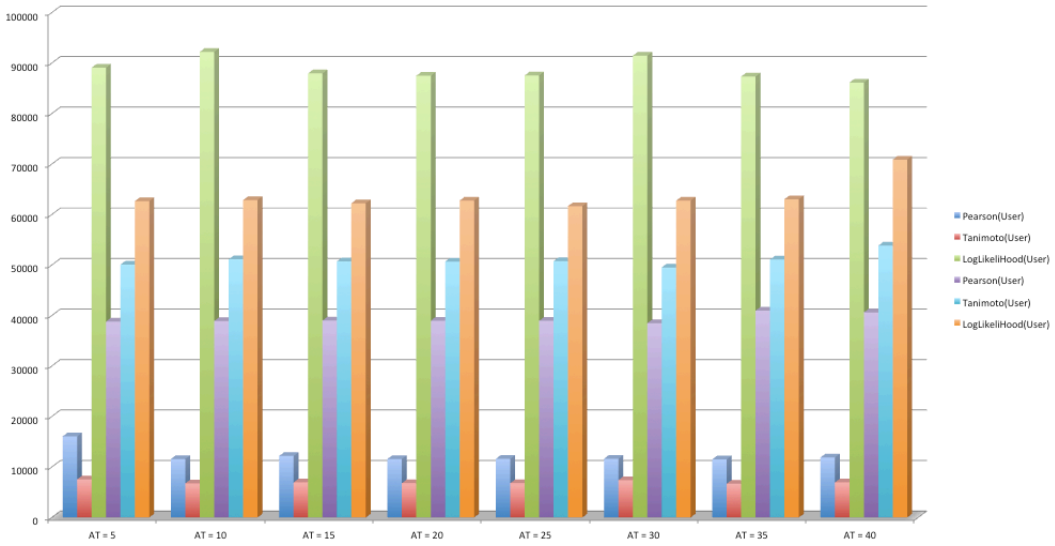

Figura 33 - Gráfico tempo decorrido por algoritmo recomendação (milissegundos)

Como se pode observar na figura 33, os tempos em milissegundos, é muito idêntico aos tempos obtidos no gráfico anterior, da recomendação e avaliação. Isso deve-se ao facto de a avaliação em termos temporal ser quase nula porque o maior esforço e impacto temporal esta presente na recomendação.

Assim e sem surpresas podemos verificar que o algoritmo de Tanimoto baseado em utilizador é o que obtém melhores resultados quando submetido a testes nas mesmas condições que os outros algoritmos.

5.4 Avaliação dos Utilizadores

De encontro a obtenção de um outro objetivo de estudo, outra das grandezas que foi avaliada foi a satisfação do utilizador com base em inquéritos. Este tipo de avaliação não foi finalizada por alguns imprevistos e uma gestão pouco cuidada do tempo. No início deste projeto foi definido um exemplo de inquérito para que os próprios utilizadores pudessem avaliar o mesmo.

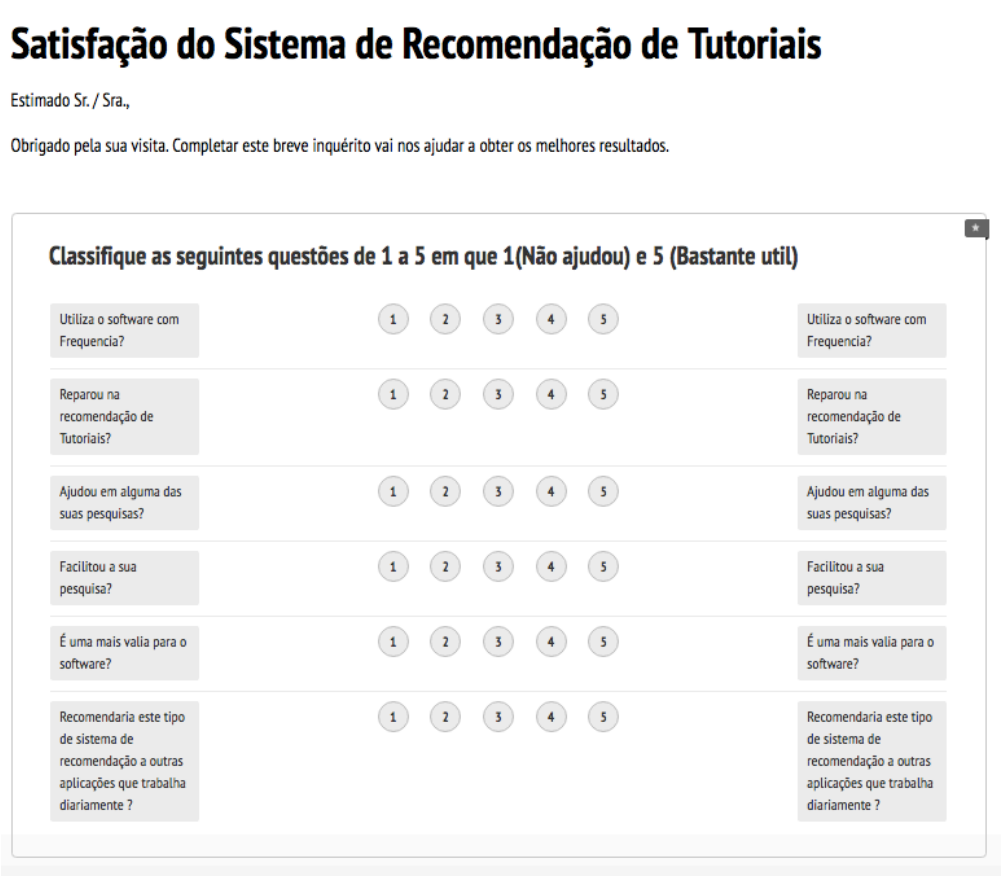

Figura 34 - Exemplo de inquérito de avaliação

A figura 34 representa o inquérito de satisfação a aplicar, como método de avaliação deste sistema junto dos utilizadores. Para a elaboração deste questionário, recorreu-se à plataforma online, acessível através do endereço.[1]

5.5 Sumário

Neste capítulo ficou vincado que para este tipo de recomendação o algoritmo de Tanimoto é o mais indicado, não só na eficiência, mas também na rapidez que é executado. Usou-se um outro conjunto de dados para que se pudesse verificar que o mesmo sistema de recomendação esta apto para outros sistemas apesar de, neste momento, estarmos a adaptar ao nosso sistema de gestão de tutoriais. Foi também apresentado neste capítulo uma grandeza de satisfação recorrendo a inquéritos de satisfação de utilizadores.

[1] http://www.survio.com/survey/d/V3T3A3C3X6Z4P5M5V

6 Conclusão

Este estudo teve como objetivos a pesquisa, escolha e implementação de um sistema de recomendação adequado a um sistema de gestão de tutoriais.

Em primeiro lugar procedeu-se ao levantamento bibliográfico de sistemas de recomendação através de revistas, artigos e blogues. Pesquisas de *frameworks* onde se listou os pontos fortes e fracos de maneira a que fosse possível escolher a mais indicada, sabendo que no futuro seria ideal também comparar estes resultados com outras *frameworks*. Só após a idealização de um sistema de recomendação, recorrendo ao uso de diagramas UML, é que se estava em condição de implementar a solução, que foi descrita neste documento. Após o cruzamento dos resultados com a informação recolhida na pesquisa efetuada relativamente a sistemas de recomendação, se pudesse concluir que foram atingidos os objetivos que foram propostos. Conclui-se, assim, com os resultados encontrados nos pontos anteriores bem como os resultados demonstrados nas tabelas e nos gráficos do ponto de avaliação, que se pode afirmar que o algoritmo mais eficaz para este problema é o algoritmo de Tanimoto baseado em utilizadores.

Através da análise de tempo/performance conclui-se que usando a mesmo máquina – 1,3 GHz Intel Core i5 com 8GB de memória RAM – , e com as mesmas condições, o algoritmo de Tanimoto baseado em utilizador conclui a tarefa pretendida num espaço de tempo mais curto do que os outros algoritmos. Do mesmo modo, e em relação à avaliação feita sobre a cobertura de resultados, conclui-se que o algoritmo de Tanimoto baseado em utilizador é também o mais eficaz porque nos testes realizados completou a tarefa com uma maior eficácia em relação aos restantes algoritmos. Este resultado define a capacidade de uma melhor qualidade na recomendação para este propósito.

Assim e depois desta análise pode-se concluir que o sistema de recomendação a usar deve ser o algoritmo de Tanimoto baseado em utilizador.

Atingindo assim o objetivo principal deste trabalho que seria desenhar uma arquitetura e desenvolver um sistema de recomendação mais rápido e eficiente este *software* de gestão de tutoriais.

6.1 Limitações

No decorrer deste trabalho houve algumas limitações encontradas, que não estando previstas, tiveram impacto na obtenção dos resultados, nomeadamente nos tempos determinados, comprometendo alguns objetivos que se tinham traçado inicialmente.

Apesar de existirem muitos artigos disponíveis relativamente a sistemas de recomendação, grande parte desses artigos não dispõe de dados que foram usados para chegar às suas conclusões, que dificulta a análise dos próprios sistemas, uma vez que não é possível usar os mesmos dados para tirar as conclusões que estavam descritas.

No início deste projeto, quando foi desenhada a arquitetura, estudou-se a possibilidade de testar outro tipo de arquitetura diferente e analisar qual delas seria a mais indicada para este sistema. A arquitetura inicial fazia a leitura de dados da base de dados através do sistema de gestão de tutoriais e a outra arquitetura deixava as responsabilidades CRUD para o sistema de recomendação alojado no servidor Mahout. Esta análise acabou por não acontecer, não se conseguindo assim escolher a melhor arquitetura para este caso.

Outra das limitações que surgiu foi a dificuldade na gestão do tempo, dado que o tempo decorrido durante a fase de testes de algoritmos para várias condições foi demasiado demorado, o que não permitiu chegar a conclusões rapidamente o que era fundamental para poder completar o sistema de recomendação na sua totalidade.

6.2 Trabalho Futuro

Devido a algumas condicionantes, retratadas na secção anterior, como a má gestão do tempo, a demora em correr os algoritmos, não foi possível finalizar na totalidade o que era proposto no início deste trabalho. Para que este trabalho atinja um nível de complexidade superior poder-se-á acrescentar algoritmos para fazer mais testes, para isso deve-se correr novos testes recorrendo aos algoritmos similaridade da distância euclidiana, similaridade city-block, similaridade de correlação Spearman para além dos testados neste trabalho.

Um dos objetivos propostos era a interligação de todos os sistemas, o que não foi possível realizar com sucesso, tarefa essa que será executada no futuro para atingirmos

em pleno os objetivos descritos inicialmente, ou seja, finalizar a interligação entre o programa de gestão de tutoriais e o sistema de recomendação.

Seria interessante que no futuro se pudesse recorrer a outras *frameworks*, que, embora tenha sido feita uma pesquisa das *frameworks* existentes só houve uma avaliação mais detalhada de uma pequena amostra selecionada através dos artigos lidos. Seria importante testar em outras *frameworks* , como, PredictionIO, Racoon Recommendation Engine, Seldon, para que se pudesse tirar um maior partido neste sistema de recomendação bem como outros sistemas de recomendação no futuro.

Também como trabalho futuro pode-se incluir nos algoritmos a pesquisa por vizinhança – *neighborhood* – podendo ser de dois tipos: NearestNUserNeighborhood que calcula vizinhos constituído pelos *n* utilizadores mais próximos de um determinado utilizador e ThresholdUserNeighborhood que calcula vizinhos constituídos por todos os utilizadores cuja semelhança com o utilizador dado atende ou excede um determinado limite.

Em relação a avaliação deste trabalho no sentido de criação de valor seria interessante que no futuro pudéssemos aprofundar a utilização do método AHP.

Além disso, será essencial, concluir este trabalho tendo a perceção do utilizador, pela aplicação do inquérito de satisfação realizado.

6.3 Apreciação Final

Após a realização deste trabalho posso concluir que houve um acréscimo de conhecimento adquirido e que foi um desafio muito importante na minha carreira académica. A inteligência artificial é um campo que me motiva e por esse motivo foi com prazer que tive a possibilidade de desenvolver um sistema de recomendação. Mas foi através da pesquisa que consegui assimilar o melhor do que se faz e se têm feito nesta área e sobretudo o que ainda se pode fazer. Embora se vá usar um sistema de recomendação para um caso muito específico, toda esta experiência que advém dos resumos de vários artigos citados no estado da arte, contribui para uma valorização pessoal e do projeto tendo adquiro técnicas ao nível da abordagem, análise, avaliação e conclusão.

Bibliografia

[1] (2016, 01). What is machine Learning. whatis.techtarget. Retirado 10, 2016, de http://whatis.techtarget.com/definition/machine-learning

[2] Jones, (2013, 12). Introduction to approaches and algorithms. IBM. Retirado 10, 2016, de https://www.ibm.com/developerworks/library/os-recommender1/

[3] Ciriaco, D. (2008, 11). O que é Inteligência Artificial?. tecmundo. Retirado 10, 2016, de http://www.tecmundo.com.br/intel/1039-o-que-e-inteligencia-artificial-.htm

[4] Pozzebon, R. (2013, 05). O que é Inteligência Artificial?. oficina da net. Retirado 10, 2016, de https://www.oficinadanet.com.br/artigo/ciencia/o-que-e-inteligencia-artificial

[5] Arruda, G. (2015, 01). Inteligência Artificial: o que é aprendizado de máquina?. manual do usuario. Retirado 10, 2016, de http://www.manualdousuario.net/inteligencia-artificial-aprendizado-de-maquina/

[6] Ekstrand, J. (2010). Collaborative Filtering Recommender Systems (ed., Vol. 4, pág.).

[7] Sorensen, S. (2010). Accuracy of Similarity Measures in Recommender Systems (1ª ed., Vol., pág.). Minnesota, Morris.

[8] A. B. Alan Said, "Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks," Universidad Autónoma de Madrid, TU-Delft The Netherlands, 2014.

[9] S. Bagchi, "Performance and Quality Assessment of Similarity Measures in Collaborative Filtering Using Mahout".

[10] (2016, 09). GraphLab Create. Turi. Retirado 10, 2016, de https://turi.com/products/create/

[11] S. Bagchi, "Performance and Quality Assessment of Similarity Measures in Collaborative Filtering Using Mahout," Indian Institute of Technology, 2015.

[12] S. Bagchi, “Performance and Quality Assessment of Similarity Measures in Collaborative Filtering Using Mahout,” Indian Institute of Technology, 2015.

[13] edureka, (2014, 09). Tanimoto Coefficient in Mahout. edureka. Retirado 10, 2016, de http://www.edureka.co/blog/tanimoto-coefficient-in-mahout/

[14] (2014). Coeficiente de correlação de Pearson. wikipedia. Retirado 10, 2016, de https://pt.wikipedia.org/wiki/Coeficiente_de_correla%C3%A7%C3%A3o_de_Pearson.

[15] (2016, 01). Mining Similarity Using Euclidean Distance, Pearson Correlation, and Filtering. humanoriented. Retirado 02, 2016, de http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/mvoget/similarity/similarity.html

[16] Tekinomo, K. (2016). City Block Distance. people revoledu. Retirado 10, 2016, de http://people.revoledu.com/kardi/tutorial/Similarity/CityBlockDistance.html

[17] (2016). Correlation (Pearson, Kendall, Spearman). statistics solutions. Retirado 10, 2016, de http://www.statisticssolutions.com/correlation-pearson-kendall-spearman

[18] (2016). MongoDB. MongoDB. Retirado 10, 2016, de https://www.mongodb.com/

[19] CofiRank, “cofirank,” [Online]. Available: https://sites.google.com/a/cofirank.org

[20] Duncan, G. (2011, 11). MyMediaLite Recommender System Library. MyMediaLite. Retirado 01, 2016, de https://channel9.msdn.com/coding4fun/blog/MyMediaLite-Recommender-System-Library

[21] Gantner, Z. (2011, 02). MyMediaLite Recommender System Library. mymedialite. Retirado 01, 2016, de http://pt.slideshare.net/zenogantner/mymedialite

[22] L. Saaty, T. (2008, 10). Decision making with the analytic hierarchy process . Services Sciences, 1, 83-98.

[23] J. Nilsson, N. (1998). Artificial Intelligence: A New Synthesis (ed., Vol., pág.). San Francisco, California: Morgan Kaufmann Publishers.

[24] (2016, 01). Precision and Recall. wikipedia. Retirado 01, 2016, de https://en.wikipedia.org/wiki/Precision_and_recall

[25] Joy, J. (2006, 07). Win-Win Negotiation. ezinearticles. Retirado 01, 2016, de http://ezinearticles.com/?Win-Win-Negotiation&id=240772

[26] (2016, 01). Coeficiente de correlação de postos de Spearman. wikipedia. Retirado 01, 2016, de https://pt.wikipedia.org/wiki/Coeficiente_de_correla%C3%A7%C3%A3o_de_postos_de_Spearman

[27] Shani, G. (2005). An MDP-Based Recommender System. Journal of Machine Learning Research , (6), 1265–1295, 1265-1267.